第 1 章　Visual C#简介

本章要点

- .NET Framework 4.5 介绍
- Visual C#程序设计语言的优点
- Visual Studio 2012 开发平台的展示

本章主要是对 C#语言基础知识的介绍，其中包括关于 C#特点的介绍以及关于 Visual Studio 开发环境的介绍。最后将给出一个简单的示例，初步学习 Windows 窗体应用程序的编写方法。

1.1 .NET Framework 4.5 介绍

目前，.NET Framework 的主流版本是 4.5，在本小节中，我们主要向读者介绍.NET Framework4.5，在下一小节中，将介绍 C# 4.5 新增的功能。

.NET Framework 是微软为开发应用程序创建的一个富有革命性的新平台。.NET Framework 发布的第一个版本是运行在 Windows 操作系统上的，以后随着技术的成熟和更新，其他操作系统，如 Linux、FreeBSD，甚至个人数字助手(PDA)类设备，都将有运行在其上的.NET Framework 版本。

.NET Framework 是.NET 的核心部分。.NET 应用程序运行时所需的所有核心服务都是由.NET Framework 提供的。.NET Framework 的核心是公共语言运行时 CLR，另外还包括了.NET 框架类库。

在.NET Framework 中，CLR 是一个公共语言运行环境。它为.NET 应用程序运行提供了各种必要的服务。所有符合公共语言规范的语言——包括 Microsoft Visual Basic、Microsoft Visual C++和其他微软编程语言以及针对.NET 平台推出的第三方语言，都可以使用这些服务。公共语言运行时解决了语言的集成问题。在逐渐以网络计算为重点的今天，公共语言运行时显得尤为重要。

.NET Framework 从底层的内存管理和构件装载一直到前端的用户界面，在各个层次上为用户提供了所有可能的支持。图 1.1 中给出了.NET Framework 的主要组成部分。

.NET Framework 的最底层是公共语言运行时 CLR。它是.NET Framework 的核心，也是其关键的功能引擎。CLR 为所有语言和环境提供了一个通用基础，使得跨语言集成成为可能。CLR 还负责内存的分配和管理、代码的即时编译、代码的装载、对象的引用计数，以及垃圾回收等操作。

CLR 之上是.NET Framework 的基本类库(Base Class

Library，BCL)。BCL实现了运行时的各种功能并通过各种命名空间为开发者提供了所需的各种高级服务。例如，Collections命名空间包括了链表、哈希表等集合类型；System.IO命名空间包含了输入/输出服务；BCL是.NET语言共享的标准类库，任何遵从.NET的编程语言都可以使用它，这些服务都在.NET框架的控制之下，为所有的语言提供了统一的类库支持。

图1.1 .NET Framework的组成

ADO.NET中主要使用DataSet(数据集)在内存中处理数据。在DataSet中，数据表格可以单独存放，也可以通过DataRelation对象在表格之间建立关联，包括一对多、多对多、甚至是同一表格的自我关联(同一表格的外键关联到同一表格的主键)，大大增加了数据处理的灵活性，再加上DataSet中的所有数据都是离线的，也就是说数据是直接存储在内存中的，因此可以降低后台DBMS(数据库管理系统)的负担，特别适合用在Web数据库应用程序的开发上。

除了使用数据库的存取方式进行数据处理外，.NET上还支持XML文档的操作，通过XmlDataDocument就可以进行XML文档的存取。添加对命名空间System.Xml的引用，就可以完成对XML文档的存取，在以后的章节中我们会具体举例如何进行操作。

.NET依旧支持对Windows应用程序的开发，以前我们在Visual Basic中经常使用的ActiveX控件，现在则由基类库中的System.Windows.Forms命名空间下的类取代(当然ActiveX控件还可以继续沿用)。新一代的Web应用程序开发则使用了ASP.NET技术，相比于原来的ASP，它延续了容易使用的优点，而且对其进行了改进。ASP.NET在2001年由微软公司推出，在结构上与前面的版本相比大不相同，几乎完全是基于组件和模块化的。Web应用程序的开发人员使用这个开发环境可以实现更加模块化的、功能更加强大的应用程序。

要使各种不同的程序语言在同一个软件平台上运行，这看起来有点不可思议。.NET Framework就实现了这个看似无解的难题。通用语言规范(Common Language Specification，CLS)，包括函数(类的方法)调用方式、参数传递方式、数据类型、异常处理方式等，任何编程语言只要符合这个规范，就可以彼此相容共存、相安无事，就好像大家不论是如何赚钱，只要符合当地的法律法规，就可以得到法律的保护。因此只要遵循CLS和只使用CLS兼容的编程语言开发组件，所获得的组件就被称为CLS兼容的组件，就可以保证其他支持CLS的程序语言都能够使用这个组件。

.NET支持多种程序设计语言，常见的有Visual C#、Visual C++、Visual Basic、Java等。.NET架构至少默认支持Visual Basic和Visual C#两种编程语言。

本书主要是面向Visual C#编程语言使用者编写的，所以在本书的内容中，只对Visual C#内容做详细的讲解。Visual C#是专门针对在.NET架构上开发应用程序而设计的新型程序设计语言，就程序语法来说，有点类似于C++，或者说更像Java。因此有着易用、灵活性大的特点，拥有完整的面向对象支持，

是.NET 平台上最常用的语言之一。

1.2 Visual C#介绍

1.2.1 Visual C#的由来

最近 20 年，C 和 C++一直被商用软件开发者普遍使用。C#的出现，为开发者提供了一个快速建立应用程序的开发平台。微软对 C#的定义是“一种类型安全、现代、简单，由 C 和 C++衍生出来的面向对象的编程语言，它是牢牢植根于 C 和 C++语言之上的，并可立即被 C 和 C++的使用者所熟悉。Visual C#的目的就是综合 Visual Basic 的高生产率和 C++的行动力”。

Visual C#是一种强大的语言，在 C++中能完成的任务利用 Visual C#也能完成。但是需要说明的是，在 Visual 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。这种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，本书将不讨论这方面的特殊内容。

今天，人们改进、开发了许多语言以提高软件生产的效率，但是这些或多或少都以牺牲 C 和 C++程序员所需要的灵活性为代价。这样的解决方案在程序员身上套上了太多的枷锁，限制了他们能力的发挥。它们不是很好地与原有的系统兼容，更为头痛的是，它们并不总是与当前的 Web 应用结合得很好。

理想的解决方案是将快速的应用开发与对底层平台所有功能的访问紧密结合在一起。程序员们需要一种环境：它与 Web 标准完全同步，并且具备与现存程序方便地进行集成的能力。除此之外，程序员们希望这种开发环境允许自己在需要时使用底层代码。

针对该问题，微软的解决方案就是推出了 Visual C#，它是一种现代的面向对象的程序开发语言，它使得程序员能够在新的微软.NET 平台上快速开发种类丰富的应用程序。.NET 平台提供了大量的工具和服务，能够最大限度地发掘和使用计算及通信能力。

由于其一流的面向对象的设计，从构建组件形式的高层商业对象到构造系统级应用程序，你都会发现，Visual C#将是最合适的选择。使用 Visual C#语言设计的组件能够用于 Web 服务。这样通过 Internet，就可以被运行于任何操作系统上的任何编程语言所调用。

任何面向对象语言的核心在于支持对类的定义和处理。类定义了新的类型，可以扩展语言以创造更适合于解决具体问题的模型。Visual C#中有声明新的类及其方法和性质的关键字，还含有实现面向对象编程的三大支柱：封装、继承和多态的关键字。

在 Visual C#中，与类的定义有关的一切都可在声明本身中找到，C#的类定义并不需要独立的头文件或 IDL(接口定义语言)文件。而且，Visual C#支

持新的XML风格的内嵌文档，大大简化了软件的在线和印刷品参考文档的制作工作。

Visual C#还支持接口(Interface)，一种与其所指定的服务的类订立合同(Contract)的方式。在Visual C#中，类只能从一个父类继承，但可以实现多个接口。在实现接口时，C#类实际上也承诺了要提供接口所规定的功能。

Visual C#还提供了对结构体(Struct)的支持，但此概念的含义与C++有显著的不同。在C#中，结构体是有严格限制的轻量级类型，实例化时比传统的类对操作系统和内存的需求都小得多。结构体不能从类继承，也不能被类继承，但它可以实现接口。

Visual C#提供了面向组件的特性，如属性(Property)、方法、事件和称为特性信息(Attribute)的声明性结构。面向组件编程是通过CLR将元数据(Metadata)与类的代码一起保存而实现的。元数据负责描述类，包括其方法和性质，以及安全要求和其他属性信息，如是否可以序列化(Serialize)；代码则包含执行功能所必需的逻辑流程。因此已编译的类是自成一体的独立单位。这样宿主环境只要能够识别类的元数据和代码，无需其他信息(如类的注册信息)，就可以使用它。使用Visual C#和CLR，可以通过自定义特性信息来给类添加自定义元数据。同样也可以用支持反射的CLR类型阅读类的元数据。

程序集(Assembly)是文件的集合，对编程人员而言就是DLL或者EXE文件。在.NET 中，程序集是重用、版本协调、安全性和部署的基本单位。CLR提供了大量处理程序集的类。

使用Visual C#开发应用程序比使用C++更简单，因为C#的语法比较简、Visual Statio 2012平台可视化更加直观、编辑方法更加灵活和方便。如前所述，Visual C#还支持使用C++式的指针和关键字直接访问内存，不过这种操作都被归入不安全的范畴，并且会警告CLR无法使用内存回收器，在指针所引用的对象被释放前不进行回收。

1.2.2 C# 4.5新增的功能

C# 4.5新增的功能主要体现在如下方面。

（1）异步读取和写入HTTP请求和响应：ASP.NET 4.5 可以读取，编写，并刷新流异步。此asynchronicity可以增量数据发送到客户端，而不必占用操作系统线程。

（2）当请求验证启用时，对读取unvalidated请求数据支持：ASP.NET 4.5 提供用于读取unvalidated请求数据，以便您可以允许用户通过选定的字段或页的标记。

（3）为WebSockets协议支持：在新 System.Web.WebSockets 命名空间中的方法提供WebSockets协议支持，可让您读取和写入字符串和二进制数据

（4）客户端脚本的绑定和缩减：ASP.NET 4.5 使用合并更快加载不同的 JavaScript 文件的绑定（通过移除不必要的字符减少 JavaScript 和 CSS 文件的大小）的缩减 。

（5）对于异步模块和处理程序支持：新 async 和 await 关键字可以方便地编写异步 HTTP 模块和异步 HTTP 处理程序。以异步开发的更新包括:

ClientDisconnectedToken:异步通知应用程序的 CancellationToken，当客户端从基础 web 服务器断开连接的。

TimedOutToken: CancellationToken，在异步通知应用程序的请求和配置的请求超时值长期运行。

ThreadAbortOnTimeout:如果希望应用程序控制计时请求行为，请将此特性设置为 false。当属性设置为 true（默认值），ASP.NET 中止为请求的线程，当该请求超时时。

Abort:使用此方法在应用程序中强制停止请求的基础 TCP 连接。所有处理 I/O 将失败。

（6）集成反 XSS 编码例程：反 XSS（脚本的跨站点)核心编码例程集成 ASP.NET 4.5。这些实例仅以前提供的作为外部库。

（7）为 OAuth 和 OpenID 支持：OAuth 和 OpenID 可以创建允许用户登录与其他站点的凭据，包括 google、雅虎、Facebook、慌张和 Windows Live 的站点。

1.3 Visual C#语言的特点

Visual C#语言的特点可以归结为以下几种：

（1）简洁的语法。

（2）精心地面向对象设计。

（3）与 Web 的紧密结合。

（4）完整的安全性与错误处理。

（5）灵活性的版本处理技术。

（6）灵活性与兼容性。

1.3.1 简洁的语法

在默认的情况下，Visual C#的代码在.NET 框架提供的“可操控”环境下运行，不允许直接内存操作。这与 C++不同，C++中会出现大量的“->”“::”操作符，这些在 Visual C#中已经不再出现，Visual C#只支持“.”，对于我们来说，现在需要理解的一切仅仅是名字嵌套而已。

语法的冗余是 C++常见的问题，比如“Const”和“#Define”、各种各样的字符类型等。Visual C#对此做了简化，只保留了常见的形式，而别的冗余形式已经从它的语法结构中清除出去。

1.3.2 精心地面向对象设计

面向对象的话题从Smalltalk开始就缠绕着任何一种现代程序设计语言，众多开发人员也越来越沉浸于面向对象思想编程给编程人员带来的便利以及给人们带来的快乐。Visual C#语言具有面向对象的语言所应有的一切特性：封装、继承、多态，这并不出奇。然而，通过精心地面向对象设计，从高级商业对象到系统级应用，已使Visual C#成为建造各种组件的最佳选择。

Visual C#的类型系统可分为值类型和引用类型，引用类型是对象，值类型可通过一个叫做装箱与拆箱的机制来完成与引用类型的转换操作，这在以后的章节中将进行更为详细的介绍。

Visual C#中只允许单继承，即每个类只允许有一个父类(亦称基类)，从而避免了类型定义的混乱。同时，Visual C#不存在全局函数、全局变量，也不存在全局常数。所有的东西，都必须封装在类之中，这样做的好处是，代码将有更好的可读性，并且命名冲突的问题也迎刃而解。

整个Visual C#的类模型是建立在.NET虚拟对象系统(Visual Object System，VOS)的基础之上，其对象模型是.NET基础架构的一部分，而不再是其本身的组成部分。

1.3.3 与Web的紧密结合

Web编程是当今编程的一大趋势与潮流，在.NET中，新的程序开发模型越来越多的解决方案需要与Web标准相结合、相统一，例如超文本标记语言(Hypertext Markup Language，HTML)和XML。由于历史的原因，现存的一些开发工具不能与Web紧密地结合。SOAP的使用使得Visual C# .NET克服了这一缺陷，大规模深层次的分布式开发从此成为可能。

由于有了Web服务框架的帮助，对程序员来说，网络服务看起来就像是C#的本地对象。程序员们能够方便地为Web服务，并允许它们通过Internet被运行在操作系统上的任何语言所调用。举个例子，XML已经成为网络中数据结构传送的标准，为了提高效率，Visual C#允许直接将XML数据映射为结构。这样就可以有效地处理各种数据。

1.3.4 完整的安全性与错误处理

语言的安全性与错误处理能力，是衡量一种语言是否优秀的重要依据。任何人都会犯错误，即使是最熟练的程序员也不例外：忘记变量的初始化，对不属于自己管理范围的内存空间进行修改，……。这些错误常常产生难以预见的后果。一旦这样的软件被投入使用，寻找与改正这些简单错误的代价也是让人难以忍受的。Visual C#的先进设计思想可以消除软件开发中的许多常见错误，并提供了包括类型安全在内的完整的安全性能。为了减少开发中的错误，Visual C#会帮助开发者通过更少的代码完成相同的功能，这不但减轻了编程人员的工作量，同时更有效地避免了错误的发生。

.NET 运行库提供了代码访问安全特性，它允许管理员和用户根据代码的 ID 来配置安全等级。当应用程序执行时，运行库将自动对它进行计算，然后给它一个权限集。根据应用程序获得的权限不同，应用程序或者正常运行，或者发生安全性异常，计算机上的本地安全设置最终决定代码所收到的权限。内存管理中的垃圾收集机制减轻了开发人员对内存管理的负担。.NET 平台提供的垃圾收集器(Garbage Collection，GC)将负责资源的释放与对象撤销时的内存清理工作。

变量是类型安全的。Visual C#中不能使用未初始化的变量，对象的成员变量由编译器负责将其置为 0，当局部变量未经初始化而被使用时，编译器将做出提醒；Visual C#不支持不安全的指向，不能将整数指向引用类型，例如对象，当进行下行指向时，Visual C#将自动验证指向的有效性；Visual C#中提供了边界检查与溢出检查功能。

1.3.5 灵活的版本处理技术

Visual C#中提供内置的版本支持来减少开发费用，使用 Visual C#将会使开发人员更加轻易地开发和维护各种商业应用。

升级软件系统中的组件(模块)是一件容易产生错误的工作。在代码修改过程中可能对现存的软件产生影响，很有可能导致程序的崩溃。为了帮助开发人员处理这些问题，Visual C#在语言中内置了版本控制功能。例如：函数重载必须被显式地声明，而不会像在 C++或者 Java 中经常发生的那样不经意地被进行，这可以防止代码级错误和保留版本化的特性。另一个相关的特性是对接口和接口继承的支持。这些特性可以保证复杂的软件能够被方便地开发和升级。

1.3.6 灵活性和兼容性

在简化语法的同时，Visual C#并没有失去灵活性。尽管它不是一种无限制的语言，比如：它不能用来开发硬件驱动程序，在默认的状态下没有指针等，但是，在学习过程中你将发现，它仍然是那样的灵巧。

如果需要，Visual C#允许你将某些类或者类的某些方法声明为非安全的，这样一来，你将能够使用指针，并且调用这些非安全的代码不会带来任何其他的问题。此外，它还提供了代理(Delegate)来模拟指针的功能。再比如说，Visual C#不能支持类对多个类的继承，但是可以通过对多个接口的继承来实现这一功能。

正是由于其灵活性，C#允许与 C 风格的需要传递指针型参数的 API 进行交互操作，DLL 的任何入口点都可以在程序中进行访问。Visual C#遵守.NET 公用语言规范(Common Language Specification，CLS)，从而保证了 Visual C#与其他语言组件间的互操作性。元数据(Metadata)概念的引入既保证了兼容性，又实现了类型安全。

1.4 VS 2012 开发环境介绍

1.4.1 VS 2012 的界面

启动 Visual Studio 2012，进入其集成开发环境，用户首先可以看见如图 1.2 所示的界面窗口，这个界面窗口中包含进行 C#程序开发的基本工具。

从界面窗口中，可以看到菜单栏、工具栏按钮、标题栏、类视图、解决方案资源管理器、属性视图窗口、代码编辑窗口。

可以单击“创建”工具按钮，进行新项目的创建；同样，我们也可以单击“打开”工具按钮，根据目录寻找我们已经写好的程序。Visual C#的初始界面上也会显示最近 6 个我们曾经打开的程序，这很方便用户的操作。

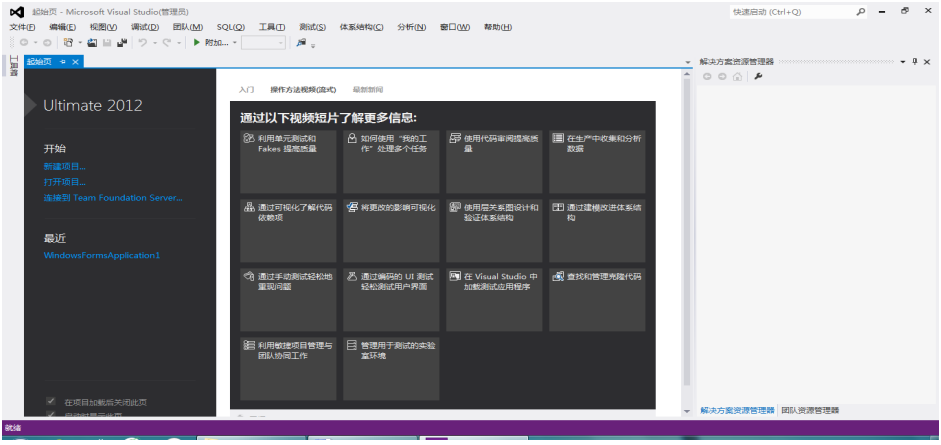


图 1.2 Visual Studio 2012 界面(Visual C#的界面)

1.4.2 菜单栏

菜单栏中列出了 C#开发环境的各个下拉菜单选项，单击某个菜单选项，便会显示出下拉菜单中的各个命令项。单击某个命令，就可以完成相应的操作。

(1) C#菜单中的命令可以分为 4 类：普通命令、带有快捷键的命令、带有子菜单的命令和可弹出对话框的命令。下面分别介绍这几种命令的使用方法。

- 普通菜单命令：普通菜单命令的选用只需直接在下拉菜单上单击命令或键入命令中带有下划线的字母即可。如“生成”菜单中的“重新生成解决方案”菜单项，“窗口”菜单中的“隐藏”菜单项都属于普通菜单命令。
- 带有快捷键的命令：快捷键亦称热键，要选用该命令，除了用第一种方法之外，还可以直接按下快捷键。例如针对“编辑”菜单中的“复制”命令，直接按 Ctrl+C 组合键即可，方法为先按住 Ctrl，然后再按住 C 键。用户应该多记住些快捷键，以节省操作时间。

- 可弹出对话框的命令：这类命令后面都带有 3 个小圆点的省略符号，如“项目”菜单中的“添加现有项...”命令，以及通过“文件”→“新建”菜单可以展开的“项目...”、“文件...”、“网站...”等命令，单击这种命令就会弹出一个对话框。图 1.3 是选择“项目”→“添加现有项...”菜单命令后弹出的对话框。
- 带有子菜单的命令：C#的菜单命令中带有子菜单的命令很多，这些命令的最后都有一个向右的黑三角，表明有子菜单。如“文件”菜单中的“新建”、“打开”、“添加”等。

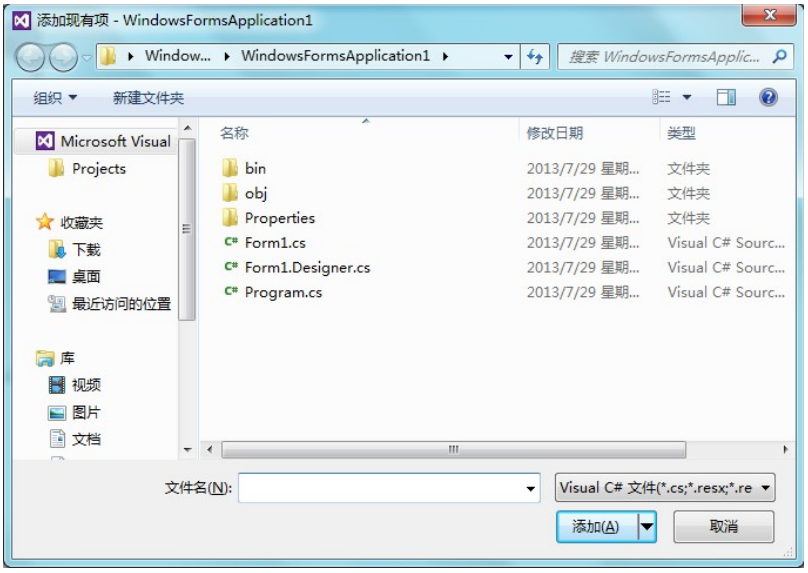


图 1.3　选择“项目”→“添加现有项...”菜单命令后弹出的对话框

(2) C#主菜单为用户提供了开发、调试以及管理应用程序的各个功能和各种工具。

- 文件：该菜单包含用于打开、保存、新建应用程序文件的各种命令。
- 编辑：该菜单包含编辑应用程序代码和窗体上各个组件的各种命令，例如删除、粘贴以及撤销、重做等。
- 视图：该菜单包含用于编辑视图、控制各个视图窗口是否显示等的命令。
- 项目：该菜单用于管理项目的文件、编译项目和进行项目设置。
- 生成：该菜单用于生成解决方案。
- 调试：该菜单用于调试和运行程序。
- 团队：该菜单用于连接到 Team Foundation Server。
- SQL：该菜单用于新建查询命令等。
- 格式：该菜单用于控件的格式设置。
- 工具：该菜单用于设置 C#环境并提供一些 C#外挂的工具。
- 测试：该菜单用于调试运行。
- 体系结构：该菜单用于新建生成关系图。

- 分析：该菜单用于启动、比较性能分析。
- 窗口：该菜单用于进行窗口的管理。
- 帮助：该菜单用于寻求 C#帮助、查询关键字，包含.NET 的说明文件。

1.4.3 标题栏

在所有窗口的最顶端，有一个蓝色的水平长条，这个蓝色条就是我们所说的标题栏。在标题栏上分别包含了 C#开发的项目名和当前打开的文件名。在标题栏的最右侧有最小化、最大化和关闭这 3 个按钮。

需要注意的是，这里的关闭按钮用来控制整个 C#环境的关闭，单击该按钮就表示退出整个 C#开发环境。在标题栏的最左侧有一个小图标，是窗口的控制菜单，单击该图标或按“Alt+空格键”就会弹出控制菜单，可以从中选择相应的命令来完成一个操作。

1.4.4 工具栏按钮

工具栏按钮位于菜单栏下方，由一些常用菜单命令的加速按钮组成，如图 1.4 所示。

工具栏上的每个按钮都有提示，将鼠标移到某个按钮上停留一秒左右，就会弹出一条提示，告诉用户与该按钮相对应的菜单命令。

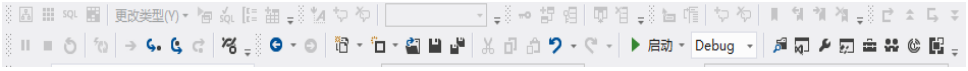


图 1.4 常用工具栏

用户可以定制自己的工具栏，直接将工具栏按钮删除或添加。具体方法是：用鼠标在工具栏或者工具栏的空白处右击，弹出一个工具栏编辑器的快捷菜单，里面有与各工具栏按钮对应的选项，通过在对应的选项上勾选实现想要的工具项。假如要把“类设计器”按钮加入到工具栏中去，则先在工具栏空白处右击，把鼠标移到【类设计器】命令上，单击，可以看到【类设计器】命令的左边已经有一个勾了，这时“类设计器”按钮也会出现在工具栏中，如图 1.5 所示。

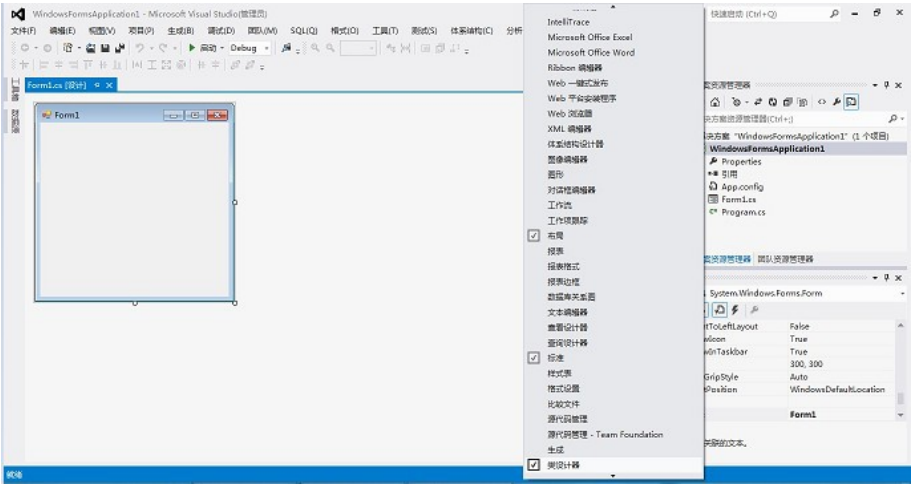


图 1.5　定制自己的工具栏

1.4.5 代码和文本编辑器

“代码和文本编辑器”在屏幕的左中部，就是我们所看到的窗口中占用面积最大的那一块。它是一种用来输入、显示及编辑代码或文本的字处理实用工具。根据该编辑器的内容，将其称为“文本编辑器”或“代码编辑器”，“文本编辑器”对应的是界面的设计状态。如果它仅包含没有关联语言的文本，则称为“文本编辑器”。如果它包含与某种语言相关联的源代码，则称为“代码编辑器”。由于该编辑器通常用于编辑代码，因此可将其称作“代码编辑器”。

可以打开多个“代码编辑器”，在不同的窗体或模块中查看代码，并在它们之间复制和粘贴代码。“窗口”菜单下列出了所有打开的“代码编辑器”。由于代码编辑器可以同时打开多个文件，所以在代码编辑器的窗口上有多个选项卡，用来控制文件的切换。每个选项卡的标签标明一个被打开的文件的文件名，如图 1.6 所示。

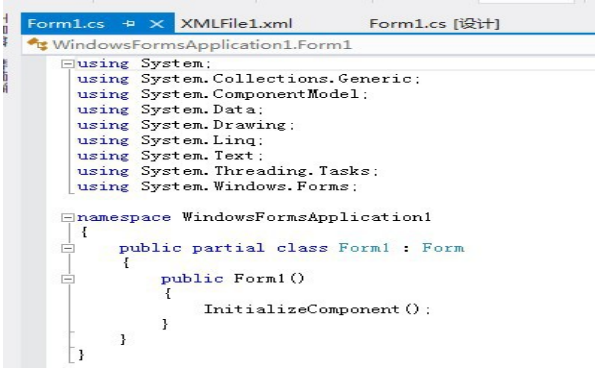


图 1.6　代码编辑器中的部分代码显示

“代码编辑器”就是我们实现各种功能的容器，我们把代码以特殊的语言法则放进“代码编辑器”，进行定制，最终生成我们所期望的结果。

可以用下列方式进入“代码编辑器”：

- 右击窗体或者是窗体上的某个控件，然后从快捷菜单中选择“查看代码”命令。
- 在解决方案资源管理器窗口中，直接右击窗体图标，从快捷菜单选中“查看代码”命令，或者是选中窗体图标，单击上面的“查看代码”快捷方式图标 <> 、或是在设计状态双击要编辑的对象即可。

1.4.6 类视图窗口和解决方案资源管理器

类视图窗口和解决方案资源管理器通常占用屏幕上的同一块空间，它们之间的切换是靠选项卡来选择的。用户也可以通过拖动窗口，把两个窗口拖到不同的位置。方法是拖动类视图窗口或者解决方案资源管理器窗口上方的蓝色长条，即可把窗口拖动到任意位置。

1. 类视图窗口

类视图能使你检查并定位到解决方案中的符号。按项目组织的符号显示在分层树视图中，以指示它们之间的包容关系。

类视图窗口默认的位置是在代码编辑窗口的右边，其作用是显示程序的数据结构。在类视图窗口中，可以清楚地看到类、函数、变量的定义和关系。在初始状态，类视图窗口中仅显示项目名称，如果单击项目名称左边的“+”符号，则会显示出更细一层的内容，即命名空间：继续单击命名空间左边的“+”符号，又会再次细化显示到 Class 级别。接着一层层展开，就可以将类的成员函数、成员变量尽收眼底，如图 1.7 所示。

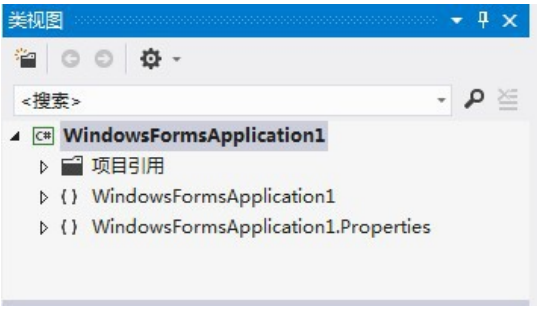


图 1.7 类视图窗口

“类视图”显示若干图标，每个图标代表不同类型的符号，如命名空间、类、函数或变量。表 1.1 给出了常见的名称以及对应的符号。

表 1.1 常见的解决方案资源管理器中的图标及其说明

图 标	说 明	图 标	说 明
{ }	命名空间		方法或函数
	接口		属性
	结构		字段或变量
	类		枚举项

图　标	说　明	图　标	说　明
	枚举		常数

2. 解决方案资源管理器窗口

解决方案资源管理器提供项目及其文件的有组织的视图，并且提供对项目和文件相关命令的便捷访问。与此窗口关联的工具栏提供适用于列表中突出显示的项的常用命令。若要访问解决方案资源管理器，可在“视图”菜单中选择“解决方案资源管理器”命令，结果如图 1.8 所示。

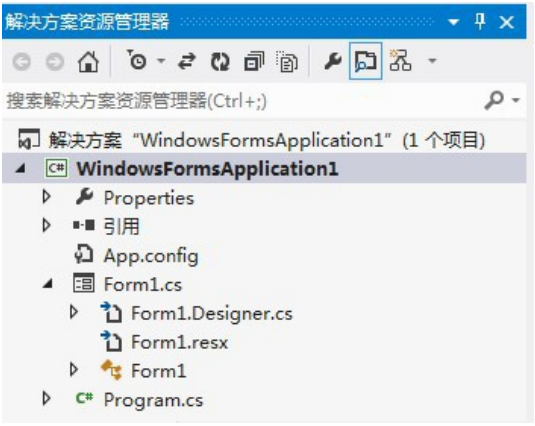


图 1.8　解决方案资源管理器窗口

解决方案资源管理器使你可以在解决方案或项目中查看项目并执行项目管理任务。它还允许使用 Visual Studio 编辑器在解决方案或项目的上下文之外处理文件。

单个解决方案及其项目以分层显示的方式出现，这种显示方式提供关于解决方案、项目和项的状态的更新信息。这可使用户同时处理若干个项目。

解决方案资源管理器非常灵活，它使用户得以独立于项目之外工作；可以在没有项目的情况下编辑和创建文件。解决方案资源管理器在“杂项文件”文件夹中显示这些文件。还可以处理仅与解决方案关联的文件。这些项显示在“解决方案项”文件夹中。

1.4.7 属性窗口

属性定义窗体、文档或控件的状态、行为和外观。多数图形控件包含可更改以定义其可视外观的属性。控件、文档和窗体还可公开一些指定它们将如何与用户进行交互，以及在运行时操作过程中需要的信息的属性。使用属性窗口来查看和设置窗体、文档或控件的设计时的属性。也可以使用属性窗口来编辑和查看文件、项目和解决方案属性。其他属性可能仅在运行时可用，可通过代码访问。属性窗口位于类视图和解决方案资源管理器下面，如果属性窗口不可见，可从“视图”菜单中选择“属性窗口”命令或按 F4 键调出。

【属性】窗口的外形如图 1.9 所示。

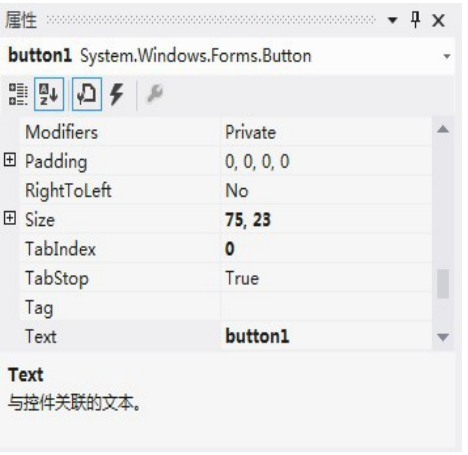


图1.9 【属性】窗口

如图1.9所示，【属性】窗口上方有5个按钮，从左到右依次介绍如下。

- 按分类顺序：按类别列出选定对象的所有属性及属性值。可以折叠类别以减少可见属性数。展开或折叠类别时，可以在类别名左边看到加号(+)或减号(-)。类别按字母顺序列出。
- 字母顺序：按字母顺序对选定对象的所有设计时属性和事件排序。若要编辑可用的属性，请在它右边的单元格中单击并输入更改内容。
- 属性：显示对象的属性。很多对象也有可以使用“属性”窗口查看的事件。
- 事件：显示对象的事件。此“属性”窗口工具栏控件仅当窗体或控件设计器在一个Visual C#项目的上下文中处于活动状态时才可用。
- 属性页：显示选定项的“属性页”对话框。“属性页”显示“属性”窗口中的可用属性的子集、同集或超集。使用该按钮可以查看和编辑与项目的活动配置相关的属性。

下面将介绍一些属性的操作。

1．使用属性窗口来设置属性

(1) 如果要修改的项未选定，请使用属性窗口最上方的“对象”下拉列表来选择。

(2) 在属性窗口中选择要修改的属性。例如，选择前景色以更改控件上文本的颜色。

(3) 指定该属性的值。

2．为多个控件设置属性值

(1) 在控件组中选择要修改的第一个控件。

(2) 选择其他要修改控件的同时按住Ctrl键。

(3) 在属性窗口中设置属性值。

这样就为选定的每个控件设置了该属性值。

1.5 案 例 实 训

1. 案例说明

本例是 C#入门的一个简单小例子，是一个 Windows 应用程序。在窗体中添加一个 Button 按钮以及一个 Label 标签，当鼠标单击按钮时，label 中显示“欢迎进入 C#世界！”。

2. 编程思路

编写 Button 按钮事件，通过对 Label 控件 Text 属性的改变，实现程序设计要求。

3.界面设计

（1）启动 Microsoft Visual Studio 2012，进入 Visual C# 2012 开发界面。

（2）选择“文件”→“新建”→“项目”，弹出如图 1.10 所示的对话框，可以看到，左边是项目类型，右边是已安装的模板，包括“Windows 窗体应用程序”、“类库”、“控制台应用程序”等模板，它们指定了要创建的应用程序的类型。

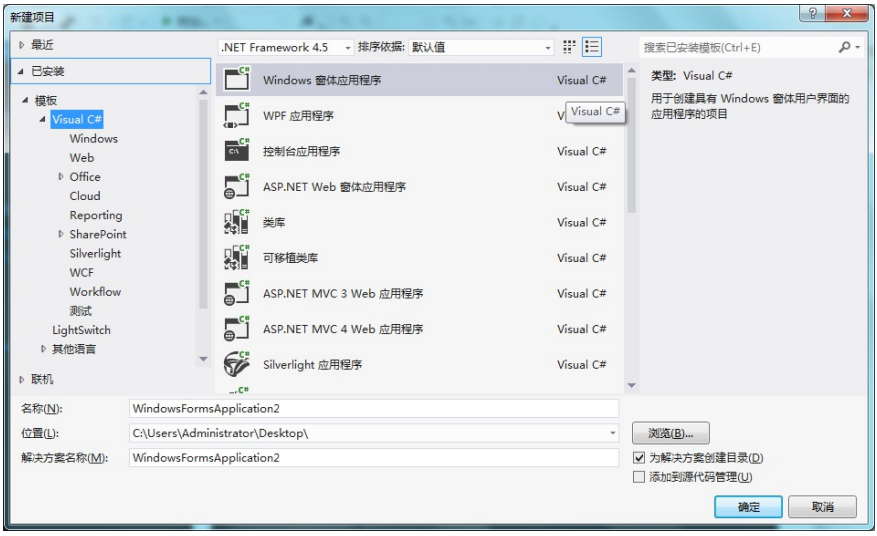


图 1.10　“新建项目”对话框

（3）在左边选择“Visual C#”，右边选择“Windows 窗体应用程序”，名称框中输入“Welcome”，并选择项目的存放位置，如图 1.11 所示。

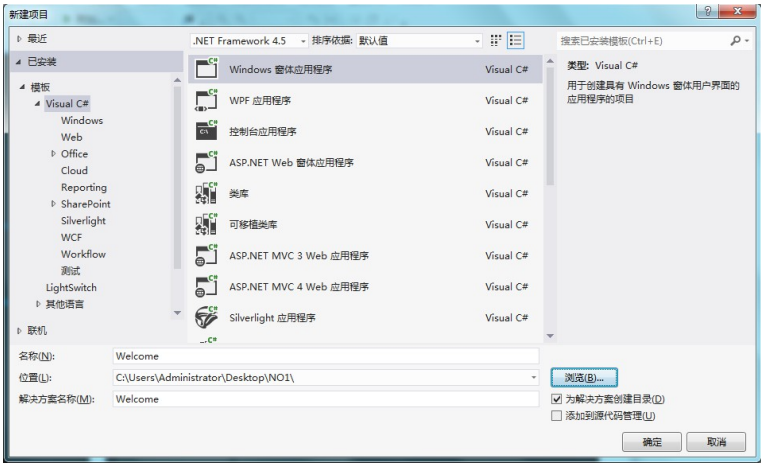


图 1.11　输入名称并选择存放位置

（4）确认“为解决方案创建目录”复选框已被选中，然后单击“确定”按钮。出现如图 1.12 所示的窗体。

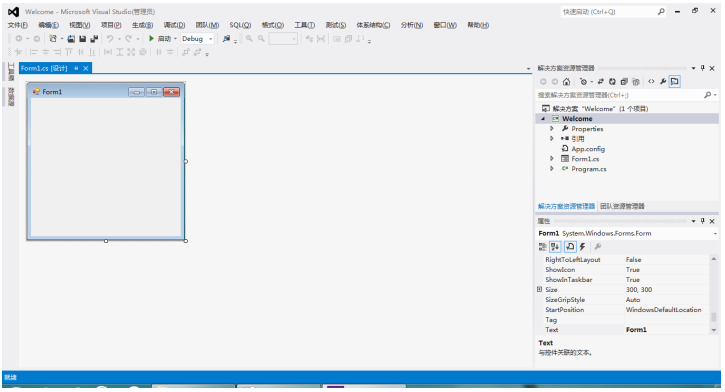


图 1.12　Welcome 项目

（5）调整窗体到合适的大小，长宽比为 2:1，然后展开工具箱中的“所有 Windows 窗体”选项卡，找到并双击控件 A Label，为窗体添加一个标签控件，这时的窗体 Form1 如图 1.13 所示。

注意：添加控件的方法一般有两种，一种是双击工具箱的控件，然后在窗体中拖动控件到合适的位置；另一种是通过鼠标的拖动，直接把控件拖动到窗体上去。

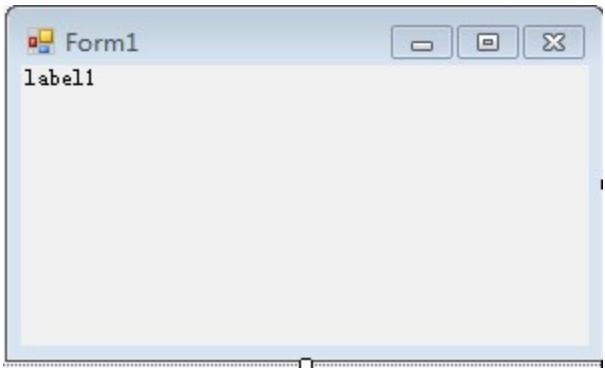


图 1.13　调整大小并添加标签后的窗体

（6）使用鼠标拖动文本框至窗体中上部，调整后的窗体 Form1 如图 1.14 所示。

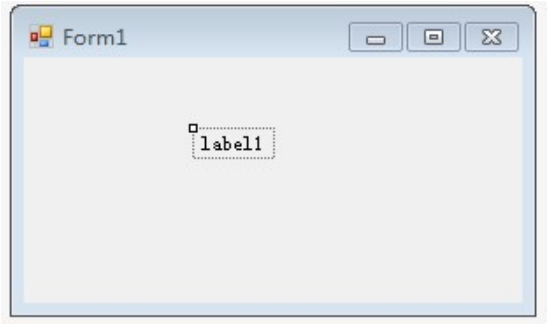


图 1.14　调整标签位置后的窗体

（7）按照同样的方法，在工具箱中找到 Button 控件，为窗体添加两个 Button（命令按钮），并调整其大小和位置，如图 1.15 所示。

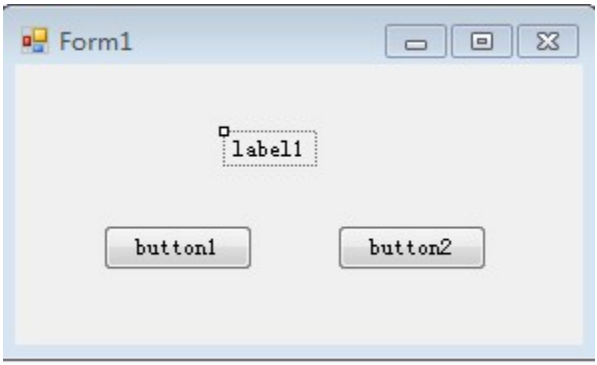


图 1.15　添加按钮控件后的窗体

4.属性设置

控件添加完毕后，接下来对窗体及窗体上各控件的属性进行设置。如果属性窗口是隐藏的，则用鼠标右击需要设置属性的对象，在弹出的下拉菜单中选择“属性”菜单命令，打开属性窗口。

窗体、标签和命令按钮的属性设置如表 1.2 所示。

表 1.2 控件的属性设置

控件类型	控件名称	属性	设置结果
Form	Form1	Text	欢迎
Label	Label1	Name	lblResult
Button	Button1	Name	btnOK
		Text	确定
	Button2	Name	btnCancel
		Text	取消

设置好窗体及各控件的属性后用户界面如图 1.16 所示。

图 1.16 设置属性后的界面

5.编写代码

双击 btnOK 按钮，在出现的代码中添加按钮单击事件的处理代码，如下：

```
private void btnOK_Click(object sender, EventArgs e)
        {
            lblResult.Text = "欢迎进入 C#世界！";
        }
```

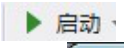
切换到用户界面窗口，再双击“取消”按钮，按照同样的方法在 btnCancel 对象的 Click 事件中加入如下代码：

```
private void btnCancel_Click(object sender, EventArgs e)
        {
            Application.Exit();
        }
```

到此，整个程序的代码全部添加完成，代码中的两个代码段（方法）都是用了鼠标的 Click（单击）事件。

在窗体和代码都设计好后，应当保存文件，以防止调试或运行程序时发生死机等意外而造成数据丢失，保存文件可以单击“文件”菜单下的“保存”或“全部保存”命令，也可单击工具栏上的“保存”命令来实现。

6.运行效果

应用程序设计的前期工作已经全部完成，下一步是调试和运行程序，运行程序的方法是：单击“调试”菜单下的“启动调试”命令，或者单击工具栏中的按 

钮，还可以按快捷键F5。运行效果如图1.17所示。

图1.17　程序运行结果

单击“确定”按钮，则会在窗体上方的标签中显示“欢迎进入C#世界！”字样，如图1.18所示。

图1.18　点击“确定”按钮的运行结果

单击“取消”按钮，则窗体关闭，并结束应用程序的运行。

1.6 小　　结

本章简要地介绍了.NET Framework，并且对C#语言的特点进行了描述。由于本章主要是面向刚刚入门的读者编写的，所以比较详细地讲解了如何顺利编写第一个C#应用程序——Welcome。这个程序是熟悉C#编程环境的最简洁的途径，初级读者并不需要对这个程序做太多的研究，因为这里毕竟只是为了熟悉一下编程环境而已，另外还要了解开发Windows应用程序的基本流程：界面设计、属性设置、编写代码、调试运行，对于学习C#语言的来说，更重要的则是后面各章将要介绍的内容。

1.7 习　　题

1. 简答题

(1) 简述 Visual C#程序设计语言的优点

(2) 开发 Windows 程序的基本流程是什么？

2. 编程题

建立一个 Windows 应用程序，在窗体上放置 2 个按钮和 1 个 TextBox 控件，将 2 个按钮的 Text 属性分别设置为“Red”、“Green”。编写程序实现通过单击不同的按钮使 TextBox 控件的文字颜色发生变化。

图 1.19　Windows 程序界面

图 1.20　点击“Red”按钮的运行结果