第 8 章　常见窗体控件的使用

本章要点

- 常见控件的使用
- 定制控件

Microsoft Visual Studio 2012 是新一代的可视化集成开发环境，所有的开发工具都被集成到一个 IDE(Integrated Development Environment)中，可以用 Visual C#创建 Windows 应用程序。

本章介绍的几乎所有的功能都是通过包含在 System.Windows.Forms 命名空间中的类来实现的。该命名空间包含了许多类和命名空间，它们都用于创建 Windows 应用程序，其中许多类都是从 System.Windows.Forms.Control 中派生而来的。

8.1 Windows 控件

8.1.1 Windows 窗体

使用 Visual Studio 2012 可以大大简化 Windows Forms 应用程序的编写，Visual Studio 2012 减少了开发人员花在界面框架上的编程时间，使开发人员可以集中精力去解决业务问题。

下面就简单介绍一下如何使用 Visual Studio 2012 来创建一个简单的 Windows Forms 应用程序。

1．创建空白窗体

创建空白窗体的操作如下。

(1) 在 Visual Studio 2012 开发环境中，选择“文件”→“新建”→“项目”命令，弹出【新建项目】对话框。

(2) 在左边选中【Visual C#】，右边选中【Windows 窗体应用程序】选项，然后在该对话框下方的【名称】文本框中，输入该项目的名称，如“MyFirstWindowsApplication”，在【位置】文本框中，输入保存该项目的位置，也可单击【浏览】按钮来选定保存位置，如图 8.1 所示。

(3) 单击【确定】按钮，在 Visual Studio 2012 的编辑窗口中将显示一个空白窗体，如图 8.2 所示。

2．设置窗体属性

上面创建了一个名为 Form1 的窗体，“Form1”出现在新建窗体的标题栏上。下面根据实际需要对窗体属性进行适当的设置。

(1) 在窗体上任意位置单击，选中要设置属性的窗体，如图 8.2 所示，窗体四周出现 3 个控点，可以拖动控点适当调整窗体大小。

(2) 选择“视图”→“属性窗口”命令，在 Visual Studio 2012 开发窗口右侧就会出现一个属性窗口，如图 8.3 所示。

(3) 在属性窗口中，列出了该窗体当前的各项属性，可以进行相应设置。不同的属性在属性窗口中的设置方式也有所不同，用户可以直接输入属性值(例如 Size、Location 和 Text 等属性)、从下拉列表中选择一个值(FormBorderStyle 和 StartPosition)或者执行更加复杂的操作。

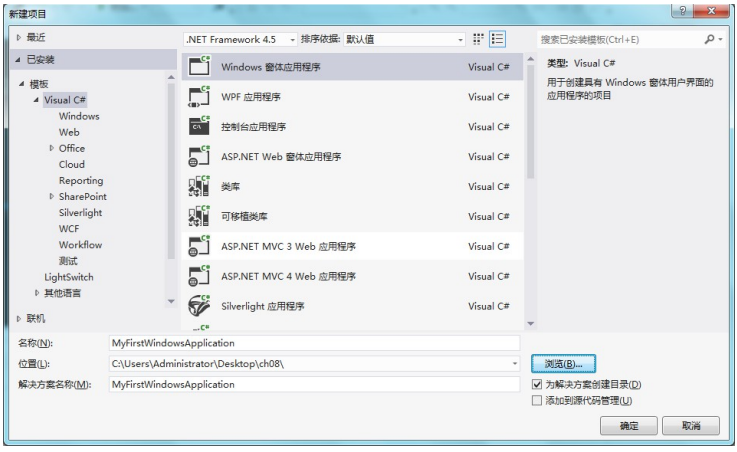


图 8.1 在【新建项目】对话框中创建 Windows 应用程序

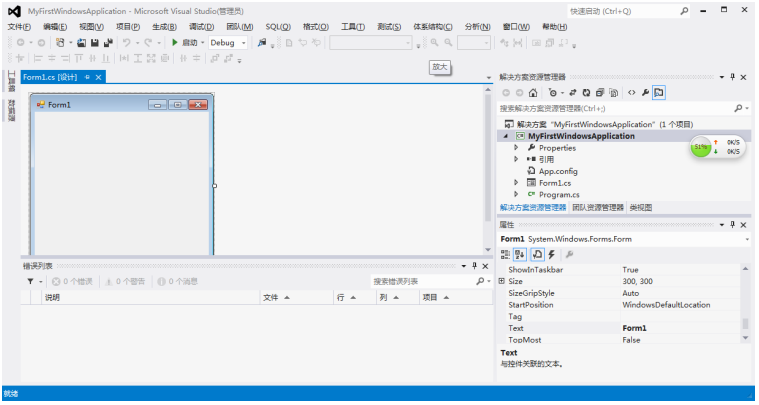


图 8.2 在 Visual Studio 2012 编辑器窗口中显示一个空白窗体

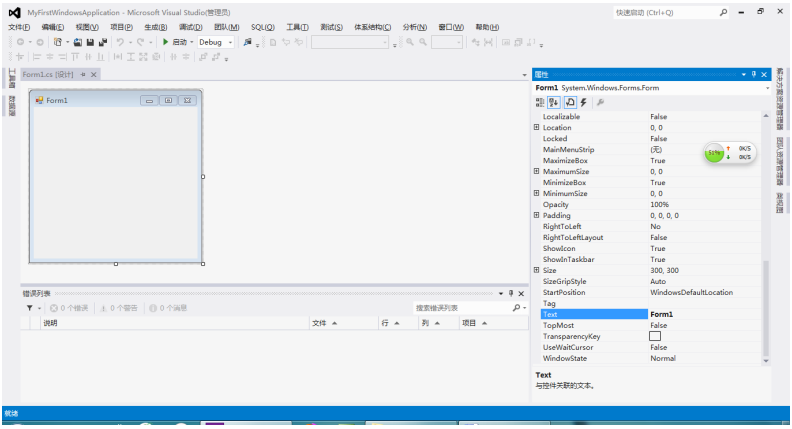


图 8.3　右侧出现【属性】窗口

8.1.2 控件的公有属性、事件和方法

.NET 中的大多数控件都派生于 System.Windows.Forms.Control 类。因此，在介绍其他各个控件之前，先来介绍一下 Control 这个类，Control 类实现了所有窗体交互控件的基本功能：处理用户键盘输入、处理消息驱动、限制控件大小等。

Control 类的属性、方法和事件是所有窗体控件所共有的，在程序设计过程中经常会遇到，所以充分了解 Control 类的成员可以为以后的窗体编程打下坚实的基础。

下面具体介绍 Control 类的各项成员。

1. Control 类的属性

所有的控件都有许多属性，用来处理控件的操作。大多数控件的基类 Control 有许多属性，其他控件要么直接继承了这些属性，要么就重写了这些属性，来提供特定的功能。

表 8.1 列出了 Control 类最常见的属性。这些属性在本章介绍的大多数控件中基本都有，所以后面就不再对它们进行详细的解释了。

表 8.1　Control 类的属性

名　称	说　明
AllowDrop	获取或设置一个值，该值指示控件是否可接受用户拖放到它上面的数据
Anchor	获取或设置控件绑定到的容器的边缘并确定控件如何随其父级一起调整大小
BackColor	获取或设置控件的背景色
BackgroundImage	获取或设置在控件中显示的背景图像
BindingContext	获取或设置控件的 BindingContext
Bottom	获取控件下边缘与其容器的工作区上边缘之间的距离(以像素为单位)
Bounds	获取或设置控件(包括其非工作区元素)相对于其父控件的大小和位置

	(以像素为单位)
CanFocus	获取一个值，该值指示控件是否可以接收焦点
CanSelect	获取一个值，该值指示是否可以选中控件
Capture	获取或设置一个值，该值指示控件是否已捕获鼠标
CausesValidation	获取或设置一个值，该值指示控件是否会引起在任何需要在接收焦点时执行验证的控件上执行验证
DataBindings	为该控件获取数据绑定
DefaultBackColor	获取控件的默认背景色
DefaultFont	获取控件的默认字体
DefaultForeColor	获取控件的默认前景色
Dock	获取或设置哪些控件边框停靠到其父控件并确定控件如何随其父级一起调整大小
Enabled	获取或设置一个值，该值指示控件是否可以对用户交互作出响应
Focused	获取一个值，该值指示控件是否有输入焦点
Font	获取或设置控件显示的文字的字体
ForeColor	获取或设置控件的前景色
Handle	获取控件绑定到的窗口句柄
HasChildren	获取一个值，该值指示控件是否包含一个或多个子控件
Height	获取或设置控件的高度
Left	获取或设置控件左边缘与其容器的工作区左边缘之间的距离(以像素为单位)
Location	获取或设置该控件的左上角相对于其容器的左上角的坐标
Margin	获取或设置控件之间的空间
MaximumSize	获取或设置大小，该大小是 GetPreferredSize 可以指定的上限
MinimumSize	获取或设置大小，该大小是 GetPreferredSize 可以指定的下限
MouseButtons	获取一个值，该值指示哪一个鼠标按钮处于按下的状态
MousePosition	获取鼠标光标的位置(以屏幕坐标表示)
Name	获取或设置控件的名称
Padding	获取或设置控件内的边距
Parent	获取或设置控件的父容器
Right	获取控件右边缘与其容器的工作区左边缘之间的距离(以像素为单位)
Size	获取或设置控件的高度和宽度
TabIndex	获取或设置在控件容器的控件的 Tab 键顺序
Text	获取或设置与此控件关联的文本
Top	获取或设置控件上边缘与其容器的工作区上边缘之间的距离(以像素为单位)
Visible	获取或设置一个值，该值指示是否显示该控件
Width	获取或设置控件的宽度

下面重点介绍几个编程中常用到的属性及其用法。

(1) Text 属性

每一个控件都有 Text 属性，是给用户查看或者输入的。Name 属性虽然也是每个控件对象都有的，不过它却是给程序员看的，常在编程中使用，作为每个控件的名字被程序员控制与操作。

Text 属性在很多控件中都是经常使用的。例如：在标签控件中显示的文字、在编辑框控件中用户输入的文字。

同样，在程序中也可以直接访问 Text 属性，用来获取和设置 Text 的值，这样就可以实现在程序运行过程中修改标题的名称，获取用户输入的数据等功能。

(2) Capture 属性

Capture 属性如果设为真，则不管鼠标是否在此控件的范围内，鼠标都被限定为只由此控件响应。

(3) Anchor 和 Dock 属性

在设计窗体时，这两个属性非常实用，.NET 中通过对这两个属性的设置，实现了用户改变窗口大小，却同时可确保窗口看起来不显得很乱。

Anchor 属性用于指定在用户重新设置窗口的大小时控件该如何响应。可以指定如果控件重新设置了自己的大小，就根据控件自己的边界锁定它，或者其大小不变，但应根据窗口的边界来锚定它的位置。

Dock 属性与 Anchor 属性是相关的。可以使用该属性指定控件应停放在容器的边框上。如果用户重新设置了窗口的大小，该控件将继续停放在窗口的边框上。

2. Control 类的方法

可以使用 Control 类的方法来获取或者设置控件的一些信息和状态。

表 8.2 列出了 Control 类最常见的方法及其功能。

表 8.2　Control 类的常见方法

名　称	说　明
Contains	检索一个值，该值指示指定控件是否为一个控件的子控件
CreateControl	强制创建控件，包括创建句柄和任何子控件
CreateGraphics	为控件创建 Graphics
Dispose	已重载。释放由 Control 使用的所有资源
DoDragDrop	开始拖放操作
DrawToBitmap	支持呈现到指定的位图
Equals	已重载。确定两个 Object 实例是否相等(从 Object 继承)
FindForm	检索控件所在的窗体
Focus	为控件设置输入焦点
FromChildHandle	检索包含指定句柄的控件
FromHandle	返回当前与指定句柄关联的控件

GetChildAtPoint	已重载。检索指定位置的子控件
GetNextControl	按照子控件的 Tab 键顺序向前或向后检索下一个控件
GetPreferredSize	检索可以容纳控件的矩形区域的大小
GetType	获取当前实例的 Type(从 Object 继承)
Hide	对用户隐藏控件
Invalidate	已重载。使控件的特定区域无效并向控件发送绘制消息
Invoke	已重载。在拥有此控件的基础窗口句柄的线程上执行代理
IsKeyLocked	确定 Caps Lock、Num Lock 或 Scroll Lock 键是否有效
PointToClient	将指定屏幕点的位置计算成工作区坐标
PointToScreen	将指定工作区点的位置计算成屏幕坐标
RectangleToClient	计算指定屏幕矩形的大小和位置(以工作区坐标表示)
RectangleToScreen	计算指定工作区矩形的大小和位置(以屏幕坐标表示)
Refresh	强制控件使其工作区无效并立即重绘自己和任何子控件
ResetText	将 Text 属性重置为其默认值
ResumeLayout	已重载。恢复正常的布局逻辑
Scale	已重载。缩放控件和任何子控件
Select	已重载。激活控件
SelectNextControl	激活下一个控件
SendToBack	将控件发送到 Z 顺序的后面
SetBounds	已重载。设置控件的边界
Show	向用户显示控件
SuspendLayout	临时挂起控件的布局逻辑
ToString	返回包含 Component 的名称的 String(如果有)。不应重写此方法
Update	使控件重绘其工作区内的无效区域

3. Control 类的事件

当用户进行某一个操作时，会引发某个事件的发生，此时就需要调用我们写好的事件处理程序代码，实现对程序的操作。在 Visual C#中，所有的机制都被封装在控件之中了，大大方便了编写事件驱动程序。表 8.3 是 Control 类的一些常见事件。

表 8.3 Control 类的常见事件

名 称	说 明
Click	在单击控件时发生
ClientSizeChanged	当 ClientSize 属性的值更改时发生
ContextMenuChanged	当 ContextMenu 属性的值更改时发生
ControlAdded	在将新控件添加到 Control.ControlCollection 时发生

DoubleClick	在双击控件时发生
DragOver	在将对象拖到控件的边界上发生
EnabledChanged	在 Enabled 属性值更改后发生
Enter	进入控件时发生
GotFocus	在控件接收焦点时发生
LostFocus	当控件失去焦点时发生
MouseCaptureChanged	当控件失去鼠标捕获时发生
MouseClick	在鼠标单击该控件时发生
MouseDoubleClick	当用鼠标双击控件时发生
MouseDown	当鼠标指针位于控件上并按下鼠标键时发生
MouseEnter	在鼠标指针进入控件时发生
MouseHover	在鼠标指针停放在控件上时发生
MouseLeave	在鼠标指针离开控件时发生
MouseMove	在鼠标指针移到控件上时发生
MouseUp	在鼠标指针在控件上并释放鼠标键时发生
MouseWheel	在移动鼠标轮并且控件有焦点时发生
Move	在移动控件时发生
SizeChanged	在 Size 属性值更改时发生
TextChanged	在 Text 属性值更改时发生
Validated	在控件完成验证时发生
Validating	在控件正在验证时发生

8.1.3 Button 控件

几乎所有的 Windows 对话框中都存在按钮控件，对于按钮的处理比较简单，通常是在窗体上添加控件，再双击它，给 Click 事件添加代码。

下面介绍 Button 控件的常用属性和事件。

Button 控件常见的属性见表 8.4，其余的属性若读者需要可以具体参考 MSDN。

表 8.4　Button 控件的常见属性

名　称	说　明
AutoSize	获取或设置一个值，该值指示控件是否基于其内容调整大小
BackColor	获取或设置控件的背景色
Enabled	获取或设置一个值，该值指示控件是否可以对用户交互作出响应
ForeColor	获取或设置控件的前景色
Image	获取或设置显示在按钮控件上的图像
Name	获取或设置控件的名称
Size	获取或设置控件的高度和宽度
Text	获取或设置按钮控件上的文本

Visible	获取或设置一个值，该值指示是否显示该控件

下面讲解一下 Button 控件的事件。

该控件最常用的事件就是 Click。只要用户在按钮上单击鼠标左键就会引发该事件。

下面的例子说明了如何使用 Button 控件，并如何定义 Click 事件。

例 8.1 Button 控件的 Click 事件演示。

创建程序的操作步骤如下。

(1) 在 Visual Studio 2012 开发环境中，选择“文件”→“新建”→“项目”命令，弹出“新建项目”对话框。

(2) 在左边选中“Visual C#”，右边选中“Windows 窗体应用程序”选项，然后在该对话框下方的“名称”文本框中，输入该项目的名称，如“ch08-1”，在“位置”文本框中，输入保存该项目的位置，也可单击“浏览”按钮来选定保存位置。

(3) 单击“确定”按钮，在 Visual Studio 2012 的编辑窗口中将显示一个空白窗体。

(4) 打开工具箱，双击 Button 控件一次，在属性面板上选择 Name 属性，改为“btnShow”，将 Text 属性改为“显示”，再双击 TextBox 控件一次，在属性面板上选择 Name 属性，改为“txtOuput”。效果如图 8.4 所示。

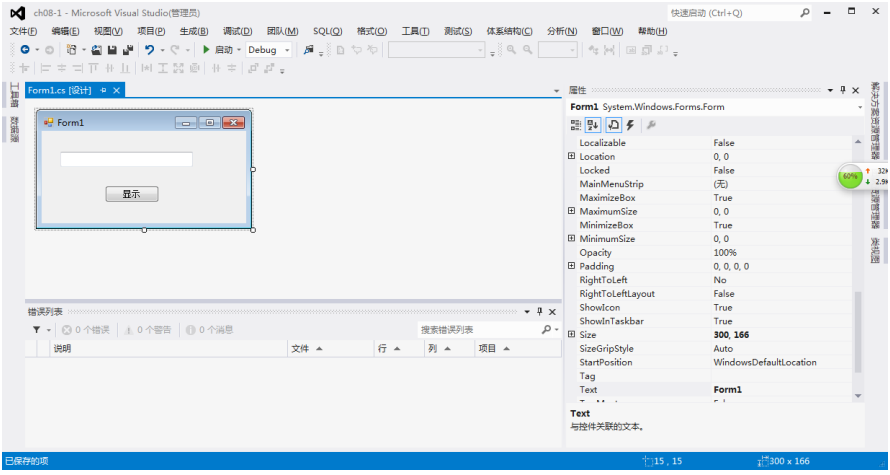


图 8.4 添加 Button 控件和 TextBox 文本框

(5) 双击该按钮控件，便可以为控件添加 Click 事件。或者在属性面板上单击事件图标，在事件列表中选中 Click 并双击，同样也可以添加 Click 事件。这样就自动添加了 Click 事件代码。

(6) 查看代码，在代码中找到以下内容。

```
private void btn_Example_Click(object sender, EventArgs e)
{
}
```

在花括号中输入 Click 事件要处理的代码：

```
txtOuput.Text = "欢迎进入 C#世界";
```

该代码的功能是在 TextBox 文本框中显示“欢迎进入 C#世界”。

程序代码：

```
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 ch08_1
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }

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

运行程序，效果如图 8.5 所示。

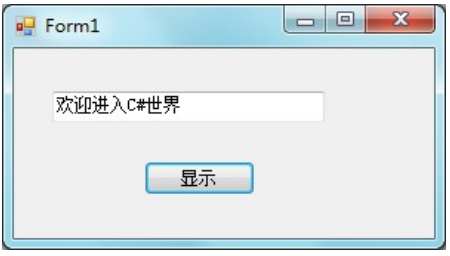


图 8.5　例 8.1 的运行结果

8.1.4 TextBox 控件

文本框(TextBox)经常用于获取用户输入或显示文本，通常用于可编辑文本，也可以设定其成为只读控件。文本框能够显示多行数据，并添加基本的格式设置。

Text 属性是文本框最重要的属性，要显示的文本就包含在 Text 属性中。Text 属性可以在设计窗口时使用属性窗口设置，也可以在运行时用代码设置

或者通过用户输入设置，同样也可以在运行时通过读取 Text 属性来获得文本框的当前内容。

(1) 属性

表 8.5 列出了 TextBox 控件的常用属性，完整的属性可以参考 MSDN。

表 8.5 TextBox 控件的常见属性

名 称	说 明
AutoSize	获取或设置一个值，该值指示当更改分配给控件的字体时，是否自动调整控件的高度。此属性与此类无关
BackColor	获取或设置控件的背景色
CausesValidation	获取或设置一个值，该值指示控件是否会引起在任何需要在接收焦点时执行验证的控件上执行验证
CharacterCasing	获取或设置 TextBox 控件是否在字符键入时修改其大小写格式
DataBindings	为该控件获取数据绑定
Enabled	获取或设置一个值，该值指示控件是否可以对用户交互作出响应
MaximumSize	获取或设置大小，该大小是 GetPreferredSize 可以指定的上限
MaxLength	获取或设置用户可在文本框控件中键入或粘贴的最大字符数
MinimumSize	获取或设置大小，该大小是 GetPreferredSize 可以指定的下限
Multiline	已重写。获取或设置一个值，该值指示此控件是否为多行 TextBox 控件
Name	获取或设置控件的名称
PasswordChar	获取或设置字符，该字符用于屏蔽单行 TextBox 控件中的密码字符
ReadOnly	获取或设置一个值，该值指示文本框中的文本是否为只读
ScrollBars	获取或设置哪些滚动条应出现在多行 TextBox 控件中
SelectedText	获取或设置一个值，该值指示控件中当前选定的文本
SelectionLength	获取或设置文本框中选定的字符数
SelectionStart	获取或设置文本框中选定的文本起始点
Text	已重写。获取或设置 TextBox 中的当前文本
TextAlign	获取或设置 TextBox 控件中文本的对齐方式
TextLength	获取控件中文本的长度
Visible	获取或设置一个值，该值指示是否显示该控件
WordWrap	指示多行文本框控件在必要时是否自动换行到下一行的开始

(2) 事件

TextBox 控件常用的事件是对文本控件中的文本进行有效性验证。如果要确保文本框中不输入无效的字符，或者只输入某个范围内的数值，就需要告诉控件的用户，输入的值是否有效。

TextBox 控件提供了表 8.6 的事件。

表 8.6 TextBox 控件的常用事件

名 称	说 明
Enter	进入控件时发生
GotFocus	在控件接收焦点时发生

Leave	在输入焦点离开控件时发生
Validating	在控件正在验证时发生
Validated	在控件完成验证时发生
LostFocus	当控件失去焦点时发生
KeyDown	在控件有焦点的情况下按下键时发生
KeyPress	在控件有焦点的情况下按下键时发生
KeyUp	在控件有焦点的情况下释放键时发生
TextChanged	在 Text 属性值更改时发生

下面是一个简单的关于文本框控件的例子。

例 8.2　演示 TextBox 控件的常见用法。

程序界面如图 8.6 所示。

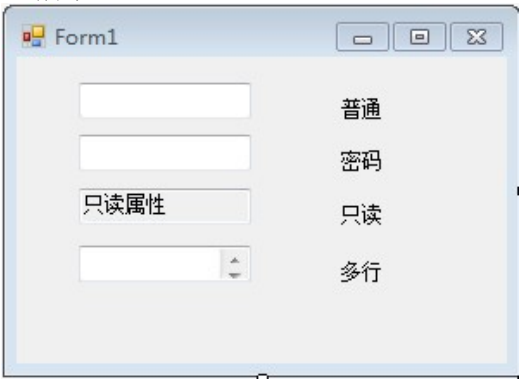


图 8.6　程序界面

新建一个 Windows 应用程序，而后再在窗体上放置 4 个 TextBox 控件以及 4 个 Label 控件(见图 8.6)。4 个 Label 控件从上到下，依次把他们的 Text 属性修改为“普通”、“密码”、“只读”和“多行”。4 个 TextBox 控件从上到下，他们的属性修改依次为：第一个不做任何修改；第二个把 Password 属性修改为“*”属性；第三个把 ReadOnly 属性修改为“True”，Text 属性修改为“只读属性”；第四个把 MultiLine 属性修改为“True”以及 ScrollBars 属性修改为“Vertical”。

运行程序，在每个 TextBox 中都试图填入一些内容，具体结果如图 8.7 所示。

从运行结果中可以看到，当对 TextBox 的 Password 属性进行设置后，在其中输入文本内容时，会出现自动隐藏，以某种符号代替之，这在密码输入栏中是必须的。MultiLine 属性设置为“True”则保证了文本框可以接收多行输入。ReadOnly 属性则令文本框无法进行内容的修改，只可以读和复制。

图 8.7 例 8.2 的运行结果

8.1.5 RadioButton 控件和 CheckBox 控件

单选按钮(RadioButton)通常成组出现，用于为用户提供两个或多个互相排斥的选项，如图 8.8 所示。单选按钮和将要介绍的复选框(CheckBox)控件类似，但也存在重要的区别，即从一组单选按钮中必须且只能选择一个，而在一组复选框中可以同时选择多个选项。

要把单选按钮组合在一起，使它们组成一个逻辑单元，必须使用 GroupBox 控件。首先在窗体上拖放一个 GroupBox 控件(组框)，再把需要的 RadioButton 按钮放在分组框的边界内，RadioButton 按钮知道如何改变自己的状态，以反映分组框中唯一被选中的选项。

复选框(CheckBox)指示某特定条件是打开的还是关闭的。当用户希望选择一个或多个选项时，就需要使用复选框。一个复选框如图 8.9 所示。

图 8.8 单选按钮

图 8.9 复选框

(1) RadioButton 控件的属性

表 8.7 只是列出了 RadioButton 控件的常用属性，完整的属性可以参考 MSDN。

表 8.7 RadioButton 控件的常用属性

属性名称	说 明
Appearance	获取或设置一个值，该值用于确定 RadioButton 的外观
AutoCheck	获取或设置一个值，它指示：在单击控件时，Checked 值和控件的外观是否自动更改
CheckAlign	获取或设置 RadioButton 的复选框部分的位置
Checked	获取或设置一个值，该值指示是否已选中控件
Enabled	获取或设置一个值，该值指示控件是否可以对用户交互作出响应

FlatStyle	获取或设置按钮控件的平面样式外观
Name	获取或设置控件的名称
Text	与控件关联的文本
TextAlign	获取或设置 RadioButton 控件上的文本对齐方式

(2) RadioButton 控件的事件

表 8.8 只是列出了 RadioButton 控件的常用事件，完整的事件可以参考 MSDN。

表 8.8　RadioButton 控件的常用事件

事件名称	说　明
Click	在单击控件时发生
CheckedChanged	当 Checked 属性的值更改时发生

(3) CheckBox 控件的属性

这个控件的属性和事件非常类似于 RadioButton 控件的属性和事件，但有两个新的属性，如表 8.9 所示。

表 8.9　CheckBox 控件的属性

属性名称	说　明
CheckState	获取或设置 CheckBox 的状态
ThreeState	获取或设置一个值，该值指示此 CheckBox 是否允许三种复选状态而不是两种

(4) CheckBox 控件事件

表 8.10 只是列出了 CheckBox 控件的常用事件，完整的事件可以参考 MSDN。

表 8.10　CheckBox 控件的常用事件

事件名称	说　明
CheckedChanged	当 Checked 属性的值更改时发生
CheckStateChanged	当 CheckState 属性的值更改时发生
Click	在单击控件时发生

下面举一个简单的关于学生注册的例子，其中有关于 RadioButton 控件以及 CheckBox 控件的使用。

例 8.3　关于 RadioButton 控件以及 CheckBox 控件的使用。

程序界面如图 8.10 所示。

在本例中，通过在文本框中输入学生的姓名以及通过对 RadioButton 和 CheckBox 控件的选择，把最终的结果输出到最下面的 TextBox 文本框中，该文本框的 MutiLine 属性设置为“True”。

图8.10 程序界面

程序代码：

```
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 ch08_3
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }

        private void btnRegister_Click(object sender, EventArgs
e)
        {
            string sex="";
            string hobby="";
            if (rdoMale.Checked)
                sex = "男";
            else
                sex = "女";
            if (ckSing.Checked)
                hobby += "唱歌 ";
            if (ckDance.Checked)
                hobby += "跳舞 ";
```

```
            if (ckDraw.Checked)
                hobby += "画画 ";
            if (ckWrite.Checked)
                hobby += "书法";
            txtOutput.Text = "学号：" + txtSno.Text + "\r\n姓
名：" + txtName.Text + "\r\n性别：" + sex + "\r\n爱好：" +
hobby;
        }
    }
}
```

程序运行的结果如图 8.11 所示。

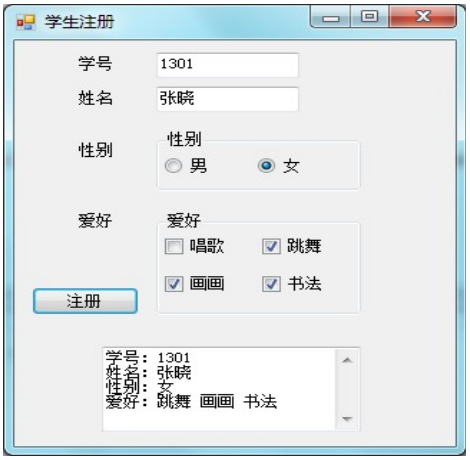


图 8.11　例 8.3 的运行结果

8.1.6 ListBox 控件

列表框用于显示一组字符串，可以一次从中选择一个或多个选项。例，在设计期间，如果不知道用户要选择的数值个数，或者列表中的值非常多时候，就应考虑使用列表框。

(1) 属性

表 8.11 列出了 ListBox 控件的常用属性。

表 8.11　ListBox 控件的常用属性

名　称	说　明
DataBindings	为该控件获取数据绑定
DataSource	获取或设置此 ListControl 的数据源
Items	获取 ListBox 的项
Name	获取或设置控件的名称
SelectedIndex	获取或设置 ListBox 中当前选定项的从零开始的索引
SelectedItem	获取或设置 ListBox 中的当前选定项
SelectedItems	获取包含 ListBox 中当前选定项的集合
SelectedValue	获取或设置由 ValueMember 属性指定的成员属性的值

SelectionMode	获取或设置在 ListBox 中选择项所用的方法
Sorted	获取或设置一个值，该值指示 ListBox 中的项是否按字母顺序排序
Text	获取或搜索 ListBox 中当前选定项的文本

(2) 方法

ListBox 控件提供许多实用的方法，可以使我们更方便地操作一个列表框。表 8.12 列出了最常用的方法。

表 8.12 ListBox 控件的方法

方法名称	说 明
ClearSelected	取消选择 ListBox 中的所有项
FindString	查找 ListBox 中以指定字符串开始的第一个项
FindStringExact	查找 ListBox 中第一个精确匹配指定字符串的项
GetItemText	返回指定项的文本表示形式
GetSelected	返回一个值，该值指示是否选定了指定的项
SetSelected	选择或清除对 ListBox 中指定项的选定
ToString	返回 ListBox 的字符串表示形式

(3) 事件

通常，ListBox 使用的事件与用户选中的选项有关，如表 8.13 所示。

表 8.13 ListBox 控件的事件

名 称	说 明
TextChanged	当 Text 属性更改时发生
SelectedIndexChanged	在 SelectedIndex 属性更改后发生

8.1.7 ComboBox 控件

与 ListBox 不同，组合框(ComboBox)从来都不能在列表中选择多个选项，但可以在 ComboBox 的 TextBox 部分输入新选项。

通常，组合框比 ListBox 节省空间，因为组合框中可见的部分只有文本框和按钮部分。

(1) 属性

ComboBox 控件包含 TextBox 和 ListBox 控件的功能，所以 ComboBox 有许多属性，这里也只列出最常见的属性，如表 8.14 所示。

表 8.14 ComboBox 控件的属性

名 称	说 明
DropDownStyle	获取或设置指定组合框样式的值
DroppedDown	获取或设置一个值，该值指示组合框是否正在显示其下拉部分
Items	获取一个对象，该对象表示该 ComboBox 中所包含项的集合
MaxDropDownItems	获取或设置要在 ComboBox 的下拉部分中显示的最大项数

MaxLength	获取或设置组合框可编辑部分中最多允许的字符数
Name	获取或设置控件的名称
SelectedIndex	获取或设置指定当前选定项的索引
SelectedItem	获取或设置 ComboBox 中当前选定的项
SelectedText	获取或设置 ComboBox 的可编辑部分中选定的文本
SelectedValue	获取或设置由 ValueMember 属性指定的成员属性的值
SelectionLength	获取或设置组合框可编辑部分中选定的字符数
SelectionStart	获取或设置组合框中选定文本的起始索引
Sorted	获取或设置指示是否对组合框中的项进行了排序的值
Text	获取或设置与此控件关联的文本

可以把 ComboBox 控件看作是结合了 TextBox、Button 以及 ListBox 功能的控件。从 ComboBox 控件的中文名称就可以看出，该控件组合了很多功能的控件。

ComboBox 控件的 DropDownStyle 属性可以进行设置，不同的设置会呈现不同的样式，具体如下。

- Simple：使得 ComboBox 的列表部分总是可见的，如图 8.12 所示。
- DropDown：这个是默认值，使得用户可以编辑 ComboBox 控件的文本框部分，必须单击右侧的箭头才可以显示列表部分，如图 8.13 所示。
- DropDownList：外观与 DropDown 的一样，不同的是用户不能编辑 ComboBox 控件的文本框部分，如图 8.14 所示。

(2) 事件

ComboBox 处理的事件主要涉及到选项的改变、下拉状态的改变、文本的改变这 3 个操作，相应的事件如表 8.15 所示。

图 8.12　Simple

图 8.13　DropDown

图 8.14　DropDownList

表 8.15　ComboBox 控件的事件列表

名　称	说　明
DropDown	当显示 ComboBox 的下拉部分时发生
SelectedIndexChanged	在 SelectedIndex 属性更改后发生

SelectedValueChanged	当 SelectedValue 属性更改时发生
KeyDown	在控件有焦点的情况下按下键时发生
KeyPress	在控件有焦点的情况下按下键时发生
KeyUp	在控件有焦点的情况下释放键时发生
TextChanged	在 Text 属性值更改时发生

例 8.4 关于 ComboBox 操作。

程序界面如图 8.15 所示。

图 8.15 程序界面

本例是要在左边的 ComboBox 中选择一个院系名称，在右边的 ComboBox 中自动添加一些被选择的院系的班级名称。

程序代码：

```
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 ch08_4
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }

        private void Form1_Load(object sender, EventArgs e)
        {
            cboGrade.Items.Add("软件与服务外包学院");
            cboGrade.Items.Add("现代港口与物流管理系");
        }

        private void cboGrade_SelectedIndexChanged(object
```

```
sender, EventArgs e)
        {
            switch (cboGrade.SelectedIndex)
            {
                case 0:
                    cboClass.Items.Clear();
                    cboClass.Items.Add("软件 1211");
                    cboClass.Items.Add("软件 1212");
                    cboClass.Items.Add("网络 1211");
                    break;
                case 1:
                    cboClass.Items.Clear();
                    cboClass.Items.Add("会计 1211");
                    cboClass.Items.Add("会计 1212");
                    cboClass.Items.Add("报关 1211");
                    break;
            }
        }
    }
}
```

程序的运行结果如图 8.16 所示。

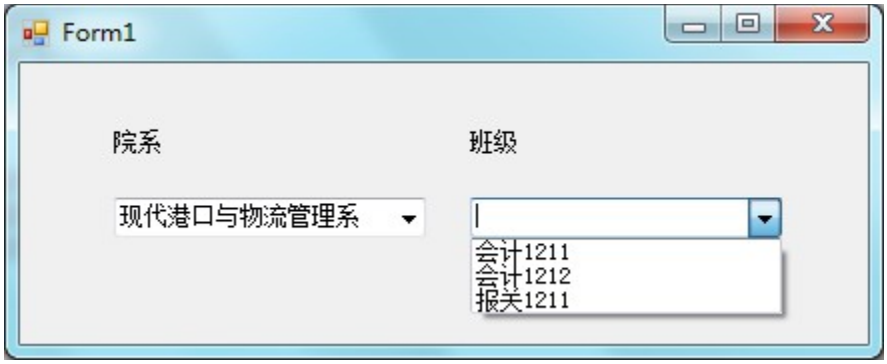


图 8.16　例 8.4 的运行结果

8.1.8 ListView 控件

ListView 是 Windows 列表视图控件，用于显示来自应用程序、数据库或文本文件的信息或者获取来自用户的信息。在标准列表视图对话框中可以进行各种查看操作，如：图标、详细视图等。

列表视图通常用于显示数据，用户可以对这些数据和显示方式进行某些控件，可以把包含在控件中的数据显示为列和行，或者显示为一列，或者先是为图标形式。

ListView 控件的主要属性就是 Items，该属性是一个包含控件所显示的项的集合，可用于在列表视图中的添加和移除项。SelectedItems 属性则包含控件中当前选定项的集合。如果将 MultiSelect 属性设置为“True”，用户

就可以同时选择多项。ListViewItem 类用于表示列表视图中的项，这些项可以包含子项，子项包含与父项相关的信息。

在应用程序中，我们经常使用方法和事件为列表视图提供附加功能。BeginUpdate 和 EndUpdate 方法用于为列表视图添加许多项，而且在每次添加项时并不显示控件的重新绘制，这样就提高了性能。

(1) 属性

ListView 的常用属性如表 8.16 所示。

表 8.16 ListView 控件的属性

名 称	说 明
Activation	获取或设置用户激活某个项必须要执行的操作的类型
Alignment	获取或设置控件中项的对齐方式
AllowColumnReorder	获取或设置一个值，该值指示用户是否可拖动列标头来对控件中的列重新排序
AutoArrange	获取或设置图标是否自动进行排列
CheckBoxes	获取或设置一个值，该值指示控件中各项的旁边是否显示复选框
CheckedIndices	获取控件中当前选中项的索引
CheckedItems	获取控件中当前选中的项
Columns	获取控件中显示的所有列标头的集合
FocusedItem	获取当前具有焦点的控件中的项
FullRowSelect	获取或设置一个值，该值指示单击某项是否选择其所有子项
GridLines	获取或设置一个值，该值指示：在包含控件中项及其子项的行和列之间是否显示网格线
HeaderStyle	获取或设置列标头样式
HoverSelection	获取或设置一个值，该值指示当鼠标指针在项上停留几秒钟时是否自动选定该项
Items	获取包含控件中所有项的集合
LabelEdit	获取或设置一个值，该值指示用户是否可以编辑控件中项的标签
LabelWrap	获取或设置一个值，该值指示当项作为图标在控件中显示时，项标签是否换行
LargeImageList	获取或设置当项以大图标在控件中显示时使用的 ImageList
MultiSelect	获取或设置一个值，该值指示是否可以选择多个项
Scrollable	获取或设置一个值，该值指示在没有足够空间来显示所有项时，是否给滚动条添加控件
SelectedIndices	获取控件中选定项的索引
SelectedItems	获取在控件中选定的项
SmallImageList	获取或设置 ImageList，当项在控件中显示为小图标时使用
Sorting	获取或设置控件中项的排序顺序
StateImageList	获取或设置与控件中应用程序定义的状态相关的 ImageList

TopItem	获取或设置控件中的第一个可见项
View	获取或设置项在控件中的显示方式

(2) 方法

ListView 控件常用的方法如表 8.17 所示。

表 8.17　ListView 控件的方法

名　称	说　明
BeginUpdate	避免在调用 EndUpdate 方法之前描述控件
Clear	从控件中移除所有项和列
EndUpdate	在 BeginUpdate 方法挂起描述后，继续描述列表视图控件
EnsureVisible	确保指定项在控件中是可见的，必要时滚动控件的内容
GetItemAt	检索位于指定位置的项

(3) 事件

ListView 控件常用的事件如表 8.18 所示。

表 8.18　ListView 控件常用的事件

名　称	说　明
AfterLabelEdit	当用户编辑项的标签时发生
BeforeLabelEdit	当用户开始编辑项的标签时发生
ColumnClick	当用户在列表视图控件中单击列标头时发生
ItemActivate	当激活项时发生

例 8.5 ListView 控件是本章比较复杂的一个控件，这里编写一个范例来说明 ListView 控件的使用方法，在窗体设计器中添加列表视图可以按照如下的步骤进行操作。

(1) 在工具箱中选中 ListView 控件，并拖动到窗体中适当位置。

(2) 单击列表视图图标，在属性窗口中将 View 属性设置为 Details。

(3) 单击属性窗口中 Columns 后的…按钮，打开【ColumnHeader 集合编辑器】对话框，如图 8.17 所示。在该窗口中单击【添加】按钮添加一个成员，并在属性窗口中设置其 Text 属性为“书名”。用同样的方法再添加两个成员，其 Text 属性分别为“作者”和“单价”。最后单击【确定】按钮回到设计器。

(4) 单击属性窗口中的 Items 后的…按钮，打开【ListViewItem 集合编辑器】对话框，如图 8.18 所示。在该窗口中单击【添加】按钮添加一个成员，并在“属性”窗口中设置其 Text 属性为“C#程序设计项目化教程”。

(5) 单击【数据】组中 SubItems 后的…按钮，打开【ListViewSubItem 集合编辑器】对话框，如图 8.19 所示。在该窗口中单击【添加】按钮添加两个成员，其 Text 属性分别为“郑广成”、“26”。最后单击【确定】按钮回到【ListViewItem 集合编辑器】对话框。这时“C#程序设计项目化教程”的数据已输入完成。

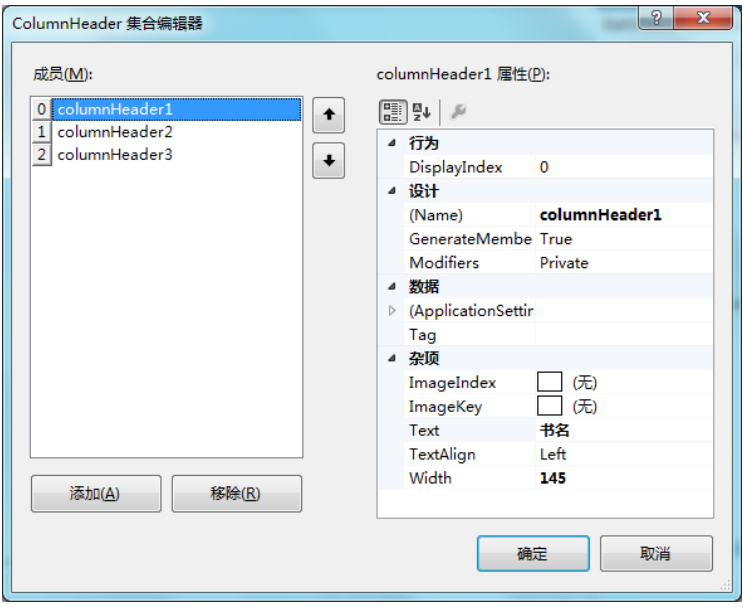


图 8.17 【ColumnHeader 集合编辑器】对话框

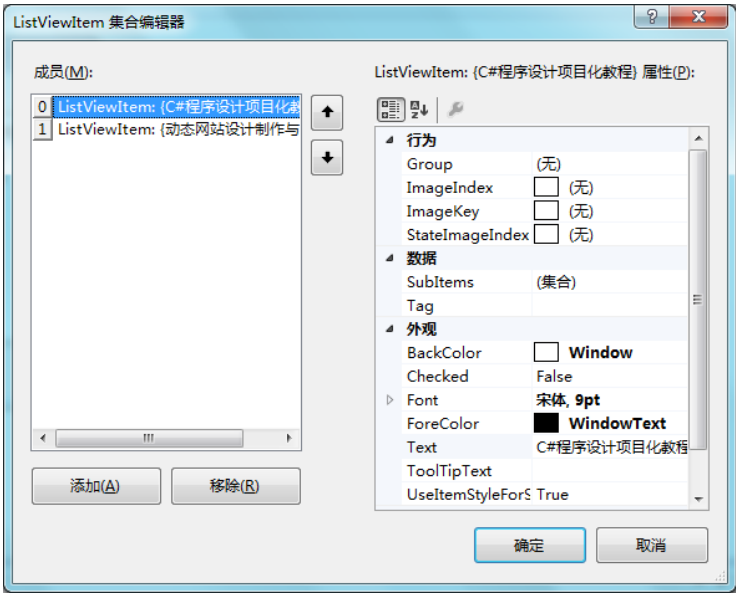


图 8.18 【ListViewItem 集合编辑器】对话框

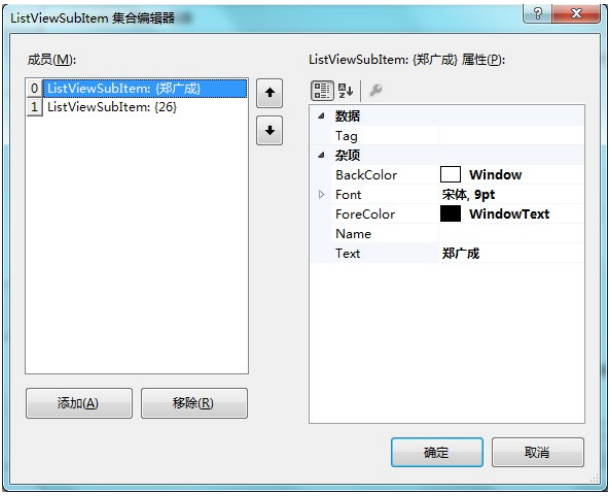


图 8.19　【ListViewSubItem 集合编辑器】对话框

(6) 用同样的方法再添加一个成员，最后单击【确定】按钮回到设计器。

(7) 按 F5 键运行应用程序，Form1 窗口出现，其中显示的列表视图如图 8.20 所示。

图 8.20　例 8.5 的运行结果

8.1.9 ToolStrip 控件

ToolStrip 控件是可以在 Windows 窗体应用程序中承载菜单、控件和用户控件的工具条。ToolStrip 控件提供丰富的设计时体验，包括就地激活和编辑、自定义布局、漂浮(即工具栏共享水平或垂直空间的能力)。尽管 ToolStrip 替换了早期版本的控件并添加了功能，但是仍可以在需要时选择保留 ToolBar 以备向后兼容和将来使用。

ToolStrip 控件有如下的功能：

- 创建易于自定义的常用工具栏，让这些工具栏支持高级用户界面和布局功能，如停靠、漂浮、带文本和图像的按钮、下拉按钮和控件、“溢出”按钮和 ToolStrip 项的运行时重新排序。

- 支持操作系统的典型外观和行为。
- 对所有容器和包含项进行事件的一致性处理，处理方式与其他控件的事件相同。
- 可以将项从一个 ToolStrip 拖到另一个 ToolStrip 内。
- 使用 ToolStripDropDown 中的高级布局创建下拉控件及用户界面类型编辑器。

ToolStrip 控件是高度可配置的、可扩展的控件，它提供了许多属性、方法和事件，可用来自定义外观和行为。以下是一些值得注意的成员。

(1) 属性

ToolStrip 的常用属性如表 8.19 所示。

表 8.19 ToolStrip 的常用属性

名 称	说 明
ContextMenuStrip	获取或设置与此控件关联的 ContextMenuStrip
ImageList	获取或设置包含 ToolStrip 项上显示的图像的图像列表
Items	获取属于 ToolStrip 的所有项
Name	获取或设置控件的名称
Stretch	获取或设置一个值，该值指示 ToolStrip 在 ToolStripContainer 中是否从一端拉伸到另一端
Text	获取或设置与此控件关联的文本

(2) 事件

ToolStrip 的常用事件如表 8.20 所示。

表 8.20 ToolStrip 的常用事件

名 称	说 明
BeginDrag	当用户开始拖动 ToolStrip 控件时发生
Click	在单击控件时发生
ContextMenuChanged	当 ContextMenu 属性的值更改时发生
ContextMenuStripChanged	当 ContextMenuStrip 属性的值更改时发生
ItemAdded	当向 ToolStripItemCollection 添加新的 ToolStripItem 时发生
ItemClicked	在单击 ToolStripItem 时发生
ItemRemoved	当从 ToolStripItemCollection 中移除 ToolStripItem 时发生
KeyDown	在控件有焦点的情况下按下键时发生
KeyPress	在控件有焦点的情况下按下键时发生
KeyUp	在控件有焦点的情况下释放键时发生
MouseClick	在鼠标单击该控件时发生
MouseDown	当鼠标指针位于控件上并按下鼠标键时发生

8.1.10 StatusStrip 控件

Windows 窗体 StatusStrip 控件在窗体中作为一个区域使用，此区域通常显示在窗口底部，应用程序可以在这里显示各种状态信息。

StatusStrip 控件上通常有 ToolStripStatusLabel 控件，用于显示指示状态的文本或图标，或者有可以用图形显示进程完成状态的 ToolStripProgressBar。

StatusStrip 控件可以显示正在 Form 上查看的对象的相关信息、对象的组件或与该对象在应用程序中的操作相关的上下文信息。通常，StatusStrip 控件由 ToolStripStatusLabel 对象组成，每个这样的对象都可以显示文本、图标或同时显示这二者。StatusStrip 还可以包含 ToolStripDropDownButton、ToolStripSplitButton 和 ToolStripProgressBar 控件。

可以使用“StatusStrip 项集合编辑器”来添加、移除和重新排序 StatusStrip 的 ToolStripItem 控件，以及查看和设置 StatusStrip 及 ToolStripItem 属性。

可以通过下列方法显示“StatusStrip 项集合编辑器”：

- 在设计器中右击 StatusStrip 控件并从快捷菜单中选择“编辑项”命令。
- 在设计器中单击 StatusStrip 控件上的智能标记，并从“StatusStrip 任务”对话框中选择“编辑项”。

在“StatusStrip 项集合编辑器”中可添加显示在下拉列表中的 ToolStripItem，即一个或多个下列控件：ToolStripStatusLabel、ToolStripProgressBar、ToolStripDropDownButton、ToolStripSplitButton。

下面介绍 StatusStrip 控件的常用属性和事件。

(1) StatusStrip 控件的属性

StatusStrip 的常用属性如表 8.21 所示。

表 8.21　StatusStrip 的常用属性

名　称	说　明
Anchor	获取或设置 ToolStrip 要绑定到的容器的边缘，并确定 ToolStrip 如何随其父级调整大小
ImageList	获取或设置包含 ToolStrip 项上显示的图像的图像列表
Items	获取属于 ToolStrip 的所有项
Name	获取或设置控件的名称
Stretch	获取或设置一个值，指示 StatusStrip 是否在其容器中从一端拉伸到另一端
Text	获取或设置与此控件关联的文本

(2) StatusStrip 控件的事件

StatusStrip 的常用事件如表 8.22 所示。

表 8.22 StatusStrip 控件的事件

名 称	说 明
BeginDrag	当用户开始拖动 ToolStrip 控件时发生
Click	在单击控件时发生
ItemAdded	当向 ToolStripItemCollection 添加新的 ToolStripItem 时发生
ItemClicked	在单击 ToolStripItem 时发生
ItemRemoved	当从 ToolStripItemCollection 中移除 ToolStripItem 时发生
KeyDown	在控件有焦点的情况下按下键时发生
KeyPress	在控件有焦点的情况下按下键时发生
KeyUp	在控件有焦点的情况下释放键时发生

8.1.11 MenuStrip 控件

MenuStrip 控件是此版本的 Visual Studio 和.NET Framework 4.5 中的新功能。使用该控件，可以轻松创建类似于 Microsoft Office 软件中那样的菜单。

MenuStrip 控件支持多文档界面(MDI)和菜单合并、工具提示和溢出。可以通过添加访问键、快捷键、选中标记、图像和分隔条，来增强菜单的可用性和可读性。

MenuStrip 控件的使用特点如下：

- 可创建支持高级用户界面和布局功能的易自定义的常用菜单，例如文本和图像排序和对齐、拖放操作、MDI、溢出和访问菜单命令的其他模式。
- 支持操作系统的典型外观和行为。
- 可以对所有容器和包含的项进行事件的一致性处理，处理方式与其他控件的事件相同。

例 8.6 下面新建一个菜单具体演示一下 MenuStrip 的使用方法。

(1) 新建一个 Windows 窗体应用程序，而后再在工具箱中双击 MenuStrip 控件或者是拖动此控件到窗体上，如图 8.21 所示。

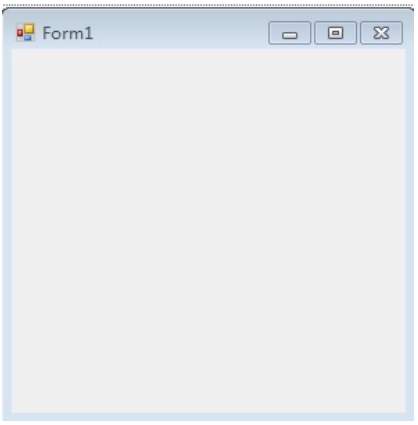


图 8.21　程序界面

(2) 单击 MenuStrip 上的“请在此输入”，就可以输入菜单文本了，MenuStrip 还将会产生下一菜单条目的提示输入。如图 8.22 所示。

图 8.22　新建 MenuStrip 示意图

(3) 右键单击 MenuStrip 控件，还可以进行分隔符等的插入，如图 8.23 所示。

(4) 还可以给菜单栏目添加图片，以方便识别和显得美观，具体操作为用右键单击 MenuStrip 控件，而后选择设置图像命令。如图 8.24 所示。

最终的结果如图 8.25 所示。

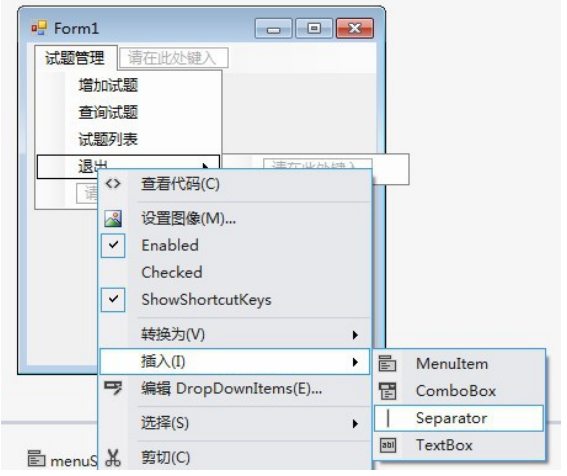
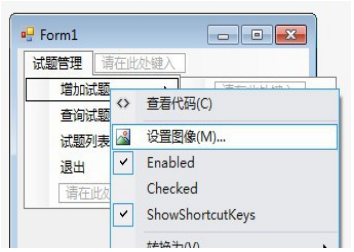


图 8.23　在 MenuStrip 上插入分隔符



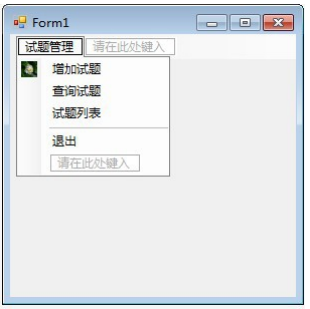


图 8.24 设置 MenuItem 的属性　　图 8.25　设置后的结果

表 8.23 显示了 MenuStrip 控件以及其关联类的一些特别重要的属性。

表 8.23　MenuStrip 控件的属性

属　性	说　明
MdiWindowListItem	获取或设置用于显示 MDI 子窗体列表的 ToolStripMenuItem
System.Windows.Forms.ToolStripItem.MergeAction	获取或设置 MDI 应用程序中子菜单与父菜单合并的方式
System.Windows.Forms.ToolStripItem.MergeIndex	获取或设置 MDI 应用程序的菜单中合并项的位置
System.Windows.Forms.Form.IsMdiContainer	获取或设置一个值，该值指示窗体是否为 MDI 子窗体的容器
ShowItemToolTips	获取或设置一个值，该值指示是否为 MenuStrip 显示工具提示
CanOverflow	获取或设置一个值，该值指示 MenuStrip 是否支持溢出功能
ShortcutKeys	获取或设置与 ToolStripMenuItem 关联的快捷键
ShowShortcutKeys	获取或设置一个值，该值指示与 ToolStripMenuItem 关联的快捷键是否显示在 ToolStripMenuItem 旁边

8.2 用户自定义控件

8.2.1 用户自定义控件概述

虽然 Visual Studio 2012 附带了大量的控件，但仍不能满足各个应用程序的特殊需要。比如说，Visual Studio 2012 自带的控件不能以我们希望的方式绘制自己，或者控件只能以某种方式使用，而我们却希望把控件的功能和界面一起封装，或者需要的控件不存在。此时，就需要自己开发一个新的控件。自定义控件的基本思想是允许开发人员生成新的功能，把现有的控件

聚集到一个公共控件上，使之可以在应用程序中重复使用，或通过组织在多个应用程序中重复使用。为此，Microsoft 提供了创建满足需要的控件方式。Visual Studio 2012 提供了一个工程类型 Windows Control Library，使用它可以创建自己的控件。

根据实际需要可以用以下 3 种方法来开发定制控件。

1. 从 Windows 窗体控件继承

开发人员可以从现有的 Windows 窗体控件继承出新的控件。这种定制方式可以保留 Windows 窗体控件所有的功能，然后根据需要添加自定义属性、方法或事件来扩展这些固定的功能。甚至在某些控件中，开发人员还可以重写基类的 OnPaint()事件，将自定义外观添加到控件的图形界面上。

当我们只需要在 Windows 窗体控件的基础上扩展一些功能，或者想为现有控件设计一个新的图形前端时，就可以从 Windows 窗体控件继承，创建定制的控件。

2. 从 UserControl 类继承

用户控件是封装在公共容器内的 Windows 窗体控件的集合。该容器包含与每个 Windows 窗体控件相关联的所有固有功能，允许开发者有选择地公开和绑定它们的属性。

当需要将若干个 Windows 窗体控件的功能合成一个可重新使用的新控件时，也可以从 UserControl 类继承，来创建定制的控件。

大多数定制控件都是继承了 System.Windows.Forms.UserControl 类。这个类包含相应的功能，为开发人员提供保存控件、提供设计界面。它比较类似于基类 Form，它提供了基本的执行方式，定制的派生类继承了这些业务功能。

3. 从 Control 类继承

当想要自定义控件的图形化表示形式，或者需要实现无法从标准控件获得的自定义功能时，开发人员就需要从 Control 类继承，来创建定制控件。Control 类提供了控件所需的所有基本功能，但不提供控件特定的功能和图形界面。从 Control 类继承来创建控件要复杂得多，用户必须为控件的 OnPaint()事件以及所需的任何功能编写代码，同时也允许用户根据自己的需要，灵活地调整控件。

控件是针对特定目的创建的，创建控件实际上也是一种编程任务，控件创建过程一般包括下列几个步骤。

(1) 确定控件要实现的目标。

(2) 确定所需要的控件类型。

(3) 将功能表示为控件及其子对象的属性、方法和事件，并指派相应的访问级别。

(4) 若控件需要自定义绘制，则为其添加对应的代码。

(5) 创建一个新的项目，对控件进行测试和调试。
(6) 在添加每个功能时，将控件添加到测试项目以试验新功能。
(7) 重复操作，改进设计。
(8) 打包和发布控件。

8.2.2 定制控件示例

本部分主要介绍前两种控件定制方式，第三种从头开始设计和绘制定制控件超出了本书的范围。在下面的内容中，我们将通过两个范例来分别说明如何通过从 Windows 窗体控件继承和从 UserControl 类继承这两种定制控件方式来创建新的控件。

1. 从 Windows 窗体控件继承

如果要扩展现有控件的功能，开发人员可以通过继承创建由现有控件导出的控件。通过这种方式，我们不仅可以保留标准的 Windows 窗体控件的所有固有的功能和可视属性，还可以加入自定义功能。

例 8.7 下面通过创建从现有 Windows 窗体控件继承的控件，介绍定制控件的一般过程。本例创建一个名为 ValueButton 的简单控件，该控件将继承标准 Windows 窗体按钮的功能，并公开一个名为 ButtonValue 的自定义属性。

(1) 在 Visual Studio 2012 中创建一个新的 C#工程，选择“Windows 控件库”，把新工程命名为 ValueButtonLib，如图 8.26 所示。

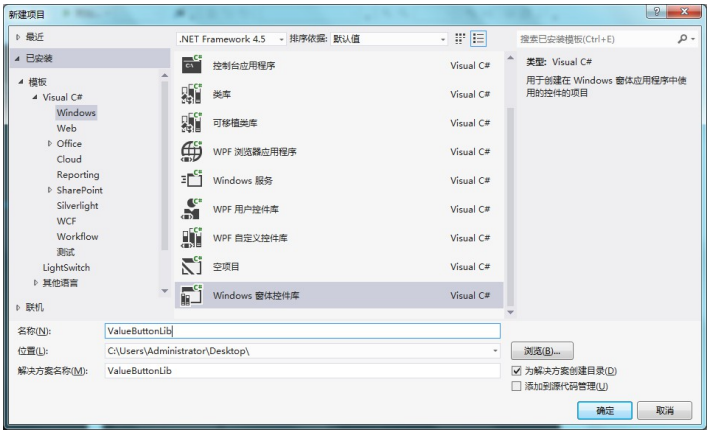


图 8.26 新建项目窗口

单击【确定】按钮，进入如图 8.27 所示的控件设计器。

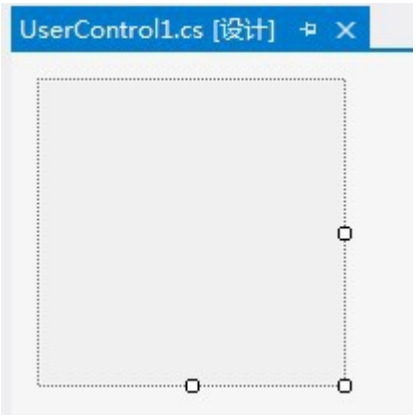


图 8.27　控件设计器

(2) 在“解决方案资源管理器”中右击“UserControl1.cs”，再从快捷菜单中选择“重命名”。将文件名更改为 ValueButton.cs。系统询问是否重命名对代码元素“UserControl1”的所有引用时，单击“是”按钮。

(3) 在“解决方案资源管理器”中右击“ValueButton.cs”，再选择“查看代码”。 找到 class 语句行 public partial class ValueButton，并将此控件继承的类型从 UserControl 更改为 Button。这允许您所继承的控件继承 Button 控件的所有功能。

(4) 在“解决方案资源管理器”中打开“ValueButton.cs”节点，以显示设计器生成的代码文件“ValueButton.Designer.cs”。找到 InitializeComponent 方法并删除分配 AutoScaleMode 属性的行。Button 控件中没有此属性。

(5) 继承的 Windows 窗体控件的可能用途之一是创建与标准 Windows 窗体控件的外观相同、但公开自定义属性的控件。在本例中，将向控件中添加名为 ButtonValue 的属性。找到 class 语句。紧接在 { 后面键入下列代码：

```
private int varValue;
public int ButtonValue
{
   get
   {
      return varValue;
   }
   set
   {
      varValue = value;
   }
}
```

(6) 在“文件”菜单上，指向“添加”，然后单击“新建项目”打开“添加新项目”对话框。在“Visual C#”节点下选择“Windows”节点，再单击“Windows 窗体应用程序”。在“名称”框中键入 Test。

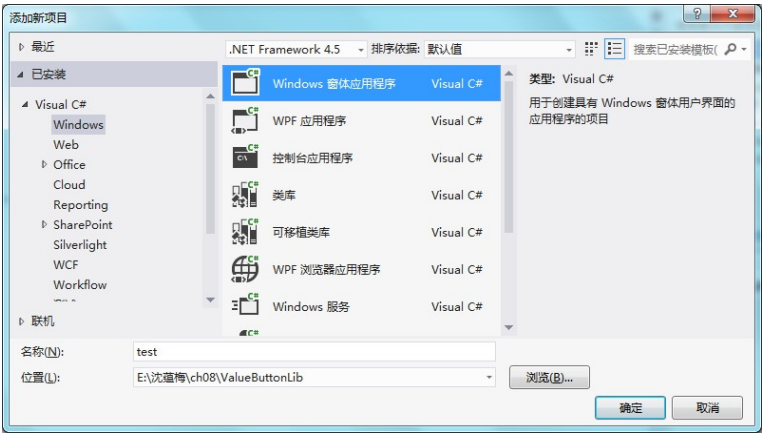


图 8.28 添加 Windows 应用程序

(7) 在解决方案资源管理器中，右键单击 test 项目的“添加引用”节点命令以打开【添加引用】对话框。如图 8.29 所示。选中“ValueButtonLib”，单击【确定】按钮。在“解决方案资源管理器”中，右击“测试”并选择“生成”。

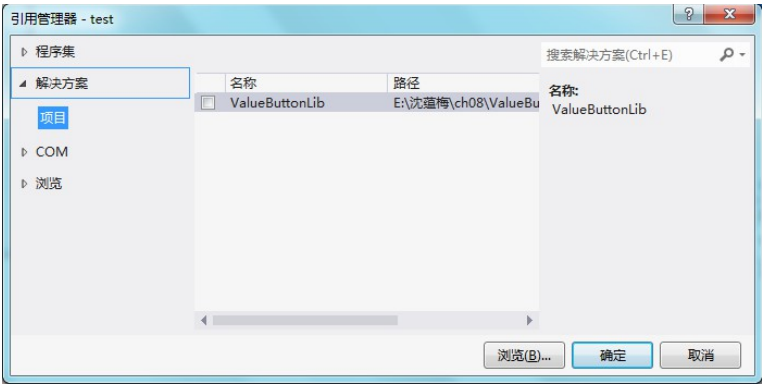


图 8.29 【添加引用】对话框

(8)在“解决方案资源管理器”中，右击“Form1.cs”，然后从快捷菜单中选择“视图设计器”。在“工具箱”中单击“ValueButtonLib 组件”。双击“ValueButton”，窗体上出现一个“ValueButton”。

(9) 右击“ValueButton”并从快捷菜单中选择“属性”。 在“属性”窗口中检查该控件的属性。注意，除增加了一个 ButtonValue 属性外，它们与标准按钮公开的属性相同。将 ButtonValue 属性设置为 5。

(10) 在“工具箱”的“所有 Windows 窗体”选项卡中，双击“标签”，将 Label 控件添加到窗体中。将标签重新定位到窗体的中央，界面如图 8.30 所示。

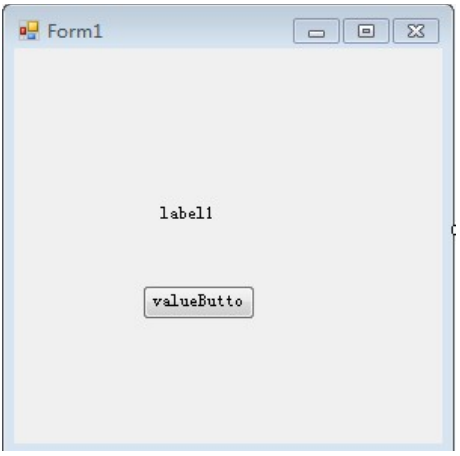


图 8.30　程序界面

(11)添加事件处理代码。双击 valueButton1 以打开代码编辑器并显示 valueButton1_Click 事件，在 valueButton1_Click 事件处理程序中输入如下代码：label1.Text = valueButton1.ButtonValue.ToString();

(12)在解决方案资源管理器中，右键单击 test 解决方案，然后从快捷菜单中选择“设为启动项目”命令。

(13)按 F5 键运行该项目，出现 Form1。单击 valueButton1 按钮，文本框中显示“5”，如图 8.31 所示。这说明我们为定制的控件添加的 valueButton1 属性已经通过 valueButton1_Click 方法传递到文本框。这样 valueButton1 控件便继承了标准的 Windows 窗体按钮的所有功能，并且公开了一个附加的自定义 ButtonValue 属性。

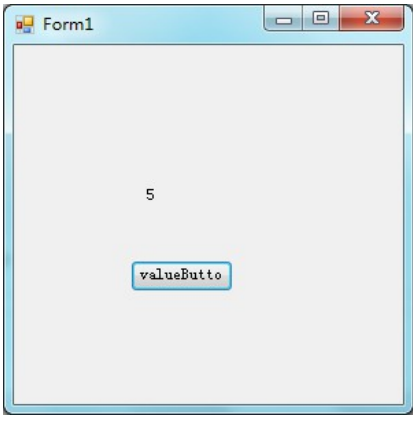


图 8.31　例 8.7 的运行结果

2. 从 UserControl 类继承

下面的示例说明了如何通过组合控件的方式来自定义控件。

例 8.8 本示例将 Label 和 Timer 两个控件绑定到一起，实现通过标签显示系统当前时间，每秒刷新一次。

启动 Visual Studio 2012，创建一个新工程。

(1) 在 Visual Studio 2012 中创建一个新的 C#工程，选择【Windows 控件库】，把新工程命名为“ClockLib”，如图 8.32 所示。

(2) 在“解决方案资源管理器”中右击“UserControl1.cs”，再从快捷菜单中选择“重命名”。将文件名更改为 Clock.cs。系统询问是否重命名对代码元素“UserControl1”的所有引用时，单击“是”按钮。

(3) 打开控件设计器的代码，找到 public partial class Clock: UserControl。默认情况下，用户控件从系统提供的 UserControl 类继承。UserControl 类提供所有用户控件所要求的功能，并实现标准方法和属性。

(4) 在用户控件中加入标签和计时器两个控件。在“解决方案资源管理器”中，切换到 Clock 控件设计器，在“工具箱”中单击“所有 Windows 窗体”选项卡，然后为 Clock 控件设计器添加一个 Label，名为 Label1 的标签控件被添加到用户控件设计器上的控件中。

在设计器中，单击 label1。在“属性”窗口中，设置属性，如表 8.24 所示。

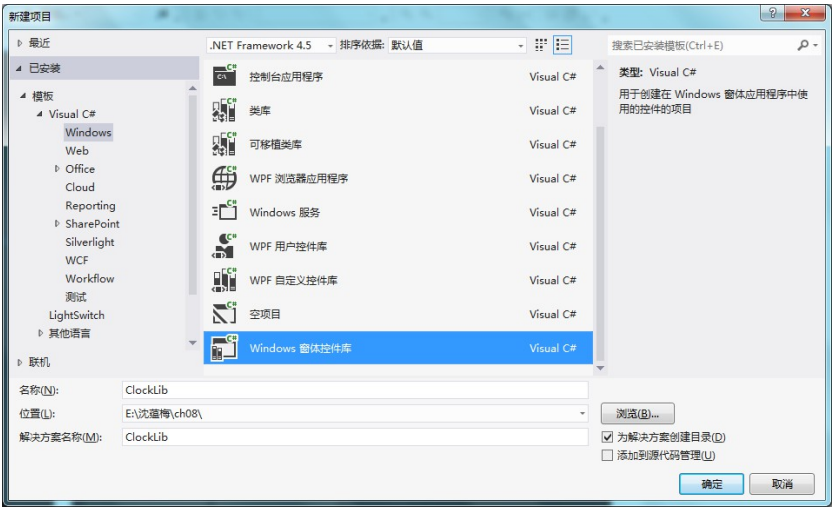


图 8.32 新建项目窗口

表 8.24 Label 的属性设置

属 性	属性描述
Name	lblDisplay
Text	(空白)
TextAlign	MiddleCenter
Font.Size	16
Fore color	Red
Back Color	Info

同样，为 Clock 控件设计器添加一个 Timer 控件，打开 timer1 的属性窗口，将其 Interval 属性设置为“1000”，Enabled 属性设置为

“true”。Timer1 每走过一个刻度，它都会运行一次 timer1_Tick 事件中的代码。Interval 属性表示前后两次刻度之间的毫秒数。

选中 timer1 控件，切换到“事件窗口”，双击“Tick”，为 time1 控件添加一个 timer1_Tick 事件。切换到代码编辑器，找到 timer1_Tick 事件的代码，将代码修改如下：

```
protected virtual void timer1_Tick(object sender, EventArgs e)
{
    //在标签中显示当前的时间
    lblDisplay.Text = System.DateTime.Now.ToLongTimeString();
}
```

修饰符从 private 更改为 protected，用 virtual 关键字修改该方法使其可被重写。

(5) 从“文件”菜单中，选择“全部保存”命令来保存项目。

(6) 生成控件。在“生成”菜单中单击“生成 ClockLib”命令，输出窗体提示生成是否成功。

(7) 创建测试项目。由于定制的控件不是独立的项目，它们必须寄宿在容器中。因此，必须提供一个运行该控件的测试项目，来进行控件的测试。

在“文件”菜单上，指向“添加”，然后单击“新建项目”打开“添加新项目”对话框。在“Visual C#”节点下选择“Windows”节点，再单击“Windows 窗体应用程序”。在“名称”框中键入 testClockLib。单击【确定】按钮，如图 8.33 所示。

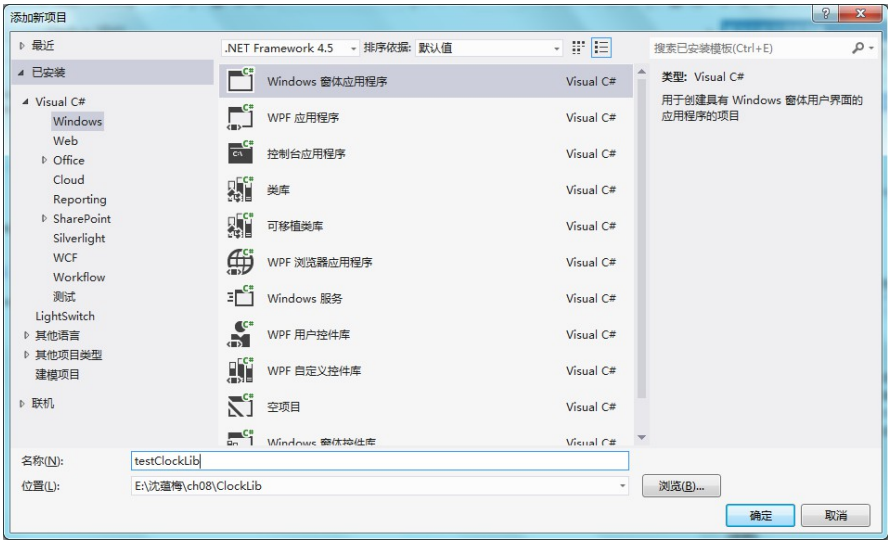


图 8.33　添加 Windows 应用程序

(8) 添加引用后，需要将新控件添加到工具箱。

在解决方案资源管理器中，右键单击 testClockLib 项目的“添加引用”节点命令以打开【添加引用】对话框。如图 8.34 所示。选中“ClockLib”，

单击【确定】按钮。在“解决方案资源管理器”中，右击“testClockLib”并选择“生成”。

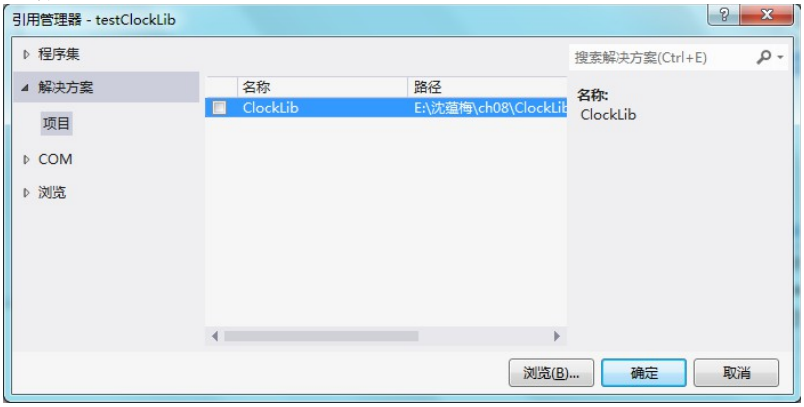


图 8.34 【引用管理集】对话框

(9)将 Clock 控件添加到 testClockLib 的窗体设计器上，并调整到适当的大小。窗体中显示一个名为“clock1”的定制控件。

(10)在解决方案资源管理器中，右击 testClockLib，然后从快捷菜单中选择“设为启动项目”命令。

(11)按 F5 键运行该项目，出现 Form1。效果如图 8.35 所示。

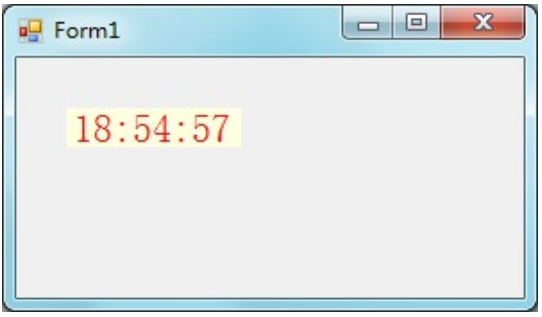


图 8.35 例 8.8 的运行结果

8.3 案 例 实 训

1. 案例说明

利用 C#制作一个简单的计算器，能够实现加、减、乘、除简单功能。

2. 编程步骤

【步骤 1】新建 windows 窗体应用程序，程序界面如图 8.36 所示。

图 8.36 程序界面

【步骤 2】窗体和窗体上各控件的属性设置，如表 8.24 所示：

表 3.4 控件属性列表

Form	Form1	Text	计算器
TextBox	TextBox1	ReadOnly	True
		Name	txtResult
	TextBox2	Name	txtA
	TextBox3	Name	txtB
Button	Button1	Name	btnCal
		Text	计算
comboBox	comboBox1	Items	+ - * /

修改属性后的窗体如图 8.37 所示：

【步骤 3】编写按钮的单击（Click）事件，代码如下。

```
private void btnCal_Click(object sender, EventArgs e)
        {
            int a = int.Parse(txtA.Text);
            int b = int.Parse(txtB.Text);
            int result=0;
            switch (comboBox1.SelectedIndex)
            {
                case 0:
                    result = a + b;
                    break;
                case 1:
                    result = a - b;
                    break;
                case 2:
                    result = a * b;
                    break;
                case 3:
                    result = a / b;
                    break;
            }
            txtResult.Text = result.ToString();
        }
```

【步骤 4】运行程序，单击计算按钮。程序运行效果如图 8.37 所示：

图 8.37 案例的运行结果

8.4 小　　结

本章首先介绍了 Windows 应用程序常用的控件及其相关的属性、方法和事件，开发人员可以使用这些控件编写复杂的应用程序，使用这些控件进行开发，可以减少开发者很多的重复性工作。

.NET 允许开发者根据实际需求创建出自己的控件，并提供了三种常用的定制控件开发方式，开发者可以根据需要，选择一种合适自己的定制方式。

8.5 习　　题

1. 选择题

(1) .NET 中的大多数控件都派生于_____类。

A. System.IO　　B. System.Data

C. System.Windows.Forms.Control　　D. System.Data.Odbc

(2) TextBox 控件的常用属性中，_____控件用来获取或设置字符，该字符用于屏蔽单行 TextBox 控件中的密码字符。

A. Name　　B. PasswordChar

C. SelectedText　　D. Text

(3) 在 RadioButton 控件的事件中，_____事件当 Checked 属性的值更改时发生。

A. CheckState　　B. ThreeState

C. CheckedChanged　　D. Click

2. 填空题

(1) 进行自定义控件开发时，根据实际需要可以用以下三种方法开发定制控件_____、_____和_____。

(2) ____和_____属性决定了一个 TextBox 控件的大小。

3. 编程题

(1)建立用户自定义控件来实现改变次数的功能，运行结果如图 8.38 所示

图 8.38 运行结果