Join the discussion @ p2p.wrox.com

涵盖Visual C# 2012

.NET开发经典名著

Beginning Visual C# 2012 Programming

C#入门经典（第6版）

2011年度
2009年度 全行业优秀畅销品种

Karli Watson　Jacob Vibe Hammer
[美] Jon D. Reid　Morgan Skinner　著
Daniel Kemper　Christian Nagel
齐立波　黄俊伟　译
黄　静　审校

清华大学出版社

.NET 开发经典名著

C#入门经典

(第 6 版)

[美] Karli Watson
Jacob Vibe Hammer
Jon D. Reid
Morgan Skinner
Daniel Kemper
Christian Nagel
著

齐立波　黄俊伟　译

清华大学出版社

北　京

Karli Watson, Jacob Vibe Hammer, Jon D. Reid, Morgan Skinner, Daniel Kemper, Christian Nagel
Beginning Visual C# 2012 Programming
EISBN：978-1-118-31441-8

北京市版权局著作权合同登记号 图字：01-2013-2566

图书在版编目(CIP)数据

C#入门经典(第 6 版) / (美) 沃森(Watson, K.) 等著；齐立波，黄俊伟 译. —北京：清华大学出版社，2013
(.NET 开发经典名著)
书名原文：Beginning Visual C# 2012 Programming
ISBN 978-7-302-34339-4

Ⅰ. ①C… Ⅱ. ①沃… ②齐… ③黄… Ⅲ. ①C 语言—程序设计 Ⅳ. ①TP312

中国版本图书馆 CIP 数据核字(2013)第 255015 号

责任编辑：王军　韩宏志
装帧设计：孔祥峰
责任校对：成凤进
责任印制：
出版发行：清华大学出版社
网　　址：http://www.tup.com.cn，http://www.wqbook.com
地　　址：北京清华大学学研大厦 A 座　　邮　　编：100084
社 总 机：010-62770175　　邮　　购：010-62786544
投稿与读者服务：010-62776969，c-service@tup.tsinghua.edu.cn
质 量 反 馈：010-62772015，zhiliang@tup.tsinghua.edu.cn
印 刷 者：
装 订 者：
经　　销：全国新华书店
开　　本：185mm×260mm　　印　　张：54.25　　字　　数：1469 千字
版　　次：2013 年 月第 6 版　　印　　次：2013 年 月第 1 次印刷
印　　数：1～5000
定　　价：98.00 元

产品编号：

译 者 序

微软 Windows 平台叱咤 IT 业界几十年，虽然如今苹果、谷歌等为代表的移动设备和互联网大军正在快速发展，但 Windows 平台仍是使用数量最多、应用最广泛的计算机平台。C# 作为这一平台上的主流开发语言，不仅自身在不断完善和发展，也引领着 IT 产业朝着新的开发目标和业务模式前进。

本人从 C#2.0 时代开始学习 Windows 上的应用程序开发，这几年开发了各种规模的项目，大到综合利用 Web 服务、后台服务和 WCF 的业务系统，小到用于提高工作效率的工具性程序。在翻译本书的过程中，学到不少之前没有注意到的知识，接触到包括 Windows 8 在内的最新开发技术，体会颇多，可归纳为以下几点。

本书讲解极其透彻，条理清晰。在翻译本书前，我认真阅读了本书大部分章节，又浏览了国外网站上读者对英文原版的评价，这些读者的感受引起了我的共鸣。精心编排和选材说明作者不仅深刻了解相关技术，还充分为读者着想，绝不是流水账式地一一罗列而已。

本书的内容全面丰富又与时俱进。本书共分五个部分，分别介绍了 C#语言基础、Windows 桌面编程、Web 编程、数据访问以及相关的 WPF 和 WCF 技术。不仅介绍 C#语言，还占用很多篇幅介绍了 Visual Sudio 编程环境的使用、如何设计用户界面、如何完整地开发出最新 Windows Store 应用程序等我们在实际项目开发中必须运用的内容，可谓面面俱到。

本书示例丰富，实用性强。本书继承了本系列书籍的优秀风格，大部分章节都精心设计了实例，而不是简单地演示功能，能起到很好的复习巩固作用，实例后的详细解释有助于读者快速掌握相关的知识点。同时，实例中提供了许多框架性代码，这些代码在实际工作中极具借鉴意义。

在翻译本书的过程中，感谢同事和朋友们给予的支持。也要感谢清华大学出版社的编辑在后期对译稿一丝不苟的校对和修改。

本书的翻译过程非常艰辛，可谓一波三折。在本书即将出版之际，激动和欣喜之余，却也多了几分惶恐。虽然自己也已经尽了很大的努力，但难免会有一些纰漏和偏颇之处。对于本书的任何想法和意见都欢迎各位读者指正。本书全部章节由齐立波、黄俊伟翻译，参与翻译活动的还有孔祥亮、陈跃华、杜思明、熊晓磊、曹汉鸣、陶晓云、王通、方峻、李小凤、曹晓松、蒋晓冬、邱培强、洪妍、李亮辉、高娟妮、曹小震、陈笑。

最后，希望本书能帮助各位开发者提升技能，开发出让用户满意的优秀应用程序！

来自全球的赞誉

★★★★★ 无与伦比的 C#书籍！ January 20, 2013

By E. Carbone

本书是我阅读过的最优秀 C#书籍。去年，我参加学习了一门 C#课程，我向讲师推荐将本书用作教材。

本书的章节编排环环相扣，可谓独具匠心。例如：

(1) 在介绍 foreach 循环之前讨论数组。

(2) 在讲解“方法”时，首先将其作为“函数”进行介绍，以便读者理解其原理。

(3) 在介绍面向对象技术之前，首先讨论调试和错误处理。

(4) 在介绍面向对象的技术后，下一章将它们捏合在一起，开始讨论类。

作者文笔优美，通俗易懂，列举了大量示例。有两处让我进一步体会到其细腻风格：(1)探讨了如何使用 Visual Studio 的任务列表和 TODO 注释；(2)简明介绍了 MVC，篇幅安排恰当，不过多涉猎，又使全书变得完整权威。

我强烈推荐 C#编程新手认真研读本书！

★★★★★ 一本卓越的书籍 January 1, 2013

By iamtron

本书示例的讲解极其透彻，如醍醐灌顶，令我茅塞顿开。

★★★★★ 编程新手必读的经典书籍 August 15, 2013

By William Kevin Beecher

一本 C#权威书籍！目前我读此书已经过半，感觉容易理解，学到不少技术，受益匪浅！每章都有“试一试”部分，让我可在实战中锻炼应用所学的新知识。在每章中都能学到不少新的编程知识，并能激起我学习下一章内容的强烈兴趣。

作 者 简 介

Karli Watson 是一名 IT 承包商和作家，目前在伦敦的金融行业从事工作。他主攻.NET(尤其是 C#)，为几家出版商编写了多本介绍该领域技术的图书。他擅长以易于理解的方式向热情求知者阐述复杂理念，并投入了大量时间研究新技术，找出可传授给其他人的新东西。

在工作之余(这种时间似乎很少)，Karli 喜欢到山上滑雪，或者尝试发表小说。他喜欢穿颜色鲜亮的衣服。他在 Twitter 上的用户名是@karlequin，也许有一天他自己会建立一个网站。

Jacob Vibe Hammer 是 Kamstrup 的一名软件架构师和开发人员，帮助公司为大型公用设施开发世界级的智能网格解决方案。自他刚能拼写 Basic 之时，他就开始了自己的编程生涯，Basic 也是他使用的第一种编程语言。从那以后，他使用过多种编程语言和解决方案架构。但进入 21 世纪后，他主要在.NET 平台上工作。如今，他主要编写 C#和 WPF 程序，以及试用 NoSQL 数据库。Jacob 是丹麦人，与妻儿一起居住在丹麦奥尔胡斯市。

Jon D. Reid 担任 IFS Metrix Service Management (www.IFSWORLD.com/Metrix)的软件技术总监。他已与他人合著了多本.NET 图书，包括 *Beginning Visual C# 2010*、*Fast Track C#*和 *Pro Visual Studio .NET* 等。

Morgan Skinner 在 1980 年就开始在学校使用汇编语言进行编程。此后他使用了许多语言编写商业程序，如 Pascal、Modula-2、VAX 宏汇编语言、Smalltalk、PowerBuilder、C、C++和 C#等。开始接触.NET 编程后不久，他于 2001 年加盟 Microsoft 公司，担任了近 10 年的应用程序开发顾问，帮助英国大大小小的公司解决问题。Morgan 于 2011 年离开 Microsoft，现在是一名独立承包商，开发一些定制系统。更多信息请访问 www.morganskinner.com。

Daniel Kemper 是一名软件架构师，拥有多个 Microsoft 证书。他擅长富 Internet 应用程序、桌面客户端和报表技术。

Christian Nagel 是微软技术代言人(Microsoft RD)、Microsoft MVP，也是 Thinktecture 的合作伙伴和 CN Innovation 的创立者。他是一位软件架构师和开发人员，为使用 Microsoft 平台开发解决方案提供培训和咨询服务。他拥有逾 25 年的软件开发经验。Christian 从 PDP 11 和 VAX/VMS 系统开始踏入计算机生涯，此后接触了各种语言和平台。自从 2000 年以来(那时.NET 还只是一个技术框架)，他就开始使用各种.NET 技术建立.NET 解决方案。目前，他主要负责指导开发可访问 Windows Azure 服务的 Windows 8 应用程序。他具备深厚的 Microsoft 技术功底，已撰写了大量.NET 图书，并获得了 Microsoft 认证培训师和专业开发人员证书。Christian 经常在国际会议发表演讲，例如 TechEd、Basta!和 Tech Days，他建立了 INETA Europe 来支持.NET 用户组。可通过网站 www.cninnovation.com 和 www.thinktecture.com 联系 Christian，他的 Twitter 用户名是@christiannagel。

技术编辑简介

Doug Holland 是 Microsoft 的 Developer and Platform Evangelism 小组的架构师，他与 Microsoft 的 ISV 战略合作伙伴携手，为 Windows 8 和 Windows Phone 8 用户提供令他们耳目一新的愉悦体验。

Richard Hopton 拥有 10 年的业务软件系统开发经验，目前专注于使用 C#为英国伦敦的一家数字媒体公司设计和构建基于 REST 的高扩展性 API 解决方案。Richard 在 Microsoft 月度 Developer Newsletter 和 MSDN Flash 上发表过文章，并在英国的众多开发社区会议上发表过演讲。

Marcel Meijer 在信息和通信技术领域的工作时间已逾 15 年。目前，他主要研究 Windows Azure、云服务、C#、软件开发和架构。他是 VX Company 的高级架构师。闲暇时间担任 SDN(Software Development Network；www.sdn.nl)的理事，负责安排 SDN Events(SDE)演讲者、遴选 SDN 大会主题以及为 SDN Magazine 整理和编辑内容。

致　　谢

感谢 Wiley 各位人士的帮助、鼓励和理解。在产品繁多而且名称不时发生变化这一现实条件下，既要快速完成本书写作，又要保证内容精确，求取二者的平衡并不容易，但是我想我们做得不错。这里特别感谢 Patrick Meader 在整个项目中(基本)保持镇定，至少比我镇定。感谢我的妻子 Donna，她几乎成功地做到了让我在写书过程中没有丧失理智。当然，还要感谢你们购买本书，祝你们在编码征程中一路好运。

——Karli Watson

前　言

C#是 Microsoft 于 2000 年 7 月推出.NET Framework 的第 1 版时提供的一种全新语言。C#从那时起迅速流行开来，成为使用.NET Framework 的桌面和 Web 开发人员无可争议的选择。他们喜欢 C#的一个原因是其派生于 C/C++的简洁明了的语法，这种语法简化了以前一些给程序员带来困扰的问题。尽管做了这些简化，但 C#仍保持了 C++原有的功能，所以现在没理由不从 C++转向 C#。C#语言并不难，也非常适合于学习基本编程技术。易于学习，再加上.NET Framework 的功能，使 C#成为开始您编程生涯的绝佳方式。

C#的最新版本 C# 5 是.NET Framework 4.5 的一部分，它建立在已有的成功基础之上，还添加了一些更吸引人的功能。Visual Studio 的最新版本 Visual Studio 2012 和开发工具的 Visual Studio Express 2012 系列也有许多变化和改进，这大大简化了编程工作，显著提高了效率。

本书将全面介绍 C#编程的所有知识，从该语言本身一直到桌面和 Web 编程，再到数据源的使用，最后是一些新的高级技术。我们还将学习 Visual Studio 2012 的功能和利用它开发应用程序的各种方式。

本书文笔优美流畅，阐述清晰，每一章都以前面章节的内容为基础，便于读者掌握高级技术。每个概念都会根据需要来介绍和讨论，而不会突然冒出某个技术术语来妨碍读者的阅读和理解。本书尽量减少使用的技术术语数量，但如果需要，将根据上下文进行正确的定义和布置。

本书的作者都是各自领域的专家，都是 C#语言和.NET Framework 的爱好者，没有人比他们更有资格讲授 C#了，他们将在您掌握从基本规则到高级技术的过程中为您保驾护航。除了基础知识外，本书还有许多有益的提示、练习、完全成熟的示例代码(可从 p2p.wrox.com 上下载)，在您的职业生涯中一定会反复用到它们。

本书将毫无保留地传授这些知识，希望读者能通过阅读本书成为最优秀的程序员。

0.1　本书读者对象

本书主要针对想学习如何使用.NET Framework 编写 C#程序的所有人。本书针对的是想要通过学习一种干净、现代、优雅的编程语言来掌握程序设计的完完全全的初学者。但是，对于熟悉其他语言、想要探索.NET 平台的人们，以及想要了解 Microsoft .NET 使用的旗舰语言的.NET 开发人员，本书同样有用。

0.2 本书内容

本书前面的章节介绍该语言本身，读者不需要具备任何编程经验。以前对其他语言有一定了解的开发人员，会觉得这些章节的内容非常熟悉。C#语法的许多方面都与其他语言相同，许多结构对所有的编程语言来说都是相通的(例如，循环和分支结构)。但是，即使是有经验的程序员也可以从这些章节中获益，理解这些技术应用于 C#的特征。

如果读者是编程新手，就应从头开始学习，了解基本的编程概念，并熟悉 C#和支持 C#的.NET 平台。如果读者对.NET Framework 比较陌生，但知道如何编程，就应阅读第 1 章，然后快速跳读后面几章，这样就能掌握 C#语言的应用方式了。如果读者知道如何编程，但以前从未接触过面向对象的编程语言，就应从第 8 章开始阅读以后的章节。

如果读者对 C#语言比较了解，就可以集中精力学习那些详细论述最新.NET Framework 和 C#语言开发的章节，尤其是集合、泛型和 C#语言的新增内容(第 11～14 章)，或者完全跳过本书的第 I 部分，从第 15 章开始学习。

本书章节的编排方式可以达到两个目的：可以按顺序阅读这些章节，将其视为 C#语言的一个完整教程；还可以按照需要深入学习这些章节，将其作为一本参考资料。

除核心内容外，从第 3 章开始，每章末尾还包含一组习题，完成这些习题有助于读者理解所学的内容。习题包括简单的选择题、判断题以及需要修改或建立应用程序的较难问题。附录 A 给出了全部习题的答案，还可在 www.wrox.com/remtitle.cgi?isbn=9781118314418 上的 Download Code 部分找到这些习题。

本书特别注重与 C# 5、.NET 4.5 的一致性。每一章都进行了彻底的检查，删掉了不太相关的内容，增加了新材料。所有代码都在最新版本的开发工具上进行了测试，所有屏幕截图都在 Windows 8 上重新截取，以提供最新的窗口和对话框。

尽管我们不喜欢承认失误，但还是修订了前几版中的错误，处理了许多其他的读者评论。我们希望不要出现太多的新错误，但一旦发现了错误，我们的 Web 专家就会联机修改它们。

本版本的亮点包括：

- 增加并改进了代码示例。
- 对桌面应用程序的讨论从使用原来的方式(Windows Forms)改为使用新方式(Wiondows Presentation Foundation)，让你紧跟技术的发展。
- 涵盖 C# 5 和.NET 4.5 的所有新内容，包括如何创建 Windows Store 应用程序。
- 十分合理地介绍高级技术，重点是适合于新手、较容易理解的内容。

0.3 本书结构

本书分为 6 个部分。

- **前言**：概述本书的内容。
- **C#语言**：介绍了 C#语言的所有内容，从基础知识到面向对象的技术，一应俱全。
- Windows **编程**：介绍如何用 C#编写桌面和 Windows Store 应用程序，如何部署它们。
- Web **编程**：描述 Web 应用程序的开发和部署。

- **数据访问**：介绍如何在应用程序中使用数据，包括存储在硬盘文件中的数据、以 XML 格式存储的数据和数据库中的数据。
- **其他技术**：讲述使用 C#和.NET Framework 的一些额外方式，包括由.NET 3.0 引入然后经.NET 4 和.NET 4.5 改进的 WCF 和 WF 技术。

下面介绍本书 5 个重要部分中的章节。

0.3.1　C#语言(第 1～14 章)

第 1 章介绍 C#及其与.NET 的关系，了解在这个环境下编程的基础知识，以及 Visual Studio 2012(VS)与它的关系。

第 2 章开始介绍如何编写 C#应用程序，学习 C#的语法，并将 C#和示例命令行、Windows 应用程序结合起来使用。这些示例将说明 C#如何快速轻松地启动和运行，并附带介绍 VS 开发环境以及本书将要使用的基本窗口和工具。

接着将学习 C#的基础知识。第 3 章介绍变量的含义以及如何操纵它们。第 4 章将用流程控制(循环和分支)改进应用程序的结构，第 5 章介绍一些高级变量类型，如数组。第 6 章开始以函数形式封装代码，这样就更易于执行重复的操作，使代码更容易让人理解。

从第 7 章开始将运用 C#语言的基础知识，调试应用程序。这包括在运行应用程序时输出跟踪信息，使用 VS 查找错误，在强大的调试环境中找出解决问题的办法。

第 8 章将学习面向对象编程(Object-Oriented Programming，OOP)。首先了解这个术语的含义，回答“什么是对象？”OOP 初看起来是较难的问题。我们将用一整章的篇幅来介绍它，解释对象的强大之处。直到该章的最后才会真正使用 C#代码。

第 9 章将理论知识应用于实践，开始在 C#应用程序中使用 OOP 时，这才体现出 C#的真正威力。在第 9 章介绍如何定义类和接口之后，第 10 章将探讨类成员(包括字段、属性和方法)，在这一章的最后将开始创建一个扑克牌游戏应用程序，这个应用程序将在几章中开发完成，它非常有助于理解 OOP。

学习了 OOP 在 C#中的工作原理后，第 11 章将介绍几种常见的 OOP 场景，包括处理对象集合、比较和转换对象。第 12 章讨论.NET 2.0 中 C#引入的一个非常有用的特性——泛型，利用它可以创建非常灵活的类。第 13 章通过一些其他技术(主要是事件，它在 Windows 编程中非常重要)继续讨论 C#语言和 OOP。最后，第 14 章介绍 C# 3.0、4 和 5 中引入的新特性。

0.3.2　Windows 编程(第 15～18 章)

第 15 章开始介绍 Windows 编程概念，理解在 VS 中如何实现 Windows 编程。该章主要关注如何使用 WPF 以图形化方式构建桌面应用程序，以及用最少的时间和精力创建高级应用程序。你将首先学习 WPF 编程的基础知识，然后在该章和第 16 章逐渐拓展相关知识。第 16 章介绍了在应用程序中如何使用.NET Framework 提供的丰富控件。

第 17 章介绍了如何创建 Windows 8 中新增的 Windows Store 应用程序。Windows Store 应用程序是一种令人兴奋的新应用程序类型，可以为用户提供愉悦的全屏体验。该章还将介绍如何准备应用程序以便在 Windows Store 中进行销售。

第 18 章讨论应用程序的部署，包括建立安装程序，以便用户快速安装和运行应用程序。

0.3.3 Web 编程(第 19～20 章)

该部分的结构与桌面编程部分类似。首先，第 19 章描述了构成最简单 Web 应用程序的控件，如何把它们组合在一起，让它们使用 ASP.NET 执行任务。接着介绍了更高级的技术、ASP.NET AJAX、各种控件、Web 上下文下的状态管理以及 Web 标准的遵循。

然后，第 20 章探讨 Web 应用程序和服务的部署，尤其是可以通过单击按钮把应用程序发布到 Web 上的 VS 特性。

0.3.4 数据访问(第 21～24 章)

第 21 章介绍了应用程序如何将数据保存到磁盘以及如何检索磁盘上的数据(作为简单的文本文件或者更复杂的数据表示方式)。该章还将讨论如何压缩数据，如何操纵旧数据(例如 CSV 文件)，如何监视和处理文件系统的变化。

第 22 章学习数据交换的事实标准 XML。之前的章节接触过 XML 几次，而该章将讨论 XML 的基本规则，论述 XML 的所有功能。

该部分其余章节介绍 LINQ(这是内置于.NET Framework 最新版本中的查询语言)。第 23 章简要介绍 LINQ。第 24 章使用 LINQ 访问数据库和其他数据。

0.3.5 其他技术(第 25 和第 26 章)

本书最后一部分将讨论.NET Framework最新版本中出现的几项新技术。第 25 章介绍 Windows Communication Foundation(WCF)，它为在企业级以编程方式跨本地网络和Interent访问信息和功能提供了许多工具。该章将介绍如何以平台无关的方式使用WCF，向Web应用程序和桌面应用程序公开复杂的数据和功能。

本书最后一章是第 26 章，介绍了 Windows Workflow Foundation(WF)，它允许在应用程序中执行工作流功能，因此可以定义一些操作，这些操作由外部的交互操作控制，按特定顺序执行，这对许多类型的应用程序都很有帮助。

0.4 使用本书的要求

本书中C#和.NET Framework的代码和描述都适用于C# 5 和.NET 4.5。除了Framework之外，不需要其他东西就可以理解本书的这个方面，但许多示例都需要使用开发工具。本书将 Visual Studio 2012 作为主要开发工具，但是，如果没有安装此工具，可以使用免费的Visual Studio Express 2012 产品系列。在本书的第I部分，可使用Visual Studio Express 2012 for Windows Desktop创建桌面和控制台应用程序。对于其余章节，可使用Visual Studio Express 2012 for Windows 8 创建Windows Store应用程序，使用Visual Studio Express for Web创建Web 应用程序，并在需要访问数据库的应用程序中使用SQL Server Express 2012。一些功能只能在 Visual Studio 2012 中使用，但这不会妨碍练习本书的示例。

可从 Wrox 网站下载示例源代码：www.wrox.com/remtitle.cgi?isbn=9781118314418。

0.5　本书约定

为了帮助读者在阅读本书的过程中获取最多信息，并随时了解当前处理的事项，本书使用了许多约定。

警告：带有警告图标的方框包含了重要且应该记住的信息，这些信息与周围的文字直接相关联。

提示：带有铅笔图标的方框表示注释、提示、暗示、技巧或对当前讨论的弦外之音。

本书通过两种方式来显示代码：

- 对于大多数代码示例，使用没有突出显示的等宽字体来表示。
- 对在当前上下文中特别重要的代码，用粗体字来强调显示。

0.6　源代码

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。

提示：

由于许多图书的标题都很类似，所以按 ISBN 搜索是最简单的，本书英文版的 ISBN 是 978-1-118-31441-8。

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。

0.7　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 wkservice@vip.163.com 发电子邮件，我们就会检查您的反馈信息，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英文版的勘误表，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。

0.8 p2p.wrox.com

要与作者和同行讨论，请加入 p2p.wrox.com 上的 P2P 论坛。这个论坛是一个基于 Web 的系统，便于您张贴与 Wrox 图书相关的消息和相关技术，与其他读者和技术用户交流心得。该论坛提供了订阅功能，当论坛上有新的消息时，它可以给您传送感兴趣的论题。Wrox 作者、编辑和其他业界专家和读者都会到这个论坛上来探讨问题。

在 http://p2p.wrox.com 上，有许多不同的论坛，它们不仅有助于阅读本书，还有助于开发自己的应用程序。要加入论坛，可以遵循下面的步骤：

(1) 进入 p2p.wrox.com，单击 Register 链接。

(2) 阅读使用协议，并单击 Agree 按钮。

(3) 填写加入该论坛所需要的信息和自己希望提供的其他信息，单击 Submit 按钮。

(4) 您会收到一封电子邮件，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消息，但要张贴自己的消息，就必须加入该论坛。

加入论坛后，就可以张贴新消息，响应其他用户张贴的消息。可以随时在 Web 上阅读消息。如果要让该网站给自己发送特定论坛中的消息，可以单击论坛列表中该论坛名旁边的 Subscribe to this Forum 图标。

关于使用 Wrox P2P 的更多信息，可阅读 P2P FAQ，了解论坛软件的工作情况以及 P2P 和 Wrox 图书的许多常见问题。要阅读 FAQ，可以在任意 P2P 页面上单击 FAQ 链接。

目　录

第 I 部分

C# 语 言

- 第 1 章　C#简介
- 第 2 章　编写 C#程序
- 第 3 章　变量和表达式
- 第 4 章　流程控制
- 第 5 章　变量的更多内容
- 第 6 章　函数
- 第 7 章　调试和错误处理
- 第 8 章　面向对象编程简介
- 第 9 章　定义类
- 第 10 章　定义类成员
- 第 11 章　集合、比较和转换
- 第 12 章　泛型
- 第 13 章　其他 OOP 技术
- 第 14 章　C#语言的改进

第 1 章

C# 简 介

本章内容：

- .NET Framework 的功能及其包含的内容
- .NET 应用程序的工作原理
- C#的概念及其与.NET Framework 的关系
- 用 C#创建.NET 应用程序的工具

本书第 I 部分将介绍使用最新版本的 C# 语言所需的基础知识。第 1 章将概述 C#和.NET Framework，包括这两项技术的含义、作用及相互关系。

首先讨论.NET Framework。这种技术包含的许多概念初看起来都不是很容易掌握的。也就是说，我们必须在很短的篇幅里介绍许多新概念，但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，本书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，包括它的起源以及与C++的类似之处。最后介绍本书使用的主要工具：Visual Studio 2012 (VS)。VS 2012 是 Microsoft 提供的一系列开发环境中最新的一个，本书将用到它的各种新功能(包括对 Windows Store 应用程序的完整支持)。

1.1 .NET Framework 的含义

.NET Framework(现在是版本 4.5)是 Microsoft 为开发应用程序而创建的一个具有革命意义的平台。这句话最有趣的地方在于它的广义性，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 的 Microsoft 版本运行在 Windows 操作系统和 Windows Phone 操作系统上，但它也有运行在其他操作系统上的版本，例如 Mono，它是.NET Framework 的开源版本(包含 C#编译器)，该版本可以运行在几个操作系统上，包括各种 Linux 版本和 Mac OS。另外，Mono 还有一些版本可以运行在 iPhone(MonoTouch)和 Android(Mono for Android，也称为 MonoDroid)智能手机上。使用.NET Framework 的一个重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 定义并未限制应用程序的类型。这是因为本来就没有限制。

可以使用.NET Framework 创建桌面应用程序、Windows Store 应用程序、Web 应用程序、Web 服务和其他各种类型的应用程序。另外注意，对于 Web 应用程序，按照定义，它们是多平台的应用程序，因为任何带有 Web 浏览器的系统都可以访问它们。

.NET Framework 的设计方式确保它可以用于各种语言，包括本书介绍的C#语言，以及 C++、Visual Basic、JScript 甚至一些旧语言，如 COBOL。为此，还推出了这些语言的.NET 版本，目前还在不断推出更多版本。所有这些语言都可以访问.NET Framework，它们彼此之间还可以通信。C#开发人员可以使用 Visual Basic 程序员编写的代码，反之亦然。

所有这些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 主要包含一个庞大的代码库，可以在客户语言(如 C#)中通过面向对象编程技术(OOP)来使用这些代码。这个库分为多个不同的模块，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，另一个模块包含网络编程的代码块，还有一个模块包含 Web 开发的代码块。一些模块还分为更具体的子模块，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，不同操作系统可以根据各自的特性，支持其中的部分或全部模块。例如，智能手机支持所有的核心.NET 功能，但不需要某些更高级的模块。

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，指定最基本类型(如 32 位带符号的整数)有助于使用.NET Framework 的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)。

除提供这个库外，.NET Framework 还包含.NET 公共语言运行库(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 使用.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 Framework 的任何一种语言)。本书用 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 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，但使用了.NET Framework，并在需要时利用了 VS 中的其他工具。

为执行 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(native code)。这种转换称为编译代码，由编译器执行。但在.NET Framework 下，此过程包括两个阶段。

1. CIL 和 JIT

在编译使用.NET Framework 库的代码时，不是立即创建专用于操作系统的本机代码，而是把代码编译为通用中间语言(Common Intermediate Language，CIL)代码，这些代码并非专门用于任何一种操作系统，也非专门用于 C#。其他.NET 语言，如 Visual Basic .NET 也会在第一阶段编译为这种语言。开发 C#应用程序时，这个编译步骤由 VS 完成。

显然，要执行应用程序，必须完成更多工作，这是 Just-In-Time(JIT)编译器的任务，它把 CIL 编译为专用于 OS 和目标机器结构的本机代码。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 CIL 代码仅在需要时才编译的事实。这种编译可以在应用程序的运行过程中动

态发生，不过开发人员一般不需要关心这个过程。除非要编写性能十分关键的代码，否则知道这个编译过程会在后台自动进行，并不需要人工干预就可以了。

过去，常常需要把代码编译为几个应用程序，每个应用程序都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 芯片组上运行得更快)，有时则是非常重要的(例如，使应用程序可以同时工作在 Win9*x* 和 WinNT/2000 环境下)。现在就不必要了，因为 JIT 编译器使用 CIL 代码，而 CIL 代码是独立于计算机、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结构，CIL 会使用合适的编译器创建所需的本机代码。

这样，开发人员需要做的工作就比较少了。实际上，可以忽略与系统相关的细节，将注意力集中在代码的功能上就够了。

读者可能遇到过 Microsoft Intermediate Language(MSIL)或 IL，MSIL 是 CIL 原来的名称，许多开发人员仍沿用这个术语。

2. 程序集

在编译应用程序时，所创建的 CIL 代码存储在一个程序集中。程序集包括可执行的应用程序文件(这些文件可以直接在 Windows 上运行，不需要其他程序，其扩展名是.exe)和其他应用程序使用的库(其扩展名是.dll)。

除包含CIL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(CIL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，我们不会遇到没有把需要的数据添加到系统注册表中这样的问题，而这在使用其他平台进行开发时这个问题常常出现。

因此，部署应用程序就非常简单了，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需要的所有信息都安装到一个地方。可以编写一些代码来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地方。在.NET Framework 中，这个地方是全局程序集缓存(Global Assembly Cache，GAC)，把代码放在这个缓存中是很简单的，只需把包含代码的程序集放在包含该缓存的目录中即可。

3. 托管代码

在将代码编译为 CIL，再用 JIT 编译器将它编译为本机代码后，CLR 的任务尚未全部完成，还需要管理正在执行的用.NET Framework 编写的代码(这个执行代码的阶段通常称为运行时(runtime))。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，不受 CLR 控制运行的应用程序属于非托管类型，某些语言(如 C++)可以用于编写此类应用程序，例如，访问操作系统的低级功能。但是在 C#中，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 处理与操作系统的任何交互。

4. 垃圾回收

托管代码最重要的一个功能是垃圾回收(garbage collection)。这种.NET 方法可确保应用程序不再

使用某些内存时，就会完全释放这些内存。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 垃圾回收会定期检查计算机内存，从中删除不再需要的内容。执行垃圾回收的时间并不固定，可能一秒钟内会进行数千次的检查，也可能几秒钟才检查一次，不过一定会进行检查。

这里要给程序员一些提示。因为在不可预知的时间执行这项工作，所以在设计应用程序时，必须留意这一点。需要许多内存才能运行的代码应自己完成清理工作，而不是坐等垃圾回收，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，先总结一下上述创建.NET 应用程序所经历的步骤：

(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) 把代码编译为 CIL，存储在程序集中，如图 1-2 所示。

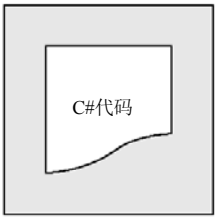

图　1-1

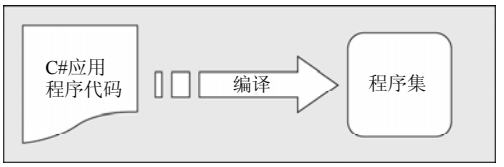

图　1-2

(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。

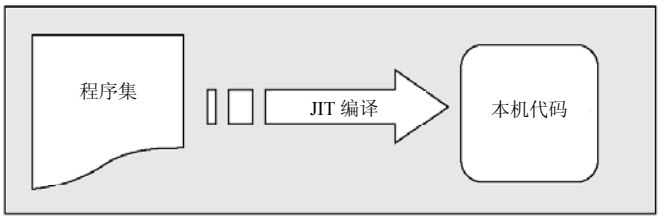

图　1-3

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

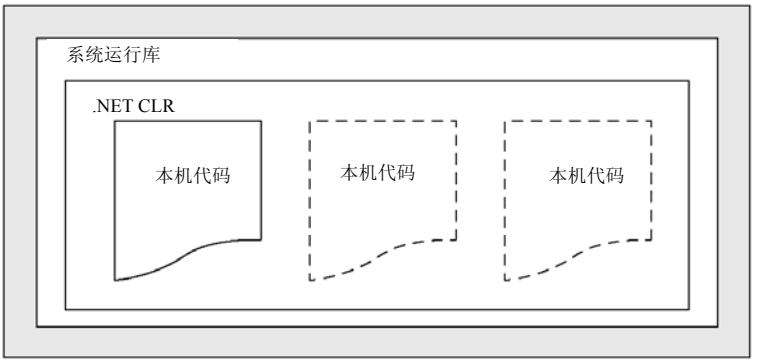

图　1-4

6. 链接

在上述过程中还有一点要注意。在第(2)步中编译为 CIL 的 C#代码未必包含在单独文件中，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(linking)，是非常有用的。原因是处理几个较小的文件比处理一个大文件要简单得多。可以把逻辑上相关的代码分解到一个文件中，以便单独进行处理，这也更便于在需要时找到特定的代码块，让开发小组把编程工作分解为一些可管理的块，让每个人编写一小块代码，而不会破坏已编写好的代码部分或其他人正在处理的部分。

1.2　C#的含义

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。因为 C#是近期发展起来的，所以吸取了以往的教训，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，因为其语法更简单。但是，C#是一种强大的语言，在 C++中能完成的任务几乎都能利用 C#完成。虽然如此，C#中与 C++高级功能等价的功能(例如直接访问和处理系统内存)，只能在标记为“unsafe”的代码中使用。这个高级编程技术存在潜在威胁(正如它的名称所暗示的)，因为它可能覆盖系统中重要的内存块，导致严重后果。因此，本书不讨论这个问题。

C#代码常比 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，这表示一旦为某个数据指定了类型，就不能转换为另一个不相关的类型。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#编写的代码通常比用 C++编写的代码长。但 C#代码更健壮，调试起来也比较简单，.NET 始终可以随时跟踪数据的类型。在 C#中，不能完成诸如“把 4 字节的内存分配给这个数据后，我们使其有 10 个字节长，并把它解释为 X”等任务，但这并不是一件坏事。

C#只是用于.NET 开发的一种语言，但它是最好的一种语言。C#的优点是，它是唯一彻头彻尾为.NET Framework 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。要使诸如 VB.NET 的语言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，至少需要不常见的语法。但 C#能使用.NET Framework 代码库提供的每种功能。而且，.NET 的每个新版本都在 C#语言中添加了新功能，满足了开发人员的要求，使之更强大。

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#使用的是.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。

- **桌面应用程序**　这些应用程序(如 Microsoft Office)具有我们很熟悉的 Windows 外观和操作方式，使用.NET Framework 的 Windows Presentation Foundation(WPF)模块就可以简便地生成这种应用程序。WPF 模块是一个控件库，其中的控件(例如按钮、工具栏和菜单等)可用于建立 Windows 用户界面(UI)。

- **Windows Store 应用程序** 这是 Windows 8 中新引入的一类应用程序。此类应用程序主要针对触摸设备设计，通常全屏运行，侧重点在于简洁清晰。创建这类应用程序的方式有多种，包括使用 WPF。
- **Web 应用程序** 它们是一些 Web 页面，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，允许进行个性化和实现安全性等。这个系统称为 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。还可以使用 Silverlight 编写在浏览器内部运行的应用程序。
- **WCF 服务** 这是一种灵活创建各种分布式应用程序的方式。使用 WCF 服务可以通过局域网或 Internet 交换几乎各种数据。无论使用什么语言创建 WCF 服务，也无论 WCF 服务驻留在什么系统上，都使用一样简单的语法。

这些类型的应用程序也可能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、ADO.NET Entity Framework 或 C#的 LINQ(Language Integrated Query)功能来实现。也可以使用许多其他资源，例如，创建联网组件、输出图形、执行复杂数学任务的工具。

1.2.2 本书中的 C#

本书第Ⅰ部分介绍 C# 语言的语法和用法，但不过分强调.NET Framework。这是必需的，因为我们不能没有一点儿 C# 编程基础就使用.NET Framework。首先介绍一些比较简单的内容，把较复杂的面向对象编程(Object-Oriented Programming，OOP)论题放在基础知识的后面论述。假定读者没有一点儿编程的知识，这些是首要的规则。

学习了基础知识后，本书还将介绍如何开发更复杂、更有用的应用程序。本书的第Ⅱ部分将讨论桌面和 Windows Store 应用程序编程，第Ⅲ部分将研究 Web 应用程序编程，第Ⅳ部分将讲述数据访问(对数据库、文件系统和 XML 数据的访问)和 LINQ，第Ⅴ部分将介绍其他一些有趣的.NET 论题。

1.3 Visual Studio 2012

本书使用 Visual Studio 2012 开发工具进行所有的 C#编程，包括简单的命令行应用程序，乃至较复杂的项目类型。VS 不是开发 C#应用程序所必需的开发工具或集成开发环境(IDE)，但使用它可以使任务更简单一些。如果愿意的话，可在基本的文本编辑器(如常见的记事本)中处理 C#源代码文件，再使用.NET Framework 中包含的命令行编译器把代码编译到程序集中。但是，为什么不使用功能完备的 IDE 呢？

下面列出一些使 VS 成为.NET 开发首选工具的功能。

- VS 可以自动执行编译源代码的步骤，同时可以完全控制编译过程中使用的任何选项。
- VS 文本编辑器为 VS 支持的语言(包括 C#)量身定制，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，这个功能称为 IntelliSense。
- VS 包括 XAML、ASP.NET 及其他 UI 语言的设计器，允许 UI 元素的简单拖放设计。
- 在 C#中，许多类型的项目都可用已有的“样板”代码来创建，不需要从头开始。各种代码文件通常已经准备好了，减少了从头开始一个项目所用的时间。

- VS 包括几个可自动执行常见任务的向导，其中很多任务可在已有的文件中添加合适的代码，在某些情况下，你甚至不需要考虑语法的正确性。
- VS 包含许多强大的工具，可以显示项目中的元素并允许在其中导航，这些元素可以是 C#源代码文件，也可以是其他资源，例如位图图像或声音文件。
- 除了在 VS 中编写应用程序外，还可以创建部署项目，以便为客户提供代码，并使客户方便地完成安装。
- 在开发项目时，VS 允许使用高级调试技巧，例如，能在代码中一次调试一条指令，并监视应用程序的状态。

C#还有许多功能，希望读者能掌握它们！

1.3.1　Visual Studio Express 2012 产品

除Visual Studio 2012外，Microsoft还提供了几个更简单的开发工具，称为Visual Studio Express 2012产品。可以在 http://www.microsoft.com/express 上免费获得它们。

各种 Express 产品可以创建所需的几乎所有 C#应用程序。在功能上它们都是 VS 的删节版本，但外观和操作方式是一样的。尽管它们提供了 VS 的许多功能，但缺少一些重要功能；不过我们仍可以在学习本书的过程中使用它们。

由于在写作本书时 Express 版本还不可用，本书使用了 Visual Studio 2012 的专业版。在写作本书时，有一个称为 Visual Studio Express 2012 for Windows Desktop 的 Express 产品预计在不久后会发布，使用它应该足以学习本书的第 I 部分。对于本书的剩余部分，使用 Visual Studio Express 2012 for Windows 8 和 Visual Studio Express 2012 for Web 应该也是可以的，但是在写作本书的时候我们不能肯定这一点一定成立。

1.3.2　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，解决方案不仅是一个应用程序，它还包含项目，可以是 WPF 项目和 Web 应用程序项目等。但是，解决方案可以包含多个项目，这样，即使相关的代码最终在硬盘上的多个位置编译为多个程序集，也可以把它们组合到一个地方。

这是非常有用的，因为它可以处理“共享”代码(这些代码放在 GAC 中)，同时，应用程序也使用这段共享代码。在使用唯一的开发环境时，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，并讨论了如何轻松创建各种功能强大的应用程序。还探讨了把用 C#等语言编写的代码转换为可运行的应用程序所需要做的工作，以及使用在.NET 公共语言运行库下运行的托管代码有什么优点。

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#开发时所使用的工具——Visual Studio 2012。

第 2 章将介绍如何运行一些 C#代码，介绍基础知识，并集中讨论 C#语言本身，而不是过多地讨论 IDE 的工作原理。

1.5　本章要点

主　　题	要　　点
.NET Framework 基础	.NET Framework 是Microsoft 最新的开发平台，目前的版本是4.5。它包括一个公共类型系统(CTS)和一个公共语言运行库(CLR)。.NET Framework 应用程序使用面向对象的编程(OOP)的方法编写，通常包含托管代码。托管代码的内存管理由.NET 运行库处理，其中包括垃圾回收
.NET Framework 应用程序	用.NET Framework 编写的应用程序首先编译为 CIL。在执行应用程序时，JIT 把 CIL 编译为本机代码。应用程序编译后，把不同的部分链接到包含 CIL 的程序集中
C#基础	C#是包含在.NET Framework 中的一种语言，它是以前的语言(如 C++)的一种演变，可以用于编写任意应用程序，包括 Web 应用程序和桌面应用程序
集成开发环境 (IDE)	可以在 Visual Studio 2012 中用 C#编写任意类型的.NET 应用程序，还可以在免费的、但功能稍弱的 Express 产品系列中用 C#创建.NET 应用程序。这两种 IDE 都使用解决方案，解决方案可以包含多个项目

第 2 章

编写C#程序

本章内容：

- Visual Studio 2012 的基础知识
- 如何编写简单的控制台应用程序
- 如何编写简单的桌面应用程序

本章源代码下载：

本章源代码的下载地址为 www.wrox.com/remtitle.cgi?isbn=9781118314418。从该网页的 Download Code 选项卡中下载 Chapter 2 Code 后，可以找到与本章示例对应的单独文件。

第 1 章占用一定的篇幅讨论了 C#是什么，它是如何适应.NET Framework 的，现在就该编写一些代码了。本书主要使用 Visual Studio 2012(VS)，所以首先介绍这个开发环境的一些基础知识。

VS 是一个庞大的复杂产品，可能会使初学者望而生畏，但使用它创建简单的应用程序是非常容易的。在本章开始使用 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 能够执行的更复杂操作，现在仅介绍基础知识。

介绍完 IDE 后，将创建两个简单应用程序。现在不要过多地考虑代码，只要应用程序可以运行即可。在这些早期的示例中熟悉了应用程序的创建过程，不久之后就会适应这个过程了。

本章将学习创建两种基本的应用程序类型：控制台应用程序和桌面应用程序。

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，所以不需要考虑按钮、菜单、用鼠标指针进行的交互等，而是在命令行窗口中运行应用程序，用更简单的方式与其交互。

第二个应用程序是使用 Windows Presentation Foundation(WPF)创建的一个桌面应用程序，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并不费力。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。

本书接下来的两个部分也使用这两种应用程序类型，但开始时主要讨论控制台应用程序。在学习 C#语言时，不需要了解桌面应用程序的其他灵活性能。控制台应用程序的简单性可以让我们集中

精力学习语法，而不必考虑应用程序的外观和操作方式。

2.1　Visual Studio 2012 开发环境

在首次加载 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。本书将使用大多数窗口，读者很快就会熟悉它们。

如果是首次运行VS，则屏幕上会显示一个首选项列表，如果用户使用过这个开发环境的旧版本，则可以在这里做出选择，这些选择会影响到很多方面，例如，窗口的布局、控制台窗口运行的方式等。所以应选择 Visual C# Development Settings，否则会发现一些地方和本书的描述不一样。注意，可用选项会随着安装 VS 时选择的选项而变化，但只要选择安装 C#，这个选项就是可用的。

如果不是第一次运行 VS，但以前选择了另一个选项，也不必惊慌。为了把设置重置为 Visual C# Development Settings，只需导入它们即可。为此，单击 Tools 菜单上的 Import and Export Settings 选项，再选中 Reset All Settings 选项，如图 2-1 所示。

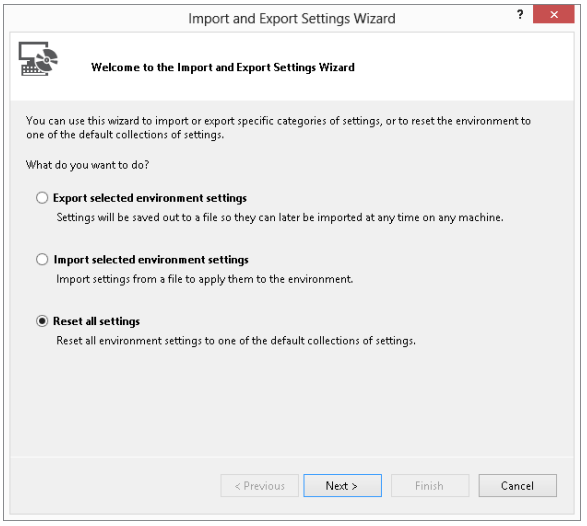

图　2-1

单击 Next 按钮，选择是否要在继续之前保存已有的设置。如果对设置进行了定制，就保存设置，否则就选择 No 按钮，再次单击 Next 按钮。在下个对话框中，选择 Visual C# Development Settings 选项，如图 2-2 所示。可用的选项可能会变化。

最后单击 Finish 按钮，应用设置。

VS 环境布局是完全可定制的，但默认设置很适合我们。在 C# Developer Settings 设置下，其布局如图 2-3 所示。

所有的代码都显示在主窗口中。在 VS 启动时，主窗口会默认显示一个提供帮助信息的 Start Page。主窗口可以包含许多文档，每个文档都有一个选项卡，单击文件名，就可以在文件之间切换。这个窗口也具有其他功能：它可以显示为项目设计的 GUI、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。

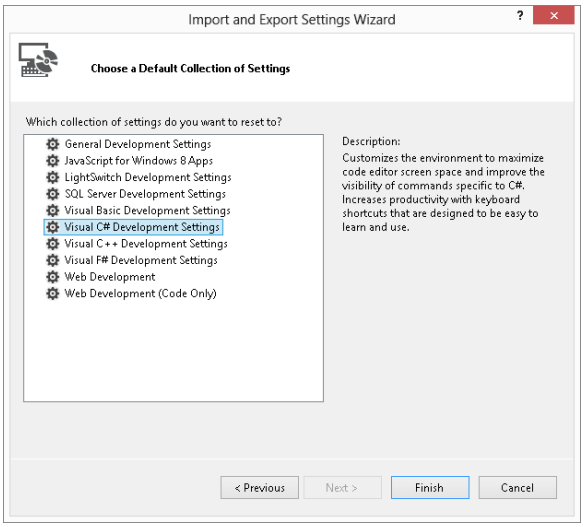

图　2-2

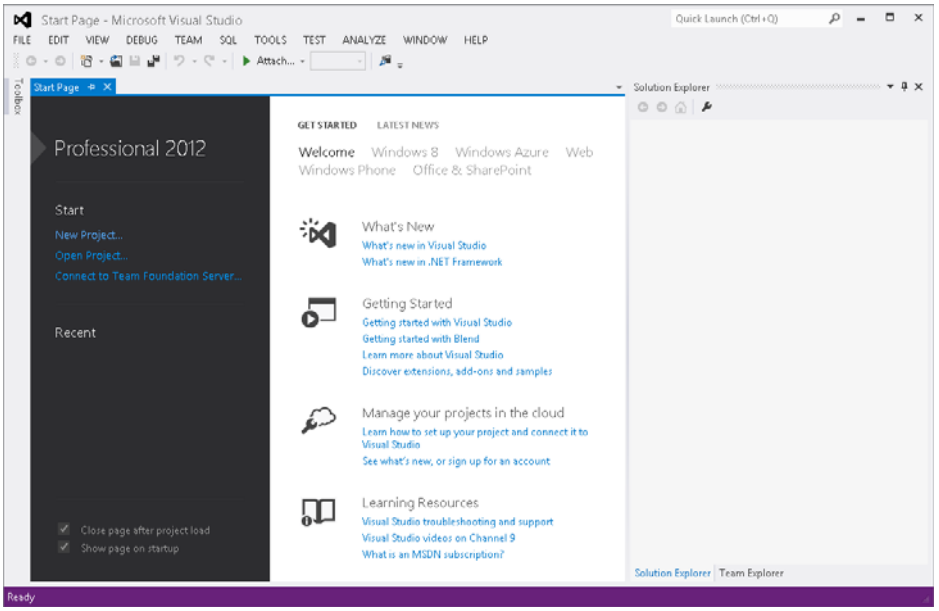

图　2-3

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，其功能包括：保存和加载文件，生成和运行项目，以及调试控件等。在需要使用这些工具栏时将会讨论它们。

下面简要描述 VS 的最常用功能：

- 单击 Toolbox 选项卡时，就会显示 Toolbox 工具栏，它们提供了桌面应用程序的用户界面构件等条目。另一个选项卡 Server Explorer 也可以在这里显示(通过 View | Server Explorer 菜单项选择它)，它包含其他许多功能，例如访问数据源、服务器设置和服务等。
- Solution Explorer 窗口显示当前加载的解决方案的信息。如上一章所述，解决方案是一个 VS 术语，表示一个或多个项目及其配置。Solution Explorer 窗口显示了解决方案中项目的各种视图，例如，项目中包含了哪些文件，这些文件中又包含了什么内容。

- Teamp Explorer 窗口显示了关于当前的 Team Foundation Server 或 Team Foundation Service 连接的信息，可用于使用源代码管理、bug 跟踪、自动生成等功能。但是，这是一个高级主题，本书中不会介绍。
- Solution Explorer 窗口之下可以显示 Properties 窗口，该窗口没有显示在图 2-3 中。稍后会看到这个窗口，因为它只在处理项目时才出现(也可以使用View | Properties Window 菜单项切换它)。这个窗口提供了更详细的项目内容视图，允许另外配置单独元素。例如，使用这个窗口可以改变桌面应用程序中按钮的外观。
- 另一个非常重要的窗口也未出现在图 2-3 中：Error List 窗口。可以使用 View | Error List 菜单项打开这个窗口，它显示了错误、警告和其他与项目有关的信息。这个窗口会持续不断地更新，但其中一些信息只有在编译项目时才出现。

这似乎需要理解很多东西，但不必担心，过不了多久就习惯了。下面首先建立第一个示例项目，它将使用上面介绍的许多 VS 元素。

VS 还可以显示许多其他窗口，它们都包含许多信息，有许多功能。其中一些窗口与上面提及的窗口共享屏幕空间，可以使用选项卡切换它们或把它们停靠在其他位置，如果有多个显示器，甚至可以分离它们，把它们放到其他显示器上显示。本书的后面会介绍其中的许多窗口，在读者自己深入探索 VS 环境时，可能还会发现更多窗口。

2.2　控制台应用程序

本书将频繁使用控制台应用程序，特别是开始时要使用这类应用程序，所以下面分步演示如何创建一个简单的控制台应用程序。

试一试：创建一个简单的控制台应用程序：ConsoleApplication1\Program.cs

(1) 选择File | New | Project菜单项，创建一个新的控制台应用程序项目，如图 2-4 所示。

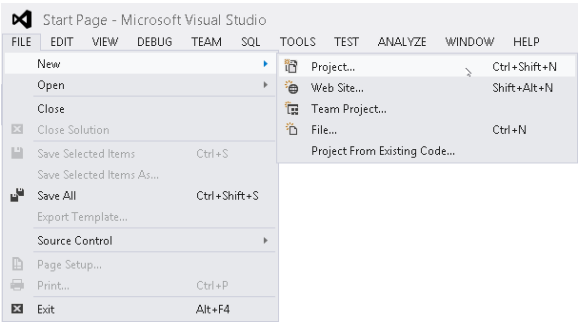

图　2-4

(2) 在显示窗口的左侧选择 Visual C# 节点，在中间窗格中选择 Console Application 项目类型，如图 2-5 所示。把 Location 文本框改为 C:\BegVCSharp\Chapter02(如果该目录不存在，会自动创建)。Name 文本框中的默认文本(ConsoleApplication1)和其他设置不变，参见图 2-5。

(3) 单击 OK 按钮。

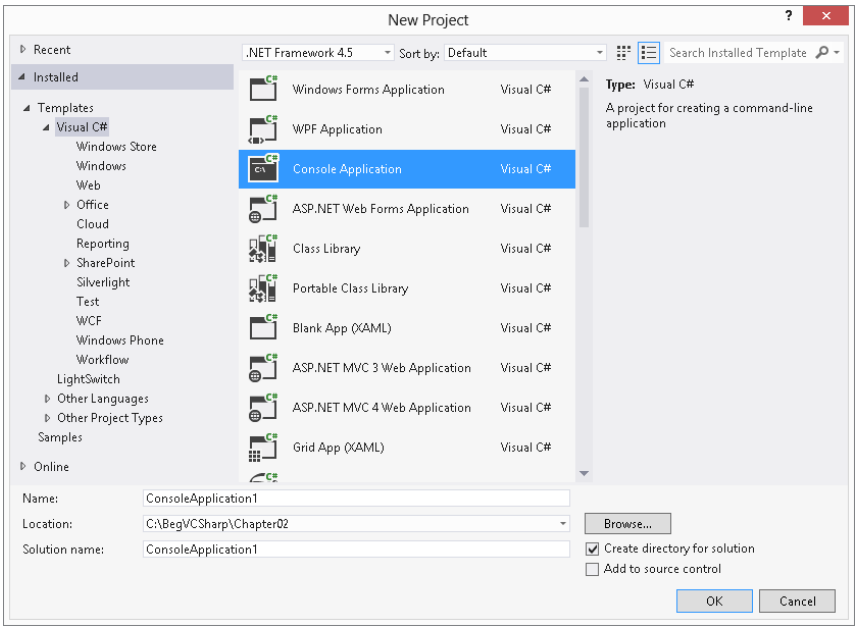

图　2-5

(4) 初始化项目后，在主窗口显示的文件中添加如下代码行：

```
namespace ConsoleApplication1
{
   class Program
   {
      static void Main(string[] args)
      {
         // Output text to the screen.
         Console.WriteLine("The first app in Beginning Visual C# 2012!");
         Console.ReadKey();
      }
   }
}
```

(5) 选择 Debug | Start Debugging 菜单项。稍后将看到如图 2-6 所示的结果。

图　2-6

(6) 按下任意键，退出应用程序(可能首先需要单击控制台窗口，以激活它)。

只有像本章前面描述的那样应用了 Visual C# Developer Settings，才会显示上述内容。例如，若应用了 Visual Basic Developer Settings，就会显示一个空的控制台窗口，应用程序的输出结果显示在 Immediate 窗口中。这种情况下，Console.ReadKey()代码也会失败，显示一个错误。如果遇到这个问题，本书中所有示例的最佳解决方案是应用 Visual C# Developer Settings，这样读者看到的结果才会与书中显示的相同。如果问题未得到解决，可以打开 Tools | Options 对话框，取消选中 Debugging | Redirect all Output Window text to the Immediate Window 选项，如图 2-7 所示。

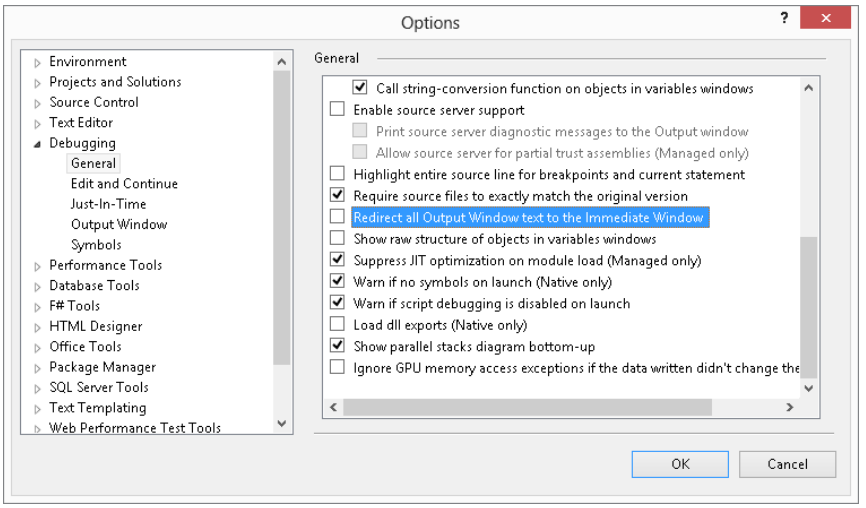

图　2-7

示例的说明

现在不仔细研究这个项目中使用的代码，而关心如何使用开发工具来启动和运行代码。显然，VS 自动完成了许多工作，简化了编译和执行代码的过程。执行这些简单的步骤还有多种方式。例如，创建一个新项目可以像前面那样使用菜单项，也可以按下 Ctrl+Shift+N 组合键，还可以单击工具栏上的相应图标。

同样，也可以采用多种方式编译和执行代码。上面使用的方法是选择 Debug | Start Debugging 菜单项，也可以按下快捷键(F5)，或者使用工具栏中的图标。使用 Debug | Start Without Debugging 菜单项(也可以按下 Ctrl+F5 组合键)还可以以非调试模式运行代码，使用 Build | Build Solution 或 F6 可以编译项目但不运行它(打开或关闭调试功能)。注意，执行项目但不调试，或者使用工具栏中的图标生成项目，只是这些图标在默认情况下没有显示在工具栏中。编译好代码后，在 Windows 资源管理器中运行生成的.exe 文件，就可以执行代码。也可以在命令提示窗口中执行，为此，应打开一个命令提示窗口，把目录改为 C:\BegVCSharp\Chapter02\ConsoleApplication1\ConsoleApplication1\bin\Debug\，键入 ConsoleApplication1，并按下回车键。

在以后的示例中，我们仅说明“创建一个新的控制台项目”或“执行代码”，用户可以选择自己喜欢的方式执行这些步骤。除非特别声明，否则所有的代码都应在启用调试的情况下运行。另外，本书中的“启动”、“执行”和“运行”等术语的含义是相同的，示例后面的讨论总是假定已经退出了示例中的应用程序。

控制台应用程序会在执行完毕后立即终止，如果直接通过 IDE 运行它们，就无法看到运行结果。为了解决上例中的这个问题，使用

```
Console.ReadKey();
```

告诉代码在结束前等待按键。后面的示例将多次使用这种技术。前面创建了一个项目，现在详细讨论开发环境中的各个组成部分。

2.2.1　Solution Explorer 窗口

首先要讨论的窗口是屏幕右上角的 Solution Explorer。与其他窗口一样，可以把它移到任何位置，或者单击其图钉图标将它设为自动隐藏。Solution Explorer 窗口与另一个有用的窗口 Class View 位于相同的位置上，使用 View | Class View 菜单项就可以显示 Class View 窗口。图 2-8 显示了展开所有节点的这两个窗口(在窗口停靠时，单击窗口底部的选项卡，就可以切换它们)。

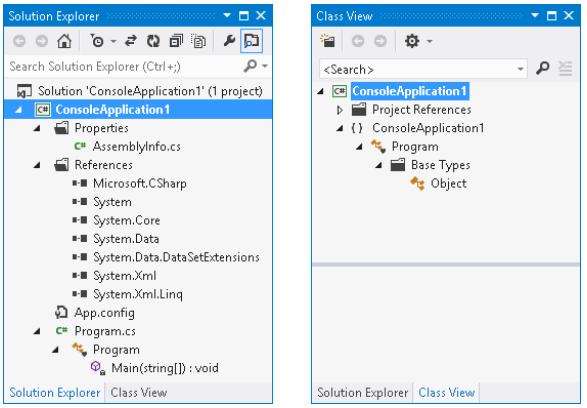

图　2-8

Solution Explorer 窗口显示了组成 ConsoleApplication1 项目的文件，包括我们在其中添加代码的文件 Program.cs、另一个代码文件 AssemblyInfo.cs 和多个引用。

所有 C#代码文件都使用.cs 文件扩展名。

此时不需要考虑 AssemblyInfo.cs 文件，它包含项目中目前我们不必关心的其他信息。

使用这个窗口可以改变主窗口中显示的代码，方法是双击.cs 文件，或右击这些文件并选择 View Code，或选中它们，单击窗口顶部的工具栏按钮。还可以对这些文件执行其他操作，例如，重命名它们，或从项目中删除它们等。在该窗口中还可以显示其他类型的文件，例如，项目资源(资源是项目使用的文件，这些文件可能不是 C#文件，如位图图像和声音文件等)。可以通过同一界面处理它们。

展开代码项(例如 Program.cs)可以查看其中包含的内容。这个代码结构概览是一个很有帮助的工具，可以用来直接定位到代码文件中的特定部分，而不必打开该代码文件并滚动到想要处理的部分。

References 项包含项目中使用的一个.NET 库列表，这个列表在后面介绍，因为标准引用很适于初学者使用。Class View 窗口显示了项目的另一种视图，可以用于查看刚才创建的代码结构。本书后面将介绍代码结构，现在使用 Solution Explorer 窗口就足够了。单击这些窗口中的文件或其他图标，Properties 窗口的内容就会发生相应变化，如图 2-9 所示。

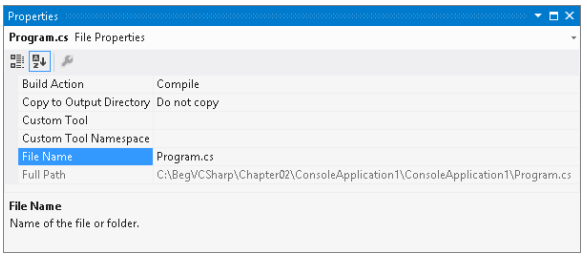

图　2-9

2.2.2　Properties 窗口

使用 View | Properties Window 菜单项就可以打开 Properties 窗口。这个窗口显示了在其上面的窗口中所选的项的其他信息。例如，选择项目中的 Program.cs 文件，就会显示如图 2-9 所示的窗口。这个窗口还显示了其他选中项的信息，例如，用户界面组件(参见本章的 2.3 节“桌面应用程序”)。

通常在 Properties 窗口中对项目的改变会直接影响代码，添加代码行，或改变文件中的内容。对于一些项目来说，通过这个窗口来操作与手动修改代码所用的时间是相同的。

2.2.3　Error List 窗口

当前 Error List 窗口(View | Error List)没有显示什么有趣的信息，这是因为应用程序没有错误。但这的确是一个非常有用的窗口。下面进行测试，从上一节添加的代码中删除某一行的分号。稍后会看到如图 2-10 所示的结果。

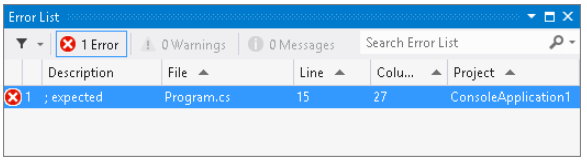

图　2-10

这次项目不会编译。

第 3 章介绍 C#语法后，您就会明白大多数代码行的末尾必须有一个分号。

这个窗口有助于根除代码中的错误，因为它会跟踪我们的工作，编译项目。如果双击该窗口中显示的错误，光标就会跳到源代码中出现错误的地方(如果包含错误的源文件没有打开，将被打开)，这样就可以快速更正错误。代码中有错误的一行会出现红色的波浪线，以便我们快速浏览源代码，找出错误。

注意错误位置用一个行号来指定。默认情况下，行号不会显示在 VS 文本编辑器中，但其实有必要显示它。为此，需要单击 Tools | Options 菜单项，选中 Options 对话框中的 Line numbers 复选框。该复选框位于 Text Editor | All Languages | General 类别中，如图 2-11 所示。

也可以在这个对话框中与各个语言对应的设置页面中针对具体语言单独修改此设置。这个对话框中还包含其他许多有用的选项，本书将使用其中几个。

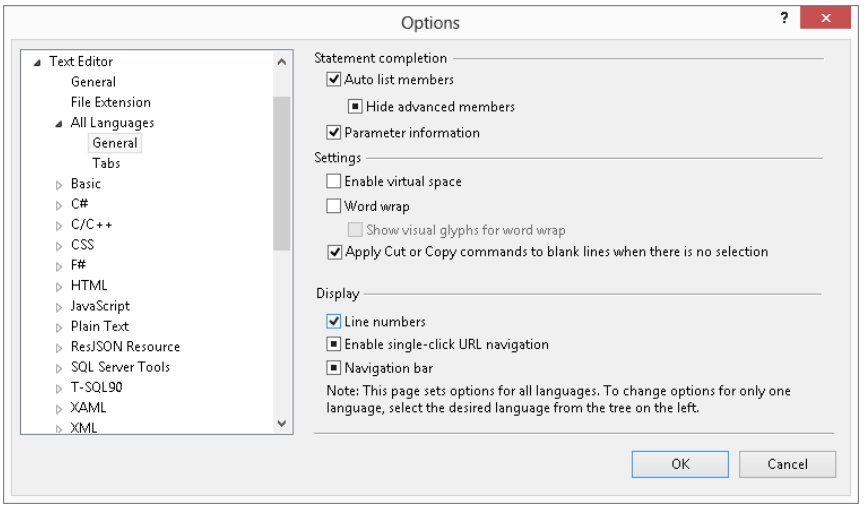

图　2-11

2.3　桌面应用程序

通常，在演示代码时，将其当作桌面 Windows 应用程序的一部分来运行，要比通过控制台窗口或命令提示符来运行更便于说明。下面用用户界面构件来组合一个用户界面。

下面的示例介绍建立用户界面的基础知识，说明如何启动和运行桌面应用程序，但并不详细讨论应用程序实际完成的工作。Microsoft 推荐使用 WPF 技术创建桌面应用程序，所以本例中使用了 WPF。本书后面会详细研究桌面应用程序，以及 WPF 到底是什么，到底可以做些什么。

试一试：创建一个简单的桌面应用程序：WpfApplication1\MainWindow.xaml 和 WpfApplication1\MainWindow.xaml.cs

(1) 在与之前相同的位置(C:\BegVCSharp\Chapter02)创建一个类型为 WPF Application 的新项目，其默认名称是 WpfApplication1。如果第一个项目仍处于打开状态，就应选择 Create New Solution 选项来启动一个新解决方案，这些设置如图 2-12 所示。

(2) 单击 OK 按钮，创建项目后，应该会看到一个新的分成两个窗格的选项卡。上面的窗格显示了一个空窗口，称为 MainWindow，下面的窗格显示了一些文本。这些文本实际上就是用来生成窗口的代码，在修改 UI 时，会看到这些文本也发生了变化。

(3) 单击屏幕左上方的 Toolbox 选项卡，然后双击 Common WPF Controls 区域中的 Button，在窗口中添加一个按钮。

(4) 双击刚才添加到窗体中的按钮。

(5) 现在应显示 MainWindow.xaml.cs 中的 C#代码。执行如下修改(为简短起见，这里只显示了文件中的部分代码)：

```
private void Button_Click_1(object sender, EventArgs e)
{
    MessageBox.Show("The first desktop app in the book!");
}
```

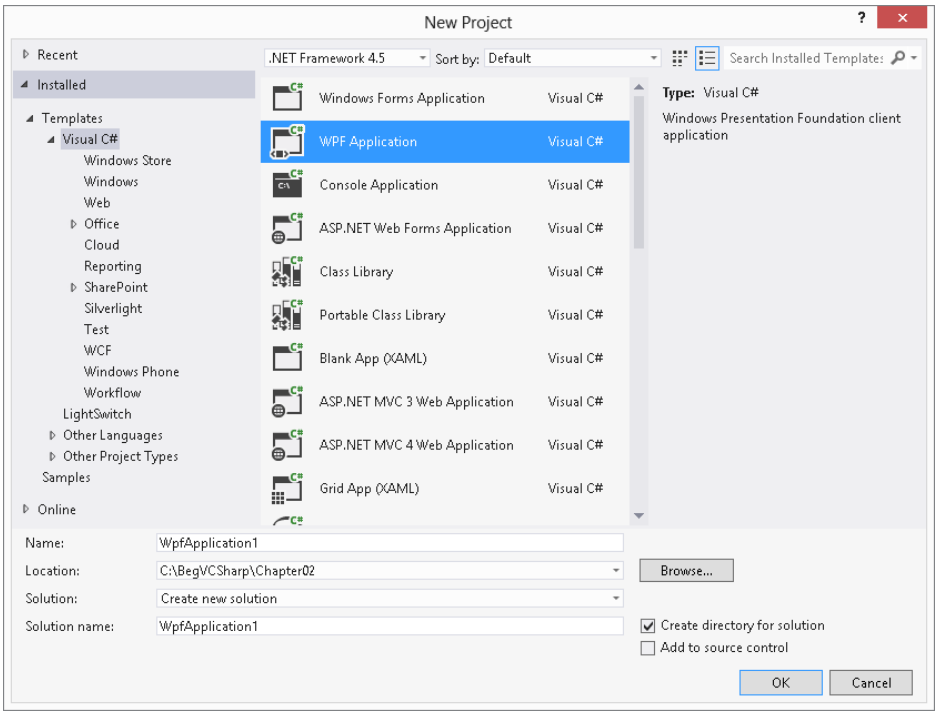

图　2-12

(6) 运行应用程序。

(7) 单击显示出来的按钮，打开一个消息对话框，如图 2-13 所示。

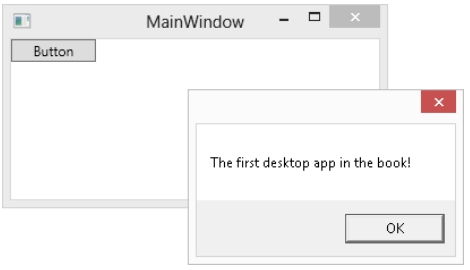

图　2-13

(8) 单击 OK。像每个标准桌面应用程序那样，单击右上角的 X 图标，退出应用程序。

示例的说明

IDE 又一次自动完成了许多工作，使我们不费吹灰之力就能完成一个实用的桌面应用程序的创建。刚才创建的应用程序与其他窗口的行为方式相同 ——可以移动、重新设置其大小、最小化等。我们不必编写任何代码来实现这种功能。我们添加的按钮也是这样。双击按钮，IDE 就知道我们想添加一些代码，当运行应用程序时，用户单击该按钮，就执行我们已经编写好的代码。只要提供了这段代码，就可以得到按钮单击的所有功能。

当然，桌面应用程序不仅限于带有按钮的普通窗口。如果看看从中选择 Button 选项的工具箱，就会看到一整套用户界面构件(称为控件)，其中一些用户可能很熟悉。本书在其他地方将使用其中的大多数用户界面构件，它们使用起来都非常简单，可以节省许多时间和精力。

应用程序的代码在 MainWindow.xaml.cs 中，看起来并不比上一节提供的代码复杂多少，Solution Explorer 窗口中其他文件的代码也不太复杂。MainWindow.xaml 中的代码(可在添加按钮的拆分窗格视图中看到)看上去也很简单。在窗口的图形化表示下，可以看到下面的代码：

```
<Window x:Class="WpfApplication1.MainWindow"
  xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
  xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
  Title="MainWindow" Height="350" Width="525">
  <Grid>
    <Button Content="Button" HorizontalAlignment="Left"
      VerticalAlignment="Top" Width="75" Click="Button_Click_1" />
  </Grid>
</Window>
```

这是一段 XAML 代码。XAML 是在 WPF 应用程序中定义用户界面的语言。

下面仔细分析一下在窗口中添加的按钮。在 MainWindow.xaml 的顶部窗格中，单击按钮一次选中它。此时屏幕右下角的 Properties 窗口显示了按钮控件的属性(控件也有属性，就像上一个示例中的文件一样)。确保应用程序当前没有运行，然后向下滚动到 Content 属性，该属性现在被设为 Button。将它设为 Click Me，如图 2-14 所示。

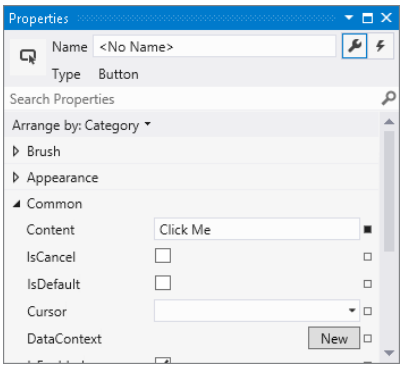

图　2-14

设计器中按钮上的文本以及 XAML 代码也会反映这种变化，如图 2-15 所示。

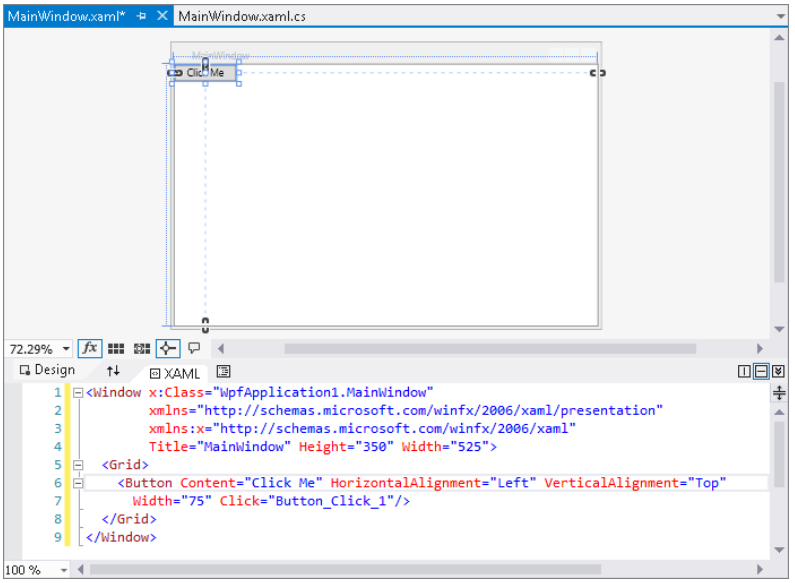

图　2-15

这个按钮具有许多属性，从按钮颜色和大小的简单格式，到某些模糊设置(如数据绑定设置，它可以建立与数据的联系)，应有尽有。如上例所述，改变属性通常会直接改变代码，这也不例外，从

XAML 代码的改变中可以看到这一点。但如果切换回 MainWindow.xaml.cs 的代码视图，是看不到代码发生变化的。这是因为 WPF 应用程序能够保持应用程序的设计(如按钮上的文本)与功能(例如单击按钮后发生的操作)的分离。

也可以使用 Windows Forms 来创建桌面应用程序。但 WPF 是一种更新的技术，能够以更灵活、更强大的方式创建桌面应用程序，而且其目的就是取代 Windows Form，所以本书中不讨论 Windows Forms。

2.4　小结

本章介绍了本书后面所使用的一些工具，快速浏览了 Visual Studio 2012 开发环境，并使用它建立了两种类型的应用程序。其中较简单的是控制台应用程序，它足以满足我们的大多数需要，便于我们集中精力学习 C#编程的基础知识。桌面应用程序比较复杂，但其可视化程度比较高，对于习惯了 Windows 环境的人而言，使用起来也比较直观。

知道如何创建简单的应用程序，就可以真正开始学习 C#了。本书后面的章节将介绍 C#的基本语法和程序结构，之后讨论更高级的面向对象方法。学习了这些内容后，就可以开始了解如何使用 C#访问.NET Framework 的功能了。

2.5　本章要点

主　题	要　点
Visual Studio 2012 设置	本书需要在第一次运行 VS 时选择 C# Development Settings 选项，或者重置它们
控制台应用程序	控制台应用程序是简单的命令行应用程序，本书主要用它演示技术。在 VS 中创建新项目时，使用 Console Application 模板就会创建新的控制台应用程序。要在调试模式下运行项目，可使用 Debug | Start Debugging 菜单项或者按下 F5
IDE 窗口	项目内容显示在 Solution Explorer 窗口中。选中项的属性显示在 Properties 窗口中。错误显示在 Error List 窗口中
桌面应用程序	桌面应用程序具备标准桌面应用程序的外观和操作方式，包括最大化、最小化和关闭应用程序等大家熟悉的图标。它们是在 New Project 对话框中用 WPF Application 模板创建的