第5章 Windows窗体设计

本章将详细讲解 Windows 窗体及窗体控件的设计与编程，包括 Form 窗体基本操作，Windows 控件设计，文本类控件设计，选择类控件设计，分组类控件设计，菜单、工具栏和状态栏控件设计等知识。每个知识点都配套经典示例，以便学生理解知识理论和掌握知识点的应用。

【本章要点】

- ☞ Form 窗体设计
- ☞ Windows 控件设计
- ☞ 分组类控件设计
- ☞ 菜单、工具栏和状态栏控件设计

5.1 Form 窗体

Windows 环境中主流的应用程序都是窗体应用程序，Windows 窗体应用程序比命令行应用程序要复杂得多，理解它的结构的基础是要理解窗体。Form 窗体也称为窗口，是 .NET Framework 的智能客户端技术。在 Windows 中，使用窗体可以显示信息、接收用户输入以及通过网络与远程计算机通信等。

使用 Visual Studio 2015 可以轻松地创建 Form 窗体程序。Form 窗体是 Windows 应用程序的基本单元，窗体都具有自己的特征，可以通过编程来设置。窗体也是对象，窗体类定义了生成窗体的模板，每实例化一个窗体类，就产生一个窗体对象。.NET Framework 类库的 System.Windows.Forms 命名空间中定义的 Form 类是所有窗体类的基类。

编写窗体应用程序时，首先需要设计窗体的外观并在窗体中添加控件。Visual Studio 2015 提供了一个图形化的可视化窗体设计器，可以实现所见即所得的设计效果，还可以快速开发窗体应用程序。

5.1.1 Form 窗体基本操作

Form 窗体的基本操作包括窗体的添加和删除、窗体的属性设置以及窗体事件和方法的应用。

【例 5.1】 本例将学习 Form 窗体的基本操作，包括窗体的添加、启动窗体的设置以及窗体的删除。启动 Visual Studio 2015，创建一个 Windows 窗体应用程序项目，命名为 case0501。

(1) 添加新窗体。

在“解决方案资源管理器”面板中右击项目名称,在弹出的快捷菜单中选择“添加”→“Windows 窗体”或者“添加”→“新建项”命令,如图 5.1 所示,弹出“添加新项”对话框。

图 5.1 添加新建项

在“添加新项”对话框中,选择“Windows 窗体”选项,如图 5.2 所示,输入窗体名称后,单击“添加”按钮,即可向当前项目添加一个新窗体。

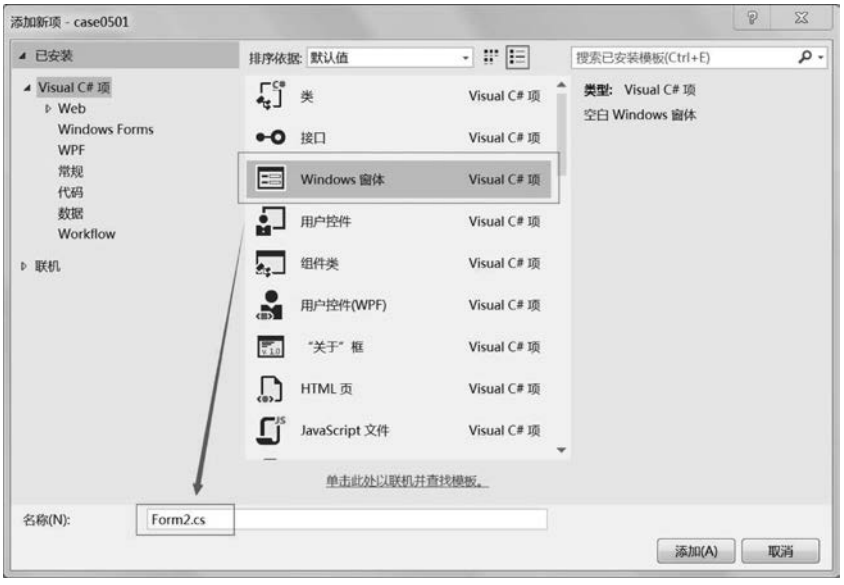

图 5.2 设置新窗体名称

(2) 设置启动窗体。

所谓启动窗体,是指当项目运行时首先显示的窗体。每个项目都默认有一个启动窗体。

向项目中添加了多个窗体以后，根据需要可以修改要首先显示的窗体。

如图 5.3 所示，case0501 项目中有两个窗体，分别是 Form1 和 Form2，程序默认启动窗体是 Form1。如果需要将 Form2 设置为启动窗体，具体做法是：打开 Program.cs 文件，找到 Main()函数中的“Application.Run(new Form1());”语句，将其修改为“Application.Run(new Form2());”即可。

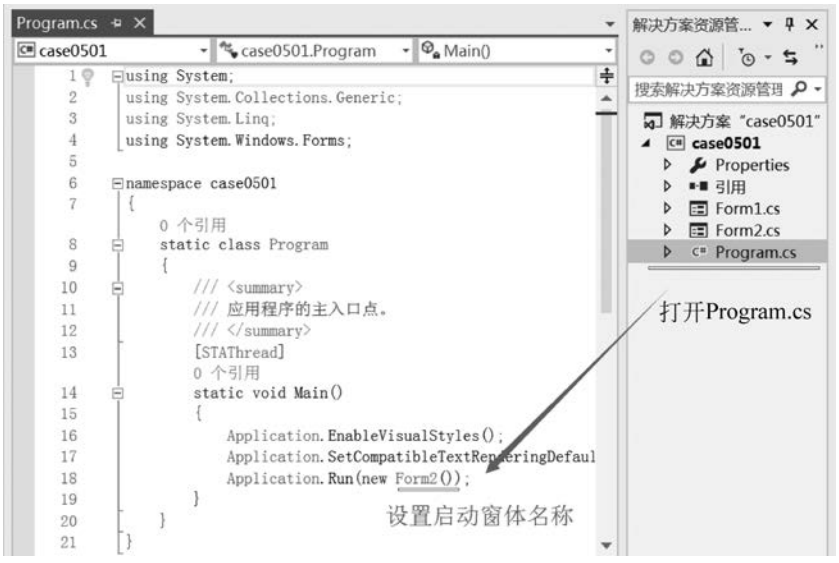

图 5.3 设置启动窗体

(3) 删除窗体。

删除窗体的方法非常简单，打开“解决方案资源管理器”面板，在要删除的窗体名称上右击，在弹出的快捷菜单中选择“删除”命令即可。

(4) Form 窗体剖析。

在 Windows 窗体应用程序项目中，每个项目都是一个命名空间，每个 Form 窗体都是一个类文件。每个 Form 窗体都由三个文件组成，以项目 case0501 中的 Form1 窗体为例，Form1 窗体文件分别包括 Form1.cs、Form1.Designer.cs 和 Form1.resx 文件。Windows 窗体的结构如图 5.4 所示。

图 5.4 Windows 窗体的结构

文件系统中的 Form 窗体文件清单和 Visual Studio 工具中的 Form 窗体文件清单分别如图 5.5 和图 5.6 所示。

Form1.cs 文件用于存放逻辑处理方法的源代码，Form1.Designer.cs 文件用于存放窗体布局的源代码，Form1.resx 文件用于存放窗体资源。这三个文件之间的关系分别介绍如下。

① Form1.cs 文件和 Form1.Designer.cs 文件。

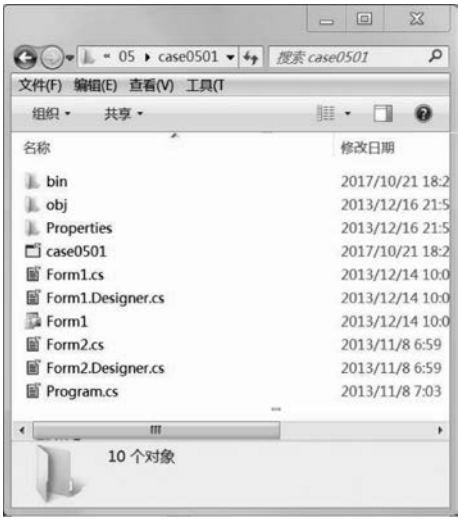

图 5.5 文件系统中的 Form 窗体文件清单

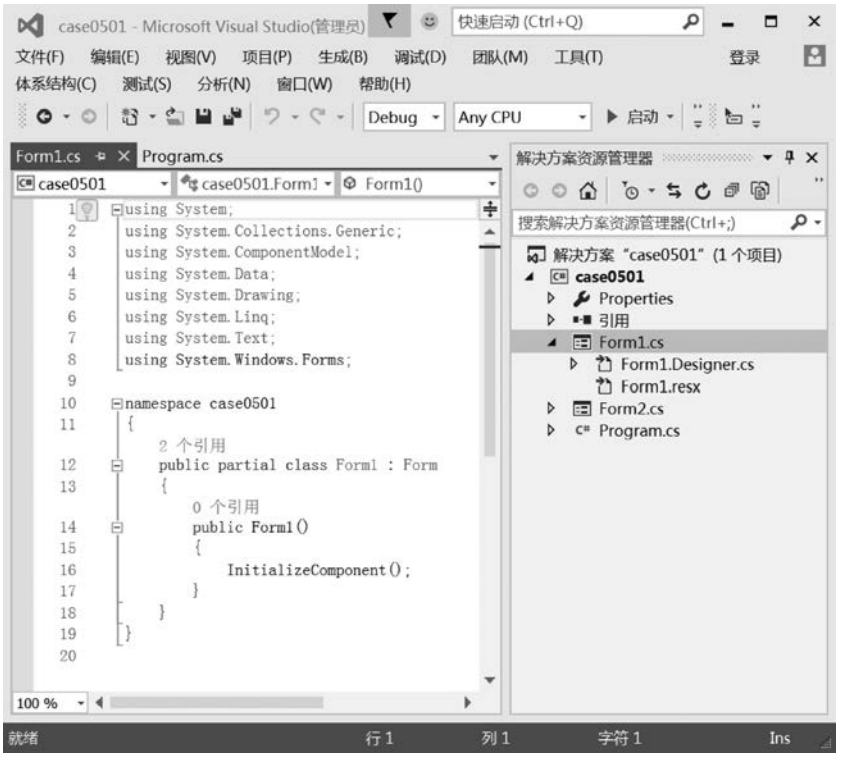

图 5.6 Visual Studio 工具中的 Form 窗体文件

Form1.cs 和 Form1.Designer.cs 是同一个类，Visual Studio 为了方便管理源代码，用 partial(部分)关键字把窗体类给拆开了，即将一个类写在两个文件中，分别定义部分类，Form1.Designer.cs 用于存放窗体布局的源代码，包括窗体的控件以及各控件的属性等信息；Form1.cs 用于存放逻辑处理方法的源代码，例如按钮的单击事件函数等。

2.1.1节已经讲解了同一个命名空间中不允许存在两个名称相同的类，但从图5.7和图5.8中可以看到这两个文件的类名都是Form1，这是否有矛盾？答案是没有矛盾。因为Form1.cs和Form1.Designer.cs对类的定义都使用了partial关键字，代表在Form1.cs中的Form1类是部分类，在Form1.Designer.cs中的Form1类也是部分类，两个文件是同一个类。

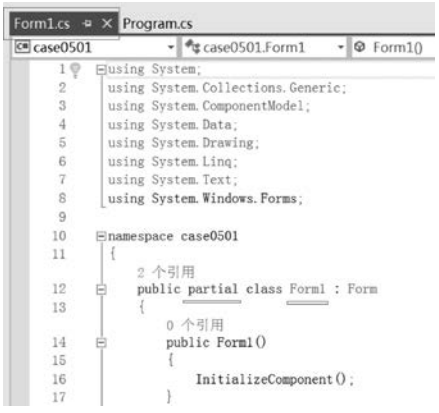

图5.7 Form1.cs文件

图5.8 Form1.Designer.cs文件

② Form1.Designer.cs文件和Form.cs[设计]。

Form1.Designer.cs文件用于存放窗体布局的源代码，在Visual Studio还有一个Form.cs[设计]的界面，该界面的作用又是什么呢？

Form.cs[设计]是可视化编程技术的结晶。所谓可视化编程，是以“所见即所得”的编程思想为原则，实现编程工作的可视化，即随时可以看到结果，程序与结果的调整同步。“可视”指的是无须编程，仅通过直观的拖放操作方式即可完成界面的设计工作。Visual Studio是目前最好的Windows应用程序开发工具。

Visual Studio工具中，如果用户需要在Form1窗体中添加控件，只需要在工具箱中选择所需要的控件，然后将其拖放到窗体中指定位置即可。例如，向Form1窗体中拖放一个按钮控件，将设置该控件显示文本为“登录”，添加控件的过程如图5.9所示。

图5.9 为窗体添加控件

此时Visual Studio工具将自动在Form1.Designer.cs生成该控件所需要源代码，如图5.10所示。也就是说，窗体上的所有控件都是通过源代码生成的，只是使用Visual

Studio 通过拖放控件来完成。

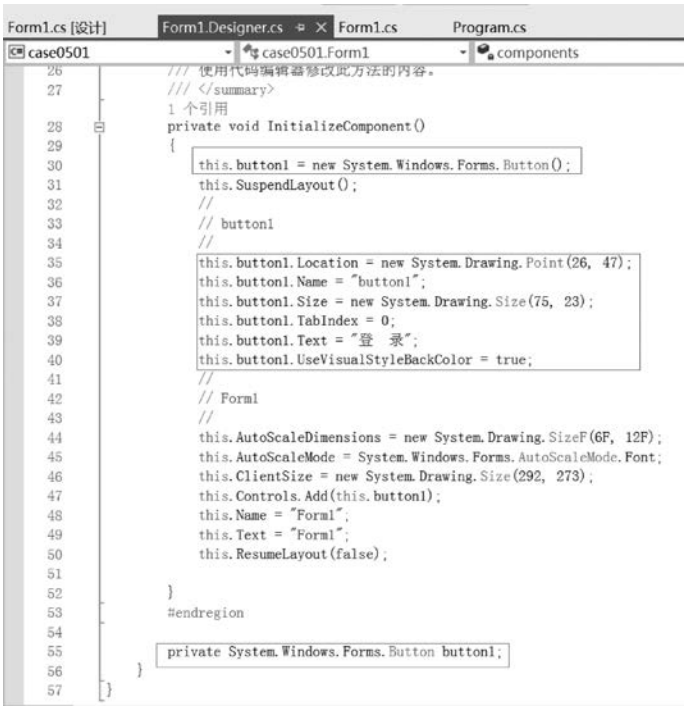

图 5.10 Form1.Designer.cs 文件中按钮的生成源代码

Form1.resx 用于存放窗体资源,如果窗体中使用图像、图标等,这些图像资源就会出现在 Form1.resx 中。

5.1.2 窗体属性、事件和方法

1. 窗体属性

WinForm 窗体包含一些基本的属性特征,包括图标、标题、位置和背景等,这些属性特征可以通过窗体的"属性"面板进行设置,也可以通过源代码实现。为了快速开发窗体应用程序,通常都是通过"属性"面板进行设置。窗体的常用属性及说明如表 5.1 所示。

表 5.1 窗体的常用属性及说明

属性名称	说明
Name	获取或设置窗体的名称
WindowState	获取或设置窗体的窗口状态,即窗体是最大化显示还是正常显示
StartPosition	获取或设置运行时窗体的起始位置,即窗体是显示在屏幕中间还是默认位置
Text	设置或返回在窗口标题栏中显示的文字
Width	获取或设置窗体的宽度
Height	获取或设置窗体的高度
ControlBox	获取或设置在该窗体的标题栏中是否显示控制框,即"最大化"和"最小化"按钮
MaximumBox	获取或设置是否在窗体的标题栏中显示"最大化"按钮
MinimizeBox	获取或设置是否在窗体的标题栏中显示"最小化"按钮

续表

属性名称	说　明
AcceptButton	获取或设置一个按钮的名称，当用户按 Enter 键时就相当于单击了窗体上的该按钮
CancelButton	获取或设置一个按钮的名称，当用户按 Esc 键时就相当于单击了窗体上的该按钮
BackColor	获取或设置窗体的背景色
BackgroundImage	获取或设置窗体的背景图像
Font	获取或设置控件显示的文本的字体
ForeColor	获取或设置控件的前景色
IsMdiChild	获取该窗体是否为多文档界面(MDI)子窗体
IsMdiContainer	获取或设置该窗体是否为多文档界面(MDI)中的子窗体的容器
Icon	设置窗体的图标(图标文件格式为.ico)
FormBorderStyle	设置窗体的边框效果
Opacity	设置窗体的不透明度，取值范围为 0%～100%，100%为完全不透明，0%为完全透明

1) 设置窗体边框效果

窗体的 FormBorderStyle 属性用于设置窗体的边框效果。FormBorderStyle 属性有 7 个属性值，其属性值及说明如表 5.2 所示。

表 5.2　FormBorderStyle 属性值及说明

属　性　值	说　明
Fixed3D	固定的三维边框，不可调整大小
FixedDialog	固定的对话框样式的粗边框，不可调整大小
FixedSingle	固定的单行边框，不可调整大小
FixedToolWindow	不可调整大小的工具窗口边框
None	无边框
Sizable	可调整大小的边框
SizableToolWindow	可调整大小的工具窗口边框

2) 控制窗体的显示位置

窗体的 StartPosition 属性用于设置加载窗体时窗体在显示器中的位置。StartPosition 属性有 5 个属性值，其属性值及说明如表 5.3 所示。

表 5.3　StartPosition 属性值及说明

属　性　值	说　明
CenterParent	窗体在其父窗体中居中
CenterScreen	窗体在当前显示器屏幕中居中
Manual	窗体位置由 Location 属性确定
WindowsDefaultBounds	窗体定位在 Windows 默认位置，其边界由 Windows 默认确定
WindowsDefaultLocation	窗体定位在 Windows 默认位置，其位置根据窗体大小指定

2. 窗体事件

现在的很多软件项目开发都被称为事件驱动型开发。所谓事件就是用户操作窗体或窗体自身状态发生变化时，所引起的一系列动作。Windows 是事件驱动的操作系统，对 Form 类的任何交互都是基于事件来实现的。Form 类提供了大量的事件用于响应对窗体执行的

各种操作。常用的窗体事件有窗体加载事件 Load、窗体关闭事件 FormClosing 等。窗体常用事件及说明如表 5.4 所示。

表 5.4 窗体常用事件及说明

事件名称	说明
Load	窗体加载事件,当窗体被打开时,将触发窗体的 Load 事件
FormClosing	窗体关闭事件,当窗体关闭时,触发该窗体的 FormClosing 事件
Activate	当窗体变为活动窗体时发生此事件,此事件比 Load 事件发生得晚
Deactivate	变为非激活
Click	当用户单击窗体时发生此事件
DragDrop	当完成一个完整的拖放动作或使用 Drag()方法时,发生此事件
Initialize	当应用程序创建 Form、MDIForm、User 控件、PropertyPage 或类的实例时发生
KeyDown	当窗体上没有能获得焦点的控件(如文本框控件)时,用户按下键盘上某个键时发生此事件

【例 5.2】 本例以系登录模块为例,学习 Form 窗体常用属性的设置,常用方法和事件的编程。

(1) 创建项目并新建窗体。

创建一个 Windows 窗体应用程序项目,命名为 case0502。系统默认创建一个 Form1 窗体,该窗体将作为系统登录窗体。因此,需要为项目添加一个新窗体作为系统主界面,将新窗体命名为 Frm_Main。

(2) 修改系统登录窗体名称。

将项目的默认窗体 Form1 作为系统登录窗体,修改该窗体名称为 Form_Login。在“解决方案资源管理器”面板中右击窗体 Form1,在弹出的快捷菜单中选择“重命名”命令,将窗体重命名为 Frm_Login 并按 Enter 键,弹出“您正在重命名一个文件。您也想在这个对源代码元素‘Form1’的所有引用的项目中执行重命名吗”的对话框,单击“是 (Y)”按钮,如图 5.11 所示。

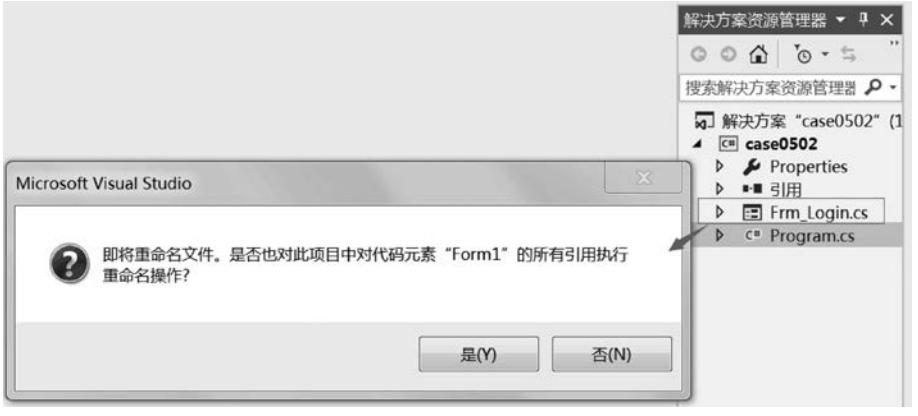

图 5.11 修改窗体名称

(3) 设置 Frm_Login 窗体属性。

在设计界面中选中 Frm_Login 窗体,在“属性”面板中设置该窗体属性。Frm_Login 窗

体的具体属性设置及说明如表 5.5 所示。

表 5.5 Frm_Login 窗体的具体属性设置及说明

属性名称	属性值	说明
Name	Frm_Login	设置窗体的名称为 Frm_Login
WindowState	normal	设置窗体的窗口状态为“正常”
StartPosition	CenterScreen	设置运行时窗体的起始位置为“屏幕中间”
Text	系统登录	设置在窗口标题栏中显示的文字为“系统登录”
Width	300	设置窗体的宽度为 300(像素)
Height	200	设置窗体的高度 200(像素)
MaximizeBox	False	在窗体标题栏中不显示“最大化”按钮
MinimizeBox	True	在窗体标题栏中显示“最小化”按钮
FormBorderStyle	FixedSingle	设置窗体边框为不可拖动大小的效果

Frm_Login 窗体的属性设置过程如图 5.12 所示。

图 5.12 Frm_Login 窗体的属性设置过程

(4) 为 Frm_Login 窗体添加控件并设置属性。

Frm_Login 窗体需要添加两个标签控件(Label)、两个文本框控件(TextBox)和两个按钮控件(Button)。添加完控件后,需要为各控件设置相应的属性。设置控件属性的过程与设置窗体属性的过程相同,各控件具体属性设置及说明如表 5.6 所示。

表 5.6 各控件具体属性设置及说明

控件	属性	属性值	控件	属性	属性值
Label1	Text	用户名:	Button1	Name	btn_Login
Label2	Text	密码:		Text	登录
TextBox1	Name	txt_username	Button2	Name	btn_Cancel
TextBox2	Name	txt_password		Text	取消
	PasswordChar	*			

控件属性设置结束后,保存项目文件并运行程序,程序运行效果如图 5.13 所示。在程序运行的系统登录窗口分别输入用户名 admin、密码 123,可以看到密码以星号“*”显示。

图 5.13 系统登录窗体界面设计

(5) 编写“登录”按钮单击事件。

关闭系统登录窗口或结束程序调试,返回项目开发环境中,双击“系统登录”窗口的“登录”按钮,进入源代码编写窗口。也可以单击“登录”按钮,在“属性”面板中选择事件选项,进入事件列表面板,双击 Click 事件,进入源代码编写窗口(Frm_Login.cs 文件),如图 5.14 所示。

在 Frm_Login.cs 文件中,Visual Studio 已经自动创建一个 btn_Login_Click()事件函数。由于还没学习数据库编程知识,在此编写直接进入系统主窗体的源代码,在 btn_Login_Click 事件函数中编写如下源代码。

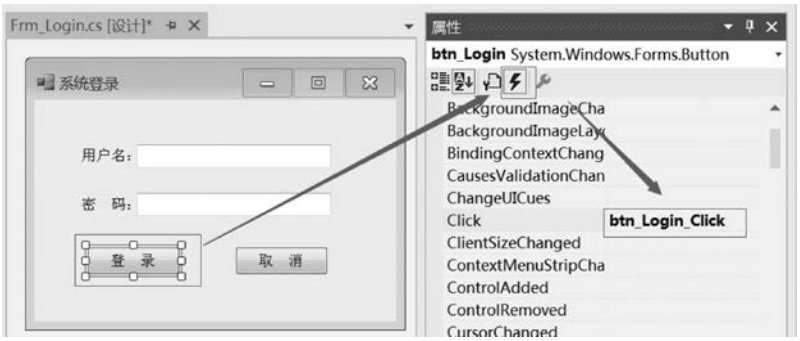

图 5.14 编写“登录”按钮 Click 事件

```
Frm_Main frm = new Frm_Main();                    //实例化主窗体类
frm.Show();                                       //显示主窗体
this.Hide();                                      //隐藏登录窗体
```

注意:此时登录窗体不能关闭,因为该窗体是程序的启动窗体,关闭该窗体后,整个程序将会结束运行。

“登录”按钮单击事件源代码编写结束后,保存项目文件并运行程序,在程序运行的“系统登录”窗口直接单击“登录”按钮,可以看到“系统登录”窗口不见了,此时显示系统主界面。在系统主界面(Frm_Main)中,单击“关闭”按钮,发现系统主窗体关闭了,但程序没有结束,原因在于系统登录窗体还没关闭(刚才单击“登录”按钮时,系统登录窗体被隐藏了)。我们希望在关闭系统主窗体时,弹出“关闭提示”对话框,确定关闭后,结束程序运行。

(6) 编写 Frm_Main 窗体关闭事件。

结束程序调试,返回项目开发环境中,在设计界面中选中 Frm_Main 窗体,在“属性”面板中选择事件选项,进入事件列表面板,双击 FormClosing 事件,进入源代码编写窗口,如图 5.15 所示。

在 Frm_Main.cs 文件中,Visual Studio 已经自动创建一个 Frm_Main_FormClosing 事件函数。在该函数中编写如下源代码。

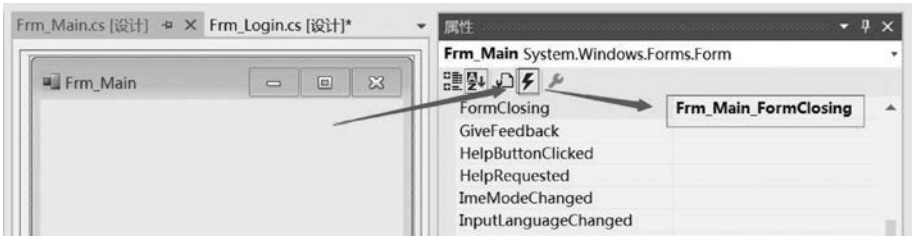

图 5.15 添加 Frm_Main 窗体关闭事件

```
//弹出确定对话框,获取选择结果
DialogResult result = MessageBox.Show("确定关闭系统?", "关闭提示", MessageBoxButtons.
YesNo, MessageBoxIcon.Question);

if (result == DialogResult.Yes)                    //如果选择"是"
{
    e.Cancel = false;
    Application.Exit();                            //关闭程序
}
```

源代码编写完成后,保存项目文件并运行程序,通过“系统登录”窗口进入系统主界面,单击系统主界面的“关闭”按钮,弹出“关闭提示”对话框,如图 5.16 所示。确定关闭后,结束程序运行。

图 5.16 窗体关闭提示

3. Windows 窗体常用方法

Windows 窗体常用方法包括 Show()、Hide()、Refresh()、Activate()、Close()和 ShowDialog()等。窗体常用方法及说明如表 5.7 所示。

表 5.7 窗体常用方法及说明

方 法	说 明
Show()	显示窗体
Hide()	隐藏窗体
Refresh()	刷新并重画窗体
Activate()	激活窗体并给予它焦点
Close()	关闭窗体
ShowDialog()	将窗体显示为模式对话框

5.2 Windows 基本控件

控件是窗体设计的基本组成单位,通过使用控件可以高效地开发 Windows 应用程序。Windows 应用程序中的控件包括文本类控件、选择类控件、分组类控件、菜单控件、工具栏控件和状态栏控件等。

5.2.1 Windows 控件

1. 控件分类

在 Visual Studio 2015 开发环境中,常用控件可以分为文本类控件、选择类控件、分组类控件、菜单控件、工具栏控件以及状态栏控件。Windows 应用程序控件的基类是位于 System. Windows. Forms 命名空间的 Control 类。Control 类定义了控件类的共同属性、方法和事件,其他的控件类都直接或间接地派生自这个基类。常用控件及说明如表 5.8 所示。

表 5.8 常用控件及说明

控　件	说　明
文本类控件	可以在控件上显示文本
选择类控件	主要为用户提供选择的项目
分组类控件	可以将窗体中的其他控件进行分组显示
菜单控件	为系统制作功能菜单,将应用程序命令分组,使它们更容易访问
工具栏控件	提供了主菜单中常用的相关命令的快捷方式
状态栏控件	用于显示窗体上的对象的相关信息,或者可以显示应用程序的信息

2. 控件对齐

在 WinForm 窗体界面设计时,可以使用控件对齐功能对控件进行摆放位置设置。选定需要对齐的两个以上的控件,再选择菜单栏中的"格式"→"对齐"命令,然后选择对齐方式,或者选择工具栏上相应的对齐按钮,如图 5.17 所示。对齐方式分别有左对齐、居中对齐、右对齐、顶端对齐、中间对齐和底部对齐。

图 5.17 控件对齐

在执行对齐之前,首先选定主导控件(首先被选定的控件就是主导控件),该组对齐控件的最终对齐位置取决于主导控件的位置。

3. 控件命名规范

在使用控件的过程中,可以使用控件的默认名称,也可以自定义控件名称。为了提高程序的可读性和规范性,建议根据控件命名规范对控件命名。目前行业内主流的通用命名规范如表 5.9 所示。

表 5.9 行业内主流的通用命名规范

控件名称	开头缩写	控件名称	开头缩写
TextBox	txt	Panel	pl
Button	btn	GropuBox	gbox
ComboBox	cbox	TabControl	tcl
Label	lab	ErrorProvider	epro
DataGridView	dgv	ImageList	ilist
ListBox	lb	HelpProvider	hpro
Timer	tmr	ListView	lv
CheckBox	chb	TreeView	tv
LinkLabel	llbl	PictureBox	pbox
RichTextBox	rtbox	NotifyIcon	nicon
CheckedListBox	clbox	DateTimePicker	dtpicker
RadioButton	rbtn	MonthCalendar	mcalen
NumericUpDown	nudown	ToolTip	ttip

推荐控件的命名格式为：控件类型单词缩写 ＋ “_” ＋ 控件用途说明。如，“姓名”文本框的命名为 txt_Name。

5.2.2 文本类控件

文本类控件主要包括按钮控件(Button)、标签控件(Label)、链接标签控件(LinkLabel)、文本框控件(TextBox)和带格式的文本框控件(RichTextBox)，如图 5.18 所示。

下面介绍最常用的几种文本类控件。

图 5.18 文本类控件

1. 按钮控件(Button)

Button 控件允许用户通过单击来执行操作，Button 控件既可以显示文本，也可以显示图像。常用的 Button 控件的事件为单击事件。

2. 标签控件(Label)

Label 控件主要用于显示用户不能编辑的文本，描述窗体上的对象(例如，为文本框、列表框等添加描述信息)。Label 控件上显示的文本信息可以直接在 Label 控件的“属性”面板中设置 Text 属性，也可以通过编写源代码来设置。

3. 文本框控件(TextBox)

TextBox 控件为用户提供信息输入接口，用于获取用户输入的数据或者显示文本。TextBox 控件通常用于可编辑文本，也可以设置为只读控件。文本框可以显示多行或密码框。文本框控件的常用属性及说明如表 5.10 所示。

表 5.10 文本框控件的常用属性及说明

属性名称	属性值	说明
ReadOnly	True | False	True 为只读状态，False 为可编辑状态
PasswordChar	*、# 等	将文本框设置为密码框时，输入密码时以掩码形式显示
Multiline	True | False	True 为多行文本框

文本框控件的常用事件及说明如表 5.11 所示。

表 5.11 文本框控件的常用事件及说明

事件名称	说明
TextChanged	文本改变事件：当文本框中的文本内容发生更改时，将触发该事件
OnEnter	获取焦点事件：当光标移入文本框时，将触发该事件
OnLeave	失去焦点事件：当光标移出文本框时，将触发该事件

4. 带格式的文本框控件(RichTextBox)

RichTextBox 控件用于显示、输入和操作带有格式的文本。RichTextBox 控件除了执行 TextBox 控件的所有功能之外，还可以显示字体、颜色和链接，也可以从文件加载文本、嵌入图像等。带格式文本框控件的常用属性及说明如表 5.12 所示。

表 5.12 带格式文本框控件的常用属性及说明

属性名称	属性值	说明
Multiline	True | False	是否多行显示
ScrollBars	Both| None| Horizontal 等	是否显示滚动条
SelectionFont	字体、大小、样式	设置字体属性

【例 5.3】 本例继续以系统登录模块为例，学习文本类控件的使用，包括各控件属性的设置，常用方法和事件的编程。

(1) 界面设计。

创建一个 Windows 窗体应用程序项目，命名为 case0503。在 Form1 窗体中添加三个 Label 控件、两个 TextBox 控件和一个 Button 控件，并为各控件设置相应的属性，如表 5.13 所示。

表 5.13 各控件具体属性设置及说明

控件名称	属性名称	属性值	控件名称	属性名称	属性值
Label1	Text	欢迎使用 QQ 软件	TextBox2	Name	txt_password
Label2	Text	QQ 号码：		PasswordChar	
Label3	Text	密码：		ForeColor	InactiveCaption
TextBox1	Name	txt_username		Text	请输入密码
	ForeColor	InactiveCaption	Button1	Name	btn_Login
	Text	请输入号码		Text	登录

(2) 编写“QQ 号码”文本框获取焦点事件。

界面设计完成后，接下来开始源代码编程，分别为“QQ 号码”和“密码”文本框编写获取焦点和失去焦点的事件。在 QQ 登录窗体中选中“QQ 号码”文本框(注意不是“QQ 号码：”标签)，在“属性”面板中选择事件选项，进入事件列表面板，双击 Enter 事件(获取焦点)，进入源代码编写窗口，如图 5.19 所示。

“QQ 号码”文本框获取焦点事件流程是：当光标移入“QQ 号码”文本框(即该文本框获取焦点)时，先判断文本框的内容是否为“请输入号码”，如果是，则将“请输入号码”这几个字清空，等待用户输入。“QQ 号码”文本框获取焦点事件流程如图 5.20 所示。

图 5.19 “QQ 号码”文本框的 Enter 事件

“QQ 号码”文本框失去焦点事件流程是：当光标移出“QQ 号码”文本框(即该文本框失去焦点)时，判断“QQ 号码”文本框的内容是否为空，如果是，则将该文本框内容恢复为“请输入号码”。“QQ 号码”文本框失去焦点事件流程如图 5.21 所示。

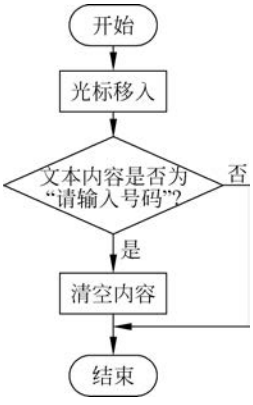

图 5.20 “QQ 号码”文本框获取焦点事件流程

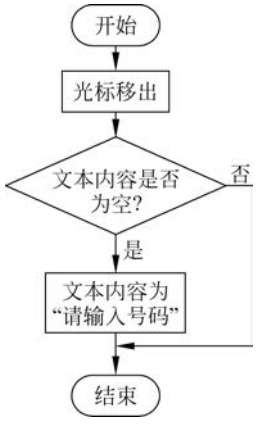

图 5.21 “QQ 号码”文本框失去焦点事件流程

“QQ 号码”文本框获取焦点事件和失去焦点事件的源代码如下。添加“文本框失去焦点”的事件为 Leave，具体方法与添加“文本框获取焦点事件”相同。

```
//"QQ 号码"文本框获取焦点事件
private void txt_username_Enter(object sender, EventArgs e)
{
    if (txt_username.Text == "请输入号码")
    {
        txt_username.Text = "";              //清空内容
    }
}

//"QQ 号码"文本框失去焦点事件
private void txt_username_Leave(object sender, EventArgs e)
{
    if (txt_username.Text == "")
    {
```

```
            txt_username.Text = "请输入号码";
        }
    }
```

源代码编写完成后,保存项目文件并运行程序,读者可以将光标移入和移出“QQ 号码”文本框,体验“QQ 号码”文本框的使用效果。

(3) 编写“密码”文本框获取焦点事件。

接着分别为“密码”文本框添加获取焦点和失去焦点的事件,具体方法与步骤(4)相同。

“密码”文本框获取焦点事件流程是:当光标移入“密码”文本框(即“密码”文本框获取焦点)时,先判断“密码”文本框的内容是否为“请输入密码”,如果是,则将“请输入密码”这几个字清空,等待用户输入密码。注意,应将“密码”文本框的 PasswordChar 属性值设为星号“*”,“密码”将以掩码形式显示。“密码”文本框获取焦点事件流程如图 5.22 所示。

“密码”文本框失去焦点事件流程是:当光标移出“密码”文本框(即“密码”文本框失去焦点)时,判断文本框的内容是否为空,如果是,则将该“密码”文本框内容恢复为“请输入密码”,注意此时应设为明码形式显示(才能看到具体文字信息)。“密码”文本框失去焦点事件流程如图 5.23 所示。

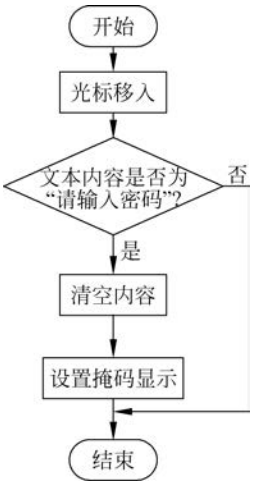

图 5.22 “密码”文本框获取焦点事件流程

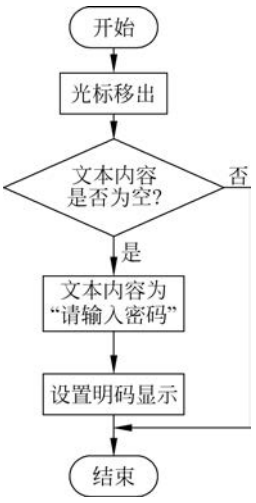

图 5.23 “密码”文本框失去焦点事件流程

“密码”文本框获取焦点事件和失去焦点事件的源代码如下:

```
//"密码"文本框获取焦点事件
private void txt_password_Enter(object sender, EventArgs e)
{
    if (txt_password.Text == "请输入密码")
    {
        txt_password.Text = "";                    //清空内容
        txt_password.PasswordChar = '*';           //设置掩码显示
    }
}
//"密码"文本框失去焦点事件
private void txt_password_Leave(object sender, EventArgs e)
```

```
{
    if (txt_password.Text == "")
    {
        txt_password.Text = "请输入密码";
        txt_password.PasswordChar = '\0';     //设置明码显示
    }
}
```

源代码编写完成后,保存项目文件并运行程序,读者可以将光标移入和移出“密码”文本框,体验“密码”文本框的使用效果。

(4) 编写“登录”按钮单击事件。

源代码如下:

```
//"登录"按钮单击事件
private void btn_Login_Click(object sender, EventArgs e)
{
    if (txt_username.Text.Trim() == "" || txt_username.Text.Trim() == "请输入号码")
    {
        txt_username.Focus();                //将光标移入"QQ号码"文本框,即获取焦点
    }
    else if (txt_password.Text.Trim() == "" || txt_password.Text.Trim() == "请输入密码")
    {
        txt_password.Focus();                //将光标移入"密码"文本框,即获取焦点
    }
    else
    {
        MessageBox.Show("可以开始连接数据库了!");
    }
}
```

源代码编写完成后,保存项目文件并运行程序,程序运行效果如图 5.24 所示。读者可以综合体验该 QQ 登录功能,理解各文本类控件的属性设置和事件原理。

5.2.3 选择类控件

选择类控件主要包括下拉列表框控件(ComboBox)、单选按钮控件(RadioButton)、复选框控件(CheckBox)、数值选择控件(NumericUpDown)和列表框控件(ListBox),如图 5.25 所示。

图 5.24 文本类控件事件

图 5.25 选择类控件

1. 下拉列表框控件(ComboBox)

ComboBox 控件用于在下拉列表框中显示数据,下拉列表框控件的常用属性及说明如表 5.14 所示。

表 5.14 下拉列表框控件的常用属性及说明

属性名称	属性值	说明
DropDownStyle	Simple	使该控件的列表部分总是可见
	DropDown	用户可以编辑该控件的文本框部分
	DropDownList	用户不能编辑该控件的文本框部分,只能选择下拉操作
DropDownWidth	像素	获取或设置该控件下拉部分的宽度
DropDownHeight	像素	获取或设置该控件下拉部分的高度
Items		获取或设置该控件中所包含项的集合
MaxDropDownItems		获取或设置要在 ComboBox 中显示的内容项

下拉列表框控件的常用事件及说明如表 5.15 所示。

表 5.15 下拉列表框控件的常用事件及说明

事件名称	说明
TextChanged	在 Text 属性值更改时触发该事件
SelectedIndexChanged	在 SelectedIndex 属性更改后触发该事件
SelectedValueChanged	当 SelectedValue 属性更改时触发该事件

【例 5.4】 本例学习下拉列表框控件的使用,重点学习 DropDownStyle 属性、Items 属性和 SelectedValueChanged 事件。创建一个 Windows 窗体应用程序,分别添加文本框、下拉列表框、标签控件和按钮控件。编写按钮单击事件,将文本框内容添加到下拉列表框中。然后编写下拉列表框的选择项改变事件,将下拉列表框的选中项的值显示在标签中。程序运行效果如图 5.26 所示。

图 5.26 下拉列表框控件使用

(1) 界面设计。

创建一个 Windows 窗体应用程序项目,命名为 case0504。在 Form1 窗体中分别添加 TextBox、ComboBox、Label 和 Button 控件。

(2) 编写按钮单击事件。

为“添加”按钮编写 Click 事件函数,具体源代码如下:

```
//按钮单击事件
private void button1_Click(object sender, EventArgs e)
{
    comboBox1.DropDownStyle = ComboBoxStyle.DropDown;   //设置下拉列表框只能选择下拉

    comboBox1.Items.Add(textBox1.Text);                 //将文本框内容添加到下拉列表框的行
}
```

(3) 编写下拉列表框的选择项改变事件。

为下拉列表框控件编写 SelectedValueChanged 事件函数,具体源代码如下:

```
//下拉列表框选择项改变时
private void comboBox1_SelectedValueChanged(object sender, EventArgs e)
{
    label1.Text = comboBox1.Text;
}
```

2. 单选按钮控件(RadioButton)

RadioButton 控件为用户提供由两个或多个互斥选项组成的选项集。当用户选中某单选按钮时,同一组中的其他单选按钮不能同时选定。单选按钮控件的常用属性及说明如表 5.16 所示。

表 5.16 单选按钮控件的常用属性及说明

属性名称	属性值	说明
Text		单选按钮显示的文本
Checked	True|False	判断单选按钮是否被选中
AutoCheck	True|False	单选按钮在选中时自动改变状态,默认为 True

单选按钮控件的常用事件及说明如表 5.17 所示。

表 5.17 单选按钮控件的常用事件及说明

事件名称	说明
CheckedChanged	当单选按钮控件的选择状态发生改变时触发该事件
Click	单击控件时触发该事件

【例 5.5】 本例学习单选按钮控件的使用,创建一个 Windows 窗体应用程序,分别添加两个标签控件、三个单选按钮控件。分别编写每个单选按钮的 CheckedChanged 事件,分别显示相应信息。程序运行效果如图 5.27 所示。

(1) 界面设计。

创建一个 Windows 窗体应用程序项目,命名为 case0505。在 Form1 窗体中添加两个 Label 控件和三个 RadioButton 控件。

(2) 编写单选按钮的 CheckedChanged 事件。

分别为三个单选按钮编写 CheckedChanged 事件

图 5.27 单选按钮控件使用

函数,具体源代码如下:

```
//已婚
private void radioButton1_CheckedChanged(object sender, EventArgs e)
{
    if (radioButton1.Checked)
    {
        label2.Text = "我已经结婚啦!";
    }
}

//单身
private void radioButton2_CheckedChanged(object sender, EventArgs e)
{
    if (radioButton2.Checked)
    {
        label2.Text = "我是单身一族哦!";
    }
}

//保密
private void radioButton3_CheckedChanged(object sender, EventArgs e)
{
    if (radioButton3.Checked)
    {
        label2.Text = "我的那一位是个秘密!";
    }
}
```

3. 复选框控件(CheckBox)

CheckBox 控件用来表示是否选取了某个选项条件,常用于为用户提供具有是/否或者是真/假值的选项。复选框控件的常用属性及说明如表 5.18 所示。

表 5.18 复选框控件的常用属性及说明

属性名称	属 性 值	说 明
TextAlign	TopLeft、TopCenter 等	设置文字对齐方式,有 9 种选择
Checked	True|False	获取或设置复选框是否被选中,值为 True 表示选中,值为 False 表示没被选中。当 ThreeState 属性值为 True 时,中间态也表示选中
CheckState	True|False	获取或设置复选框的状态。当 ThreeState 属性值为 False 时,取值有 Checked 或 Unchecked。当 ThreeState 属性值为 True 时,取值还有 Indeterminate,复选框显示为浅灰色选中状态,表示该选项下的子选项未完全选中
ThreeState	True|False	获取或设置复选框是否能表示三种状态。属性值为 True 表示可以表示三种状态(选中、没选中和中间态),属性值为 False 表示两种状态(选中和没选中)

复选框控件的常用事件及说明如表 5.19 所示。

表 5.19 复选框控件的常用事件及说明

事件名称	说明
CheckedChanged	当复选框的 Checked 属性发生改变时,引发该事件
CheckedStateChanged	当 CheckedState 属性改变时,引发该事件

4. 数值选择控件(NumericUpDown)

NumericUpDown 控件是一个显示和输入数值的控件,该控件提供一对上下箭头,用户可以单击上下箭头选择数值,也可以直接输入。NumericUpDown 控件的常用属性及说明如表 5.20 所示。

表 5.20 NumericUpDown 控件的常用属性及说明

属性名称	说明
Maximum	设置最大数值,如果输入值大于该值,则自动把输入值改为设置的最大值
Minimum	设置最小数值,如果输入值小于该值,则自动把输入值改为设置的最小值
value	获取或设置 NumericUpDown 控件中显示的数值
DecimalPlaces	设置在小数点后显示几位,默认值为 0 位
ThousandsSeparator	设置是否每隔 3 个十进制数字位就插入一个分隔符,默认情况下为 False

5. 列表框控件(ListBox)

ListBox 控件用于显示一个列表,用户可以从中选择一项或多项。如果选项总数超出可以显示的项数,则控件会自动添加滚动条。ListBox 控件的常用属性及说明如表 5.21 所示。

表 5.21 ListBox 控件的常用属性及说明

属性名称	属性值	说明
HorizontalScrollbar	True | False	显示水平滚动条
ScrollAlwaysVisible	True | False	始终显示垂直滚动条
SelectionMode	MultiExtended	选择多项,可使用 Shift 键、Ctrl 键和箭头键来进行选择
	MultiSimple	选择多项
	None	无法选择项
	One	只能选择一项

ListBox 控件的常用方法及说明如表 5.22 所示。

表 5.22 ListBox 控件的常用方法及说明

方法	说明
Items.Add()	向 ListBox 控件中添加项目
Items.Remove()	将 ListBox 控件中选中的项目移除
Items.Clear()	清空 ListBox 控件的所有项目

5.2.4 分组类控件

分组类控件主要包括分组框控件(GroupBox)、容器控件(Panel)和选项卡控件(TabControl),

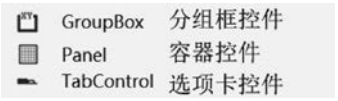

图 5.28 分组类控件

如图 5.28 所示。

1. 分组框控件(GroupBox)

GroupBox 控件主要为其他控件提供分组,按照控件的分组来细分窗体的功能。GroupBox 控件以边框形式出现,可以设置标题,但没有滚动条。

2. 容器控件(Panel)

Panel 控件用于为其他控件提供可识别的分组,Panel 控件默认情况下没有边框。当一个窗体中需要显示两组单选按钮时,则可以使用 Panel 控件将其中一组单选按钮放在其中,从而与另一组单选按钮进行逻辑隔离。

【例 5.6】 本例学习 Panel 控件的使用,当一个 WinForm 窗体中需要使用两组单选按钮时(如性别和婚姻),可以使用 Panel 控件将单选按钮进行分组。

创建一个 Windows 窗体应用程序,命名为 case0506。先添加一个 Label 控件(显示“性别:”)、两个 RadioButton 控件(分别显示“男”和“女”)。接着添加一个 Panel 控件,在控件中继续添加一个 Label 控件(显示“婚姻:”)、两个 RadioButton 控件(分别显示“已婚”和“未婚”),界面设计如图 5.29 所示。

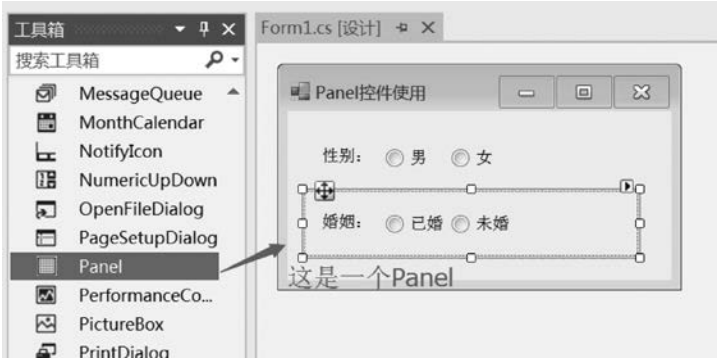

图 5.29 添加 Panel 控件

保存项目文件并运行程序,程序运行效果如图 5.30 所示。此时“性别”单选按钮组和“婚姻”单选按钮组可以独立选择。

图 5.30 Panel 控件的使用

3. 选项卡控件(TabControl)

TabControl 控件可以添加多个选项卡,然后在选项卡上添加子控件。这样就可以把窗体设计成多页,使窗体的功能划分为多个部分。选项卡中可包含图片或其他控件。选项卡控件还可以用来创建用于设置一组相关属性的属性页。TabControl 控件包含选项卡页,TabPage 控件表示选项卡,TabControl 控件的 TabPages 属性表示其中的所有 TabPage 控件的集合。TabPages 集合中 TabPage 选项卡的顺序反映了 TabControl 控件中选项卡的顺序。TabControl 控件的常用属性及说明如表 5.23 所示。

表 5.23 TabControl 控件的常用属性及说明

属性名称	属性值	说明
Alignment	Top|Bottom Left|Right	设置标签在标签控件的显示位置，默认为控件顶部
Appearance	Normal Buttons FlatButtons	设置标签显示方式，分为按钮或平面样式
HotTrack	True|False	当鼠标指针滑过控件标签时的外观是否改变
Multiline	True|False	是否允许多行标签
RowCount	True|False	获取当前显示的标签行数
SelectedIndex		获取或设置选中标签的索引号
SelectedTab		获取或设置选中的标签
TabCount		获取标签总数
TabPages		控件中 TabPage 对象集合

TabControl 控件的常用方法及说明如表 5.24 所示。

表 5.24 TabControl 控件的常用方法及说明

方法	说明
TabPages.Add()	向 TabControl 控件中添加新的选项卡
TabPages.Remove()	将 TabControl 控件中选中的选项卡移除
TabPages.Clear()	清空 TabControl 控件的所有选项卡

5.2.5 菜单控件

菜单是窗体应用程序主要的用户界面要素，主菜单控件(MenuStrip)支持多文档界面、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。菜单控件包括主菜单控件(MenuStrip)和快捷菜单控件(ContextMenuStrip)，如图 5.31 所示。

图 5.31 菜单控件

【例 5.7】 本例学习菜单控件的使用，创建菜单控件，添加“文件”“编辑”“视图”等菜单项，分别设置各菜单项的快捷键。

创建一个 Windows 窗体应用程序，命名为 case0507。在 Form1 窗体中添加一个 MenuStrip 控件。输入各菜单项信息，顶级菜单中分别输入“文件”“编辑”“视图”。在“文件”菜单项的下级菜单中分别输入“新建”“打开”分隔线“-”“保存”和“退出”菜单项，界面设计如图 5.32 所示。

图 5.32 添加菜单项

接着根据需要为各菜单项设置相应访问快捷键，选中菜单项，在“属性”面板中设置属性 ShortcutKeys 的值。例如“新建”菜单项的访问快捷键为 Ctrl+N，设置过程如图 5.33 所示。

保存项目文件并运行程序,即可看到菜单控件运行效果如图 5.34 所示。

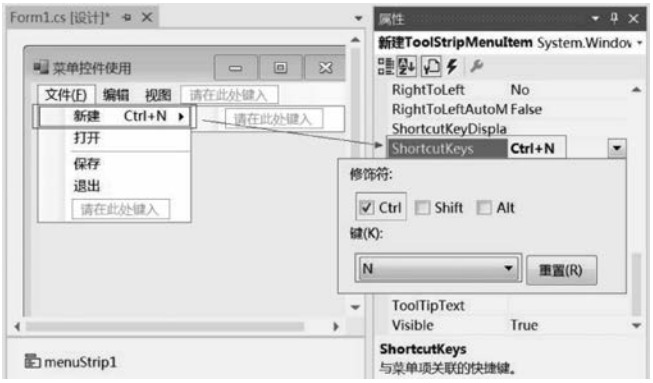

图 5.33 设置菜单项访问快捷键

图 5.34 菜单控件运行效果

5.2.6 工具栏控件

工具栏控件(ToolStrip)用于创建具有 Windows 系统产品风格的工具栏及其他用户界面元素。工具栏是一个控件容器,一个工具栏控件可以添加多个项目,工具栏项目包括按钮控件(Button)、标签控件(Label)、具有下拉效果的按钮控件(SplitButton 和 DropDownButton)、分隔符(Separator)、下拉列表框控件(ComboBox)、文本框控件(TextBox)和进度条控件(ProgressBar)。工具栏控件如图 5.35 所示。

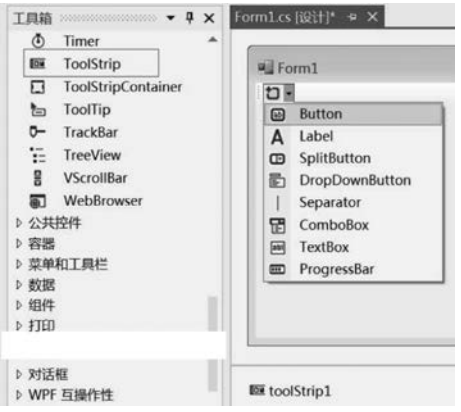

图 5.35 工具栏控件

【例 5.8】 本例学习工具栏控件的使用,创建一个工具栏控件,分别添加按钮、标签、文本框等控件。

创建一个 Windows 窗体应用程序,命名为 case0508。在 Form1 窗体中添加一个工 ToolStrip 控件。添加各工具栏项目控件:以"新建"按钮为例,添加一个按钮控件并设置其属性,为属性 Image 选择相应的图标文件,属性 DisplayStyle 设置为 ImageAndText,属性 TextImageRelation 设置为 ImageAboveText,界面设计如图 5.36 所示。依此方法添加并设计其他控件。保存项目文件并运行程序,即可看到工具栏控件运行效果如图 5.37 所示。

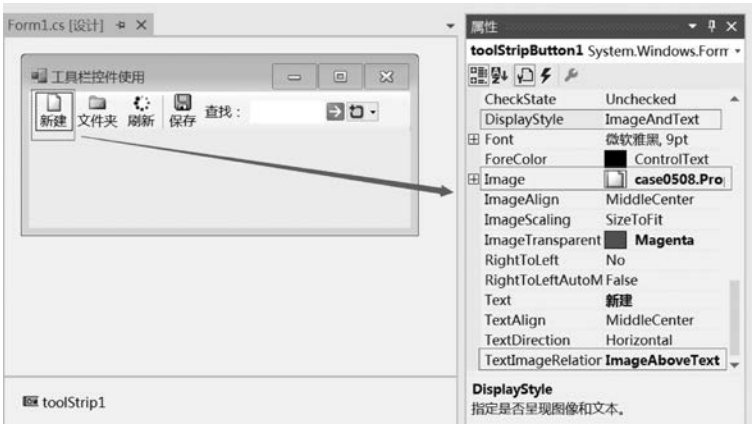

图 5.36 添加工具栏项目控件

5.2.7 状态栏控件

状态栏控件(StatusStrip)通常位于窗体的最底部,用于显示窗体上对象的相关信息,或者显示应用程序的信息。StatusStrip 控件包含标签控件(Label)、进度条控件(ProgressBar)和具有下拉效果的按钮控件(SplitButton 和 DropDownButton)。状态栏控件如图 5.38 所示。

图 5.37 工具栏控件运行效果

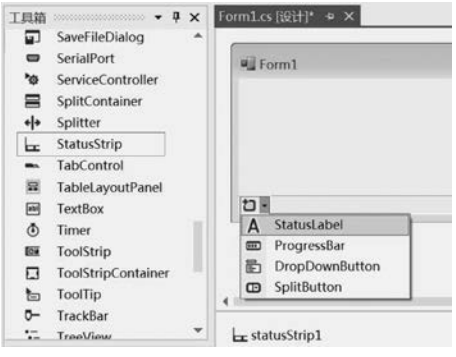

图 5.38 状态栏控件

5.3 综合案例

5.3.1 案例 5.1 只允许输入数字的文本框

在开发一些应用软件时,要求用户录入数据,根据录入数据的类型和具体情况需要对数据进行处理。例如,在录入年龄时,要求录入的数据必须是数字。为了防止用户录入错误的数据,需要在文本框中对录入的数据进行处理,如果录入的数据不符合要求,则给出相应的提示信息。

【技术要点】

(1) TextBox 控件的 KeyPress 事件用来在控件获取焦点的情况下按下键盘中的按键时触发。控制文本框中只能输入数字主要通过 TextBox 控件的 KeyPress 事件实现的。其

语法格式如下：

```
private void textBox1_KeyPress(object sender, KeyPressEventArgs e)
{
   [语句块]
}
```

KeyPressEventArgs是事件数据，其属性KeyChar用于获取键盘按下键所对应的ASCII字符。

(2) 键盘的ASCII字符。

ASCII码是目前计算机中用得最广泛的字符编码集，已经被国际标准化组织(ISO)定为国际标准，称为ISO 646标准。键盘的ASCII字符集如表5.25所示。

表5.25 键盘的ASCII字符集

十进制	字　符	十进制	字　符	十进制	字　符
0	nul(空字符)	30	re(记录分离符)	60	<
1	soh(标题开始)	31	us(单元分隔符)	61	=
2	stx(正文开始)	32	sp(空格)	62	>
3	etx(正文结束)	33	!	63	?
4	eot(传输结束)	34	"	64	@
5	enq(请求)	35	#	65	A
6	ack(收到通知)	36	$	66	B
7	bel(响铃)	37	%	67	C
8	bs(退格)	38	&	68	D
9	ht(水平制表符)	39	`	69	E
10	nl(换行键)	40	(	70	F
11	vt(垂直制表符)	41	)	71	G
12	ff(换页键)	42	*	72	H
13	er(Enter键)	43	+	73	I
14	so(不用切换)	44	,	74	J
15	si(启用切换)	45	-	75	K
16	dle(数据链路转义)	46	.	76	L
17	dc1(设备控制1)	47	/	77	M
18	dc2(设备控制2)	48	0	78	N
19	dc3(设备控制3)	49	1	79	O
20	dc4(设备控制4)	50	2	80	P
21	nak(拒绝接收)	51	3	81	Q
22	syn(同步空闲)	52	4	82	R
23	etb(传输块结束)	53	5	83	S
24	can(取消)	54	6	84	T
25	em(介质中断)	55	7	85	U
26	sub(替补)	56	8	86	V
27	esc(换码(溢出))	57	9	87	W
28	fs(文件分割符)	58	:	88	X
29	gs(分组符)	59	;	89	Y

续表

十进制	字　符	十进制	字　符	十进制	字　符
90	Z	103	g	116	t
91	[	104	h	117	u
92	\	105	i	118	v
93	]	106	j	119	w
94	^	107	k	120	x
95	_	108	l	121	y
96	'	109	m	122	z
97	a	110	n	123	{
98	b	111	o	124	|
99	c	112	p	125	}
100	d	113	q	126	~
101	e	114	r	127	del
102	f	115	s		

【实现步骤】

创建一个 Windows 窗体应用程序，命名为 case0509。在 Form1 窗体中添加 Label 和 TextBox 控件。界面设计完成后，接下来开始源代码编程，为文本框编写 KeyPress 事件，如图 5.39 所示。

图 5.39　编写文本框 KeyPress 事件

文本框 KeyPress 事件的具体源代码如下：

```
private void textBox1_KeyPress(object sender, KeyPressEventArgs e)
{
    //如果输入的不是数字,也不是 Backspace 键,也不是 Enter 键
    if (!char.IsDigit(e.KeyChar) && e.KeyChar != 8 && e.KeyChar != 13)
    {
        e.Handled = true;              //不接受此输入,即输入无效
    }
}
```

保存项目文件并运行程序，程序运行效果如图 5.40 所示。此时文本框只能接受数字数据，不能输入非数字字符。

图 5.40　只允许输入数字的文本框

5.3.2　案例 5.2　带查询功能的下拉列表框

下拉列表框控件可以方便地显示多项数据内容，通过设置 ComboBox 控件的 AutoCompleteSource 属性和 AutoCompleteMode 属性，可以从 ComboBox 控件中查询已存在的项，自动完成控件内容的输入。当用户在 ComboBox 控件中输入一个字符时，ComboBox 控件会自动列出最有可能与之匹配的选项，如果符合用户的要求，则直接确认，从而加快用户输入。

【实现步骤】

创建一个 Windows 窗体应用程序，命名为 case0510。在 Form1 窗体中添加一个 ComboBox 控件。界面设计完成后，接下来开始源代码编程，为 Form1 窗体编写窗体加载事件(Load)，打开窗体时，自动为下拉列表框添加列表项。具体源代码如下：

```
//窗体加载事件
private void Form1_Load(object sender, EventArgs e)
{
    //为下拉列表框添加列表项
    comboBox1.Items.Add("可视化编程技术");
    comboBox1.Items.Add("PHP 程序设计");
    comboBox1.Items.Add("软件工程");
    comboBox1.Items.Add("电子商务概论");
    comboBox1.Items.Add("计算机导论");
}
```

为下拉列表框编写 TextChanged 事件，当用户在 ComboBox 控件中输入一个字符时，ComboBox 控件会自动列出最有可能与之匹配的选项。具体源代码如下：

```
private void comboBox1_TextChanged(object sender, EventArgs e)
{
    if (comboBox1.Text != "")
    {
        comboBox1.AutoCompleteMode = AutoCompleteMode.SuggestAppend;
        comboBox1.AutoCompleteSource = AutoCompleteSource.ListItems;
    }
}
```

保存项目文件并运行程序，程序运行效果如图 5.41 所示。

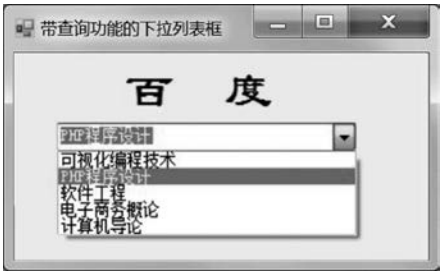

图 5.41 带查询功能的下拉列表框

5.3.3 案例 5.3 逐渐显示的软件启动界面

很多软件都带有欢迎界面，例如 Visual Studio、Photoshop、Office 等，当打开这些软件时，首先会出现一个逐渐显示的欢迎界面。本案例将使用 Timer 控件实现一个逐渐显示的软件启动界面。

【实现步骤】

(1) 创建一个 Windows 窗体应用程序，命名为 case0511。在设计界面中选中 Form1 窗体，在“属性”面板中设置该窗体属性。Form1 窗体的具体属性设置及说明如表 5.26 所示。

表 5.26 Form1 窗体的具体属性设置及说明

属性名称	属 性 值	属性名称	属 性 值
StartPosition	CenterScreen	FormBorderStyle	None
Width	424	BackgroundImage	选择图像资源
Height	222	Opacity	0%

(2) 添加 Timer 控件。从工具箱中选中 Timer 控件，将其拖放到 Form1 窗体中，并且设置 Timer 控件的属性 Enabled 为 True，属性 Interval 为 100，即时间间隔为 100ms(即为 0.1s)，如图 5.42 所示。

图 5.42 添加 Timer 控件

(3) 选中 timer1 控件,为其编写 Tick 事件函数,编写源代码如下。

```
private void timer1_Tick(object sender, EventArgs e)
{
    if (this.Opacity <= 1.0)
    {
        this.Opacity = this.Opacity + 0.02;                    //窗体的不透明度逐渐增加
    }
}
```

(4) 保存项目文件并运行程序,可以看到窗体正在逐渐显示,程序运行效果如图 5.43 所示。

图 5.43 逐渐显示的软件启动界面

5.3.4 案例 5.4 状态栏实时显示时间

使用过 Windows 系列操作的用户都会记得,Windows 操作系统状态栏右侧及时显示当前时间。本案例使用 Timer 控件实现在状态栏中实时显示时间。

【实现步骤】

创建一个 Windows 窗体应用程序,命名为 case0512。添加一个 StatusStrip 控件,并且在状态栏中添加两个 Label 控件。接着添加一个 Timer 控件,并且设置 Timer 控件的属性 Enabled 为 True,属性 Interval 为 1000,即时间间隔为 1000ms(即为 1s 执行一次 Tick 事件),如图 5.44 所示。

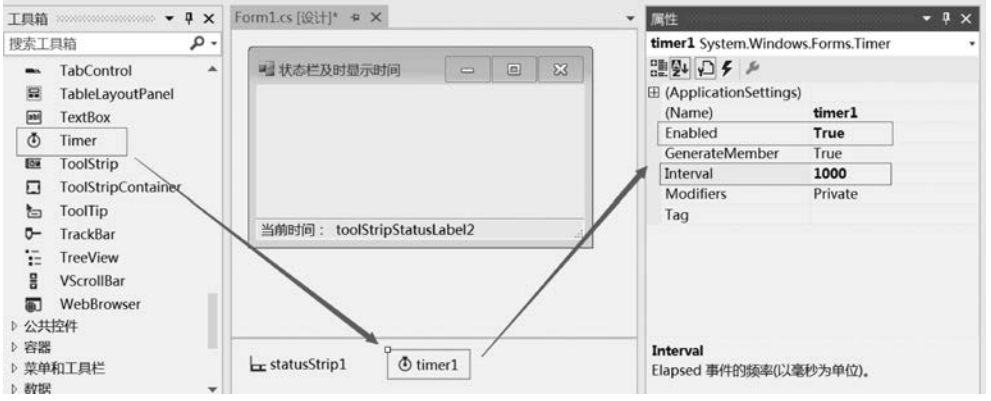

图 5.44 添加 Timer 控件

选中 timer1 控件，为其编写 Tick 事件函数，编写源代码如下：

```
private void timer1_Tick(object sender, EventArgs e)
{
    //将当前时间赋值给标签控件
    toolStripStatusLabel2.Text = DateTime.Now.ToString();
}
```

保存项目文件并运行程序，可以看到窗体状态栏中及时显示当前时间，程序运行效果如图 5.45 所示。

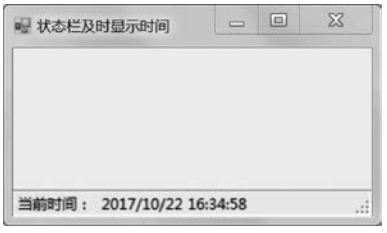

图 5.45 状态栏实时显示时间

5.4 习题

一、选择题

1. 如果将窗体的 FormBoderStyle 设置为 None，则(　　)。

A. 窗体没有边框并不能调整大小　　B. 窗体没有边框但能调整大小

C. 窗体有边框但不能调整大小　　D. 窗体是透明的

2. 若有语句：label1.Text="C#.NET"，默认情况下，在执行本语句之后标签控件的 Name 属性和 Text 属性的值分别为(　　)。

A. "label1" "C#.NET"　　B. "label1" "Text"

C. "label1" "label1"　　D. "Text" "label1"

3. (　　)事件是在窗体被装入工作区时自动触发的事件。

A. Click　　B. Load　　C. Closed　　D. Move

4. 如果要在窗体中始终显示系统的当前时间，应该使用的控件是(　　)。

A. CheckBox　　B. Panel　　C. RadioButton　　D. Timer

5. 下面(　　)源代码可以显示一个消息框。

A. MessageBox.Show();　　B. Dialog.Show();

C. Form.Show();　　D. Form.ShowDialog();

6. Windows 窗体属性中，用于获取或设置运行时窗体的初始显示位置的属性是(　　)。

A. Name　　B. WindowsState

C. StartPosition　　D. Text

7. 窗体中有一个表示物品数量的文本框 txt_Count，下面源代码(　　)可以获得文本

框中的数量值。

A. int count = txt_Count;

B. int count = int.Parse(txt_Count.Text);

C. int count = txt_Count.Text;

D. int count = Convert.ToInt32(txt_Count);

8. 若有语句：text2.Text="ASP.NET",默认情况下,在执行本语句之后标签控件的Name属性和Text属性的值分别为(　　)。

A. "text2" "ASP.NET"　　B. "text2" "Text"

C. "text2" "text2"　　D. "Text" "Name"

9. 下面关于Windows窗体的属性描述,不正确的选项是(　　)。

A. Name获取或设置窗体的名称

B. WindowState获取或设置是否在窗体的标题栏中显示“最小化”按钮

C. StartPosition获取或设置运行时窗体的起始位置,即窗体是显示在屏幕中间还是默认位置

D. Text设置或返回在窗口标题栏中显示的文字

10. 以下选项中,不属于文本类控件的是(　　)。

A. 标签控件(Label)

B. 文本框控件(TextBox)

C. 带格式的文本控件(RichTextBox)

D. 工具栏控件(ToolStrip)

11. 以下选项中,不属于选择类控件的是(　　)。

A. 按钮控件(Button)　　B. 下拉列表框控件(ComboBox)

C. 复选框控件(CheckBox)　　D. 单选按钮控件(RadioButton)

12. 下面关于文本框控件的事件描述,错误的是(　　)。

A. TextChange表示文本改变事件,当文本框中的文本内容发生更改时,将触发该事件

B. OnEnter表示获取焦点事件,当光标移入文本框时,将触发该事件

C. OnLeave表示失去焦点事件,当光标移出文本框时,将触发该事件

D. OnFormClosing表示文本框关闭事件,当文本框关闭时,将触发该事件

13. 在C#的Windows窗体应用程序中,下面(　　)选项不是一个Form窗体包含的文件。

A. Form.cs　　B. Form.Designer.cs

C. Form.resx　　D. Program.cs

14. 以下选项中,不属于分组类控件的是(　　)。

A. 分组框控件(GroupBox)　　B. 容器控件(Panel)

C. 文本框控件(TextBox)　　D. 选项卡控件(TabControl)

二、填空题

1. 设置控件为不可用的属性是_____,设置控件为不可见的属性是_____。

2. 在一个带“\”的字符串前加上_____字符,可使“\”失去转义符功能。

3. 修改控件的名称，需要设置该控件的_____属性。

4. 在 Windows 窗体应用程序中，每个 Form 窗体文件都包括三个文件，分别是_____文件、_____文件和_____文件。

5. C#程序中，用于弹出一个对话框的源代码是_____。

三、判断题

1. (　　)在 Windows 窗体应用程序中，可以使用 Show()方法来打开一个窗体。

2. (　　)窗体的 StartPosition 属性用于设置加载窗体时窗体在显示器中的位置。

3. (　　)Windows 应用程序控件的基类是位于 System. Windows. Forms 命名空间的 Control 类。

4. (　　)窗体的 FormBorderStyle 属性用于设置窗体的边框效果。

5. (　　)Windows 是事件驱动的操作系统，对 Form 类的任何交互都是基于事件来实现的。