第 1 章

C# 简 介

本章内容：

- .NET Framework
- .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

本章源代码下载

本章源代码的下载网址为 www.wrox.com/go/beginningvisualc#2015programming。从该网页的 Download Code 选项卡中下载 Chapter 1 Code 后，可以找到与本章示例对应的单独文件。

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。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话题。

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 2015 (VS 2015)。VS 2015 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(现在最新版本是 4.6)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，详见 http://www.mono-project.com。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、云/Web 应用程序、Web API 和其他各种类型的应用程序。另外注意，对于 Web、云和 Web API 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。要获得这些语言的列表，可以访问 https://msdn.microsoft.com/library/aa292164.aspx。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)。现在就没必要了，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。

提示：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

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，访问操作系统的底层功能。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行完成清理工作，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 应用程序所需的步骤：

(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。

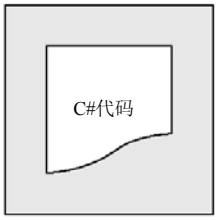

图　1-1

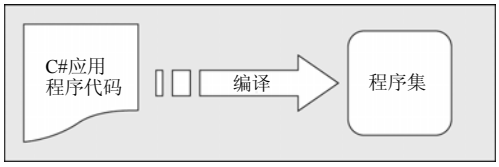

图　1-2

(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。

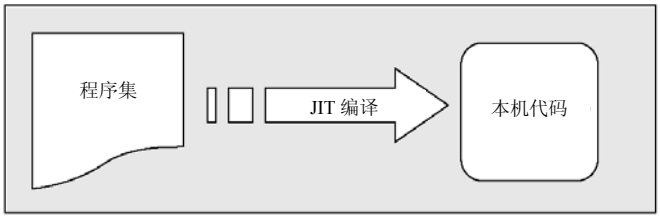

图　1-3

(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。

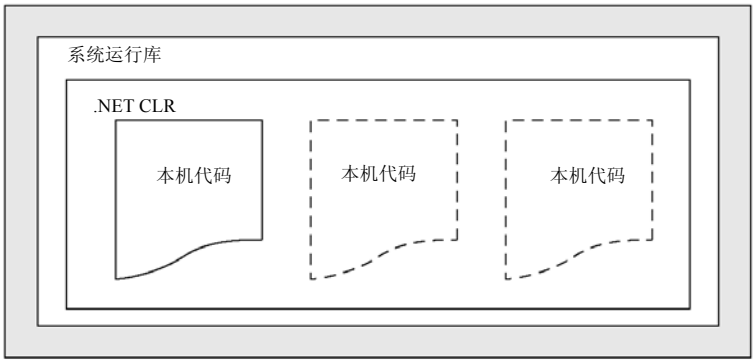

图　1-4

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。C#吸取了以往语言失败的教训，考虑了其他语言的许多优点，并解决了它们存在的问题。

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。顾名思义，这个高级编程技术存在潜在威胁，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 引入的一类新的应用程序。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。
- **云/Web 应用程序** .NET Framework 包括一个动态生成 Web 内容的强大系统——ASP.NET，允许进行个性化和实现安全性等。另外，这些应用程序可以在云中驻留和访问，例如 Microsoft Azure 平台。
- **Web API** 这是建立 REST 风格的 HTTP 服务的理想框架，支持许多客户端，包括移动设备和浏览器。
- **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)主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，这些是首要原则。

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。本书第Ⅱ部分将研究基于云的 Web 应用程序编程，第Ⅲ部分将讲述数据访问(对 ORM 数据库、文件系统和 XML 数据的访问)和 LINQ，第Ⅳ部分将详细讨论桌面和 Windows Store 应用程序编程。

1.3 Visual Studio 2015

本书使用 Visual Studio 2015 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？

1.3.1　Visual Studio Express 2015 产品

除 Visual Studio 2015 外，Microsoft 还提供了几个更简单的开发工具，称为 Visual Studio Express 或 Community 2015 产品。可以在 https://www.visualstudio.com/en-us/downloads/download-visual-studio-vs 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。

注意：由于在写作本书时 Express 版本还不可用，本书使用了 Visual Studio 2015 的企业版。在写作本书时，有一个称为 Visual Studio Express 2015 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 2015 for Windows 10 和 Visual Studio Express 2015 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 Cloud/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被编译为多个程序集，也可以把它们组合到一处。

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　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 4.6。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编程(OOP)的方法论编写，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

(续表)

主　题	要　点
C#基础	C#是包含在.NET Framework 中的一种语言，它是已有语言(如 C++)的一种演变，可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)	可在 Visual Studio 2015 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