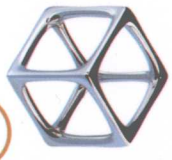
源代码免费下载



商业开发
代码库系列

Visual C#.NET

案例开发集锦

（第二版）

马　煜　陈海军　朱朝阳　编著
朱晓冰　康祥顺　审校

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY
http://www.phei.com.cn

商业开发代码库系列

Visual C#.NET案例开发集锦

（第二版）

马　煜　陈海军　朱朝阳　编著

朱晓冰　康祥顺　审校

電子工業出版社

Publishing House of Electronics Industry

北京 · BEIJING

内 容 简 介

本书围绕.NET框架提供的类库，详细探讨如何使用Visual C#.NET 2005开发应用程序，每个技术要点均通过具体的案例来解析，从Visual C#.NET 2005的基本开发环境到高级线程管理，内容涵盖了Visual C#.NET应用开发、控件操作、Windows窗体编程、图形图像和多媒体应用与打印、文件目录与输入输出、系统维护、线程和进程与同步、网络应用、数据库编程、ASP.NET开发与Web编程、XML操作等11个方面的近百个案例。此书为读者提供了.NET框架的知识手册，根据此书中的源码，读者可以结合自己的实际，快速、高效、灵活地设计出专业级的应用程序，而且所有的案例基本上都可以直接嵌入到读者自己的程序中去。

本书适合对Visual C#.NET感兴趣的大中专学生、软件开发人员以及Visual C#.NET产品爱好者阅读，尤其适合希望精通Visual C#.NET编程的读者。

图书在版编目（CIP）数据

Visual C#.NET案例开发集锦/马煜，陈海军，朱朝阳编著.—2版.—北京：电子工业出版社，2008.4
（商业开发代码库系列）
ISBN 978-7-121-06022-9

Ⅰ. V… Ⅱ. ①马…②陈…③朱… Ⅲ. 语言—程序设计 Ⅳ. TP312

中国版本图书馆CIP数据核字（2008）第018906号

责任编辑：李 莹
印 刷：北京天竺颖华印刷厂
装 订：三河市金马印装有限公司
出版发行：电子工业出版社
北京市海淀区万寿路173信箱 邮编：100036
北京市海淀区翠微东里甲2号 邮编：100036
开 本：787×1092 1/16 印张：29.5 字数：750千字
印 次：2008年4月第1次印刷
定 价：48.00元

凡所购买电子工业出版社图书有缺损问题，请向购买书店调换。若书店售缺，请与本社发行部联系，联系及邮购电话：（010）88254888。
质量投诉请发邮件至zlts@phei.com.cn，盗版侵权举报请发邮件至dbqq@phei.com.cn。
服务热线：（010）88258888。

前　言

C#是微软推出的一门简单优雅、面向对象、类型安全、平台独立的新型组件编程语言。它是Visual Studio中的一部分，其语法风格源自C/C++家族，在保持C/C++灵活性的基础上，为程序员带来了有效的RAD开发方式。它不仅能用于Web服务程序的开发，而且还能开发强大的系统级程序。

C#是完全依靠Windows的最完美的产物。那个困扰Java SDK、MFC和SET的数据库已成为过去，你想放入C#的任何东西（无论何种语言），只要是在Windows的.NET子系统下建立和包装的，都可以使用Windows的运行库。

.NET是继Java之后最重要的平台级开发框架，如同整个.NET平台一样，C#语言如今已经是计算机世界中不容忽视的初生牛犊，随着越来越多的程序员开始在.NET Framework上进行应用程序开发，C#成为了首选的语言。

对C#语言感兴趣的乐知者，也许你们在学习中会发现，学习C#语言编程在深谙了C#语言的语法后，还需要对.NET框架中类的调用方法进行深入的了解。本书编者在了解了开发者这样的需求后，结合C#开发者在实际工作中遇到的问题，用百余个详细的案例解释了.NET框架中类的调用方法。本书既可作为C#语言的教程，也可以结合为.NET框架的类调用手册，本书中的精彩案例也能由学习者结合到自己开发的程序中。

本书通过100个编程案例，介绍了以下技术内容。

第1章：用4个案例说明了在编制C#应用程序时，开发者会遇到的开发环境中存在的问题。

第2章：详细给出了10种常用控件的使用与操作，结合实际案例不仅说明了默认的控件的执行方法，也给出了如何通过设置实现一些并不是默认的效果但又非常常用的操作；不但有.NET框架中提供的常用控件，也实现了用户自定义控件。

第3章：用4个案例讲解了Windows窗口应用程序开发中如何获得单文档程序和多文档程序的特点以及如何利用这些特点。

第4章：用21个案例说明了C#中图形、图像、多媒体与打印的实现。图像、视频、声音和打印是Windows操作系统的最大特点，针对.NET框架提供的强大的图形、图像操作类库，我们在本章中给出了详尽的解释。

第5章：用15个案例说明了如何对文件系统进行操作、如何读写文本文件、如何读写Word文档，绘制Excel三维曲线、获得文件和文件夹属性、复制和删除文件夹等。

第6章：用5个案例说明了如何操作系统日志、读写注册表、获取硬件信息等。

第7章：用8个案例介绍了多线程和分布式编程的特点，主要包括创建新线程、线程间如何通信、多线程同步的实现。

第8章：用9个案例介绍了网络通信的TCP/IP协议、UDP协议、SMTP协议的开发实现。

第9章：用7个案例介绍了数据库开发过程中经常用到的一些技巧和编程方法，包括连接数据源、执行SQL命令或存储过程、如何从SQL Server查询到XML文档、操作水晶报表等。

第10章：用9个案例说明了ASP.NET和Web编程，包括页面之间的信息传递、ASP.NET中如何调用JavaScript事件、ASP.NET调用正则表达式、创建Web Service应用程序等。

第11章：用5个案例说明了C#中对XML如何操作，包括创建XML文档、分割XML文档以及查询XML中的内容等。

本书主要由马煜主持编写，参加编写的还有陈海军、朱朝阳、邵天增、尚冬娟、冯妍、董强、李海军、汤丹丹、朱晓冰、刘强、王继元、卜润友、高晶文、王芳、姜云、孙明、王亚等。由于作者水平有限，书稿不中免有纰漏，恳请读者批评指正。

本书案例运行环境为Visual Studio 2005，数据库管理工具为SQL Server 2000。

为方便读者阅读，若需要本书配套资料，请登录“华信教育资源网”（http://www.hxedu.com.cn），在“资源下载”频道的“图书资源”栏目下载。

目　录

第1章

Visual C#.NET应用开发

本章内容

- 创建控制台应用程序
- 创建Windows应用程序
- 存取命令行参数
- 条件编译

案例1　创建控制台应用程序

案例运行效果与操作

首先，我们使用C#创建这样一个应用程序，它能够实现在命令行控制台接收输入并显示输出的功能。因为这样的应用程序用户界面非常简单，所以对于学习C#开发非常理想。控制台应用程序对于极少需要或不需要用户交互的实用工具程序也非常有用。

在本例中，你可以跟随我们创建你的第一个控制台应用程序，这是一个非常简单的程序，用户通过命令窗口输入自己的姓名，回车后控制台上会显示“XX欢迎您来到C#的世界！”，执行结果如图1-1所示。

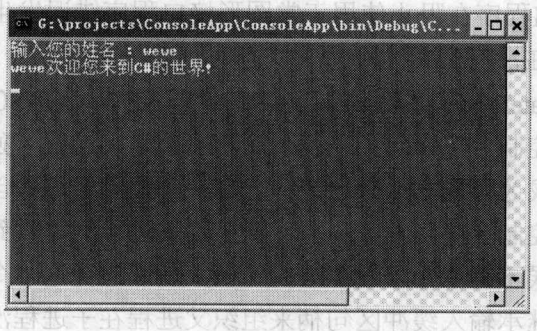


图1-1　控制台应用程序执行效果图

知识要点

控制台管理字符模式程序的输入和输出（程序不需要提供它们自己的图形用户界面）。控制台函数使访问不同级别的控制台成为可能。高级控制台I/O函数使程序可以从标准输入，以重新得到存储在控制台输入缓冲区中的键盘输入；这些函数也能够使程序向标准输出，或

将错误显示在标准的控制台缓冲区中。高级控制台函数也支持重定向标准的句柄和不同I/O功能的控制台模式。低级I/O函数能够使程序接收详细的键盘输入和鼠标事件，以及控制台窗口和用户交互的事件，同时，也能控制屏幕输出。

控制台对简单字符模式程序提供高级的支持，其通过读写标准输入和输出以及标准错误函数来和用户交互。控制台也提供复杂低级支持，例如：直接访问控制台屏幕缓冲区，以及接收额外输入信息（例如鼠标输入）。

·什么是控制台

控制台是一个用来提供字符模式I/O的接口。正是因为有这种处理器独立的机制存在，才使得导入一个存在的字符模式程序或创建一个新的字符模式工具和程序变得更为容易。

控制由输入缓冲区和一到多个屏幕缓冲区组成。输入缓冲区包含一个输入记录的序列，序列中是输入事件的信息。输入队列既可以包含键按下和松开键事件，也包括鼠标事件（指针移动和鼠标键按下或释放），以及用来影响活动屏幕区域大小的用户动作。屏幕缓冲区是一个控制台窗口的二维字符数组和彩色数据。所有的处理能共享一个控制台。

系统在启动一个控制台程序的时候创建一个控制台，控制台程序是一个字符模式的程序，入口是main函数。例如，系统在其他命令处理器中会创建一个新的控制台。当命令处理器开始一个新的控制台程序时，用户能指定系统是为新的程序创建一个新的控制台，还是从命令处理器控制台继承。

一个程序可以使用下面的方法来创建一个控制台：

1. GUI或控制台程序可以使用CreateProcess函数并带有CREATE_NEW_CONSOLE标志来创建一个带有新控制台的控制台程序（默认情况下，控制台程序从它的父控制台中继承，并且不能保证输出可以被程序接收）。

2. GUI或控制台进程没有附着到一个控制台上时，可以使用AllocConsole函数来创建一个新的控制台（GUI程序在创建的时候不附着到控制台上，控制台进程在使用DETACHED_PROCESS标志的CreateProcess函数创建的时候也不附着到控制台上。

典型情况下，一个程序在错误发生并请求用户交互的时候使用AllocConsole来创建一个控制台。例如，一个GUI程序在阻止使用正常图形接口程序错误发生的时候能创建一个控制台，或一个控制台进程在没有正常地和用户交互时创建一个控制台来显示错误。

进程可以在调用CreateProcess的时候指定CREATE_NEW_CONSOLE标志来创建一个控制台。用这个方法创建的控制台可访问子进程，而不可访问父进程。独立的控制台对于父子进程与用户不冲突的交互成为可能。如果这个标志在一个控制台进程创建的时候没有指定，两个进程都附着到相同的控制台上，并且不能保证正确的进程能接收到提供给它的输入。程序可以在创建子进程的时候不继承输入缓冲区的句柄，从而避免这种令人迷惑的情况发生，或者同时只有一个子进程继承输入缓冲区句柄来组织父进程在子进程没有完成的时候读控制台输入。

创建一个新控制台的结果是产生一个新的控制台窗口，也包括独立的屏幕缓冲区。同时，新控制台关联的进程可以使用GetStdHandle函数来得到新的控制台输入和屏幕缓冲区的句柄。这些句柄使进程可以访问控制台。

当一个进程使用CreateProcess函数时，它可以指定一个STARTUPINFO结构，该结构的成员控制为子进程创建的第一个新控制台的特性。如果CREATE_NEW_CONSOLE标志被指定，

STARTUPINFO结构在调用CreateProcess时会影响一个控制台的创建；它也影响子进程后来使用AllocConsole来创建控制台。下面的控制台特性可以指定：

1. 新控制台窗口的大小，字符单元。
2. 新控制台窗口的位置，屏幕像素坐标。
3. 新控制台屏幕缓冲区的文本和背景颜色属性。
4. 新控制台窗口的TITLE BAR上显示的名字。

如果STARTUPINFO值没有指定，系统则使用默认的值。子进程可以使用GetStartupInfo函数来判断STARTUPINFO结构中的值。

进程不能改变控制台窗口在屏幕上的位置，但下面的控制台函数可以用来设置和获得STARTUPINFO结构的其他属性，如表1-1所示。

表1-1 获得STARTUPINFO结构的其他属性的控制台函数

函数	描述
GetConsoleScreenBufferInfo	返回窗口大小、屏幕缓冲区大小和颜色属性
SetConsoleWindowInfo	改变控制台窗口的大小
SetConsoleScreenBufferSize	改变控制台屏幕缓冲区的大小
SetConsoleTextAttribute	设置颜色属性
SetConsoleTitle	设置控制台窗口的标题
GetConsoleTitle	获得控制窗口的标题

进程可以使用FreeConsole函数来分离继承的控制台或通过AllocConsole创建的控制台。

本节中我们将生成一个在屏幕上显示“欢迎您来到C#的世界！”的小程序，其显示功能由一个.cs文件写成。

实现步骤

（1）打开VS 2005编译器，选择主菜单“文件”➤“新建”➤“项目”，在弹出的“新建项目”窗口中，选择“控制台应用程序”，在窗口下方的“名称”文本框中输入ConsoleApp，如图1-2所示。

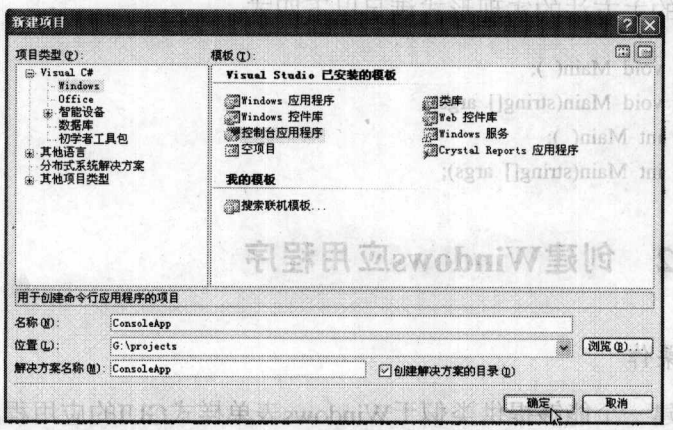


图1-2 “新建项目”对话框

（2）默认情况下，系统创建一个控制台应用项目，鼠标指向系统的Main( )方法，在这里你就可以向Main( )方法输入你所需要的代码了：

```
using System;
using System.Collections.Generic;
using System.Text;
namespace ConsoleApp
{
    class Program
    {
        //显示一个提示符并读入屏幕上的输入
        public static string ReadString(string msg)
        {
            Console.Write(msg);
            return System.Console.ReadLine( );
        }
        //在屏幕上显示一条信息
        public static void WriteString(string msg)
        {
            System.Console.WriteLine(msg);
        }
        //控制台应用程序的主方法
        public static void Main( )
        {
            // Prompt the reader to enter their name
            string name = ReadString("输入您的姓名 : ");
            // Welcome the reader to the C# Cookbook
            WriteString( name + "欢迎您来到C#的世界! " );
            Console.ReadLine( );
        }
    }
}
```

（3）编译、运行程序。

注意：确保你的主方法的实现形式满足以下四式：

```
public static void Main( );
public static void Main(string[] args);
public static int Main( );
public static int Main(string[] args);
```

案例2　创建Windows应用程序

案例运行效果与操作

如果你需要创建一个能够提供类似于Windows表单样式GUI的应用程序，那么本例将带领你创建一个简单的Windows表单样式的应用程序。本例中的程序执行后，出现“欢迎窗口”

用户在该窗口中输入用户姓名并单击button1按钮后，会出现一个消息窗口，表示对用户的欢迎，如图1-3所示。

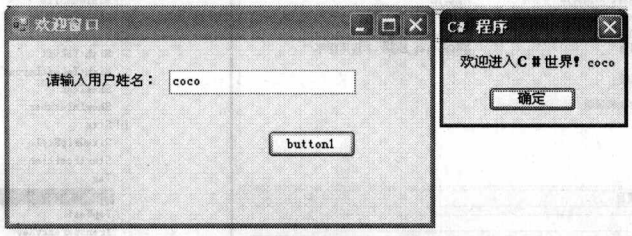


图1-3 C#窗口应用程序执行效果图

知识要点

确定你所有的原码文件中只有一个实现静态方法Main( )，在Main( )方法中创建一个继承自System.Windows.Forms.Form类的类实例，这就是你的项目中的主窗体。将你的主窗体对象传给System.Windows.Forms.Application的静态方法Run( )。

创建Windows应用程序的基本思路是：

• 创建项目中所需要的每个继承自System.Windows.Forms.Form类的窗体类。

• 在每个窗体类中声明本窗体所使用的控件成员，如Button、Lable等。将这些控件成员声明成private或protected，这样其他成员元素就不能直接对这个窗体中的控件元素进行操作。

• 在你的窗体类中声明这个控件需要处理的事件，例如Button控件的click事件或TextBox的key presses事件。这些方法会依据.NET事件模式提供private或protected事件。当然，在这些方法中你可以定义应用程序中所使用的函数。

• 构造器创建了Form窗体以及窗体的每个控件的初始状态（大小、颜色、位置、内容等）。构造器会创建窗体中每个控件的各事件的方法句柄。

• 在项目的主窗体中声明静态方法Main( )，这个方法是应用程序的入口；创建主窗体的实例，将这个实例传递给静态static Application.Run( )方法。static Application.Run( )方法使主窗体可见，并且开始一个标准的Windows消息循环线程，这个线程负责将用户的输入传递到应用程序的事件中。

本节中我们创建一个WelcomeForm窗口应用程序，用户输入用户名后，显示一个消息对话框，以欢迎用户进入C#世界。

实现步骤

（1）打开VS 2005编译器，选择主菜单“文件”➤“新建”➤“项目”，在弹出的“新建项目”窗口中，选择“Windows应用程序”，在窗口下方的“名称”文本框中输入“WelcomeForm”，如图1-4所示。

（2）系统将自动创建一个新窗口，修改该窗口的Text属性为“欢迎窗口”，如图1-5所示。

（3）从工具箱中选择Label控件，拖放到主窗体中，如图1-6所示。

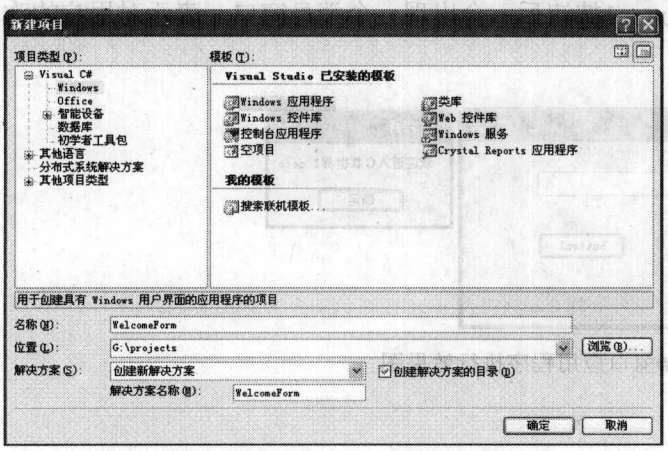


图1-4 新建Windows应用程序项目

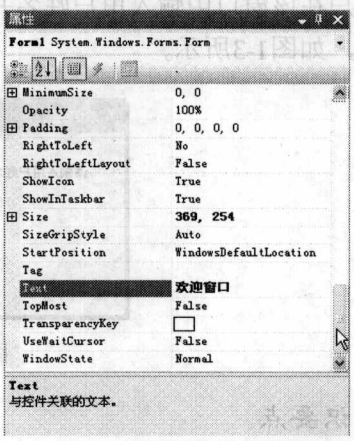


图1-5 设置主窗体的Text属性

（4）以同样的方式在窗口中放入一个TextBox控件和一个Button控件。创建好的主窗体的样式如图1-7所示。

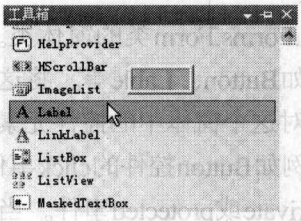


图1-6 从工具箱中选择Label控件

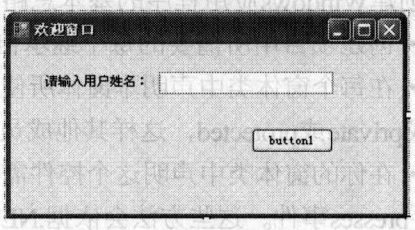


图1-7 主窗体的样式

（5）VS 2005系统为创建窗口的初始化自动设置代码，代码在Form1.Designer.cs文件中，如下所示：

```
namespace WelcomeForm
{
    partial class Form1
    {
        /// <summary>
        /// 必需的设计器变量
        /// </summary>
        private System.ComponentModel.IContainer components = null;
        /// <summary>
        /// 清理所有正在使用的资源
        /// </summary>
        /// <param name="disposing">如果应释放托管资源，为true；否则为false。</param>
        protected override void Dispose(bool disposing)
        {
            if (disposing && (components != null))
            {
                components.Dispose( );
            }
            base.Dispose(disposing);
        }
```

```
#region Windows 窗体设计器生成的代码
/// <summary>
/// 设计器支持所需的方法
/// 不要使用代码编辑器修改此方法的内容
/// </summary>
private void InitializeComponent( )
{
    this.label1 = new System.Windows.Forms.Label( );
    this.textBox1 = new System.Windows.Forms.TextBox( );
    this.button1 = new System.Windows.Forms.Button( );
    this.SuspendLayout( );
    //
    // label1
    //
    this.label1.AutoSize = true;
    this.label1.Location = new System.Drawing.Point(30, 29);
    this.label1.Name = "label1";
    this.label1.Size = new System.Drawing.Size(101, 12);
    this.label1.TabIndex = 0;
    this.label1.Text = "请输入用户姓名：";
    //
    // textBox1
    //
    this.textBox1.Location = new System.Drawing.Point(137, 26);
    this.textBox1.Name = "textBox1";
    this.textBox1.Size = new System.Drawing.Size(160, 21);
    this.textBox1.TabIndex = 1;
    //
    // button1
    //
    this.button1.Location = new System.Drawing.Point(222, 79);
    this.button1.Name = "button1";
    this.button1.Size = new System.Drawing.Size(75, 23);
    this.button1.TabIndex = 2;
    this.button1.Text = "button1";
    this.button1.UseVisualStyleBackColor = true;
    this.button1.Click += new System.EventHandler(this.button1_Click);
    //
    // Form1
    //
    this.AutoScaleDimensions = new System.Drawing.SizeF(6F, 12F);
    this.AutoScaleMode = System.Windows.Forms.AutoScaleMode.Font;
    this.ClientSize = new System.Drawing.Size(361, 159);
    this.Controls.Add(this.button1);
    this.Controls.Add(this.textBox1);
    this.Controls.Add(this.label1);
    this.Name = "Form1";
    this.Text = "欢迎窗口";
    this.ResumeLayout(false);
```

```
            this.PerformLayout( );
        }
        #endregion
        private System.Windows.Forms.Label label1;
        private System.Windows.Forms.TextBox textBox1;
        private System.Windows.Forms.Button button1;
    }
}
```

（6）双击Button控件进入代码编辑窗口，输入以下代码：

```
private void button1_Click(object sender, EventArgs e)
    {
        MessageBox.Show("欢迎进入C#世界！"
            + textBox1.Text, "C#程序");
    }
```

（7）编译、运行程序。

案例3 存取命令行参数

案例运行效果与操作

在应用程序执行的时候如何去存取命令行的参数呢？C#中命令行参数的存取和C语言与C++语言的命令行参数的存取又有什么不同呢？在本节中我们用例子来说明。

本例是一个命令行应用程序，我们在命令行中输入：

```
CommandLineArguments "cat \"dog\"    elephant" lion "'tiger    puma'
```

程序运行的结果如图1-8所示。

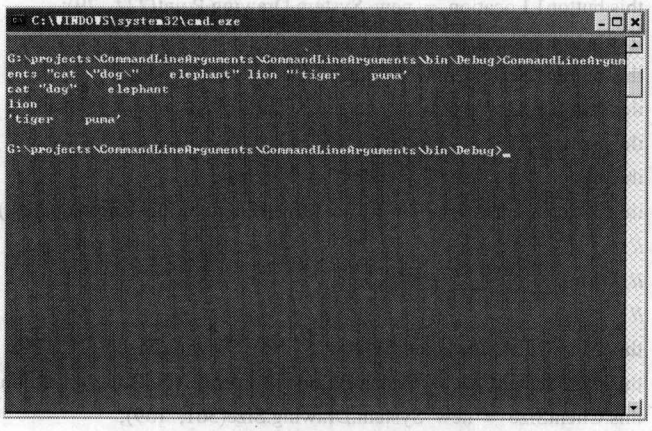


图1-8 存取命令行参数演示

知识要点

Main( )方法是程序的入口点，你将在那里创建对象和调用其他方法。一个C#程序中只能有一个入口点。该方法在类或结构的内部声明，它必须为静态方法，而不应为公共方法。