第 1 部分

C#实用教程

第 1 章

C#概述

C#是以.NET 为应用开发平台而全新设计的一种现代编程语言，随着微软公司.NET 战略进入开发人员的视野后，C#很快成为 Windows 应用开发语言中的宠儿。本章将提供给读者 C#快速入门的指导，帮助读者对 C#有一个准确的定位，并能对自己的学习做好安排。

1.1 C#语言简介

1.1.1 C#的优势

作为编程语言，C#是简单的、完全面向对象的，且是类型安全的语言。最重要的是，它是一种现代编程语言。在类、名字空间、方法重载和异常处理等方面，C#去掉了 C++中的许多复杂性，借鉴和修改了 Java 的许多特性，使其更加易于使用，不易出错。

下面列举了 C#在设计上较其他语言的优势。

1．简单性

C#在设计上的首要目标就是简单性。没有指针是 C#的一个显著特性。在默认情况下，用户使用一种可操控的代码进行工作时，一些不安全的操作，如直接地进行内存存取，将是不允许的。

在 C#中，不再需要记住那些源于不同处理器结构的数据类型，C#在 CLR 层面统一了它们，使得基于.NET 平台上的不同语言拥有相同的数据类型。我们可以将每种类型看成一个对象，不管它是初始数据类型还是完全的类。

整型和布尔型是完全不同的，这意味着 if 判别式的结果只能是布尔型，如果是别的类型，编译器则会报错。混淆比较和赋值运算的错误将不会再发生。

2．现代性

许多在传统语言中必须由用户自己来实现或者没有的特性，都成为 C#语言中的基础部分。增加的金融类型对于企业级编程语言来说是很受欢迎的一个附加类型，用户可以使用一个新的 decimal 类型进行货币计算。

3．高度面向对象

C#语言面向对象的程度比 Java 还要高，除了支持面向对象的所有概念：封装、继承和多态外，还增加了委托、事件、集合、泛型等诸多新概念，并且整个 C#的对象模型是建立在.NET 虚拟对象系统（Virtual Object System，VOS）之上的，是基础架构的一部分，而不仅

仅是编程语言的特性。

C#中没有孤立的全局函数、变量或常数。每个对象都必须封装在一个类中，要么作为一个实例成员（通过类的实例对象访问），要么作为一个静态成员（通过类型来访问），这样就使用户的代码具有更好的可读性，同时减少了发生命名冲突的可能。

多重继承的优劣一直是面向对象领域争论最多的话题之一，因此 C#只允许继承一个基类，如果需要多重继承，可以使用接口。

4．类型安全性

C#实施了最严格的类型安全机制来保护它自身及其垃圾收集器。因此，程序员必须遵守关于变量的一些规定，比如不能使用未初始化的变量，否则编译器将会警告用户；对于对象的成员变量，编译器负责将它们置零；当数组实际上只有 n–1 个元素时，不可能访问到它“额外”的元素 n，这使重写未经分配的内存成为不可能。C#允许在应用级或语句级检查算术运算操作中的溢出，当溢出发生时会出现一个异常。此外，C#中传递的引用参数都是类型安全的。

5．版本处理

动态链接库（Dynamic Link Library，DLL）是一种被编译为二进制代码的函数库，在程序运行时才被调入内存执行，而不是在编译时链接到可执行程序内部，这样就在二进制级别实现了代码共享，而不必在每个应用程序中编译一个副本，如果 DLL 中的代码更新了，只需替换相应的 DLL 文件即可。然而，这就带来了 DLL 版本的问题，不同版本的 DLL 可能与调用程序不兼容，同一版本的 DLL 也不能同时为不同的调用程序服务，造成应用程序出现无法预料的错误，于是不得不更换文件名来保存同一 DLL 的多个版本——这种混乱的状态被称为“DLL 地狱”。不过令人欣慰的是，C#提供了一种版本处理机制，使得程序员可以确保当他开发的类库升级时，会与已有的应用保持二进制级别上的兼容。

1.1.2　.NET 框架

.NET 框架（.NET Framework）可以建立.NET 应用程序。使用.NET 开发的程序需要在.NET Framework 下才能运行。.NET Framework 体系结构主要包括下列五大部分：

（1）程序设计语言及公共语言规范（CLS）；

（2）应用程序平台（ASP.NET 及 Windows 应用程序等）；

（3）ADO.NET 及类库；

（4）公共语言运行时（CLR）；

（5）程序开发环境（Visual Studio）。

.NET 框架的体系结构如图 1.1 所示。构建在 Windows 操作系统之上的是公共语言运行时（CLR），其作用是负责执行程序，提供内存管理、线程管理、安全管理、异常处理、通用类型系统与生命周期监控等核心服务。在 CLR 之上的是基类库，提供许多类与接口。

.NET Framework 类库是以命名空间（Namespace）方式来组织的，命名空间与类库的关系就像文件系统中的目录与文件的关系一样，如用于处理文件的类属于 System.IO 命名空间。

在.NET Framework 基础上的应用程序主要包括 ASP.NET 应用程序和 Windows Forms 应用程序，其中 ASP.NET 应用程序又包含了 Web Forms 和 Web Service，它们组成了全新的

Internet 应用程序；而 Windows Forms 则是传统的窗口应用程序。

在.NET Framework 之上，无论采用哪种语言编写的程序，都先被编译成中间语言 IL，IL 经过再次编译后才生成机器码，完成 IL 到机器码编译任务的是 JIT（Just In Time）编译器。上述处理过程如图 1.2 所示。

图 1.1 .NET 框架结构

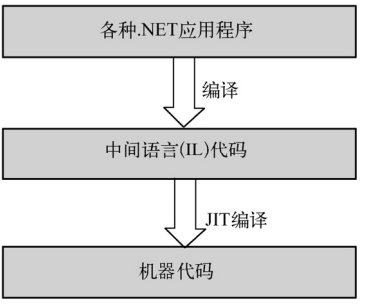

图 1.2 .NET 应用程序的编译过程

随着.NET 技术的不断发展，.NET Framework 的发展也经历了几个阶段，从早期的.NET Framework 1.0/1.1 发展到 2.0、3.0、3.5、4.0、4.5、4.6，功能越来越强。.NET Framework 的最新版本为 4.7。

Microsoft Visual Studio（简称 VS）是微软.NET 平台的集成开发环境（IDE），其功能强大，整合了多种开发语言（包括 Visual Basic、Visual C++、Visual C#、Visual F#），集代码编辑、调试、测试、打包、部署等功能于一体，大大提高了开发效率。本书介绍 Visual C#，最新平台为 Visual Studio 2017。

对初学者来说，并不能明显体会到.NET Framework 功能的改进与增强，但对于 Visual C# 开发人员来说获得的的确是真金白银。它使用户在开发环境下解决问题变得越来越容易，运行的性能越来越高，部分自己原来通过程序才能解决的问题不再需要，原来编程都很难解决的问题变得容易解决。

1.2 Visual C#（2015/2017）开发环境

1.2.1 Visual C#环境安装与设置

VS 目前总共有三个版本：社区版（Community）、专业版（Professional）和企业版（Enterprise）。其中，企业版的功能最为强大，本书将以 Visual Studio 2015 Enterprise 版本作为程序开发的平台，Visual Studio 2017 版本与 2015 版本基本使用方法大致相同。

1．Visual Studio 安装

本书在 32 位 Windows 7 旗舰版操作系统上安装 VS，步骤如下。

（1）从网上下载 VS 的安装包文件（如 vs2015.ent_chs.iso），双击该文件解压，双击其中的 vs_enterprise.exe 启动安装向导。

（2）在“选择安装位置”栏单击...指定 VS 的安装路径（本书采用默认）；在“选择安装类型”下选中“自定义”选项（如图 1.3 所示），单击“下一步”按钮。

（3）在“选择功能”页，列出了 VS 包含的全部功能组件选项，其中前面打钩的是选中需要

安装的组件。除了勾选 Visual Studio，为了使 VS 支持数据库应用开发，请确保勾选 Microsoft SQL Server Data Tools 项（如图 1.4 所示）。其他选项用户可根据实际开发的需要决定勾选或取消勾选。确认选择后，单击“下一步”按钮继续。

图 1.3　自定义安装类型

图 1.4　选择功能组件

（4）在“选定的功能”页，列出了上一步选中的所有组件让用户进一步确认，确认无误后，单击“安装”按钮开始安装进程。若发现有漏选或希望增加安装其他组件，可单击“上一步”回退至“选择功能”页，补充选择。

（5）然后，向导进入正式的安装进程，根据用户的选择依次从安装包中解压获取各个组件并应用（安装上），两个进度条分别指示了当前组件和总的安装进度。因为选择的安装选项不同，VS 的安装时间差别很大，在此期间请不要关闭、重启计算机，最好也不要进行任何其他操作，请读者耐心等待。

（6）出现“安装已完成”页，由于操作系统兼容性缘故，通常并非所有的功能都能成功安装，在下方问题列表中会列出未能成功安装的组件，请用户稍做检查。若不是 VS 的核心功能组件，即使未能成功安装也不会影响接下来的正常使用，可忽略（例如，本书用 Windows 7 系统，所以 Windows 10 的 SDK 不能安装），如图 1.5 所示。单击“立即重新启动”按钮，重启计算机。

（7）重启计算机后，单击桌面“开始”→“所有程序”→“Visual Studio 2015”项，出现“欢迎使用”页，在此页上可申请微软官方的开发人员账户，但作为初学者来说，没有必要申请专业开发账户，直接单击下方的“以后再说”，启动 VS 2015 环境，如图 1.6 所示。

（8）由于是初次启动，系统会弹出对话框让用户设置初始的开发环境，如图 1.7 所示。从“开发设置”下拉列表中选择 Visual C#，表示初始设置为 C#语言的编程环境，开发环境的颜色主题选为“蓝色”。然后，单击“启动 Visual Studio”按钮。

图 1.5　安装完成

图 1.6　启动 VS 2015 环境

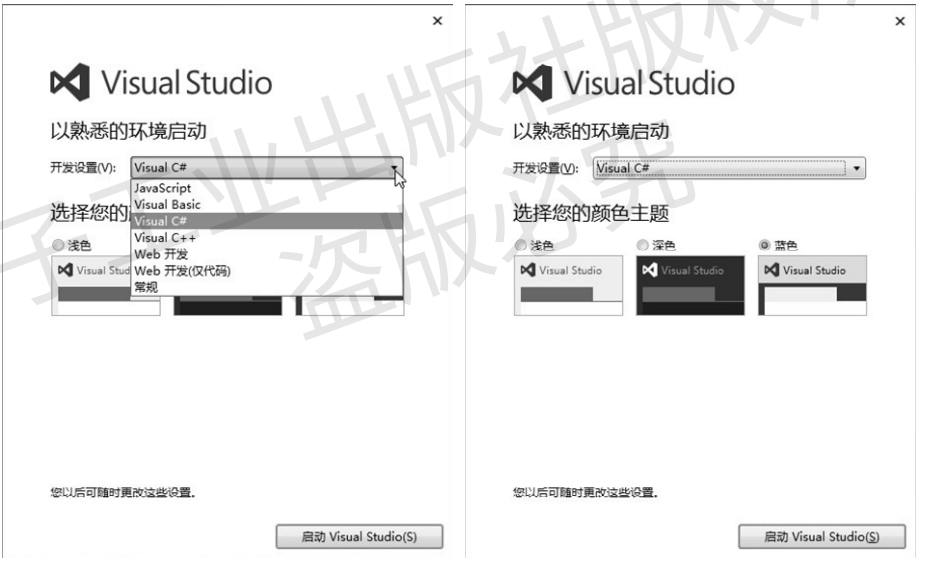

图 1.7　设置初始开发环境

（9）系统开始为第一次使用 VS 做准备，稍等几分钟，待启动完成即可进入 VS 2015 开发环境。

2．Visual C#环境设置

除了可在安装完成后，于初次启动时指定初始环境，用户还可在任何时候重置开发环境，步骤如下。

（1）选择主菜单“工具”→“导入和导出设置”，在“导入和导出设置向导”对话框中点选“重置所有设置”，单击“下一步”按钮，如图 1.8 所示。

（2）在“保存当前设置”页选择“否，仅重置设置，从而覆盖我的当前设置”，单击“下一

步”按钮，如图 1.9 所示。

图 1.8　重置 VS 开发环境

图 1.9　覆盖当前设置

（3）在“选择一个默认设置集合”页的“要重置为哪个设置集合？”列表中选择“Visual C#”，单击“完成”按钮，设置成 C#的编程环境，如图 1.10 所示。

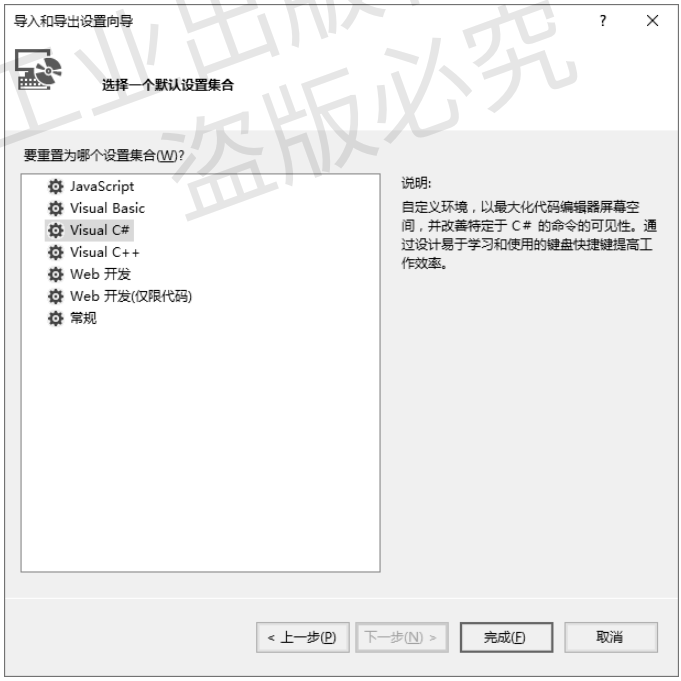

图 1.10　设置为 C#开发环境

3．Visual C#开发环境

经过配置后，打开 VS 2015 主窗口，显示“起始页”界面，如图 1.11 所示。

图 1.11 “起始页”界面

在“起始页”界面中，允许用户新建或打开项目。若要打开已有项目，可单击最近的项目列表中的某个项目名称；也可以依次单击菜单“文件”→“打开”→“项目/解决方案”，在弹出的“打开项目”对话框中选择要打开的项目。

1.2.2 Visual Studio 项目管理

为了能有效地管理各类应用程序的开发，VS 2015 提供了两类“容器”：一是项目，二是解决方案。那么，它们是什么？又是如何管理的呢？

1．项目与解决方案

VS 2015 开发的程序可以表现为多种应用类型，如控制台应用程序、Windows 窗体应用程序、WPF 应用程序、ASP.NET Web 应用程序、类库等。而 VS 2015 的“项目”以逻辑方式管理、生成和调试构成应用程序的诸多项，包括创建应用程序所需的引用、数据链接、文件夹和文件等。“项目”的输出通常是可执行程序（.exe）、动态链接库（.dll）文件或模块等。

解决方案是一类相关项目的集合，一个解决方案可包含多个项目。VS 2015 还为解决方案提供了指定的文件夹，用于管理和组织该方案下的各种项目和项目组。同时，在该文件夹下还有一个扩展名为.sln 的解决方案文件。

2．解决方案资源管理器

作为查看和管理解决方案、项目及其关联项的界面，“解决方案资源管理器”是 VS 2015 开发环境的一部分。它将方案中所有关联的项以“树视图”的形式分类显示。针对 Visual C#，这些项包括 Properties（程序集属性）、引用（名字空间）、App.config（应用配置）和.cs 文件（源文件）等，单击节点名称图标前的“▷”或“◢”符号，或双击图标，将显示或隐藏节点下的相关内容，如图 1.12（左）所示。

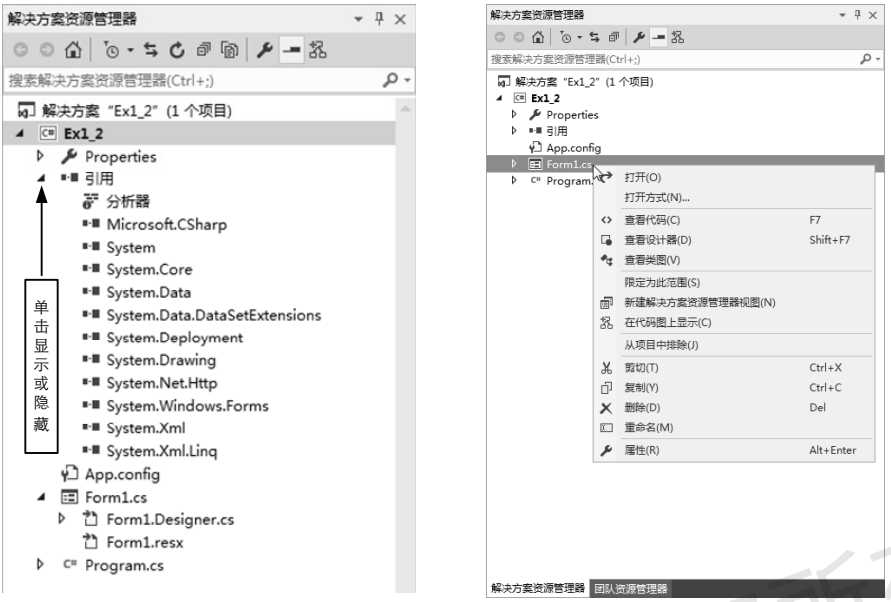

图 1.12 “解决方案资源管理器”窗口

“解决方案资源管理器”窗口的顶部有几个工具图标。其中，用来显示“树视图”中所选项的相应“属性页”对话框；用来显示所有的文件，包括那些已经被排除的项和在正常情况下隐藏的项；而用来查看代码图，帮助理解复杂的代码。

需要说明的是，选择的节点项不同，对应的窗口顶部出现的工具图标也不同。同时，右键单击节点显示的快捷菜单也各不相同。例如，右键单击 Form1.cs 节点，弹出如图 1.12（右）所示的快捷菜单，从中可选择相应的命令和操作。

1.2.3 Visual Studio IDE 界面元素

1．标题栏

标题栏是 VS 2015 窗口顶部的水平条，它显示的是应用程序的名字。在默认情况下，用户建立一个新项目后，标题栏显示的是如下信息：

WindowsFormsApplication1 – Microsoft Visual Studio（管理员）

其中，“WindowsFormsApplication1”代表解决方案名称。随着工作状态的变化，标题栏中的信息也随之改变。当处于调试状态时，标题栏显示：

WindowsFormsApplication1（正在调试） – Microsoft Visual Studio（管理员）

在上面的标题信息中，第一个括号中的“正在调试”表明当前的工作状态处于“调试阶段”。当处于运行状态时，该括号中的信息为“正在运行”，表明当前的工作状态处于“运行阶段”。

2．菜单栏

标题栏的下面是主菜单。菜单是 Visual C#编程环境的重要组成部分，开发者要完成的主要功能都可以通过菜单或与菜单对应的工具栏按钮及快捷键来实现。在不同的状态下，菜单栏中菜单项的个数是不一样的。

启动 VS 2015 后，在建立项目前（即“起始页”状态下），菜单栏有 11 个菜单：文件、编辑、视图、调试、团队、工具、体系结构、测试、分析、窗口和帮助。当建立或打开项目后，如果

当前活动的窗口是窗体设计器，则菜单栏中有与此相关的 14 个菜单；而如果当前活动的是代码窗口，则菜单栏中另有与之相关的 13 个菜单。

每个菜单包含若干个子菜单（项），灰色的选项是不可用的；菜单名后面“( )”中的字母为键盘访问键，某些菜单项后显示的字母组合为快捷键。例如，“新建项目”的操作可以先按 Alt+F 组合键打开“文件”菜单，再按 N 键，或直接按 Ctrl+Shift+N 组合键，如图 1.13 所示。

图 1.13　文件菜单的快捷访问

（1）文件菜单（File）

文件菜单用于对文件进行操作，如新建和打开项目，以及保存和退出等。文件菜单如图 1.13 所示，对应的主要功能见表 1.1。

表 1.1　文件菜单功能表

菜　单　项	功　　能
新建	包括新建项目、网站和文件等
打开	包括打开项目/解决方案、网站和文件等
关闭	关闭当前项
关闭解决方案	关闭打开的解决方案
保存选定项	保存对选定项的修改，文件名不变
将选定项另存为	将选定项另存为其他文件名
全部保存	保存当前打开的所有项目
导出模板	将项目或项导出为可用作将来项目的基础的模板
源代码管理	包括查找/应用标签、从服务器打开、工作区等
账户设置	登录到微软 Visual Studio 官网，在线管理和发布程序代码
退出	退出 VS 2015 集成开发环境

（2）视图菜单（View）

视图菜单用于显示或隐藏各功能窗口或对话框。若不小心关闭了某个窗口，可以通过选择视图菜单项恢复显示。视图菜单同时还控制工具栏的显示，若要显示或关闭某个工具栏，只需单击“视图”→“工具栏”，找到相应的工具栏，在其前面打钩或去掉钩即可。视图菜单如

图 1.14 所示，其主要功能见表 1.2。

表 1.2　视图菜单主要功能表

菜　单　项	功　　能
解决方案资源管理器	打开解决方案资源管理器窗口
服务器资源管理器	打开服务器资源管理器窗口
类视图	打开类视图窗口
对象浏览器	打开对象浏览器窗口
工具箱	打开工具箱窗口
其他窗口	打开命令、Web 浏览器、属性管理器等其他窗口
工具栏	打开或关闭各种快捷工具栏
属性窗口	打开用户控件的属性页

（3）项目菜单（Project）

项目菜单只有在打开某个项目后才会显现，如图 1.15 所示，主要用于向程序中添加或移除各种元素，如窗体、控件、组件和类等。菜单中的功能使用较简单，其中两个重要菜单项见表 1.3。

图 1.14　视图菜单

图 1.15　项目菜单

表 1.3　项目菜单功能表

菜　单　项	功　　能
添加 Windows 窗体	向项目中添加新窗体
添加服务引用	添加一个 Web 服务引用或添加 WCF 服务引用

（4）格式菜单（Format）

格式菜单用于在设计阶段对窗体中各个控件进行布局。使用它可以对所选定的对象进行格式调整，在设计多个对象时用来使界面整齐划一。格式菜单如图 1.16 所示，主要功能见表 1.4。

图 1.16　格式菜单

表 1.4　格式菜单功能表

菜　单　项	功　　能
对齐	所有选中的对象对齐
使大小相同	所有选中的对象按宽或高统一尺寸
水平间距	对所有选中的对象水平间距统一调整
垂直间距	对所有选中的对象垂直间距统一调整
窗体内居中	对象在窗体中居中对齐
顺序	对象按前、后顺序放置
锁定控件	使所选中的控件锁定，不能调整位置

（5）调试菜单（Debug）

调试菜单用于选择不同调试程序的方法，如逐语句、逐过程、设断点等。调试菜单如图 1.17 所示，对应主要功能见表 1.5。

图 1.17　调试菜单

表 1.5　调试菜单功能表

菜　单　项	功　　能
开始调试	以调试模式运行
开始执行（不调试）	不调试，直接运行
逐语句	一句一句运行
逐过程	一个过程一个过程运行
新建断点	用于设置新断点
删除所有断点	清除所有已设置的断点

（6）工具菜单（Tools）

工具菜单用于选择设计程序时使用的一些工具，例如，可用来添加/删除工具箱项、连接数据库、连接服务器等。工具菜单如图 1.18 所示。

（7）生成菜单（Build）

生成菜单主要用于生成能运行的可执行程序文件。生成之后的程序可以脱离开发环境独立运行，也可以用于发布程序。

（8）帮助菜单（Help）

学会使用帮助菜单是学习和掌握 C#的捷径。可以通过内容、索引和搜索的方法寻求帮助，帮助菜单如图 1.19 所示。

图 1.18　工具菜单

图 1.19　帮助菜单

（9）其他菜单

菜单栏中还有“编辑”和“窗口”等菜单，它们的功能与标准 Windows 程序的基本相同，在此不再详细介绍。

另外，除了菜单栏中的菜单外，若在不同的窗口中单击鼠标右键，还可以得到相应的专用快捷菜单，也称为上下文菜单或弹出菜单。

3．工具栏

单击工具栏上的按钮，则执行该按钮所代表的操作。Visual C#提供了多种工具栏，并可根据需要定义用户自己的工具栏。默认情况下，Visual C#中只显示标准工具栏和布局工具栏，其他工具栏可以通过“视图”→“工具栏”命令打开（或关闭）。每种工具栏都有固定和浮动两种形式，把鼠标光标移到固定形式工具栏中没有图标的地方，按住左键向下拖动鼠标，即可把工具栏变为浮动的，而如果双击浮动工具栏的标题，则又还原为固定工具栏。

默认的工具栏如图 1.20 所示，这是启动 Visual C#之后显示的默认工具栏，当鼠标停留在工具栏按钮上时会显示出该按钮的功能提示。

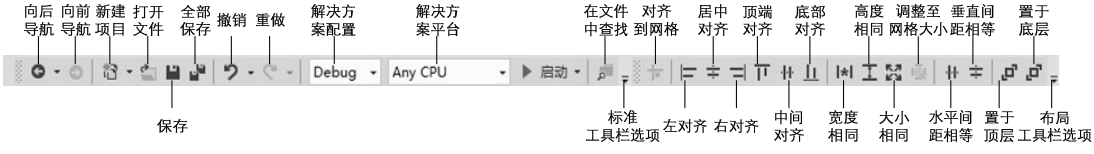

图 1.20　默认工具栏

工具栏中常用按钮的功能见表 1.6。

表 1.6　工具栏按钮功能表

名　称	功　能
新建项目	相当于文件菜单中“新建”→“项目”菜单项
打开文件	相当于文件菜单中“打开”→“文件”菜单项
保存	相当于文件菜单中“保存”菜单项
全部保存	相当于文件菜单中“全部保存”菜单项

续表

名　称	功　能
撤销、重做	对应编辑菜单中“撤销”、“重做”菜单项
启动	相当于调试菜单中“开始调试”菜单项
在文件中查找	相当于编辑菜单中“查找和替换”→“在文件中查找”菜单项
对齐到网格、左对齐、居中对齐、右对齐、顶端对齐、中间对齐、底部对齐	对应格式菜单中“对齐”子菜单下的各同名项
使宽度相同、使高度相同、使大小相同、调整至网格大小	分别对应格式菜单中“使大小相同”→“宽度”、“高度”、“两者”和“调整至网格大小”等子菜单项
使水平间距相等、使垂直间距相等	分别对应格式菜单中“水平间距”→“相同间隔”，“垂直间距”→“相同间隔”菜单项
置于顶层、置于底层	分别对应格式菜单中“顺序”→“置于顶层”、“置于底层”子菜单项

4．工具箱

工具箱（Toolbox）提供了一组控件，用户在设计界面时可以选择所需要的控件放入窗体中。工具箱位于屏幕的左侧，如图 1.21 所示，默认情况下是自动隐藏的，当鼠标接近工具箱“敏感”区域单击时，工具箱会弹出，鼠标离开后又会自动隐藏。

从图 1.21 可以看出，工具箱是由众多控件组成的，为便于管理，常用的控件被分门别类地放在“所有 Windows 窗体”、“公共控件”、“容器”、“菜单和工具栏”、“数据”、“组件”、“打印”、“对话框”、“WPF 互操作性”、“常规”共 10 个选项卡中，如图 1.22 所示。比如，在“所有 Windows 窗体”选项卡中，存放了常用的命令按钮、标签、文本框等控件。10 个选项卡中存放的控件见表 1.7。

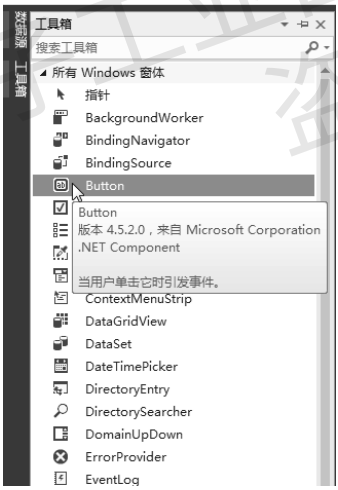

图 1.21　控件工具箱

图 1.22　工具箱选项卡

表 1.7　工具箱选项卡中存放的控件

选项卡名称	内 容 说 明
所有 Windows 窗体	存放 Windows 程序界面设计所有的控件
公共控件	存放常用的控件
容器	存放容器类的控件
菜单和工具栏	存放菜单和工具栏类控件

续表

选项卡名称	内 容 说 明
数据	存放操作数据库的控件
组件	存放系统提供的组件
打印	存放打印相关的控件
对话框	存放各种对话框控件
WPF 互操作性	存放 WPF 相关的控件
常规	保存了用户常用的控件（包括自定义控件）

选项卡中的控件不是一成不变的，用户可以根据需要增加或删除。在工具箱窗口中单击鼠标右键，在弹出的菜单中选择“选择项”，会弹出一个包含所有可选控件的“选择工具箱项”对话框，如图 1.23 所示，通过勾选或取消勾选其中的各类控件，即可添加或删除选项卡中的控件。

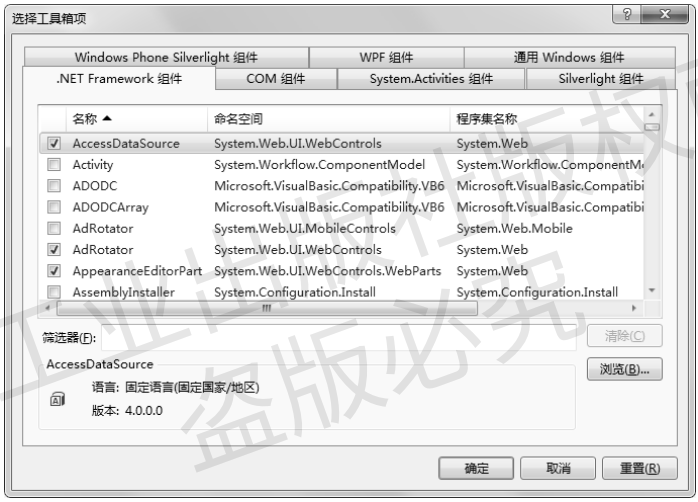

图 1.23 “选择工具箱项”对话框

5．窗口

除前面提到过的“解决方案资源管理器”窗口之外，VS 2015 还有“属性”窗口、“窗体设计器”窗口等诸多功能窗口，它们都可由用户通过“视图”菜单来设置显现或隐藏。

（1）窗体设计器

“窗体设计器”窗口简称窗体（Form），是用户自定义窗口，用来设计应用程序的图形界面，它对应的是程序运行的最终结果。各种图形、图像、数据等都是通过窗体或其中的控件显示的。

（2）属性窗口

属性窗口位于“解决方案资源管理器”的下方，用于列出当前选定窗体或控件的属性设置，属性即对象的特征。图 1.24 是名称为 Form1 的窗体对象的属性。

属性的显示方式有 2 种，图 1.24 是按“分类顺序”排列各个属性的，而图 1.25 则是按“字母顺序”排列各个属性的，在属性窗口的上部有一个工具栏，用户可以通过单击其中相应的工具按钮来改变属性的排列方式。

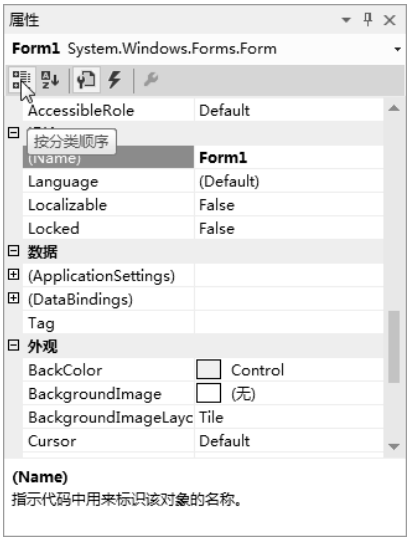

图 1.24　属性窗口（按分类排序）

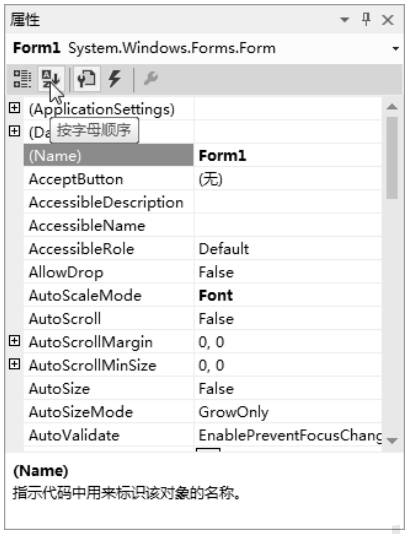

图 1.25　属性窗口（按字母排序）

类和名称空间位于属性窗口的顶部，其下拉列表中的内容是应用程序中每个类的名字及类所在的名称空间。随着窗体中控件的增加，这些对象的有关信息将添加到命名空间框的下拉列表中。

（3）代码窗口

代码窗口与窗体设计器窗口在同一位置，但被放在不同的标签页中，如图1.26 所示，其中Form1 窗体的代码窗口的标题是 Form1.cs。代码窗口用于输入应用程序代码，又称为代码编辑器，包含项目列表框、对象列表框、成员列表框和代码编辑区。项目列表框显示此源文件所属的项目，对象列表框显示和该窗体有关的对象清单，成员列表框显示对象列表框中所选中对象的全部事件，代码编辑区用于编辑对应事件的程序代码。

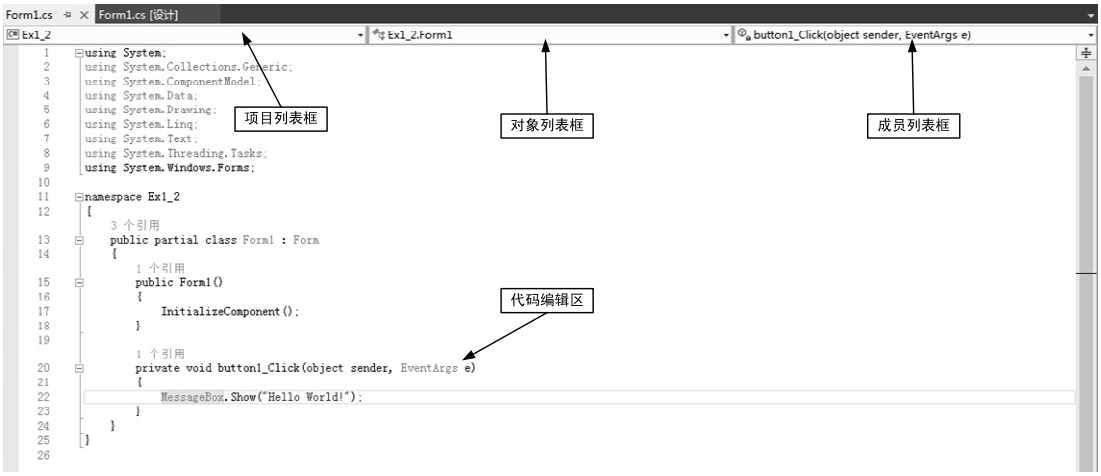

图 1.26　代码窗口

除了上述几种窗口外，在集成环境中还有其他一些窗口，包括输出、命令、任务列表等，将在本书后续章节中介绍。

1.3　最简单的 C#程序

1.3.1　C#项目的创建与分类

VS 2015 可用于多种类型的快速程序开发，如基于 Web 的应用程序、基于 WPF 应用程序、基于 Windows 的应用程序、控制台应用程序和移动应用程序等。

单击“文件”→“新建”→“项目”按钮，系统弹出“新建项目”对话框，如图 1.27 所示。

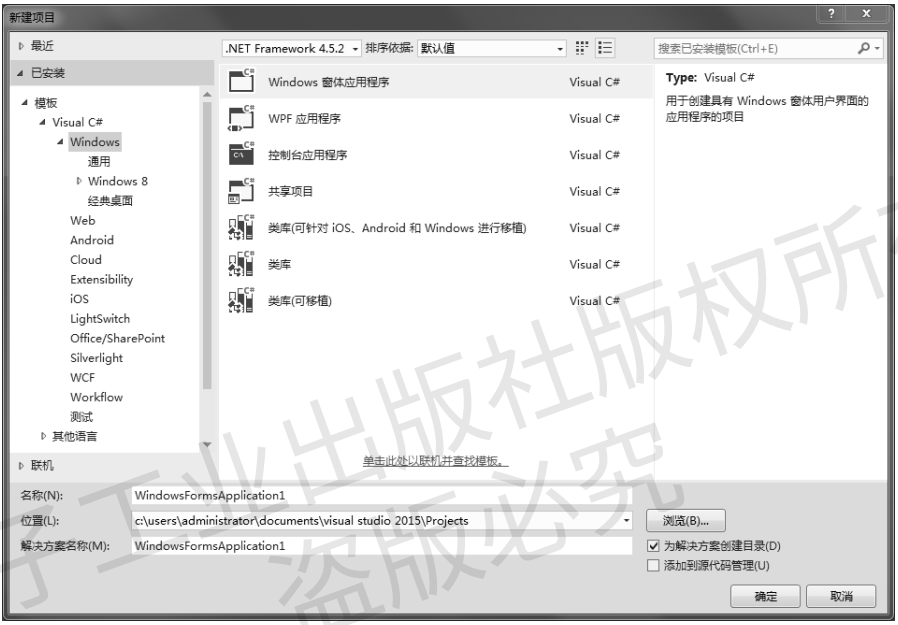

图 1.27　“新建项目”对话框

在“模板”栏中选择模板类型，选择模板后，在“名称”栏中输入项目的名称，在“位置”栏中输入（选择）保存项目的路径，在“解决方案名称”栏中输入解决方案的名称，单击“确定”按钮即可进入项目开发工作区。新建立的项目都保存在设定的解决方案中，一个解决方案中可以包含一个或多个项目。在默认情况下，解决方案的名称与项目名称相同，而且保存项目和解决方案的文件夹名就是项目名称。

最常见的 C#项目有两大类：**控制台应用程序**和 **Windows 窗体应用程序**，对于每一种类型 VS 2015 都提供了默认模板。

1.3.2　第一个控制台应用程序

【例 1.1】 在控制台窗口中输出“Hello World!”字样。

在.NET 开发环境中新建一个控制台应用程序项目后，在源代码文件中输入如下语句：

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
namespace Ex1_1
```

```
{
    class Program
    {
        static void Main(string[] args)
        {
            Console.WriteLine("Hello World!");
        }
    }
}
```

将此项目命名为 Ex1_1，然后打开“命令提示符”程序，进入目录“C:\Users\Administrator\Documents\Visual Studio 2015\Projects\Ex1_1\Ex1_1\bin\Debug”，输入 Ex1_1.exe 后回车，可以看到运行结果出现在控制台窗口中，显示“Hello World!”字样，如图 1.28 所示。

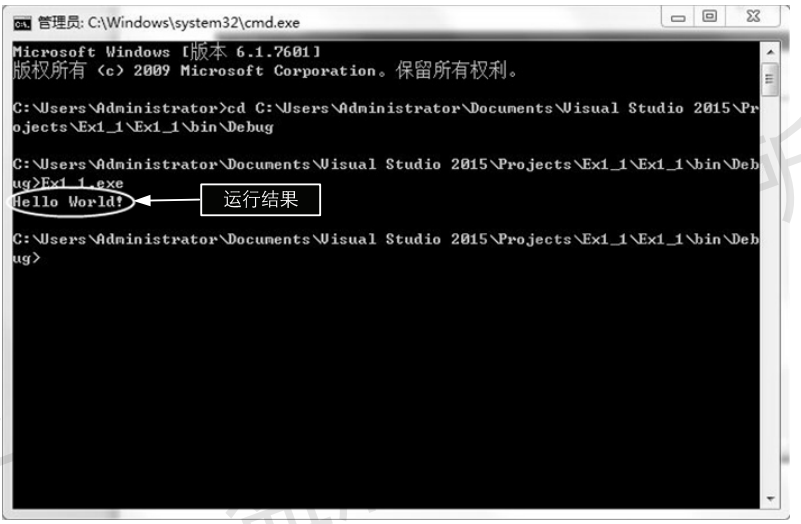

图 1.28　控制台中程序运行结果

（1）命名空间

在上面代码中，以 using 关键词开始的是命名空间导入语句。命名空间是为了防止相同名字的不同标识符发生冲突而设计的隔离机制。比如用户开发了一个二维的图形组件，将该组件命名为 Point，而另一个用户开发的一个三维图形组件恰好也命名为 Point，这时，如果在应用程序中同时使用这两个组件，那么在编译时编译器将无法判断引用哪一个组件。通过将组件的命名放在不同的命名空间中就可以加以区别，要使用哪一个组件，通过 using 关键字打开其所在的命名空间即可。在 C#中（确切地说是在.NET 框架类库中）使用了一种树状的类似于“中国→江苏→南京”这样的地址编码方式来对命名空间进行管理，通过引入命名空间，就可以用 MyClass.Point 和 YourClass.Point 这样的方式对相同名称的标识符进行识别，即使是同时使用这两个组件，编译器也不会迷惑。

在.NET 框架类库中提供的不同组件都被包含在一定的命名空间中，所以要使用这些组件必须通过 using 关键字打开相应的命名空间使得相应的标识符对编译器可见，如果没有使用 using 关键字，那么相应的标识符就应包含完整的命名空间路径。

（2）完全面向对象

C#是一种面向对象语言，所以不会有独立于类的代码出现，应用程序的入口也必须是类的方法。C#规定以命名为 Main 的方法作为程序的入口。方法的代码使用“{}”符号作为起始

标识符，static 关键字是对方法的修饰，使得这个方法在类的实例被建立之前就可被调用，因为在程序入口的时候还不会有任何类的实例生成。Main 前面的关键字 void 代表该方法没有返回值，这与 C/C++和 Java 是一样的。

方法中的代码“Console.WriteLine("Hello World!");”是调用了.NET 框架类库中对象的方法来向控制台输出信息。可以看出，本程序的核心代码所实现的功能全部来自.NET 框架类库，而 C#只是提供了一个语法框架，C#开发实际就是用 C#语言将.NET 框架类库中的组件加以组织，实现应用程序的业务逻辑。

1.3.3　第一个 Windows 窗体程序

【例 1.2】 显示含有“Hello World!”字样的对话框。

在“新建项目”对话框中，选择“Windows 窗体应用程序”模板，将此项目命名为 Ex1_2，单击“确定”按钮后，将进入 C#的 Windows 编程窗体设计工作区，如图 1.29 所示。

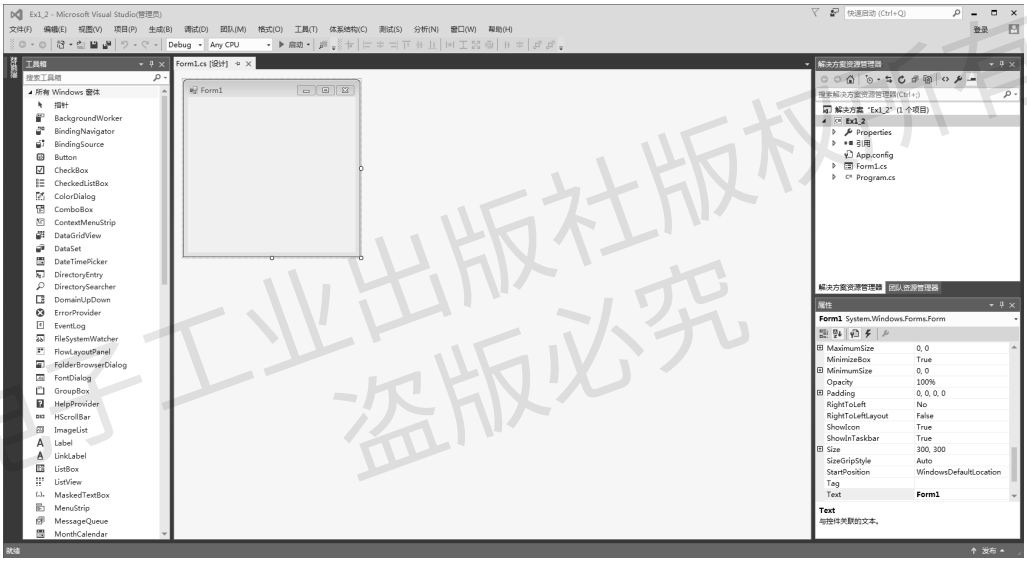

图 1.29　C# Windows 窗体设计环境

中间工作区的左上方是窗体设计器，设计器窗口的标题是 Form1.cs [设计]。

在建立一个新的项目后，系统将自动建立一个窗体，其默认名称和标题为 Form1。

在设计应用程序时，根据用户需要，从工具箱中选择所需要的控件，然后在窗体的工作区中画出相应的控件对象，这样就完成了窗体的界面设计。

将窗体 Form1 调整为合适的大小，从工具箱中选择 Button 按钮控件并将其拖曳到 Form1 窗体中，双击此按钮，在代码窗口中添加代码，代码如下：

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
```

```
using System.Windows.Forms;
namespace Ex1_2
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }
        private void button1_Click(object sender, EventArgs e)
        {
            MessageBox.Show("Hello World!");
        }
    }
}
```

按 F5 快捷键运行此程序，结果如图 1.30 所示。

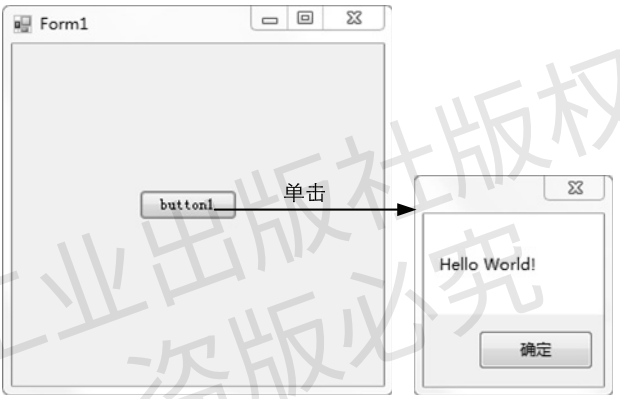

图 1.30　Windows 窗体程序运行结果