# 窗体应用程序
# （二）

王忠民

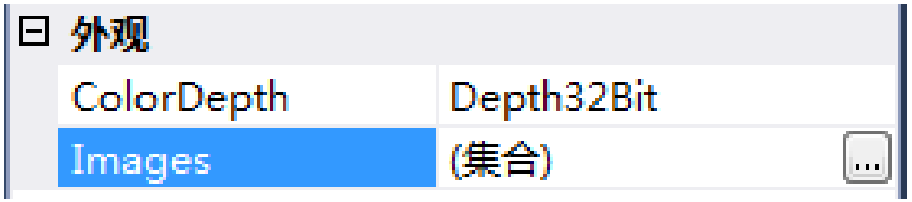
# 3 组件

# ImageList 控件

ImageList 控件提供了一个集合，可以在图像列表中存储任意大小的图像，但是每个图像大小应相同（否则显示比例不对）。

| 属性名 | 说明 |
| --- | --- |
| Images | 它是一个集合，存储在此 ImageList 控件中的图像 |
| ImageSize | 获取或设置该控件中各个图像的大小，默认为 16×16，但可以取 1~256 之间的值 |
| ColorDepth | 用来呈现图像的颜色数，它的默认值是 Depth8Bit |

**例：** 点击按钮，从 ImageList 中选择图像显示在 PictuBox 中。
首先要设置 Imagelist 的主要属性（尤其是 Images 集合）





```
private void button1_Click(object sender, EventArgs e)
      {   pictureBox1.Image = imageList1.Images[1];  }

private void button2_Click(object sender, EventArgs e)
      {  pictureBox1.Image = imageList1.Images[0];  }
```

# Timer 控件

又叫定时器，可以让程序每隔一定时间重复做一件事情。主要属性：

- Enabled　是否定时引发事件
- Interval　事件发生的频率，以**毫秒**为单位

Timer 控件的事件只有一个，就是 Tick 事件，每当指定间隔时发生
常用的两种方法： Start() 方法和 Stop() 方法，分别启动和停止计时器。

**例：**
在窗体中添加 ImageList 控件，保存一组图片。
再添加 PictureBox 控件用于显示图像。
添加 Timer 控件，设置其 Interval 为 1000，在 Tick 事件中实现图片 1 秒钟轮换一次。
添加“开始”和“取消”按钮分别用于控件计时开始和停止。


```
int count = 1;

private void timer1_Tick(object sender, EventArgs e)
  {
     if (count <= imageList1.Images.Count)
     {
        pictureBox1.Image = imageList1.Images[count - 1];
        count++;
     }
  }

private void button1_Click(object sender, EventArgs e)
  { timer1.Start(); }

private void button2_Click(object sender, EventArgs e)
  { timer1.Stop(); }
```

# 4　对话框

# OpenFileDialog

**常用属性：**

- InitialDirectory 对话框的初始目录
- Filter 对话框中显示的文件筛选器。例：
  “Word 文件 |*.doc| 文本文件 (*.txt)|*.txt| 所有文件 (*.*)|*.*"
- FilterIndex 在对话框中选择的文件筛选器的索引，如果选第一项就为 1
- FileName 第一个在对话框中显示的文件或最后一个选取的文件
- FileNames 获取对话框中所有选定的文件名，字符串数组
- Title 将显示在对话框标题栏中的字符
- AddExtension 是否自动添加默认扩展名
- DefaultExt 默认扩展名
- ShowHelp 启用 " 帮助 " 按钮
- Multiselect 是否可以选择多个文件

**常用事件：**

- FileOk 当用户点击 " 打开 " 或 " 保存 " 按钮时要处理的事件
- HelpRequest 当用户点击 " 帮助 " 按钮时要处理的事件

**常用方法：**

ShowDialog()： 返回类型 DialogResult ，如果用户在对话框中单击“确定”，则为 DialogResult.OK ；否则为 DialogResult.Cancel

```
private void button1_Click(object sender, System.EventArgs e)
{
   if (openFileDialog1.ShowDialog() == DialogResult.OK )
            textBox1.Text = openFileDialog1.FileName;
}
```

# SaveFileDialog

与 OpenFileDialog 相似。

与 OpenFileDialog 不同的属性：

OverwritePrompt　提示是否覆盖已有的文件

# ColorDialog

**常用属性：**

| | |
|---|---|
| AllowFullOpen | 允许用户定制颜色 |
| FullOpen | 打开定制颜色选项。如果 AllowFullOpen 为 false ，则 FullOpen 不起作用 |
| CustomColors | 使用 CustomColors 属性可以预置一个定制颜色数组，并可以读取用户定义的定制颜色。 int 型数组 |
| Color | 获取或设置用户选定的颜色。类型： Color |

```
private void button1_Click(object sender, EventArgs e)
    {
        if (colorDialog1.ShowDialog() == DialogResult.OK)
            textBox1.BackColor = colorDialog1.Color;
   }
```

# FontDialog

**常用属性：**

Font　设置或获取对话框中的字体
Color　设置或获取对话框中的字体
MaxSize　最大可用字号
MinSize　最小可用字号
ShowApply
ShowEffects
ShowHelp

```
private void button1_Click(object sender, EventArgs e)
  {
     if (fontDialog1.ShowDialog() == DialogResult.OK)
         {
            textBox1.Font = fontDialog1.Font;
            textBox1.ForeColor = fontDialog1.Color;
         }
  }
```

# FolderBrowserDialog

用来选择一个文件夹，从而读取这个文件夹下面的文件。

**常用属性：**

RootFolder　根文件夹的位置。对话框中只显示指定的根文件夹及其下层的所有子文件夹。

SelectedPath　对话框中的文件夹路径。 类型为 string

ShowNewFolderButton　对话框中包括“新建文件夹”按钮

```
private void  btn_Click(object sender, EventArgs e)
    {    if (folderBrowserDialog1.ShowDialog() == DialogResult.OK )
                textBox1.Text = folderBrowserDialog1.SelectedPath;
    }
```

# 5　菜单和工具栏

# MenuStrip

Form 的属性 MainMenuStrip 设置为本组件的名称，即可实现窗体与菜单的关联。


常用属性：
Items ： 所有菜单项的集合

可以设定每个菜单项的属性：
ShortcutKeys
ShowShortcutKeys
DropDownItems 下级菜单项

各菜单项的 Text ： 显示下划线，单字符命令等同于点击。
**例：** Text 设定为“关于 &A”

常用事件： Click

设定某个菜单项的程序

1、为该项的 Click 事件编程。

2、若其功能与某按钮相同，则在设计器中找到该菜单项的 Click 事件，在其右边点选相应按钮的事件响应程序。

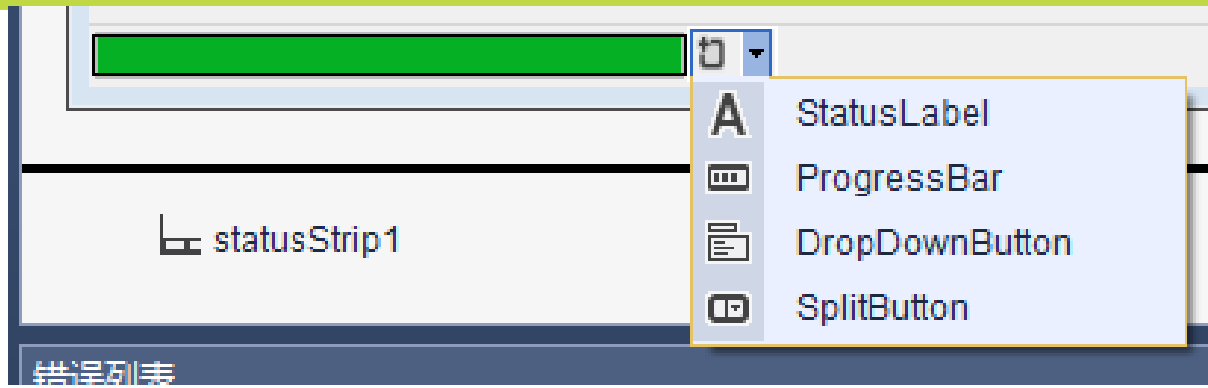
3、若其功能与某按钮相同，可编程使二者共享代码：

```
m1ToolStripMenuItem.Click += new System.EventHandler(button1_Click);
```

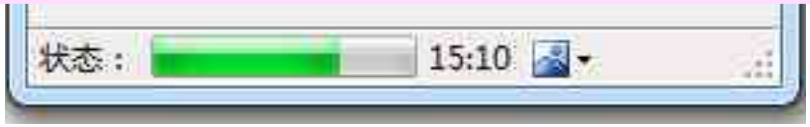
# StatusStrip

**状态栏**

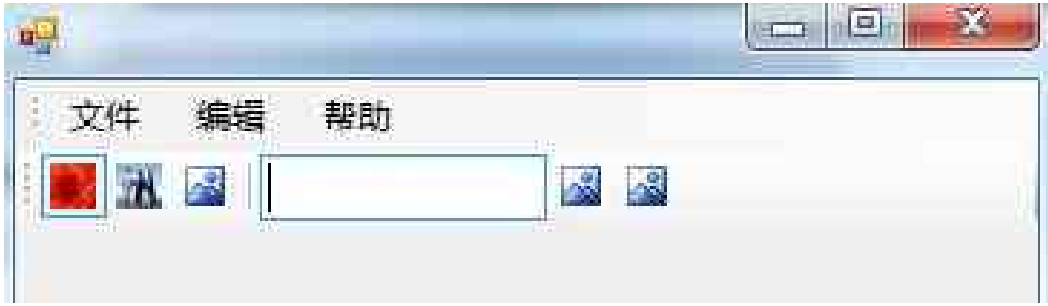
可添加标签、进度条等




```
private  int  progress = 0;        //  需添加 timer ，设置 interv=500 ，并 start
private void  timer1_Tick(object sender, EventArgs e)
 {
     progress= progress<100 ? progress+20 : 0;
     toolStripProgressBar1.Value = progress ;
  }
```



# ToolStrip




**工具条**

可以添加多种形式的工具：
Label , Buttoon , Combobox , Separator , …

各自的属性可单独设置：
DisplayStyle: ImageAndText , Image , Text , None
Image :
Text： 字符串
TextImageRelation: Overlay , TextBeforeImage , ImageBeforeText , TextAboveImage , ImageAboveText


**作业 4 ：**

编制自己的浏览器（地址输入栏， GO ，后退，前进，刷新）。

在地址栏输入地址后，按回车键也可以转到相应网页。

点击链接时，新页面在本浏览器打开，不创建新窗口。

地址栏始终显示当前网页地址。

```
private void webBrowser1_NewWindow(object sender, CancelEventArgs e)
    {
        e.Cancel = true; // 防止 IE 弹窗
        string  url = webBrowser1.StatusText;// 获取鼠标点击的 URL
        webBrowser1.Navigate(url);
    }
```

# 6 容器

# Panel 和 GroupBox

Panel 控件可以称作面板控件，它主要用来将控件分组，它是一个能够包含其他控件的控件。例如，可以将用户性别“男”和“女”放在一个面板中。默认情况下它是不显示的。

GroupBox 控件也叫组合框控件，和 Panel 控件非常相似，也可以用来包含其他的容器控件。

二者的不同之处：Panel 控件没有标题，但是可以显示滚动条；而 GroupBox 控件有边框，可以显示标题，但是不能显示滚动条。


GroupBox 属性如表所示：

| 属性名 | 说明 |
| --- | --- |
| AutoSize | 指定控件是否自动调整自身的大小以适应其内容的大小 |
| FlatStyle | 获取或设置控件的平面样式外观 |
| TabStop | 指定用户能否使用 Tab 键将焦点放到该控件上 |

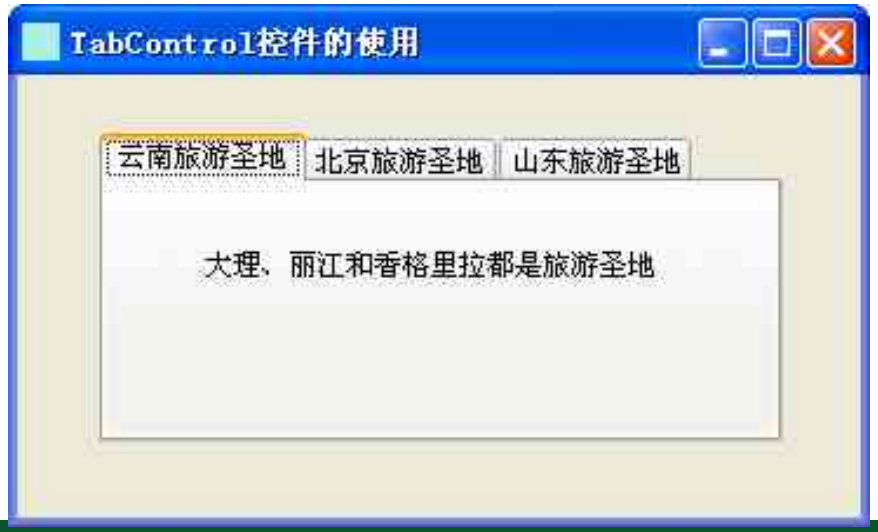
需要注意的是：在为控件进行分组的时候需要标题时使用 GroupBox 控件；如果不需要标题但是需要使用滚动条时就用 Panel 控件；如果都不要求的情况下使用任意一个控件都能达到分组的效果。

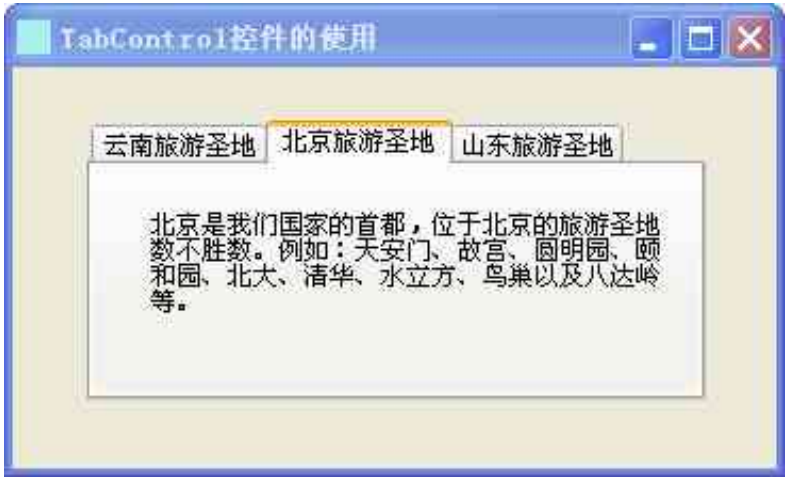
# TabControl 控件

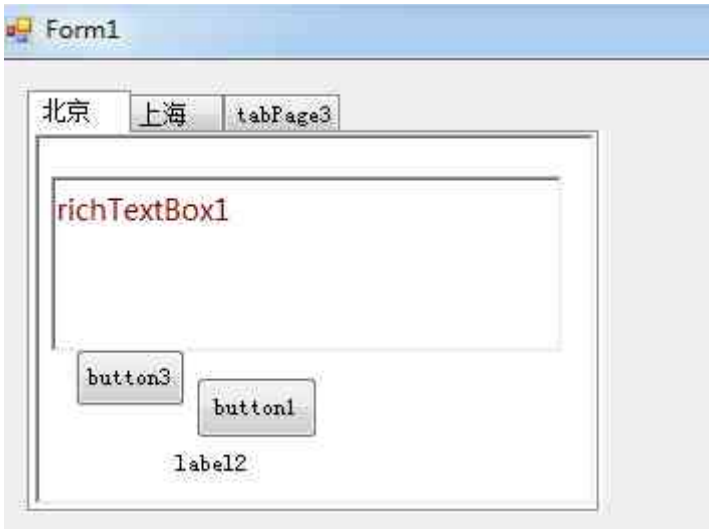
TabControl 控件又称为选项卡控件，可以显示多个选项卡，每一个选项卡都包含一个选项卡页（ TabPage ）控件。 TabControl 控件是所有 TabPage 的集合。该控件包括 4 个常用属性：

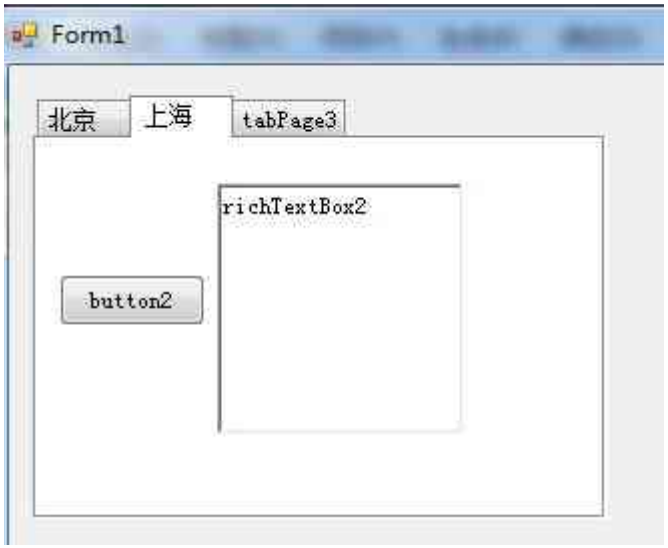
- SelectedIndex　当前选定的选项卡页的索引
- SelectedTab　当前选定的选项卡页
- TabPages　TabControl 控件中所有选项卡页的集合
- RowsCount　获取控件的选项卡条中当前正显示的行数

例如添加新的窗体，在窗体中添加 TabControl 控件，然后为 TabPages 属性中的 Text 属性的值分别设置为“云南旅游圣地”、“北京旅游圣地”和“山东旅游圣地”。然后分别添加一个 Label 控件显示旅游圣地信息，最终运行效果如图 8 和图 9 所示。





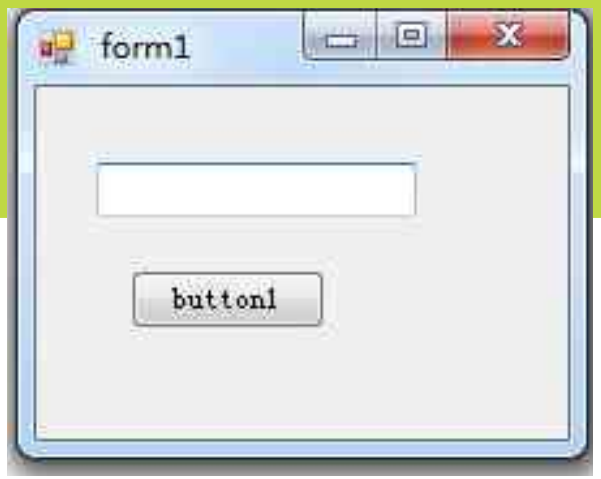

各 TabPage 中的控件分别编程，或交叉访问，与此前的方法一样。例如：

```
private void button2_Click(object sender, EventArgs e)
   {
      richTextBox1.LoadFile("G:\\shiyan.rtf");      // LoadFile 只能读 rtf 文件
   }
```

# 7　窗体的其它操作

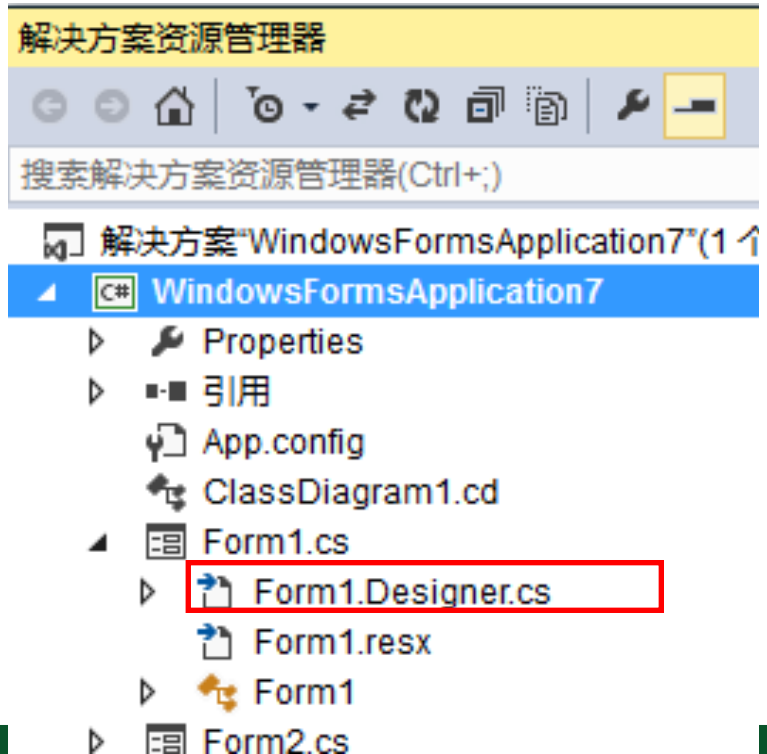
# 窗体之间的交互

*若要使窗体中的控件可被外部访问，需要将该控件设置为 public 。*





在解决方案资源管理器中，找到“ Form1.Designer.cs”，打开代码




把 Form1 中 textBox1 的 private 改为 public

```
Windows 窗体设计器生成的代码

public System.Windows.Forms.TextBox textBox1;
private System.Windows.Forms.BindingSource bir
private System.Windows.Forms.Button button1;
```

在 Form1 中打开 Form2 ：

```
private void button1_Click(object sender, EventArgs e)
      {
         Form2   f2 = new   Form2();
         f2.Show(this);       // this 成为 f2 的 owner ；若 Show() 则 f2 无
owner
      }
```

使窗体最大化最小化：

修改其 WindowState 属性

（ Minimized 、 Maximized 、 Normal ）

例：

```
f2.WindowState = FormWindowState.Minimized;
```

Form2 中添加代码：

```
private   Form1   f1;

private  void  Form2_Load(object sender, EventArgs e)
{
   f1 = (Form1)this.Owner;
}
private  void  button1_Click(object sender, EventArgs e)
{
   f1.Show();
   f1.textBox1.Text = Convert.ToString(DateTime.Now);
}

private void button2_Click(object sender, EventArgs e)
{
   f1.Hide();  //  隐藏，但还可以用 Show() 显示出来。 Close 关闭
}
```