第 3 章　C#.NET常用控件使用

3.1　编写形成性考核系统登录程序

在管理信息系统中，为了提高系统的访问安全性，几乎在系统启动时都要设计身份验证，通过用户登录界面向系统用户提供友好的身份验证窗体。怎么设计这些身份验证窗体，本小节通过形成性考核系统登录界面设计展示相关实现技术。

3.1.1　能力目标

通过编写一个窗体登录界面程序，掌握 Windows 控件的应用，设计界面如图 3-1 所示：

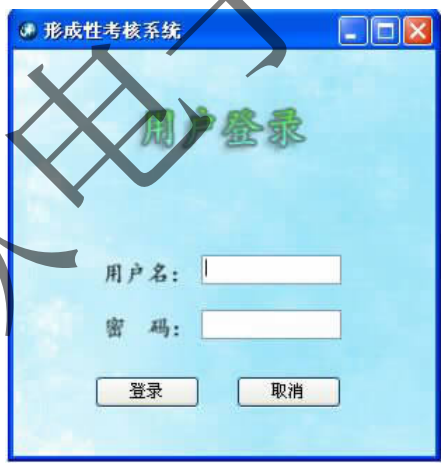

图 3-1　形成性考核系统登录界面

图 3-1 形成性考核系统登录界面所示的形成性考核系统登录界面包括的主要控件有：Form 窗体控件，TextBox 文本框控件，Label 标签控件和 Button 按钮控件。

3.1.2　能力形成过程

要实现图系统登录功能，主要分以下步骤实现：

1.设计窗体界面

该系统登录窗体界面，主要设计的控件有：Label 标签控件，Button 按钮控件，TextBox 文本框控件。

2.实现系统登录核心代码

```
private void button1_Click(object sender, EventArgs e)
    {
        try
        {
            if (txtUser.Text.Trim() == "" || txtPw.Text.Trim() == "")
            {
                MessageBox.Show("用户名或密码为空!");
                txtUser.Focus();
                return;
            }
            Encryption en = new Encryption();
            string pw = en.UserMd5(txtPw.Text.Trim());
            DataSet ds = new DataSet();
            DataBase db = new DataBase();
            string sqlStr = "select * from users where user_id= '" + txtUser.Text.Trim() + "'
        or user_rename='" + txtUser.Text.Trim()+"'";
            ds = db.GetDataFromDB(sqlStr);
            if (ds.Tables[0].Rows[0].ItemArray[2].ToString() == pw)
            {
                frmMain f = new frmMain();
                //frmMain ob_FrmMain = new frmMain();
                PublicClass.userInfo[0] = ds.Tables[0].Rows[0].ItemArray[0].ToString();
                //PublicClass.userInfo[1] = ds.Tables[0].Rows[0].ItemArray[2].ToString();
                PublicClass.userInfo[2] = ds.Tables[0].Rows[0].ItemArray[3].ToString();
                DataBase db1 = new DataBase();
                DataSet ds1 = new DataSet();
                if (PublicClass.userInfo[2] == "教师")

                    ds1 = db1.GetDataFromDB("select teacher_name from teachers where teacher_
                id='" + PublicClass.userInfo[0] + "'");
                    PublicClass.userInfo[1] = ds1.Tables[0].Rows[0].ItemArray[0].ToString
                ();
                }
                else if (PublicClass.userInfo[2] == "学生")
                {
                    ds1 = db1.GetDataFromDB("select s_name from students where s_no='" +
                PublicClass.userInfo[0] + "'");
                    PublicClass.userInfo[1] = ds1.Tables[0].Rows[0].ItemArray[0].ToString
                ();
                }
                else
```

```
            {
                PublicClass.userInfo[1] = "管理员";
            }
            f.Show();
            this.Hide();
            txtUser.Text ="";
            txtPw.Text ="";
        }
        else
        {
            MessageBox.Show("用户名或密码错误,请重新输入!","错误");
            txtUser.Text ="";
            txtPw.Text ="";
            txtUser.Focus();
        }
    }
    catch
    {
        MessageBox.Show("请输入正确的用户名和密码!","错误");
    }
    }
```

3.1.3 能力相关知识

在图 3-1 形成性考核系统登录界面所示的系统登录界面实现过程中,使用了控件 Form、Button、Label 及 TextBox 等,下面就这几种控件的使用进行详细介绍。

1.Form 窗体

Form 窗体控件是进行可视化程序设计界面设计的重要载体、一种最重要的容器控件。Visual Studio .NET提供了非常实用且详细的 Form 控件属性、方法、事件等。

(1)Form 属性。

①AutoScroll:获取或设置一个值,该值指示当控件内容大于它的可见区域时是否显示自动滚动条,值为 True 时显示,否则不显示,默认值为 False。

②BackColor:获取或设置窗体的背景颜色。

③BackgroundImage:获取或设置窗体的背景图片,在形成性考核系统的主界面中就是使用了背景图片这一属性。

④CancelButton:获取或设置用户按【Esc】键时单击的按钮控件。在形成性考核系统中,将退出按钮设置成该属性的值,所以只要按【Esc】键就可以实现退出。

⑤Cursor:获取或设置窗体上的鼠标移动光标形状。

⑥Enable:获取或设置窗体是否可用,默认值为 True,表示可用,一般无需修改该属性的值。

⑦Font:获取或设置窗体上的字体。包括字体的大小、形状等。

⑧ForeColor：获取或设置窗体上显示文字的颜色。

⑨FormBorderStyle：获取或设置窗体边框及标题栏的外观。

⑩Icon：用来设置窗体标题栏上的图标。

⑪ImeMode：确定对象被选中时的 IME(输入法编辑器)状态；该属性在实际开发过程中比较重要，例如：该窗体上只支持英文输入法，就可以设置该属性为 off，当然如果要支持输入法只需将属性值设置为 on。

⑫IsMdiContainer：获取或设置窗体是否是 MDI 容器，默认为不是即为 False，如果进行 MDI 设置时，只需将该属性值改为 True 即可。

⑬KeyPreview：确定窗体上的键盘事件是否已向窗体注册，默认值为 False，表示窗体不支持键盘事件。在项目实际开发应用中，许多操作要求支持键盘操作，这时，就需要将该属性值设置为 True。

⑭MaxmizeBox：用来控制窗体最大化按钮，默认值为 True；表示窗体支持最大化，False 表示不支持最大化。

⑮MinimizeBox：用来控制窗体最小化按钮，默认值为 True；表示窗体支持最小化，False 表示不支持最小化。

⑯ShowIcon：指示窗体标题栏中的图标是否显示，True 表示显示，否则表示隐藏。

⑰ShowInTaskbar：获取或设置一个值，该值指示是否在 Windows 任务栏中显示窗体，True 表示显示，否则表示隐藏。

⑱Size：获取或设置窗体大小。

⑲StartPosition：获取或设置窗体在计算机屏幕上启动的位置。属性值 CenterParent：窗体在其父窗体中居中；属性值 CenterScreen：窗体在当前显示窗口中居中，其尺寸在窗体大小中指定；属性值 Manual：窗体的位置由 Location 属性确定；属性值 WindowsDefaultBounds：窗体定位在 Windows 默认位置，其边界也由 Windows 默认决定；属性值 WindowsDefaultLocation：窗体定位在 Windows 默认位置，其尺寸在窗体大小中指定。

⑳Text：获取或设置窗体标题栏标题内容。

㉑TopMost：指示当前窗体是否在其他窗体之上，True 表示在其他窗体上，否则不是。

㉒TransparencyKey：获取或设置将表示窗体透明区域的颜色，在进行窗体个性化设计时该属性非常重要，多媒体编程章节中将会使用，在此不再举例。

㉓WindowState：获取或设置窗体状态，分为正常 Normal、最大化 Maximized、最小化 Minimized。

(2)Form 事件。

Form 窗体事件是完成 Form 功能的重要组成部分，有关 Form 窗体常用事件及功能说明如表 3-1 所示。

表 3-1 Form 窗体常用事件及功能说明

事件名称	说　明
Activated	当使用代码激活或用户激活窗体时发生
BackColorChanged	当 BackColor 属性的值更改时发生
BackgroundImageChanged	当 BackgroundImage 属性的值更改时发生
Click	在单击控件时发生
ClientSizeChanged	当 ClientSize 属性的值更改时发生
Closed	关闭窗体后发生
Closing	在关闭窗体时发生
Deactivate	当窗体失去焦点并不再是活动窗体时发生
Disposed	添加事件处理程序以监听组件上的 Disposed 事件
DockChanged	当 Dock 属性的值更改时发生
DoubleClick	在双击控件时发生
EnableChanged	在 Enabled 属性值更改后发生
Enter	进入控件时发生
FontChanged	在 Font 属性值更改时发生
ForeColorChanged	在 ForeColor 属性值更改时发生
FormClosed	关闭窗体后发生
FormClosing	关闭窗体前发生
GotFocus	在控件接收焦点时发生
HelpButtonClicked	单击“帮助”按钮时发生
HelpRequested	当用户请求控件帮助时发生
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键时发生
KeyUp	在控件有焦点的情况下释放键时发生
Leave	在输入焦点离开控件时发生
Load	在第一次显示窗体前发生
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

事件名称	说　明
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
Paint	在重绘控件时发生
PreviewKeyDown	在焦点位于此控件上的情况下，当有按键动作时发生(在 KeyDown 事件之前发生)
Resize	在调整控件大小时发生
Scroll	用户或代码滚动工作区时发生
Shown	只要窗体是首次显示就发生
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
VisibleChanged	在 Visible 属性值更改时发生

2.Label 标签

Label 控件在实际开发中使用较多，主要向用户显示系统信息且不允许用户修改。Label 控件的一些常规属性与 Form 窗体的常规属性一致，这里我们不再描述。

3.TextBox 文本框

(1)TextBox 属性。

TextBox 控件是用户向计算机系统输入数据的一种重要途径，它的丰富属性也让用户操作该控件变得更加简单、方便。下面是 TextBox 控件的一些特有属性：

①CausesValidation：获取或设置一个值，该值指示当 TextBox 控件在获得焦点时是否会引起执行验证。

②CharacterCasing：指示控件中的 Text 内容是保持不变还是转换为大写或小写，Normal 表示保持不变，Upper 表示转为大写，Lower 表示转为小写。

③MaxLength：指示控件允许最大输入字符数，默认值为 0，表示没有长度限制。

④Multiline：表示控件是否允许多行编辑，True 表示允许，False 表示不允许。

⑤PasswordChar：指示当前文本框为单行密码框，并设置其显示密码的字符，例如：“*”或其他的字符作为密码显示字符。

⑥ReadOnly：设置或获取当前控件编辑状态是否为编辑状态还是只读状态，True 表示为只读，False 表示编辑状态。

⑦ScrollBars：获取或设置哪些滚动条应出现在多行 TextBox 控件中。None 表示不显示滚动条，Horizontal 表示只显示水平滚动条，Vertical 表示只显示垂直滚动条，Both 表示水平和垂

直都显示。

⑧Text：表示控件编辑的内容。

⑨TextAlign：表示编辑文本内容对齐的方式，分为 Left 表示为左对齐、Center 表示居中对齐、Right 表示右对齐。

⑩WordWrap：指示多行编辑控件是否自动换行，True 表示自动换行，False 表示手动换行。

(2)TextBox 事件。

除了其他控件拥有的常用事件外，TextBox 控件常用事件如下：

```
Private void  TextBox_TextChanged(object sender, EventArgs e)
{
    事件体
}
```

TextBox_TextChanged 用来监视文本变化。

```
Private void  TextBox_Validating(object sender, CancelEventArgs e)
{
    事件体
}
```

TextBox_Validating 用来验证文本框里值的有效性。

4.Button 命令控件

Button 命令控件是项目开发中最常用的控件之一，是用户发送指令到计算机执行的一种常见方法。最常见的是用户单击计算机进行功能响应。

(1)Button 属性。

Button 按钮属性与 Form 等控件的属性有很多类似，这里介绍几个常用的属性。

①BackgroundImage：用来设置 Button 按钮的背景图片，我们在按钮上看到的背景图片一般就是通过这种方式来实现的。

②BackgroundImageLayout：设置 Button 按钮背景图片的布局方式，属性值为 Center 时，图像在控件的矩形工作区中居中显示，属性值为 None 时，图像沿控件的矩形工作区顶部左对齐，属性值为 Stretch 时，图像沿控件的矩形工作区拉伸，属性值为 Tile 时，图像沿控件的矩形工作区平铺，属性值为 Zoom 时，图像在控件的矩形工作区中居中放大。

③FlatStyle：获取或设置按钮控件的平面样式外观，属性值为 Flat 时，该控件以平面显示，属性值为 Popup 时，该控件以平面显示，直到鼠标指针移动到该控件为止，此时该控件外观为三维，属性值为 Standard 时，该控件外观为三维，属性值为 System 时，该控件的外观由用户的操作系统决定。

④Image：按钮上面显示的图片，需要指定图片的来源。

⑤ImageAlign：按钮上图片对齐的方式，共分为左上角、左中、左下角、中上、中、中下、右上角、右中、右下角九个方位。通常和 Image 属性配合使用。

⑥ImageIndex：按钮上显示的图片在 ImageList 中的索引。

⑦ImageList：获取按钮上显示图片的图片源，需要 ImageList 控件支持，要与 ImageList 控件配合使用。

⑧Text：设置或获取按钮上显示的文本。

⑨TextAlign：按钮上显示文本的对齐方式，具体值与 ImageAlign 值一致。该属性与 Text 属性配合使用。

(2)Button 事件。

Button 事件同其他事件一样有很多事件，其中用得最大的为 Click 单击事件，当用户单击按钮时产生，具体格式如下：

```
Private void  Button_Click(object  sender ,  EventArgs  e)
{
    事件体
}
```

5.ComboBox 组合框

ComboBox 控件是用来显示或向系统进行信息输入的重要控件之一。除具备 Form 窗体的一些常规属性外，还有如下的特殊属性：

(1)ComboBox 属性。

①Anchor：设置控件在其容器控件中的位置，取值分别为：Top(上)、Bottom(下)、Left(左)、Right(右)。

②Dock：实现控件在其容器控件中的布局，它将其容器控件分成五块，分别为上、下、左、右、中；取值分别为上、下、左、右、中及 None，当取值为上时，控件在其容器控件的最上面，其他类似，而值为 None 时表示控件布局不受控制，即 Dock 属性失效。

③DropDownStyle：控制组合框的外观和功能。属性值为 DropDown 时，文本部分可编辑，用户必须单击箭头按钮来显示列表部分，这是默认样式，属性值为 DropDownList 时，用户不能直接编辑文本部分，用户必须单击箭头按钮来显示列表部分，属性值为 Simple 时，文本部分可编辑，列表部分总可见。

④DropDownWidth：用来设置下拉显示列表的宽度，以像素为单位。

⑤Enabled：是否允许对当前控件进行操作，如果取值为 True，表示可以进行操作，False 表示不可以对控件进行操作，默认值为 True。

⑥FlatStyle：确定控件的显示风格。属性值为 Flat 时，该控件以平面显示，属性值为 Popup 时，该控件以平面显示，直到鼠标指针移动到该控件为止，此时该控件外观为三维，属性值为 Standard 时，该控件外观为三维，属性值为 System 时，该控件的外观是由用户的操作系统决定的。

⑦FormatString：格式说明符，指示 ComboBox 的 Text 属性显示的格式，但需要数据绑定后才起作用。

⑧FormattingEnabled：该属性指示 FormatString 是否起作用，True 为有效，False 为无效。

⑨Items：设置或获取控件中下拉选择数据项，该属性是为组合框控件提供下拉数据源，在开发过程中动态添加数据项时可用该属性。

⑩Sorted：指示控件中的选项是否进行排序，True 为排序，False 为不排序。

⑪Text：控件显示的内容。

⑫Visible：指示控件是隐藏还是显示，True 表示显示，False 表示隐藏。

(2)ComboBox 事件。

如表 3-2 所示列出了 ComboBox 的常用事件及功能说明。

表 3-2 ComboBox 的常用事件及功能说明

事件名称	说明
DropDown	当显示 ComboBox 的下拉部分时发生
DropDownClosed	在 ComboBox 的下拉部分不再可见时发生
DataSourceChanged	当 DataSource 更改时发生
SelectedIndexChanged	在 SelectedIndex 属性更改后发生
SelectedValueChanged	当 SelectedValue 属性更改后发生
Validated	在控件完成验证时发生
Validating	在控件正在验证时发生

【课堂实训 3-1】编写形成性考核系统登录程序

实训性质：程序设计。

实训目的：

(1)熟悉 Form 窗体，TextBox 控件，Label 控件，Button 控件，ComboBox 控件的使用；

(2)学会控件中数据类型的使用；

(3)完成形成性考核系统中登录模块用户名自动匹配。

实训环境：

Window XP/7、Visual Studio .NET 2008。

实训内容：

(1)打开 VS.NET，选择菜单“文件”→“打开”→“项目/解决方案”命令，在“打开项目”窗口中找到形成性考核系统对应的项目文件并打开，解决方案资源管理器窗口中的内容如图3-2所示。

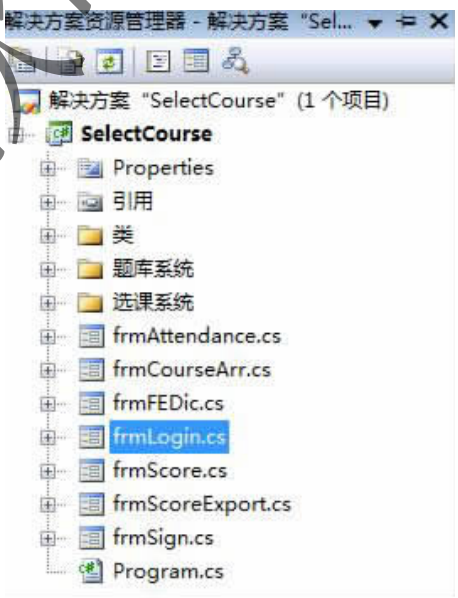

图 3-2 资源管理器窗口

(2)双击 frmLogin 窗体,在原窗体上将用户名的 TextBox 控件改为 Combobox 控件,如图 3-3 所示。

图 3-3　改为 Combobox 控件

(3)在 ComboBox 控件属性栏选择“Items”项,如图 3-4 所示。

图 3-4　ComboBox 控件属性栏选择“Items”项

弹出如图所示的对话框,在对话框中加入相应的用户名,如图 3-5 所示。

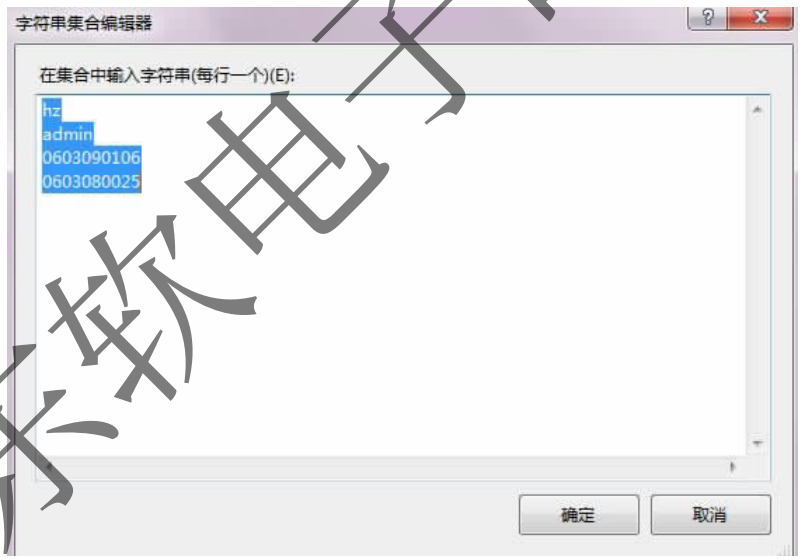

图 3-5　字符串集合编辑器

(4)添加代码如下。

```
private string strTemp = "";
……
private void comboBox1_KeyPress(object sender, KeyPressEventArgs e)
    {
        strTemp = strTemp + e.KeyChar;
```

```
    }

    private void comboBox1_TextChanged(object sender, EventArgs e)
    {
        if (comboBox1.FindString(strTemp, 0) >= 0)
        {
            comboBox1.SelectedIndex = comboBox1.FindString(strTemp);
            comboBox1.Select(strTemp.Length, comboBox1.Text.Length);
        }
        else
            strTemp = comboBox1.Text;
        }
```

运行效果如图 3-6 所示。

图 3-6 运行效果

说明:上面给出的代码只完成了用户名的模糊查找,并不能完成登录的功能,读者可根据提供的源代码进行相应的修改,完成具有用户名模糊查找的登录功能。另外,本模块实现过程采用直接输入的方式添加用户名,读者可采用文件读写的方式完成更加实用的用户登录功能。文件读写相关内容可参考下一章节的内容。

思考:用户登录模块是信息系统中的常用模块,该模块中可以加入验证码功能,读者可以思考如何加入验证码完成安全性更高的系统登录模块。

3.2 编写形成性考核系统主界面程序

对于用户来说,通过身份验证仅仅是系统安全验证的体现,进入系统后,系统的易操作性显得犹为重要,如何设计出友好、易操作的系统主界面是系统设计的一个重要环节,本小节通过案例进行系统主界面设计与程序编写。

3.2.1 能力目标

编写如图 3-7 所示的系统主界面。

图 3-7　系统主界面

图 3-7 所示的系统主界面主要包括的控件有：MenuStrip 菜单控件，StatusStrip 状态栏控件，Timer 控件，ContextMenuStrip 上下文菜单控件。

3.2.2　能力形成过程

实现图 3-7 所示的界面设计，可按以下步骤完成：

1.主界面设计

①Form 窗体作为主容器控件，窗体 Form 作为 MDI 容器，即为多文档操作界面，如图 3-8 所示。

图 3-8　设置窗体是否是 MDI 容器

②设置窗体属性 Text 值为“形成性考核系统”，如图 3-9 所示。

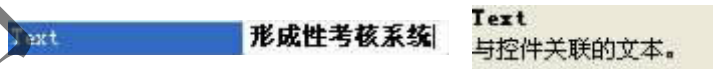

图 3-9　设置窗体属性 Text 值为“形成性考核系统”

③设置 BackgroundImage 属性为指定“图片文件”，如图 3-10 所示。

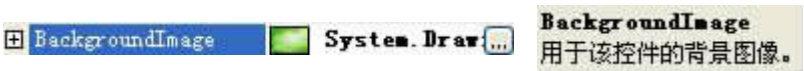

图 3-10　设置 BackgroundImage 属性为指定图片文件

④设置 Icon 属性为指定“图标文件”，如图 3-11 所示。

图 3-11　设置 Icon 属性为指定图标文件

⑤设置 StartPosition 属性为“CenterScreen”，如图 3-12 所示。

图 3-12 设置 StartPosition 属性为 CenterScreen

⑥设置 WindowState 属性为“Maximized”，如图 3-13 所示。

图 3-13 设置 WindowState 属性为 Maximize

⑦设置 MainMenuStrip 为“MenuStripl”，如图 3-14 所示。

图 3-14 设置 MainMenuStrip 为 MenuStrip1

2.菜单设计

使用 MenuStrip 控件作为图的主菜单；

设置 Items 属性添加各项需要的菜单；

设置 Image 属性添加菜单所需的图标；

设置 ShortcutKeys 属性完成各项菜单快捷键设置；

设置 ShowShortcutKeys 属性为 True 表示允许使用快捷键；

设置 ToopTipText 属性表示该项菜单的操作提示。

3.上下文菜单设计

使用 ContextMenuStrip 控件作为图的上下文菜单；

设置 Items 属性添加各项需要的菜单。

4.状态栏设计

使用 StatusStrip 控件作为图的状态栏；

设置 Items 属性添加所需的相关内容；

设置“当前用户”项，设置 Image 添加图片，ImageAlgin 设置为 MiddleLeft，Text 设置为“当前用户”，Spring 设置为 True；

设置“当前时间”项，设置 Text 为“当前时间”，Spring 设置为 True。

5.时间设置

使用 Timer 控件作为图的时间功能实现；

设置 Enable 属性为 True；

编写 Timer 中的 Tick 事件程序。

6.部分功能实现代码

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Text;
using System.Windows.Forms;
```

```
namespace SelectCourse
{
    public partial class frmMain : Form
    {
        public static Form frmParent;

        public frmMain()
        {
            InitializeComponent();
        }

        private void 退出 ToolStripMenuItem1_Click(object sender, EventArgs e)
        {
            PublicClass.frm.Close();
        }

        private void frmMain_Load(object sender, EventArgs e)
        {
            frmParent =this;
            //labelTitle.BackColor = Color.Transparent;
            //labelText.BackColor = Color.Transparent;

            slUserName.Text ="  当前用户:" + PublicClass.userInfo[0] + "!";
            purview();

        }

            private void 注销用户 ToolStripMenuItem_Click(object sender, EventArgs e)
            {
                if (MessageBox.Show("是否注销当前用户?", "系统询问", MessageBoxButtons.
                YesNo, MessageBoxIcon.Question) == DialogResult.Yes)
                {
                    this.Close();
                    PublicClass.frm.Show();
                }
            }

            private void 用户管理 ToolStripMenuItem_Click(object sender, EventArgs e)
            {
                frmUser f = new frmUser();
                f.MdiParent =this;
                f.Show();
```

```
    }

    private void purview()
    {
        if (PublicClass.userInfo[2] ! = "管理员")
        {
            学院管理 ToolStripMenuItem.Visible = false;
            专业管理 ToolStripMenuItem.Visible = false;
            toolStripMenuItem9.Visible =false;
            toolStripSeparator3.Visible =false;
            toolStripMenuItem12.Visible =false;
            班级管理 ToolStripMenuItem.Visible = false;
        }
        if (PublicClass.userInfo[2] == "学生")
        {
            进行考试 ToolStripMenuItem.Visible = true;
            形成性考核 ToolStripMenuItem.Visible = false;
            toolStripMenuItem1.Visible =false;
            toolStripMenuItem7.Visible =false;
            toolStripMenuItem4.Visible =true;
        }
    }

    private void toolStripMenuItem5_Click(object sender, EventArgs e)
    {
        frmYours frmY = new frmYours();
        frmY.MdiParent =this;
        frmY.Show();
    }

    private void 退出 ToolStripMenuItem2_Click(object sender, EventArgs e)
    {
        PublicClass.frm.Close();
    }

    private void 注销 ToolStripMenuItem_Click(object sender, EventArgs e)
    {
        if (MessageBox.Show("是否注销当前用户?", "系统询问", MessageBoxButtons.
YesNo, MessageBoxIcon.Question) == DialogResult.Yes)
        {
            this.Close();
            PublicClass.frm.Show();
```

```
    }
}

private void 显示主窗体 ToolStripMenuItem_Click(object sender, EventArgs e)
{
    this.Show();
}

private void 学院管理 ToolStripMenuItem_Click(object sender, EventArgs e)
{
    frmDept f = new frmDept();
    f.MdiParent =this;
    f.Show();
}

private void 专业管理 ToolStripMenuItem_Click(object sender, EventArgs e)
{
    frmMajor f = new frmMajor();
    f.MdiParent =this;
    f.Show();
}

private void 课程管理 ToolStripMenuItem_Click(object sender, EventArgs e)
{
    frmCourse frmC = new frmCourse();
    frmC.MdiParent =this;
    frmC.Show();
}

private void 选课管理 ToolStripMenuItem_Click(object sender, EventArgs e)
{
    frmSC frmSc = new frmSC();
    frmSc.MdiParent =this;
    frmSc.Show();
}

private void 学生管理 ToolStripMenuItem_Click(object sender, EventArgs e)
{
    frmStudent fs = new frmStudent();
    fs.MdiParent =this;
    fs.Show();
}
```

```
private void 实训信息 ToolStripMenuItem1_Click(object sender, EventArgs e)
{
    frm_tra_dic f_tra_dic = new frm_tra_dic();
    f_tra_dic.MdiParent =this;
    f_tra_dic.Show();
}

private void 实训安排 ToolStripMenuItem1_Click(object sender, EventArgs e)
{
    frm_tra_arr f_tra_arr = new frm_tra_arr();
    f_tra_arr.MdiParent =this;
    f_tra_arr.Show();
}

private void 分组管理 ToolStripMenuItem1_Click(object sender, EventArgs e)
{
    frmTeams f_teams = new frmTeams();
    f_teams.MdiParent =this;
    f_teams.Show();
}

private void 课程成绩权重模块管理 ToolStripMenuItem_Click(object sender, EventArgs e)
{
    frm_weight f_weight = new frm_weight();
    f_weight.MdiParent =this;
    f_weight.Show();
}

private void 实训打分 ToolStripMenuItem_Click(object sender, EventArgs e)
{
    frmTraScore f = new frmTraScore();
    f.MdiParent =this;
    f.Show();
}

    private void 试题录入 ToolStripMenuItem_Click(object sender, EventArgs e)
{
    frmAQ f = new frmAQ();
    f.ShowDialog();
}

private void 试题信息 ToolStripMenuItem_Click(object sender, EventArgs e)
```

```
{
    frmQuestions f = new frmQuestions();
    f.MdiParent =this;
    f.Show();
}

private void 试题类型 ToolStripMenuItem_Click(object sender, EventArgs e)
{
    frmType fT = new frmType();
    fT.MdiParent =this;
    fT.Show();
}

private void 组合试卷 ToolStripMenuItem_Click(object sender, EventArgs e)
{
    frmChoiceClass frmCC = new frmChoiceClass();
    frmCC.ShowDialog();
}

private void 试卷编辑 ToolStripMenuItem_Click(object sender, EventArgs e)
{
    frmSelPapers frmSP = new frmSelPapers();
    frmSP.MdiParent =this;
    frmSP.Show();
}

private void 进行考试 ToolStripMenuItem_Click(object sender, EventArgs e)
{
frmSelectExam fse = new frmSelectExam();
    fse.MdiParent =this;
    fse.Show();
}

private void 进行评卷 ToolStripMenuItem_Click(object sender, EventArgs e)
{
    frmPapers frmP = new frmPapers();
    frmP.ShowDialog();
}

private void 成绩审核 ToolStripMenuItem_Click(object sender, EventArgs e)
{
frmCheckScore f = new frmCheckScore();
```

```
        f.MdiParent =this;
        f.Show();
    }

    private void toolStripMenuItem3_Click(object sender, EventArgs e)
    {
        frmAttendance f = new frmAttendance();
        f.MdiParent =this;
        f.Show();
    }

    private void toolStripMenuItem4_Click(object sender, EventArgs e)
    {
        frmScore f = new frmScore();
        f.MdiParent =this;
        f.Show();
    }

    private void toolStripMenuItem6_Click(object sender, EventArgs e)
    {
        frmFEDic f= new frmFEDic();
        f.MdiParent =this;
        f.Show();
    }

    private void toolStripMenuItem8_Click(object sender, EventArgs e)
    {
        frmFEScore f = new frmFEScore();
        f.MdiParent =this;
        f.Show();
    }

    private void 成绩导出 ToolStripMenuItem_Click(object sender, EventArgs e)
    {
        frmScoreExport f = new frmScoreExport();
        f.MdiParent =this;
        f.Show();
    }

    private void 班级管理 ToolStripMenuItem_Click(object sender, EventArgs e)
    {
        frmClass f = new frmClass();
```

```
            f.MdiParent =this;
            f.Show();
        }

        private void toolStripMenuItem12_Click(object sender, EventArgs e)
        {
            frmTeacher f = new frmTeacher();
            f.MdiParent =this;
            f.Show();
        }

        private void toolStripMenuItem13_Click(object sender, EventArgs e)
        {
            frmCourseArr f = new frmCourseArr();
            f.MdiParent =this;
            f.Show();
        }
    }
}
```

3.2.3　相关能力知识

1.MenuStrip 菜单栏

(1)MenuStrip 属性。

①BackColor:设置或获取 MenuStrip 控件的背景颜色。

②BackgroundImage:设置或获取 MenuStrip 控件的背景图片。

③BackgroundImageLayout:设置 MenuStrip 的背景图片的布局。具体的布局方式与 Form 控件相同,可参照使用。

④ContextMenuStrip:设置 MenuStrip 控件的上下文菜单控件。

⑤ImageScalingSize:设置各菜单图标显示大小。

⑥Items:设置或获取 MenuStrip 控件中的各菜单项。对于菜单项的添加、删除及修改可以直接在设计菜单中操作。菜单编辑如图 3-15 所示。

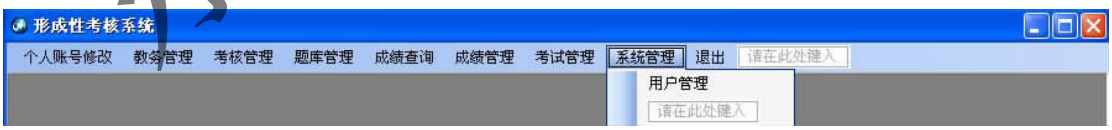

图 3-15　菜单编辑

另外,还有一种方式可以完成菜单编辑,即通过 Items 项集合编辑器设计向导,项集合编辑器如图 3-16 所示。

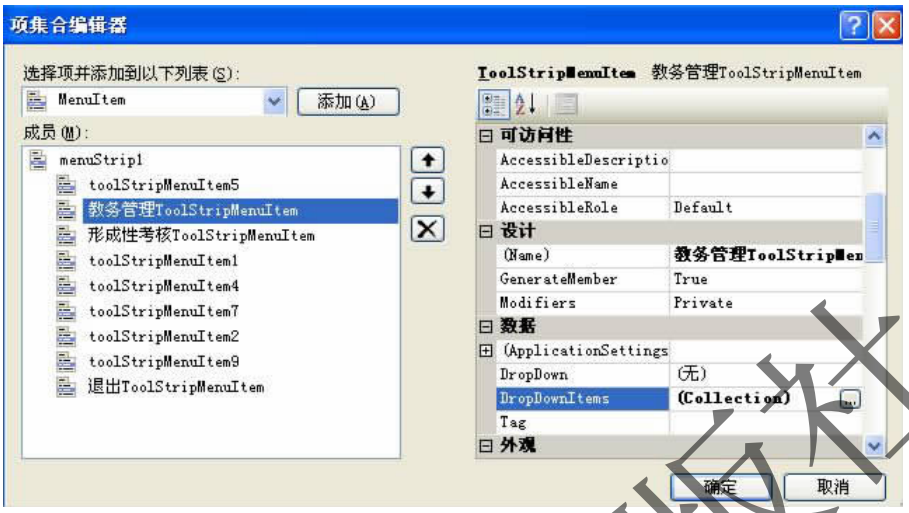

图 3-16　项集合编辑器

通过项集合编辑器就可以完成对菜单的添加、修改及删除操作。

⑦ShowItemToolTips:设置或获取各菜单项是否显示 ToolTips 控件,如果值为 True,表示使用 ToolTips,否则不应用 ToolTips 控件。

⑧Stretch:获取或设置一个值,该值指示 MenuStrip 是否在它的容器中从一端拉伸到另一端。

⑨Text:获取或设置与此控件关联的文本。

⑩TextDirection:获取或设置在 ToolTips 上绘制文本的方向。

(2)MenuStrip 事件。

MenuStrip 菜单控件最常用的事件就是响应用户的单击事件。单击事件格式如下:

```
private void toolStripMenuItem_Click(object sender, EventArgs e)
{
    事件体;
}
```

其他事件读者可参照 MSDN 学习。

2.ToolStrip 工具栏

(1)ToolStrip 工具栏控件在实际项目应用开发过程中很常用,主要为用户提供快捷的操作方式,这些操作一般是通过应用属性、方法和事件实现的。

①Items:设置工具栏中的各项。设置方式可以直接编辑 ToolStrip 控件。共分为 Button、Label、SplitButton、DropDownButton、Separator、ComboBox、TextBox、ProgressBar 八种类型。

我们可以通过 Items 工具栏项集合编辑器完成工具栏的编辑,如图 3-17 所示。

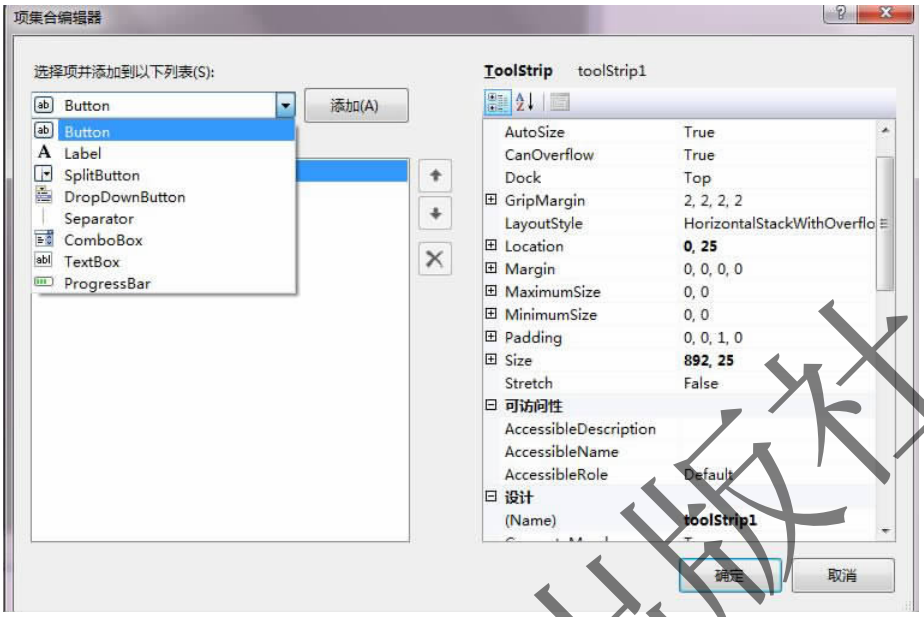

图 3-17　通过 Items 工具栏项集合编辑器完成工具栏的编辑

②LayoutStyle:指定 ToolStrip 布局方向,Flow 表示根据需要指定项按水平方向或垂直方向排列,HorizontalStackWithOverflow 表示指定项按水平方向进行布局且必要时会溢出,StackWithOverflow 表示指定项按自动方式进行布局,Table 表示指定项的布局方式为左对齐。

③Stretch:指定 ToolStrip 在漂浮容器中是否从一端伸展到另一端。

(2)ToolStrip 事件。

ToolStrip 主要事件如下:

```
Private void  toolStripComboBox1_Click(object  sender, event  Args e)
{
    //Items 为 ComboBox 的控件
}
Private void  toolStripTextBox1_Click(object  sender, event  Args e)
{
    //Items 为 TextBox 的控件
}
Private void  toolStripProcessBar1_Click(object  sender, event  Args e)
{
    //Items 为 ProcessBar 的控件
}
Private void toolStripButton1_Click(object  sender, event  Args e)
{
    //Items 为 Button 的控件
}
```

除此之外,ToolStrip 中有很多的事件,而 Items 中的不同项也有很多事件,这些事件有与其他控件相同的地方,例如:Items 中 TextBox 与 TextBox 控件就有很多相同的事件,Button、

ComboBox 也一样，读者可以从中参照相关的控件事件进行学习。

3.ContextMenuStrip 上下文菜单

ContextMenuStrip 上下文菜单，也称为浮动菜单，其最大的好处就是用户可以通过该控件进行方便、快捷的操作。

(1)ContextMenuStrip 属性。

ContextMenuStrip 控件与 MenuStrip 控件非常相似，可参照 MenuStrip 控件学习。

(2)ContextMenuStrip 事件。

同样可参照 MenuStrip 控件，要特别注意 Click 事件的使用。

4.StatusStrip 状态栏

StatusStripu 状态栏在很多地方起到的作用是向用户提供一些提示信息，例如：Windows 操作系统的状态栏里会提示当前正在操作的内容，C# 的 StatusStrip 状态栏也不例外。

StatusStrip 有如下属性：

(1)autoSize：是否允许控件中的各项进行自动调整自身大小以适应其内容的大小。

(2)Items：设置 StatusStrip 中的显示项，可以直接在界面设计时设置 StatusStrip。

(3)ShowItemToolTips：是否显示提示信息控件，如果值设置为 True，表示显示提示控件，每个 Items 中的对象都包含 ToolTipsText 属性，用来显示该对象的一些用户使用提示信息；反之为 False，表示不适用 ToolTips。

(4)Stretch：该属性使用与工具栏 ToolStrip 类似，请参照 ToolStrip 中的 Stretch。

5.Timer 控件

Timer 提供以指定的时间间隔执行方法的机制。可以根据 Timer 指定某个程序段或某个软件在指定时间段执行，当然也可以重复执行这些程序段或软件。例如：主界面上的时间日期的显示就是通过 Timer 控件完成的。

(1)Timer 属性。

①Enabled：设置是否启用 Timer 控件，True 表示启用，False 表示禁用。

②Interval：设置程序间隔执行的时间段，以毫秒为单位，1s=1000ms。

(2)Timer 事件。

Timer 控件最常用的事件是 Tick 事件，格式如下：

```
private void timer1_Tick(object sender, EventArgs e)
{
    //事件体
}
```

将需要重复执行的事件写在 Tick 事件中即可，要注意的是如果 Enabled 属性设置为 False，Tick 事件是不会执行的，Timer 的默认值为 False，所以使用时要改为 True。

【课堂实训 3-2】编写形成性考核系统主界面程序

实训性质：程序设计。

实训目的：

(1)熟悉 MenuStrip 控件，ToolStrip 控件，ContextMenuStrip 控件，StatusStrip 控件，Timer 控件的使用；

(2)完成系统主界面用户工具栏控件程序编写；

(3)完成系统主界面状态栏控件时间显示程序编写。

实训环境：

Window XP/7、Visual Studio .NET 2008。

实训内容：

(1)打开 VS.NET，选择菜单“文件”→“打开”→“项目/解决方案”命令，在“打开项目”窗口中找到形成性考核系统对应的项目文件并打开，解决方案资源管理器窗口中的内容如图 3-18 所示。

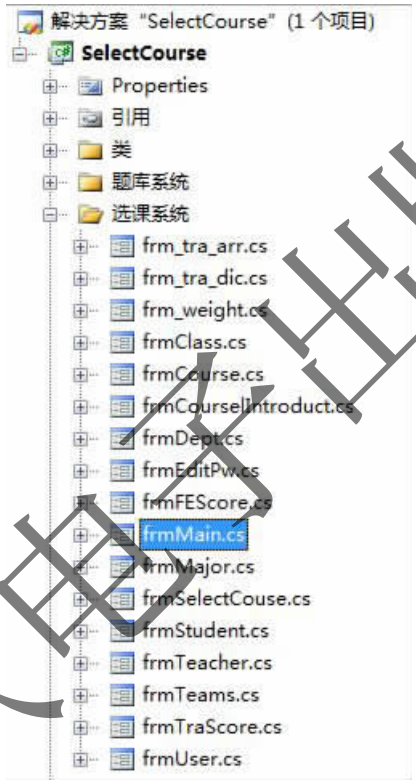

图 3-18　解决方案资源管理器窗口中的内容

(2)双击 frmMain 窗体，在原窗体上增加用户工具栏 ToolStrip 控件，如图 3-19 所示。

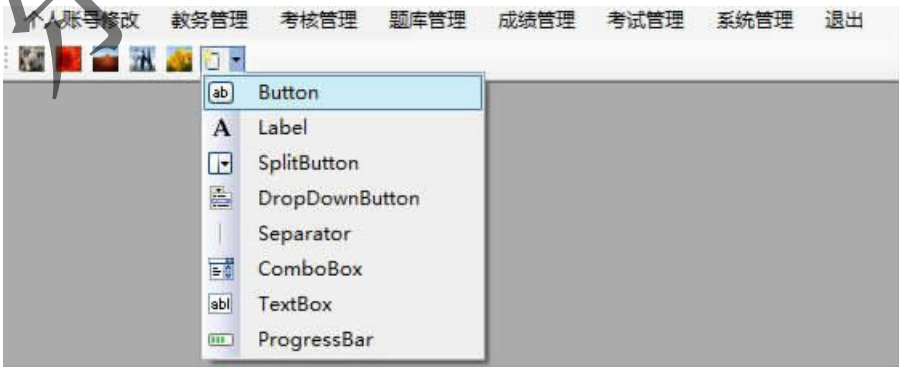

图 3-19　在原窗体上增加用户工具栏 ToolStrip

我们这里使用的是 ToolStrip 中的 Button 类型添加工具栏，读者可以根据需要添加其它类型的工具栏。图标的设置只需在 TooStrip1 的属性栏中进行设置即可，如图 3-20 所示。

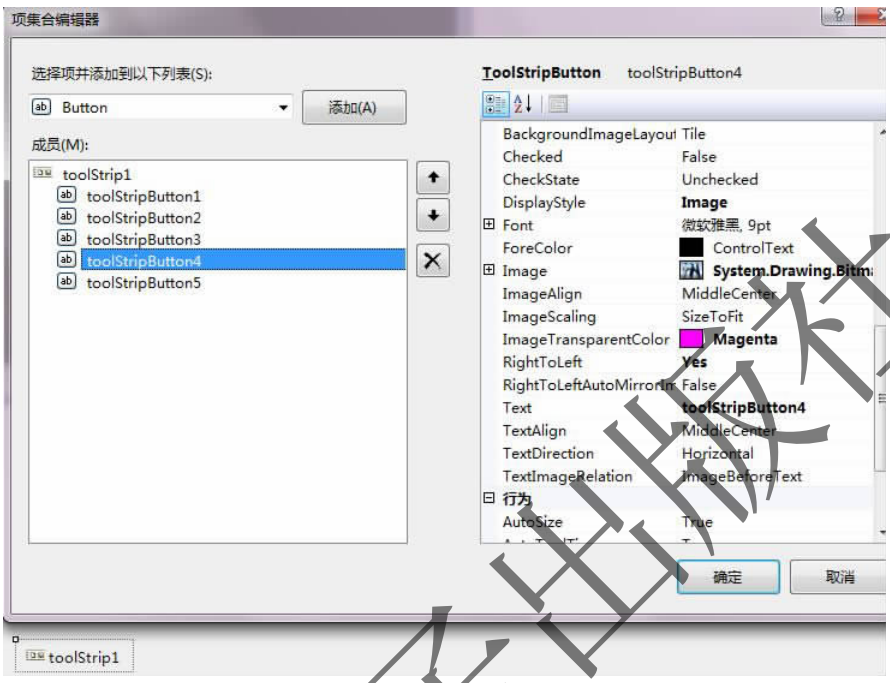

图 3-20　在原窗体上增加用户工具栏 ToolStrip

每个工具栏按钮所对应的代码，一般直接调用菜单栏中的相应的功能代码，这里我们有 5 个工具栏按钮，分别对应的菜单栏功能代码如下所示。

```
private void toolStripButton1_Click(object sender, EventArgs e)
{
    分组管理 ToolStripMenuItem1_Click(sender,  e);
}

private void toolStripButton3_Click(object sender, EventArgs e)
{
    课程成绩权重模块管理 ToolStripMenuItem_Click(sender, e);
}

private void toolStripButton2_Click(object sender, EventArgs e)
{
    实训打分 ToolStripMenuItem_Click(sender, e);
}

private void toolStripButton4_Click(object sender, EventArgs e)
{
    成绩审核 ToolStripMenuItem_Click(sender, e);
}
```

```
private void toolStripButton5_Click(object sender, EventArgs e)
{
    成绩导出 ToolStripMenuItem_Click(sender,e);
}
```

(3)在原窗体用户状态栏控件 StatusStrip1 中,增加 StatusLabel 类型的组件,如图 3-21 所示。

图 3-21　增加 StatusLabel 类型的组件

在状态栏属性中修改 toolStripStatusLabel2 的 Name 属性为 slTimerName,如图 3-22 所示。

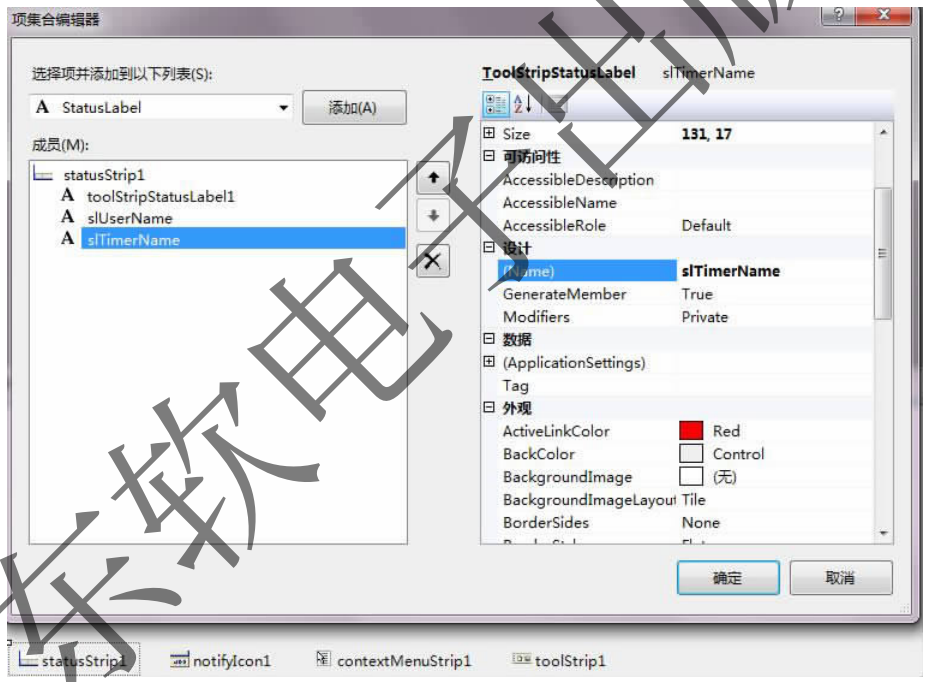

图 3-22　修改 toolStripStatusLabel2 的 Name 属性为 slTimerName

(4)在原窗体中增加 Timer 组件,并将 Timer 组件的 Enabled 属性值设置为 True。如图 3-23所示。

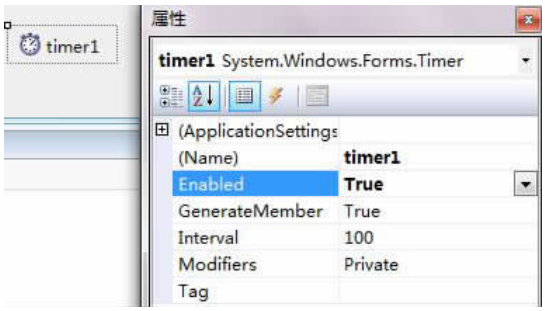

图 3-23　并将 Timer 组件的 Enabled 属性值设置为 True

在 Timer 控件中增加的代码如下。

```
private void timer1_Tick(object sender, EventArgs e)
    {
        slTimerName.Text ="当前时间:" + DateTime.Now.Year + "-" + DateTime.Now.Month + "-" +
DateTime.Now.Day + " " + DateTime.Now.Hour + ":" + DateTime.Now.Minute + ":" + DateTime.Now.Second;
    }
```

修改后的程序运行结果如图 3-24 所示。

图 3-24　修改后的程序运行结果

本章小结

本章以形成性考核系统中的登录模块及主界面模块为例，讲述了 C#.NET常用控件的使用，包括按钮控件，文本控件，标签控件，菜单控件，工具栏控件，状态栏控件等。