第 1 章　C# 入门基础

王忠民

内容简介

C#（ C Sharp ）是微软公司在 2000 年 6 月发布的一种新的编程语言，也是微软为 .NET Framework 量身订做的程序语言。它拥有 C/C++ 的强大功能以及 Visual Basic 简易使用的特性。本章详细介绍 C# 的入门知识，包括 .NET Framework 和程序集的概念、命名空间以及如何安装 .NET Framework 的开发环境等内容。

通过本章的学习，用户可以了解 C# 、 .NET Framework 和 Visual Studio 的相关知识，并且可以使用 Visual Studio 创建第一个控制台应用程序和窗体应用程序。

.NET 平台全新的语言

- 安德斯·海尔斯伯格（ Anders Hejlsberg ， 1960.12~ ），丹麦人，发明了 Delphi 、 C# 两种著名编程语言
- Turbo Pascal 编译器的主要作者
- Delphi 之父
- C# 之父
- 1996 年从 Borland 公司到微软
- 主持了 Visual J++ 的开发
- 微软 .Net 的首席架构师， .Net 概念的发起人之一

1.1 C# 语言简介

C# 与 Java 有着惊人的相似，它包括了单一继承、界面、与 Java 几乎相同的语法、和编译成中间代码再运行的过程。

C# 是一种安全的、稳定的、简单的、优雅的，由 C 和 C++ 衍生出来的面向对象的编程语言。它继承了 C 和 C++ 强大功能的同时去掉了一些它们的复杂特性，并且它综合了 VB 简单的可视化操作和 C++ 的高运行效率，以其强大的操作能力、优雅的语法风格、创新的语言特性和便捷的面向对象编程的支持成为 .NET 开发的首选语言。如表 1 列出了 C# 语言和 C++ 语言的主要区别。

	C#	C++
编译目标	编译为中间语言代码，执行时再通过 JIT 将需要的模块临时编译成本地代码	直接编译为本地可执行的代码
内存管理	采用垃圾回收机制自动回收不再使用的内存	显式删除动态分配给堆的内存
指针	基本不使用指针	可以大量的使用指针
库	依赖于 .NET Framework 类库	依赖于以继承和模板为基础的标准库
继承	只能单继承，但是可以实现多个接口	允许多继承

1.2 .NET Framework 简介

任何程序语言的运行都需要一个开发环境，C# 语言的开发环境就是 .NET Framework ，简称为 .NET ，也叫 .NET 框架。

本节介绍 .NET Framework 的相关知识，包括 .NET 的概念、功能体现和 .NET Framework 的重要组件等内容。

1.2.1 .NET Framework 概述

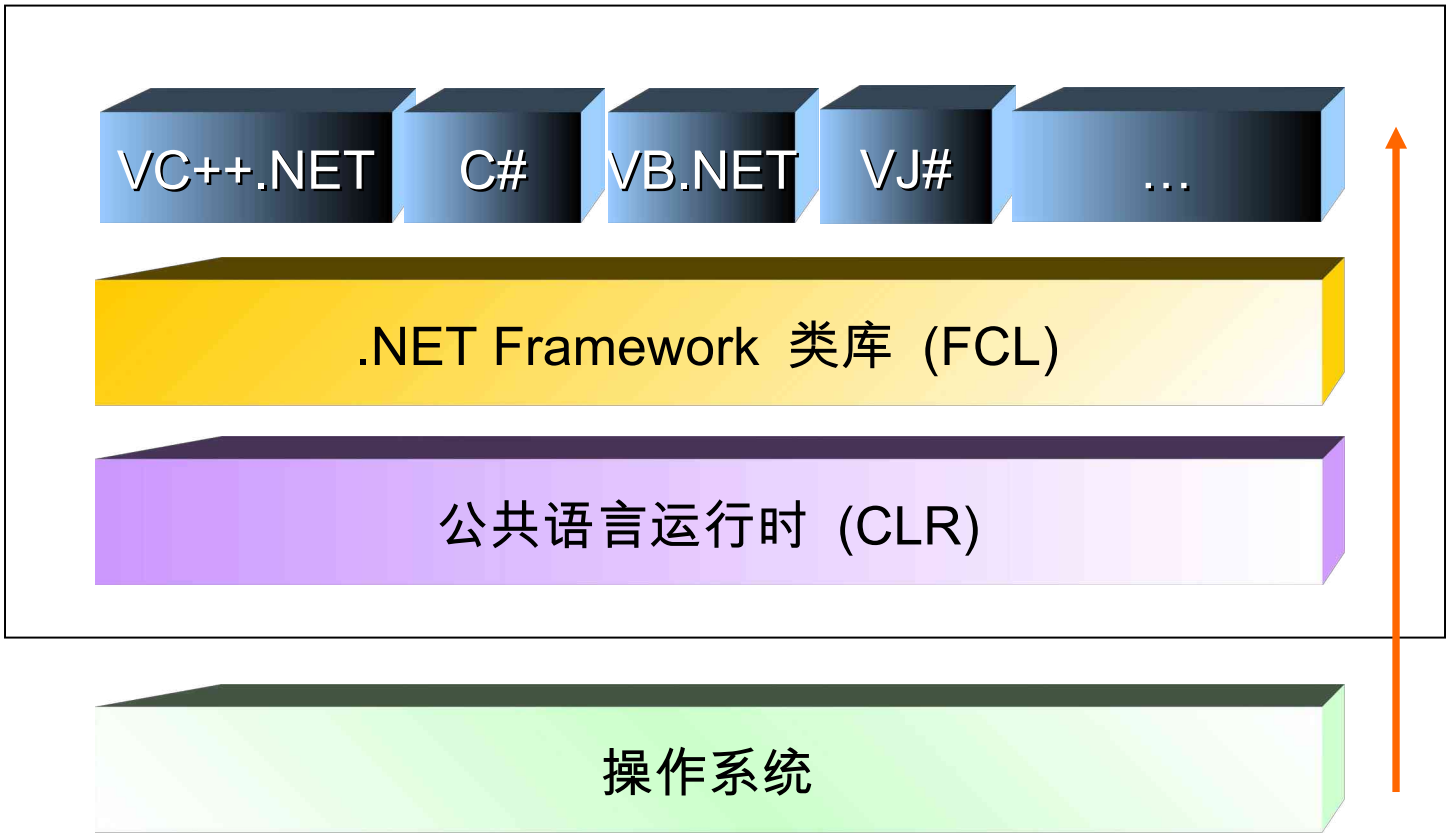
.NET Framework 是由微软开发，致力于敏捷软件开发、快速应用开发、平台无关性和网络透明化的软件开发平台。它以公共语言运行库为基础，支持多种语言（如 C# 、 VB 、 C++ 和 Python 等）的开发。

.NET Framework 的功能非常强大，主要体现在以下方面：

- 提供一个面向对象的编程环境，完全支持面向对象编程
- 提供一个将软件部署和版本控制冲突最小化的代码执行环境
- 提供一个可提高代码执行安全性的代码执行环境
- 提供一个可消除脚本环境或解释环境性能问题的代码执行环境
- 对 Web 应用和 Web Service （ Web 服务）提供强大支持

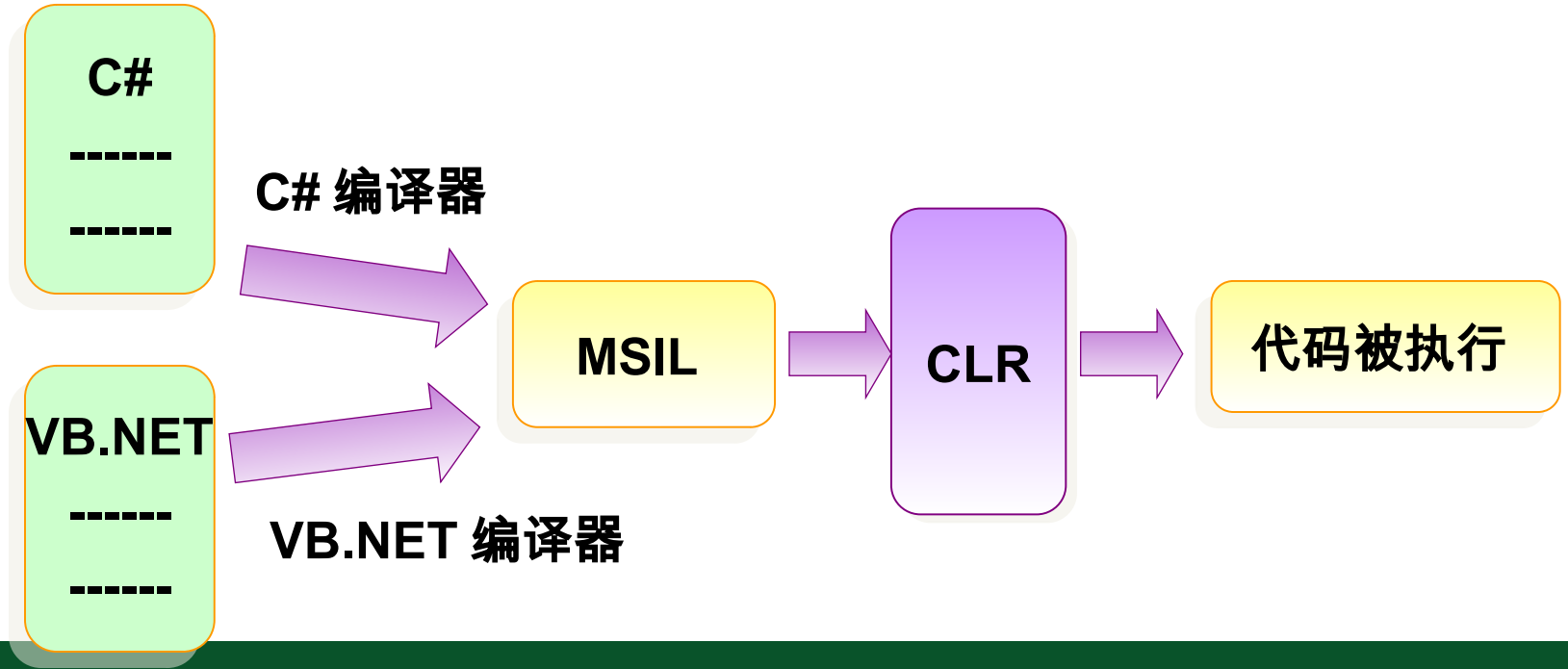
.NET Framework 主要有两个组件：公共语言运行时和 .NET Framework 类库。公共语言运行时是 .NET Framework 的基础。

.NET 框架的核心



C# 所开发的程序源代码并不是编译成能够直接在操作系统上执行的二进制本地代码。与 Java 类似，它被编译成为中间代码——MSIL（ Microsoft Intermediate Language ）。然后通过 .NETFramework 的虚拟机（通用语言运行时 CLR ）执行。 执行时， .Net Framework 将中间代码翻译成为二进制机器码，使它得到正确的运行。

因此虽然最终的程序仍然具有“ .exe” 的后缀名，但是如果计算机上没有安装 .Net Framework ，将不能够被执行。



1.2.2 公共语言运行时

公共语言运行时（ Common Language Runtime ，简称 CLR ）是 .NET 的基础，是所有 .NET 应用程序运行时的环境和编程基础。包括两部分：

1、CTS（ Common Type System ，通用类型系统）

CTS 定义了如何在运行库中声明、使用和管理类型，同时也是运行库支持跨语言集成的重要组成部分，用于解决不同语言的数据类型不同的问题。例如： VB.NET 的 Integer 和 C# 的 int 型都被编译成 Int32 。

CTS 执行的主要功能如下：

- 建立一个支持跨语言的集成、类型安全和高性能代码执行的框架
- 提供一个支持完整实现多种编程语言的面向对象的模型
- 定义各语言必须遵守的规则，有助于确保用不同语言编写的对象能够交互作用

2、CLS（ Common Language Specification ，公共语言规范）

CLS 是指确定公共语言运行库如何定义、使用和管理类型的规范。它定义了 .NET 平台下各种语言必须支持的最小特征，以及各语言之间实现互操作所需要的完备特征。（例如：是否区分大小写）

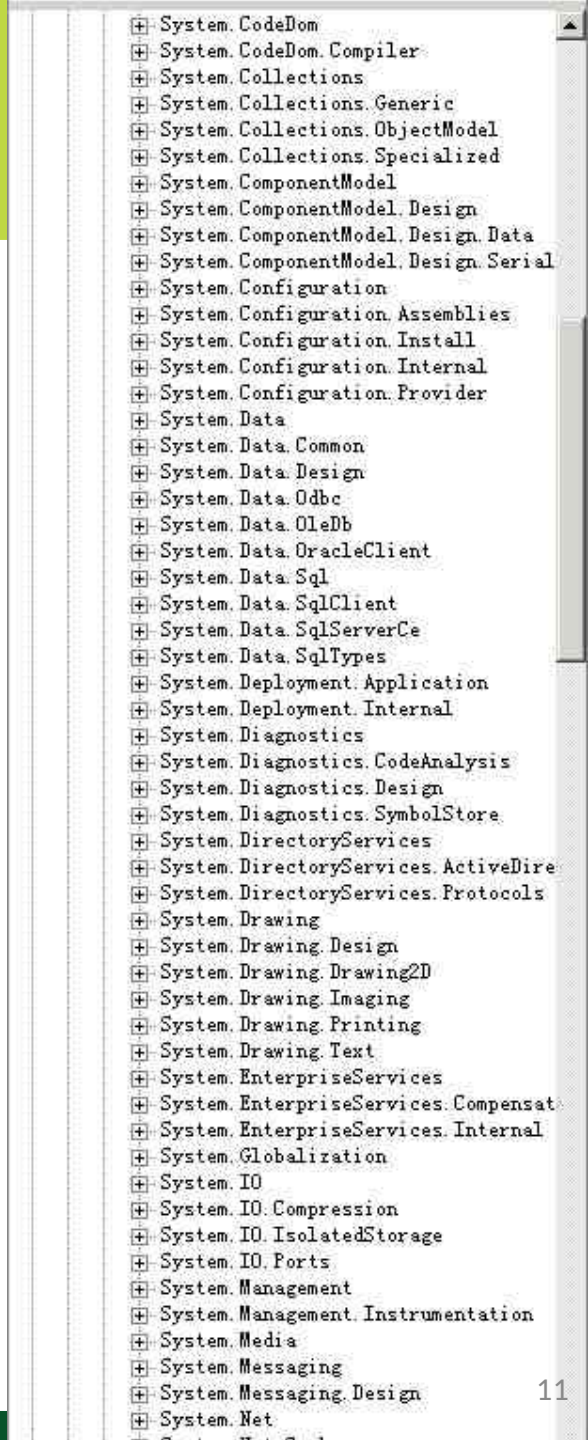
1.2.3 类库

.NET Framework 类库是一个综合性的面向对象的可重用类型集合，它是一个由 Windows 软件开发工具包中包含的类、接口和值类型所组成的库。

.NET 框架类库提供了大量实用的类，是开发程序时的重要资源。

类库

- 庞大的类库数量
  - 170 多个命名空间，上千个类
- 功能齐全，方便使用
  - 对文件的基本操作、对网络的访问
  - 安全控制、对图形的操作……



1.3 程序集

.NET Framework 类库由许多**程序集**组成，它提供了多种功能，如读取和写入文件、从数据库保存和检索信息以及提供窗体的功能等。

下表列出了类库中常用的程序集。

程序集名称	**说明**
System.dll	定义数据类型，如 Int 和 Long
System.Windows.Forms.dll	包含桌面应用程序的窗体组件，以及创建这些窗体的组件
System.XML.dll	包含处理文档所必需的组件
System.Drawing.dll	包含用于向输出设备绘制各种图形（如直线、椭圆等）的组件
System.Data.dll	定义组成 ADO.NET 的组件

通常，静态的程序集可以由 4 个元素组成：

- 程序集清单 包含程序集元数据，它是必需的内容
- 类型元数据
- 实现这些类型的 Microsoft 中间语言（ MSIL ）代码
- 资源集

程序中的元素分组有几种方法：（ 1 ）将所有元素分组到单个文件中。（ 2 ）如果希望组合不同语言编写的模块并优化应用程序的下载过程，可以创建多文件程序集。

图 4

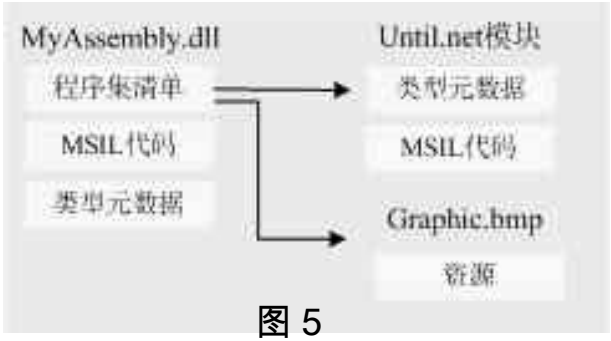


图 5

图 4 和图 5 分别表示单文件程序集和多文件程序集的结构。

图 5 中三个文件属于一个程序集。对于文件系统而言，这 3 个文件是独立的文件，但是 Until.net 被编译为一个模块，它不包含任何程序集信息。当创建了程序集后，该程序集清单被添加到 MyAssembly.dll ，指示程序集与 Until.net 模块和 Graphic.hmp 的关系。

1.4 命名空间

.Net FrameWork 中的类都包含在命名空间里面。

在使用 .NET Framework 类库时，常常会引入一些相应的命名空间。命名空间使用关键字 namespace 表示，它提供了一个组织相关类和其他类型的方式，它是一种逻辑组合，而不是物理组合。

可以将命名空间理解为组，组中包含的是具有相同或类似功能的类。每一个程序集可以包含一个或多个组，如 System.dll 中就包含了 System.Int16 、 System.Int32 和 System.String 类等。

一个程序集中可以包含多个命名空间，一个命名空间也可以在多个程序集中。

程序开发过程中常用的命名空间

命名空间	说明
System	可叫做根命名空间，它包含 .NET Framework 类库中的其他所有命名空间
System.Data	包含提供数据访问功能的命名空间和类
System.Drawing	包含了提供与 Windows 图形设备接口的接口类
System.IO	包含了用于读写数据流 / 文件和普通输入 / 输出（ I/O ）功能的类
System.Windows.Forms	定义包含工具箱中的控件及窗体自身的类
System.Net	包含了用于网络通信的类或命名空间
System.Xml	包含用于处理 XML 数据的类
System.Text	包含表示 ASCII 、 Unicode 和 UTF-8 等字符编码的类

- System.Data // 命名空间提供对表示 ADO.NET 结构的类的访问。通过 ADO.NET 可以生成一些组件，用于有效管理多个数据源的数据。
- System.Data.Common // 命名空间包含由各种 .NET Framework 数据提供程序共享的类。
- System.Data.Odbc // 用于 ODBC 的 .NET Framework 数据提供程序。
- System.Data.OleDb // 用 OLE DB 的 .NET Framework 数据提供程序。
- System.Data.Sql // 命名空间包含支持 SQL Server 特定功能的类。
- System.Data.OracleClient // 是用于 Oracle 的 .NET Framework 数据提供程序。
- System.Data.SqlClient // 是 SQL Server 的 .NET Framework 数据提供程序。
- System.Data.SqlTypes // 命名空间为 SQL Server 中的本机数据类型提供类。

- System.Drawing // 命名空间提供了对 GDI+ 基本图形功能的访问。
- System.Drawing.Design // 命名空间包含扩展设计时用户界面 (UI) 逻辑和绘制的类。
- System.Drawing.Drawing2D // 命名空间提供高级的二维和矢量图形功能
- System.Drawing.Imaging // 命名空间提供高级 GDI+ 图像处理功能
- System.Drawing.Text // 命名空间提供高级 GDI+ 排版功能。
- System.Globalization // 命名空间包含定义区域性相关信息的类，这些信息包括语言、国家 / 地区、使用的日历、日期、货币和数字的格式模式以及字符串的排序顺序。我们可以使用这些类编写全球化（国际化）应用程序

- System.IO // 命名空间包含允许读写文件和数据流的类型以及提供基本文件和目录支持的类型。
- System.Management // 提供对大量管理信息和管理事件集合的访问，这些信息和事件是与根据 Windows 管理规范 (WMI) 结构对系统、设备和应用程序设置检测点有关的
- System.Net // 命名空间为当前网络上使用的多种协议提供了简单的编程接口
- System.Net.Mail // 命名空间包含用于将电子邮件发送到简单邮件传输协议 (SMTP) 服务器进行传送的类

- System.Timers // 提供 Timer 组件，可以指定间隔引发事件。
- System.Web // 提供使得可以进行浏览器与服务器通信的类和接口。
- System.Web.Caching // 提供用于缓存服务器上常用数据的类。
- System.Web.Configuration // 包含用于设置 ASP.NET 配置的类。
- System.Web.Handlers // 命名空间包含的 HTTP 处理程序类用于处理对 Web 服务器发出的请求。
- System.Web.Services // 命名空间由使您可以用 ASP.NET 和 XML Web services 客户端来创建 XML Web services 的类组成
- System.Web.UI // 命名空间提供的类和接口可用于创建 ASP.NET 服务器控件以及用作 ASP.NET Web 应用程序用户界面的 ASP.NET 网页
- System.Xml // 命名空间为处理 XML 提供基于标准的支持。

除了系统提供的命名空间外，用户也可以自定义命名空间。自定义命名空间需要使用关键字 namespace 。它的定义规则如下：

- 命名空间名可以是任何合法的标识符，也可以包含句号“ .”
- 无论用户是否显示声明命名空间，编译器都会添加一个默认的命名空间。该未命名的命名空间存在于每一个文件中
- 命名空间隐式具有公共访问权，并且是不可修改的
- 在两个或更多的声明中可以定义一个命名空间

引用命名空间： using ……

命名空间与程序集的区别

命名空间用于对类型进行逻辑分组（针对开发人员）。

程序集则是程序的物理分组（针对安装和部署），对应于一个 dll 或 exe 文件。

采用两种分组的好处

由不同的部门共同开发同一个命名空间的不同组件时，各部门可以把他们开发的东西编译成各自的 dll 文件。使用时把几个 dll 一起引用即可。

一个命名空间里包含的类型很多，而有很多是用不上的。如果把一个命名空间编译成一个 dll ，则会降低运行效率。

有时需要对一个命名空间内的类型区别对待。例如，所开发的软件分普通版、专业版、高级版，越高的版本具有越高的权限（当中包含了一些高级的类），这时可以把高级的功能放到一个单独的 dll 来控制。

.NET 框架可构建的应用类型

- Windows 应用程序
- 控制台应用程序
- Windows 服务
- Web 应用程序
- Web service 应用
- 面向 office 应用
- 智能设备应用

1.5 安装 Visual Studio

Visual Studio 和 .NET Framework 的关系如图 6 所示。

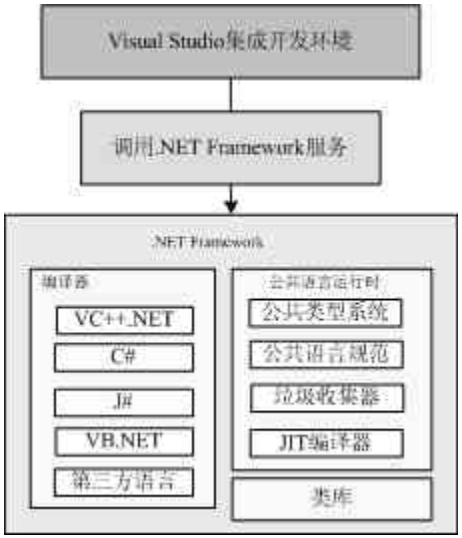


图 6

从图 6 中可以看出，Visual Studio 依赖于 .NET Framework 提供的服务。这些服务包括 Microsoft 公司或者第三方提供的语言编译器。用户在执行 .NET Framework 语言开发的应用程序时必须安装 .NET Framework ，不过 .NET Framework 会在安装 Visual Studio 程序时自动安装。

1.7 创建第一个控制台应用程序

在本案例中新建控制台应用程序，然后输出一句话：“大家好，这是我的第一个项目。”

（1）单击【开始】|【程序】|Microsoft Visual Studio | Microsoft Visual Studio 的命令，打开【起始页 -Microsoft Visual Studio】的对话框。

（2）在起始页中单击【新建项目】选项或者单击【文件】|【新建】|【项目】选项，弹出【新建项目】对话框。

（3）在【新建项目】对话框中，左侧选择 Visual C# 选项，中间选择【控制台应用程序】选项，然后输入要添加的应用程序名称，也可以重新设置路径。如图 17 所示。全部完成后，单击【确定】按钮。

1.7 创建第一个控制台应用程序

（4）打开新添加的项目，在 Program.cs 文件的 Main() 方法中添加要输出的内容。具体代码如下：

```
static void Main(string[] args)
{
   Console.WriteLine(“ 大家好，这是我的第一个项目。 ");
}
```

（5）单击【调试】|【启动调试】选项或直接单击 F5 运行。

为了看清控制台的输出，可加上：

Console.ReadLine();

Console 类

输出到控制台

Console.WriteLine(输出的值); 向控制台直接输出后换行
Console.WriteLine();
Console.Write(输出的值); 向控制台输出，不换行
Console.Write();
Console.Beep(); 通过控制台扬声器播放提示音

Console.WriteLine(" 输出的格式字符串 ", 变量列表);
Console.Write(" 输出的格式字符串 ", 变量列表);

例： Console.WriteLine(“ 团队名称是 {0} ，包括 {1},{2} 等组员”，groupName,strName[1],strName[2]]);

括号中包含两类参数：“格式字符串”和变量列表。格式字符串中的 {0} 、 {1} 、 {2} 、 {3} 叫做**占位符**，代表后面依次排列的变量表，从 0 开始，依次类推，完成输出。

Console 类

从控制台输入

Console.ReadLine() 返回字符串型数据，能读多个字符也可换行读取。
可以把返回值直接赋给字符串变量，如：

```
string strname=Console.ReadLine();
```

有时需要从控制台输入数字，就要做数据转换，如：

```
int num = int.Parse(Console.ReadLine());
int num = Convert.ToInt32(Console.ReadLine());
```

上面两句代码效果相同，可以根据自己的习惯选择任意一种。

Console.Read(), 返回首字符的 ASCII 码（ int 类型），只能读取第一个字符。

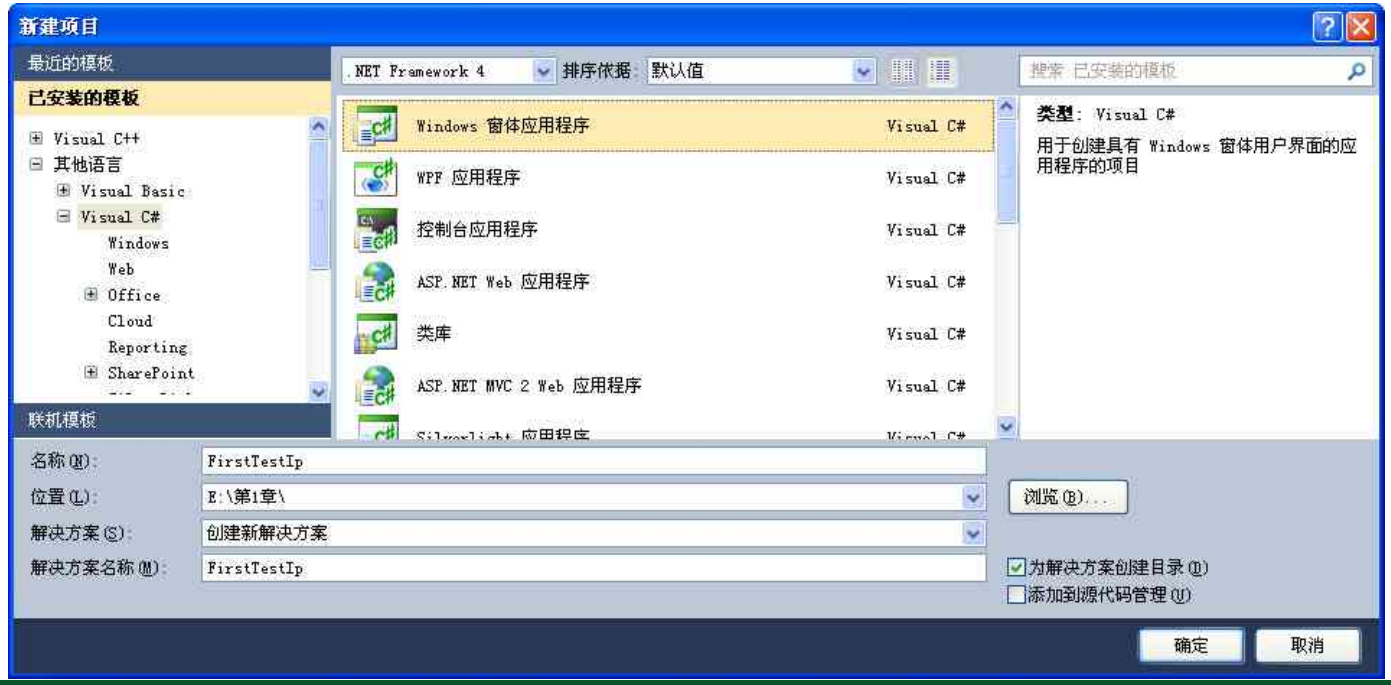
- 例： Console.WriteLine(Console.ReadLine())

1.8 创建第一个窗体应用程序

例：创建窗体应用程序，单击窗体上的按钮检测用户输入的 IP 地址是否合法。

（1）打开 Visual Studio ，然后单击【文件】|【新建】|【项目】选项，弹出【新建项目】对话框。

（2）在【新建项目】对话框中，中间选择【Windows 窗体应用程序】选项，然后输入保存窗体应用程序的名称，选择保存的路径。输入完成后，单击【确定】按钮。



1.8 创建第一个窗体应用程序

（3）将默认生成名称为 Form1 的窗体重新命名为 TestIP ，从左侧【工具箱】中向该窗体添加 Label 控件、 TextBox 控件和 Button 控件。

（4）使用 Ping 类和 PingReply 类实现检测 IP 地址的功能。为【测试】按钮的 Click 事件添加如下的代码：

```
using System.Net.NetworkInformation;
private void button1_Click(object sender, EventArgs e)
{
   Ping testPing = new Ping();                          // 创建 Ping 类的实例对象
   PingReply reply = testPing.Send(textBox1.Text); / / 调用 Send() 方法
   if (reply.Status == IPStatus.Success)          // 判断 IP 地址是否合法
   {
      string message = string.Format(" 地址： {0} 连接测试成功！ ",
textBox1.Text);
      MessageBox.Show(message);
   }else{
      string message = string.Format(" 地址： {0} 连接测试失败！ ",
textBox1.Text);
      MessageBox.Show(message);
   }
}
```

上面的程序运行时，若输入的是非法字符串，会出错，故修改为：

```
private void button1_Click(object sender, EventArgs e)
    {
       Ping testPing = new Ping();
       try
       {
            PingReply reply = testPing.Send(textBox1.Text);
            if (reply.Status == IPStatus.Success)
            {
               string message = string.Format(" 地址 : {0}  连接成功 !",
textBox1.Text);
               MessageBox.Show(message);
             } else
             {
               string message = string.Format(" 地址 : {0} 失败 !", textBox1.Text);
               MessageBox.Show(message);
             }
       } catch (Exception ex)
       {
          MessageBox.Show(“ 错了 ....“ + + ex.Message);
       };
   }
```

.NET Framework 开发中的拼写：

- 帕斯卡拼写法， PascalCase 第一个单词的首字母大写
- 骆驼拼写法， camelCase 第一个单词的首字母小写

对于普通的参数变量使用 camelCase ，常量采用全部大写 RED_VALUE ，其它的使用 PascalCase

解决方案、项目

项目可以表现为多种类型，如控制台应用程序， Windows 应用程序，类库（ Class Library ），Web 应用程序， Web Service ， Windows 控件等等。

应用程序都有一个主程序入口点，即方法 Main() 。如果经过编译，应用程序都会被编译为 .exe 文件既然是 .exe 文件，就表明它是可以被执行的。

而类库、 Windows 控件等，则没有 Main() ，仅提供一些功能给其他项目调用，不能直接执行，会被编译为 .dll 文件。

建立解决方案后，会建立一个扩展名为 .sln 的文件。

在解决方案里添加项目，要在“ File” 菜单中，选择“ Add Project” 。添加的项目，可以是新项目，也可以是已经存在的项目。

例：解决方案中包含了两个项目，一个是控制台应用程序，一个是类库。类库提供一些基本的功能，如 Print() 方法。我们常常把一些共用的方法，放到类库中。这样其他的应用程序就可以去调用它。

代码窗口鼠标右键可打开设计器。

视图 → 解决方案资源管理器。　　解决方案中添加多个项目，设置启动项目