窗体应用程序

王忠民

内容简介

窗体是一个窗口或对话框，是存放各种控件的容器，可用来向用户显示信息。

C# 中窗体主要分为两种类型：

- 普通窗体： 又称单文档窗体，它又可以分为模式窗体和无模式窗体。
- MDI 父窗体： 又称多文档窗体，其中可以放置普通子窗体。

1 Windows 窗体控件概述

窗体控件是用户可与之交互以便输入或操作数据的对象。

窗体提供了许多控件和组件，大多数的控件都派生于 Control 类，Control 类定义了控件的基本功能，所以许多属性和事件都相同。 这些控件还有自己的属性、方法和事件，便于控件适合于特定的用途。

窗体控件包括很多种，如：公共控件、对话框、组件、菜单、容器等。

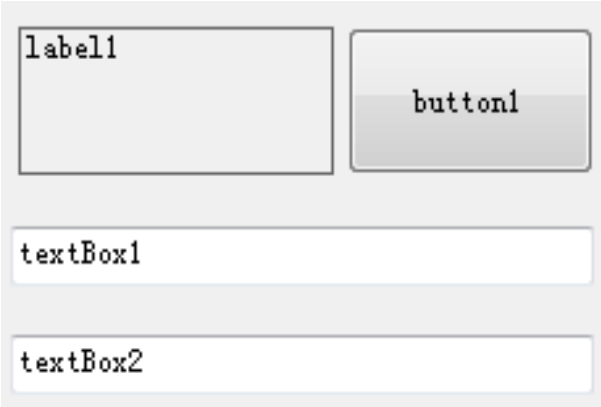
大多数控件共有的常见属性

属性名	说明
Name	用来标识控件的名称
Text	控件上的文本内容
Size	控件的大小（像素为单位）
Location	控件左上角相对于其容器左上角的坐标
Font	控件中文本的字体
BackColor	控件或组件的背景色
ForeColor	前景色，用于显示文本
Cursor	指针移过控件时显示的光标
Enabled	是否启用该控件，默认 True
Visible	确定控件是可见的还是隐藏的，默认为 True

控件共有的常见事件

事件名	说明
Click	单击组件触发该事件
MouseUp	在鼠标指针在组件上方并释放鼠标按钮时发生
MouseMove	鼠标移过组件时发生
MouseLeave	在鼠标离开控件的可见部分时发生
Validated	验证控件后发生
Validating	在控件验证时发生
VisibleChanged	在更改控件的可见性时发生

```
 private void label1_MouseLeave(object sender, EventArgs e)
        {    textBox1.Text = "mouse leave";   }
private void label1_MouseEnter(object sender, EventArgs e)
        {   textBox1.Text = "mouse enter";    }
private void label1_MouseDown(object sender, MouseEventArgs e)
        {   textBox1.Text = "mouse down";      }
private void label1_MouseUp(object sender, MouseEventArgs e)
        {   textBox1.Text = "mouse up";        }
private void label1_DoubleClick(object sender, EventArgs e)
        {   textBox2.Text += "*";          }
private void button1_MouseMove(object sender, MouseEventArgs e)
        {   textBox1.Text = "mouse move on button1";  }
private void button1_Click(object sender, EventArgs e)
        {   if (textBox1.BackColor == Color.Yellow)
                textBox1.BackColor = Color.LightPink;
            else
                textBox1.BackColor = Color.Yellow;
        }
private void textBox1_BackColorChanged(object sender, EventArgs e)
        { textBox2.Text = " textBox1:"+textBox1.BackColor;}
```



设定字体、字号：

```
textBox2.Font = new   Font("宋体",20, FontStyle.Bold);
```

Regular , **Bold ,** *Italic ,* Underline , ~~Strikeout~~

设定位置：

```
label1.Location = new  Point(100, 250);
```

或者

```
label1.Top = 50;
label1.Left = 10;
```

设置光标形状：

```
label1.Cursor = Cursors.Cross;
```

Cross, Arrow, Hand, Help , AppStarting

编程向窗体中添加控件

```
private void button1_Click(object sender, EventArgs e)
        {
            TextBox   box1 = new   TextBox();    // name
            box1.Location = new   Point(50, 250);
            box1.Text = "hello .....";
            this.Controls.Add(box1);
            Controls.Remove(label1);
}
```

改变窗体背景色

```
private void button4_Click(object sender, EventArgs e)
        {
            this.BackColor = Color.DarkBlue;
        }
```

2　公共控件

Button 控件

Button 控件表示为简单的按钮，派生于 ButtonBase 类。它通常呈现为一个矩形按钮，允许用户通过单击来执行某些操作。

属性名	说明
Image	该属性用于在控件上显示的图像
ImageAlign	该属性用于设置控件上的图像显示在什么地方
ImageIndex	在控件上显示的 ImageList 中的图像的索引
FlatStyle	确定当用户将鼠标移动到控件上并单击时该控件的外观

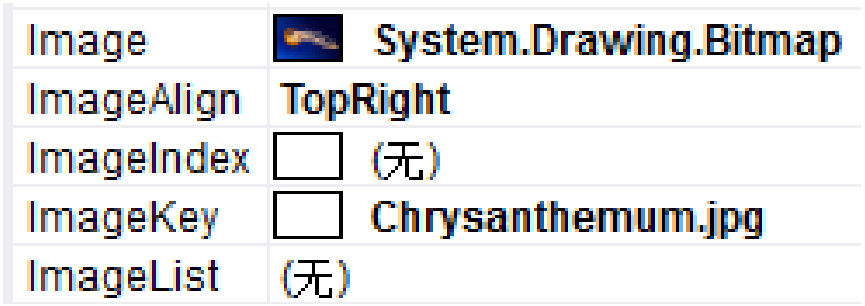
FlatStyle 属性可以控制按钮控件的外观， FlatStyle 是一个枚举类型，它有 4 个枚举值。具体说明如下所示：

- Standard　默认值，设置控件外观为三维
- Flat　该控件以平面显示
- Popup　该控件以平面显示，当鼠标指针移动到该控件时外观为三维
- System　该控件的外观是由用户的操作系统决定的

Label 控件

Label 控件主要用于显示用户不能编辑的文本或图像。用户常常使用它对窗体上的其他各种控件进行说明或标注，例如“用户名”、“用户密码”和“出生日期”等。

属性名	说明
TextAlign	确定文本控件的显示位置
Image	设置要在控件上显示的图像
Font	设置控件中文本的字体
AutoSize	是否根据字号自动调整大小
BorderStyle	用于设置控件边框的样式
ImageAlign	设置显示图像在控件的什么位置



LinkLabel 控件

可以使用 Label 控件的地方，都可以使用 LinkLabel 控件。它可将文本的一部分指向某个对象或 Web 页的链接。 LinkLabel 除了具有 Label 控件的所有属性、方法和事件以外，还有针对超链接的属性和事件：

属性名	说明
LinkArea	设置激活链接的文本区域 （ start,length ）
LinkColor	设置超链接处于默认状态时的颜色
ActiveLinkColor	确定当用户单击超链接的颜色

与 Lable 控件相比， LinkLabel 控件多了一个 LinkClicked 事件。

```
private void linkLabel1_LinkClicked(object sender,
LinkLabelLinkClickedEventArgs e)
  {  System.Diagnostics.Process.Start("http://www.baidu.com/");
  }
```

TextBox 控件

TextBox 控件用于获取用户输入的信息或向用户显示文本。通常用于可编辑文本，不过也可成为只读控件。 TextBox 控件可以显示多行。

属性名	说明
Multiline	控件的文本是否能够跨越多行
ScrollBars	如 Multiline 设置为 True ，指示控件显示哪些滚动条
WordWrap	如 Mulitiline 设置为 True ，指示控件是否自动换行
ReadOnly	设置文本框是否是只读
AcceptsReturn	True 则多行编辑控件中允许输入回车符， false 则在回车时相当于按下窗体的默认回车按钮。
AcceptsTab	制表符作为多行编辑控件 (Mutiline 为 true) 的制表符输入，还是控制窗体中控件焦点转移
MaxLength	限制在 TextBox 控件中输入的最大字符数量
UseSystemPasswordChar	文本是否以默认的密码字符显示，默认为 False
PasswordChar	指示密码输入显示的字符
CharacterCasing	是否会改变输入的大小写。该值有 3 个： Normal（默认）、 Upper 和 Lower

BackColor , ForeColor , Font , ImeMode …

常用方法：

AppendText 向文本框的当前文本追加文本。
Clear 从文本框控件中清除所有文本。
ClearUndo 从该文本框的撤消缓冲区中清除关于最 近操作的信息。
Copy 将文本框中的当前选定内容复制到剪贴板。
Cut 将文本框中的当前选定内容移动到剪贴板中。
Focus 为控件设置输入焦点。
Hide 对用户隐藏控件。
Show 向用户显示控件。
Paste 把剪贴板的内容粘贴到文本框中。
Select 选择控件中的文本。
SelectAll 选定文本框中的所有文本。
Undo 撤消文本框中的上一个编辑操作。

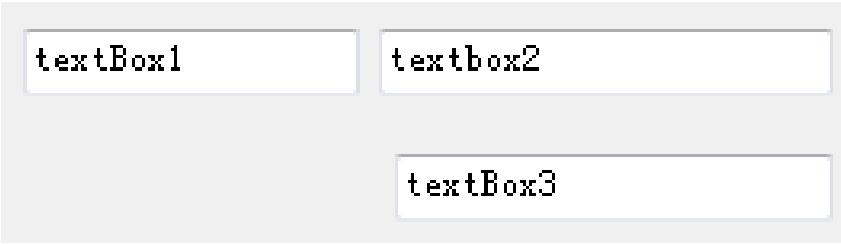
常用事件：

TextChanged　文本框的内容发生变化
Click
KeyDown　有按键按下
KeyUp　　有按键抬起
MouseDown
MouseUp
MouseEnter
MouseLeave
MouseDoubleClick
MouseMove
MouseHover　鼠标在控件内悬停了一定时间

```
private void  textBox1_KeyUp(object sender, KeyEventArgs e)
      {     textBox2.Text = textBox1.Text;        }
private void textBox1_MouseHover(object sender, EventArgs e)
      {     textBox2.Text = “box1 hover!”;        }
```

关于“验证”

如果属性 CauseValidation 设为 True ， 则：
控件内容验证时触发 Validating 事件
控件内容验证成功后触发 Validated 事件



与焦点有关的事件顺序：
Enter　// 进入控件时发生
GotFocus　// 在控件接收焦点时发生
Leave　// 输入焦点离开控件时发生
Validating　// 控件数据校验时发生（未成功）
Validated　// 数据校验后发生（无论成功与否）
LostFocus　// 失去焦点时发生

```
private void textBox1_Validating(object sender, CancelEventArgs e)
  {   if (textBox1.Text != "ok")
            textBox1.Focus();
   }
private void textBox1_Validated(object sender, EventArgs e)
   {  string  s = DateTime.Now.ToString();
      textBox3.Text = s;            }
```

关于 IME （ Input Method Editors ）

ImeMode 常用取值：

- On 打开中文输入法
- Off 关闭中文输入法
- Disable 禁止从键盘打开输入法

```
textBox1.ImeMode = System.Windows.Forms.ImeMode.On;
```

MaskedTextBox 控件

MaskedTextBox 控件是一个增强的 TextBox 控件。通过使用 Mask 属性，不需要在应用程序中编写任何的验证逻辑。从指定的列表中选择预定义的掩码说明，或者选择“自定义”定义一个自定义掩码，即可指定下面的输入：

- 可选和必需的输入字符
- 应该直接出现在 MaskedTextBox 中的字符，或者说掩码的原义字符
- 掩码中给定位置所需要的输入类型，例如只允许输入字母或数字
- 输入字符的处理，例如字符的大小写转换

当 MaskedTextBox 控件运行显示时会将掩码表示为一系列提示字符和可选的原义字符，表示一个必需或可选输入的每个可编辑掩码位置都显示为单个提示字符。 MaskedTextBox 控件的常用属性如表所示。

属性名	说明
Mask	设置控件此控件允许的输入的字符串（见下页）
PromptChar	指定自定义提示字符
HidePromptOnLeave	当控件失去输入焦点时用户能否看到提示字符，默认为 False
HideSelection	当编辑控件失去焦点时，应隐藏选定内容
TextMaskFormat	指示在从 Text 属性中返回字符串时是否包含原义字符和（或）提示字符

常用事件： TextChanged　控件中的文本发生变化时 。

```
 private void maskedTextBox1_TextChanged(object sender, EventArgs e)
{       textBox1.Text = maskedTextBox1.Text;     }
```

利用 Validating 验证其是否按规定填写： MaskFull 属性可以判断所有必要输入和选择输入是否都已经完成， MaskComplete 属性可以判断所有必要输入都已经完成。

```
private void maskedTextBox1_Validating(object sender, CancelEventArgs e)
    {
            if (maskedTextBox1.MaskFull)   //  .MaskCompleted
              {
               }
   }
```

RadioButton 控件

RadioButton 控件是一个单选按钮，也称为选项按钮，表示从多个可选项中选择一项，即只允许用户从几个选项中选择一个答案。例如选择用户性别时“男”和“女”就是单选按钮。

可以使用分组框或面板把一组单选按钮组合起来，确保只有一个单选按钮能被选中，且跟其它组的 RadioButton 状态无关。

属性名	说明
Checked	该控件是否已经选中，默认为 False
CheckAlign	RadioButton 控件中复选框与控件文字的相互位置
Appearance	默认为 Normal ，按通常情况显示。也可显示为 Button
FlatStyle	确定当用户将鼠标移动到控件上并单击时该控件的外观

RadioButton 控件最常用的事件是 CheckedChanged 和 Click ：

- 控件的 Checked 属性值更改时会触发 CheckedChanged 事件（如果不是点击的该控件，就不会触发 Click 事件）
- 每次单击 RadioButton 控件都会引发 Click 事件（连续点击该控件不会触发 CheckedChanged 事件）

CheckBox 控件

CheckBox 控件表示复选框，表示某个选项是否被选中。例如用户可以选择多个爱好，这些爱好可以使用 CheckBox 控件。它和 RadioButton 控件的相同处在于都是表示用户的真 / 假选项；而不同之处在于 RadioButton 控件一次只能选择一个单选按钮， CheckBox 控件则表示可以选择任何的数量。

属性名	说明
Checked	表示该控件是否已经选中，默认为 False ，选中后为 True 。其值变化会与 CheckState 取值联动。
CheckState	获取或设置 CheckBox 的状态，默认为 Unchecked ，选中后变为 Checked 。 如果 ThreeState 属性被设置为 True ，则 CheckState 还可能返回 Indeterminate 。
ThreeState	指示 CheckBox 是否会允许三种状态（ Checked 、 Unchecked 、 Indeterminate 。），而不是两种状态

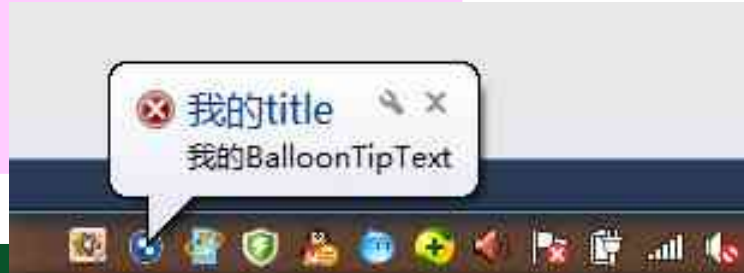
NotifyIcon 控件

NotifyIcon 控件指定通知区域中创建图标。 常见的属性如下：

- ◆BalloonTipIcon　与气球状工具提示关联的图标。它的值有 4 个，分别是 None（默认值）、 Info 、 Warnning 和 Error
- ◆BalloonTipText　与气球状工具提示关联的文本
- ◆BalloonTipTitle　与气球状工具提示关联的标题
- ◆ContextMenuStrip　当用户右击该图标时显示的菜单
- ◆Icon　系统栏中显示的图标，并且该属性只接受 .ico 格式的文件
- ◆Text　当鼠标悬停在该图标上时显示的文本
- ◆Visible　确定该控件是可见的还是隐藏的

每个 NotifyIcon 控件都在状态区域显示一个图标。
某个事件触发后，调用 ShowBalloonTip 方法使气球提示显示出来：

```
private void textBox1_TextChanged(object sender, EventArgs e)
 {
   notifyIcon1.ShowBalloonTip(1000);
 }
```



RichTextBox 控件

RichTextBox 控件用于显示、输入和操作格式文本，是一个可编辑控件，又称为富文本控件。RichTextBox 控件除了做 TextBox 控件所做的每件事外，还可以显示字体、颜色和连接，从文件加载文本和加载嵌入图像，以及查找指定的字符串。另外，RichTextBox 控件还可以打开、存储 .rtf 格式的文件、ASCII 文本格式文件及 Unicode 编码格式的文件。

属性名	说明
CanFocus	该值指示控件是否可以接收焦点
CanRedo	控件内发生的操作是否可以重新应用
CanUndo	该值指示在文本框控件中能否撤销前一操作
SelectionIndent	应用到当前选定文本或插入点的左边的当前缩进距离
SelectionRightIndent	该控件右边缘与选中文本或当前插入点添加的文本的右边缘之间的距离
SelectedText	获取或设置控件内选中的文本（无格式）
SelectedRtf	获取或设置控件内选中的 rtf 格式文本
SelectionFont	获取或设置当前选定文本或插入点的字体
SelectionColor	获取或设置当前选定文本或插入点的文本颜色
SelectionLength	当前选定文本的长度

RichTextBox 控件文本的任何部分，都可以指定格式。被选定的文本可以设置字符和段落格式，例如把文本改为粗体或斜体、改变颜色，创建上标和下标。设置左右缩进和悬挂式缩进，调整段落的格式等。

RichTextBox 控件支持对象的嵌入。插入到控件中的每个对象，都代表 OLEObject 对象。用这样的控件，就可以创建包含其它文档或对象的文档。例如，可创建这样的文档，它有一个嵌入的 Microsoft Excel 电子数据表格、或 Microsoft Word 文档、或其它已在系统中注册的 OLE 对象。

为了把一个对象插入到 RichTextBox 控件中，只需简单地拖动一个文件，或拖动另一应用程序（如 Microsoft Word ）所用文件的突出显示的区域，然后将所拖内容直接放入控件。

常用的方法：

AppendText(string s) 追加字符串到控件的内容。

Clear() 清空

LoadFile (string path)

SaveFile (sting path)

Find(string s) 返回值 int ，即所查找字符串的起始位置

Focus （ ）把焦点移到该控件

```
private void button1_Click(object sender, EventArgs e)
   { if (this.richTextBox1.SelectionLength > 0)
        {
          Font font = new  Font(FontFamily.GenericMonospace, 10, FontStyle.Bol
          this.richTextBox1.SelectionFont = font;
        }
         else MessageBox.Show(" 请先选择要编辑的文本 ");      }
 private void button2_Click(object sender, EventArgs e)
   { if (this.richTextBox1.SelectionLength > 0)
           this.richTextBox1.SelectionColor = Color.Red;
        else  MessageBox.Show(“ 请先选择要编辑的文本” );     }
 private void button3_Click(object sender, EventArgs e)
   {   this.richTextBox1.SaveFile (" G:\\ 实验文件 .rtf");     }
private void button4_Click(object sender, EventArgs e)
      {  richTextBox1.LoadFile("G:\\test.rtf") ; }
```


```
private void button1_Click(object sender, EventArgs e)
    { richTextBox1.Copy();  } // 把选中部分 copy 到剪切板
private void button2_Click(object sender, EventArgs e)
    { richTextBox1.Cut(); } //  剪切选中部分
private void button3_Click(object sender, EventArgs e)
    { richTextBox2.Paste(); } //  粘贴到右边的控件
private void button4_Click(object sender, EventArgs e)
    { richTextBox2.Clear(); } // 清空
private void button5_Click(object sender, EventArgs e)
    {  richTextBox2.AppendText( richTextBox1.SelectedText ); }
private void button6_Click(object sender, EventArgs e)
    {  string s = richTextBox1.SelectedRtf;  //rtf 格式
       richTextBox2.SelectedRtf = s;   }
```

PictureBox 控件

PictureBox 控件用于显示 bmp 、 gif 、 jpeg 、图标等图形，也称为图片框。

属性名	说明
Image	在该控件中显示的图像
ImageLocation	获取或设置要在 PictureBox 中显示的图像的路径或 URL
InitialImage	获取或设置在加载主图像时显示在该控件中的图像
SizeMode	控制 PictureBox 将如何处理图像位置和控件大小

SizeMode 属性返回值是 PictureBoxSizeMode 枚举类型：

- Normal 默认值，图像被置于 PictureBox 控件的左上角。如果图像比包含它的 PictureBox 大，则该图像将被裁剪掉
- AutoSize PictureBox 自动调整大小，使其等于所包含的图像大小
- StretchImage 图像被拉伸或收缩（不按原始比例），以适合 PictureBox 的大小
- Zoom 图像按其原有大小比例被增加或减小，以适应 PictureBox
- CenterImage 图像在 PictureBox 中居中显示（如果图像比 PictureBox 大，则图片外边缘将被裁剪掉）

常用方法：

public void Load (string url)　把 url 的图片文件加载显示到控件。调用 Load 方法可将 ImageLocation 属性设置为 url 的值。

pictureBox1.Load ("g:\\leaf.jpg");

ListBox 控件

ListBox 控件可以称为列表框，列表中的每一个元素都被称为一个项。该控件用于显示一个完整的列表项，用户可从中选择一个或多个选项，并添加、删除一项或多项。

属性名	说明
Items	列表框中的所有选项，使用此属性可增加和删除选项
SelectedIndex	列表框中当前选定项目的索引号，如果可以一次选择多个选项，此属性包含选中列表中的第一个选项
SelectedItem	获取当前选中的项
SelectedItems	获取当前选中项的集合
SelectionMode	指示列表框将是单项选择、多项选择还是不可选择

SelectionMode 属性用于设置列表框的选择模式，它有 4 个值：

- None　不能选择任何选项
- One　一次只能选择一个选项
- MultiSimple　鼠标点击选择多个选项
- MultiExtended　可以选择多个选项，还可以使用 Ctrl 键、 Shift 键和箭头进行选择

ListBox 的常用方法：

方法名	说明
FindString	查找控件中以指定字符串开始的第一个项
ClearSelected	取消控件中的全部选项
GetSelected	返回一个表示是否选择一个选项的值
SetSelected	选择或清除控件中制定项的选择
GetItemText	返回指定项的文本表示形式

常用的事件：
SelectedIndexChanged 事件，当选中选项的索引改变时触发该事件。

```
private void btnAdd_Click(object sender, EventArgs e)
      {
         if (string.IsNullOrEmpty(textBox1.Text))
            MessageBox.Show(" 请输入城市名 ");
         else
         {
            listBox1.Items.Add(textBox1.Text);
            textBox1.Text = “”;    // 清空文本框，便于输入
          }
      }
private void btnDel_Click(object sender, EventArgs e)
      {
         if (listBox1.SelectedItem != null)
            listBox1.Items.RemoveAt(listBox1.SelectedIndex);
      }
private void btnPre_Click(object sender, EventArgs e)
      { if (listBox1.SelectedIndex > 0)  listBox1.SelectedIndex--; }
private void btnNex_Click(object sender, EventArgs e)
      { if (listBox1.SelectedIndex <listBox1.Items.Count -1)  listBox1.SelectedIndex++; }

// 显示所选项的文本内容：
private void button1_Click(object sender, EventArgs e)
    {    textBox1.Text = listBox1.GetItemText( listBox1.SelectedItem);   }
```

CheckedListBox 控件

CheckedListBox 控件显示一个 ListBox 控件，但是每项的左边显示一个复选框。可以选择一项或多项，它除了可以使用 ListBox 控件的所有属性和方法外，还有一些自己的属性和方法。

属性	说明
CheckedIndices	CheckedListBox 中选中索引的集合
CheckedItems	CheckedListBox 中选中项的集合
ThreeDCheckBoxes	获取或设置一个值，指示复选框是否有 Flat 或 Normal 的 ButtonState
SelectionMode	指示列表框将是单项选择、多项选择还是不可选择

CheckedListBox 控件中常用的特有方法有 4 个：

- bool GetItemChecked(int idx) 返回值表示指定索引项是否被选中
- SetItemChecked() 将指定索引项 CheckState 的属性值设置为 Checked
- GetItemCheckState()

返回指定索引项的复选状态（ Checked/Unchecked ）

- SetItemCheckState() 设置指定索引处项的复选状态

```
private void button1_Click(object sender, EventArgs e)
  {
      if (checkedListBox1.GetItemCheckState(1) == CheckState.Checked)
          MessageBox.Show("1,checked");
  }
```

ListView 控件

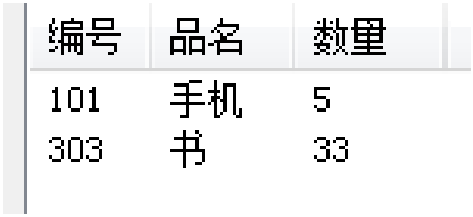
ListView 控件又称列表视图，利用它可以创建类似 Windows 资源管理器右边窗口的界面。

属性名	说明
Items	ListView 控件中所有项的集合
Columns	详细信息视图中显示的列
FullRowSelect	当选中一项时，它的子项是否同该项一起突出显示
MultiSelect	是否允许选择多项
SelectedItems	选中的项的集合
View	指定显示 5 种视图中的一种：List、Detail、LargeIcon、SmallIcon、Title

Items 属性表示包含在控件中的所有项的集合，它的每一项都是一个 ListViewItem（列表视图项），可以使用 Add() 方法向列表视图中添加一项，用 Remove、RemoveAt 删除项。

ListView 控件最常用的事件是 MouseDoubleClick 事件，双击鼠标时就会触发该事件。

例 1 ：手工设置 ListView 的属性。



①创建 ListView ；
②设置控件的 View 为 Detail ；
③点击控件的 Columns 的集合，设置每列的抬头；
④点击控件的 Items 的集合，设置每行的编号，
再分别点击其 SubItems 集合，设置各行内容。

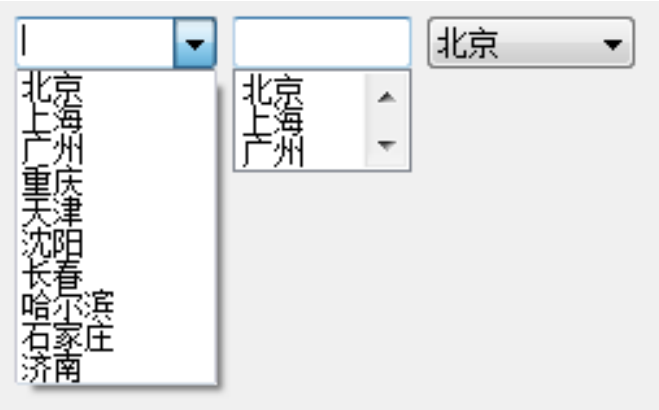
例 2 ：程序修改 ListView 。

```
private void button1_Click(object sender, EventArgs e)
 {    ListViewItem   lvi = new  ListViewItem("305");
      lvi.SubItems.Add(" 签字笔 ");
      lvi.SubItems.Add("53");
      listView1.Items.Add(lvi);
  }
```

```
listView1.Items.RemoveAt(1);
```

ComboBox 控件

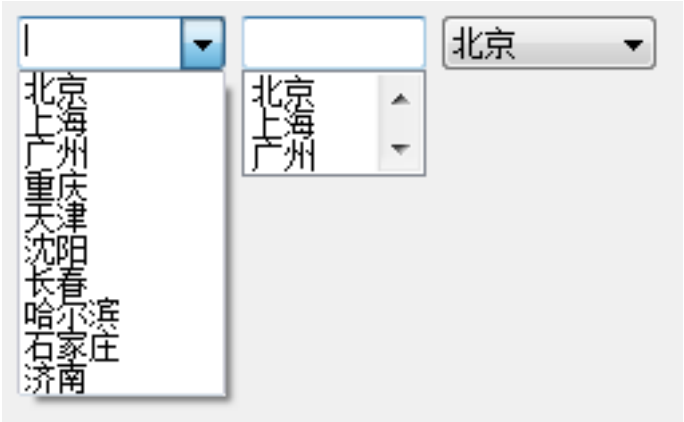
ComboBox 用于在下拉组合框中显示数据。分两个部分显示：一部分是允许用户输入列表项的文本框；另一部分是列表框。



属性名	说明
DropDownStyle	确定控件要显示的样式，默认值是 DropDown
Items	ComboBox 控件中的所有列表选项
SelectedIndex	当前选中选项的索引值
SelectedItem	当前选中的选项
SelectedText	文本框部分被选中的文本
SelectionStart	获取或设置 ComboBox 控件选中文本的起始索引
MaxLength	允许输入的控件文本框的最大字符个数

DropDownStyle 表示控件的样式，它的值有 3 个：

选项	下拉列表	编辑输入	自动定位
DropDown	点开列表选择	可编辑输入	
Simple	从列表滚动选择	可编辑输入	根据输入自动定位
DropDownList	点开列表选择	不能编辑输入	



常用的事件： SelectedIndexChanged

常用操作： comboBox1.Items.Add() , RemoveAt ,

例： (1) 当选项发生变化时，在 textBox1 显示选项的索引号，在 textBox2 显示选项字符串。 (2) 在 comboBox1 手工输入文本后，按 button1 将在 textBox1 中显示其文本。

```
private void comboBox1_SelectedIndexChanged(object sender, EventArgs e)
 {
   textBox1.Text = Convert.ToString(comboBox1.SelectedIndex);
   textBox2.Text = (string)comboBox1.SelectedItem;  //  取得所选的字符串
 }

private void button1_Click(object sender, EventArgs e)
 {
    textBox1.Text = comboBox1.Text;
 }
```

DateTimePicker

常用属性：

属性	说明
Value	获取或设置控件的日期 / 时间值， DateTime 类型。 Value.Year , Value.Month , Value.Day
Text	此属性返回的字符串等效于具有相应格式的 Value 值
Format	控件取值的格式： Long , Short , Time , Custom
CustomFormat	格式字符串见下表

常用事件： ValueChanged

格式字符串	描述
d	一位数或两位数的天数。
dd	两位数的天数。 一位数天数的前面加一个零。
ddd	三个字符的星期几缩写。
dddd	完整的星期几名称。
h	12 小时格式的一位数或两位数小时数。
hh	12 小时格式的两位数小时数。 一位数数值前面加一个 0 。
H	24 小时格式的一位数或两位数小时数。
HH	24 小时格式的两位数小时数。 一位数数值前面加一个 0 。
m	一位数或两位数分钟值。
mm	两位数分钟值。 一位数数值前面加一个 0 。
M	一位数或两位数月份值。
MM	两位数月份值。 一位数数值前面加一个 0 。
MMM	三个字符的月份缩写。
MMMM	完整的月份名。
s	一位数或两位数秒数。
ss	两位数秒数。 一位数数值前面加一个 0 。
t	一个字母 AM/P.M. 缩写 (AM 显示为“ A”) 。
tt	两个字母 AM/P.M. 缩写 (AM 显示为“ AM”) 。
y	一位数的年份（ 2001 显示为“ 1” ）。
yy	年份的最后两位数（ 2001 显示为“ 01” ）。
yyyy	完整的年份（ 2001 显示为“ 2001” ）。

MMMM dd, yyyy – dddd　　十二月 16, 2014 - 星期二

MonthCalendar

可以选择某个范围的日期

属性	说明
MaxSelectionCount	可以选择的总天数
SelectionStart	所选取的开始日期
SelectionEnd	所选取的结束日期
FirstDayOfWeek	控件显示的左边第一列是星期几

常用事件： DateChanged ，选定日期时发生

ProgressBar

常用属性：

- Minimum
- Maximum
- Step
- Value ： int 类型，默认初值为 0

常用方法：

PerformStep 使控件的 Value 增加 Step

例句：

```
progressBar1.Maximum = 100;// 设置最大长度值
progressBar1.Value = 0;// 设置当前值
progressBar1.Step = 5;// 设置每次增长多少
progressBar1.Value += 5;
progressBar1.PerformStep();
```

ToolTip

常用属性：

IsBalloon ： True 或 False ，是否以气球形式显示
ToolIcon ： None, Info, Warning, Error
AutoPopDelay: 指针在工具提示区域保持静止时，工具提示窗口保持可见的时间长度（ ms ）
InitialDelay: 在工具提示窗口显示之前，指针必须在工具提示区域内保持静止的时间长度

注意：
设定每个控件上需要显示的工具提示内容，方法：
（ 1 ）在设计器中分别设定各控件的属性：“ toolTip1 上的 TooTip”
（ 2 ）或编程设定：

```
    private void Form1_Load(object sender, EventArgs e)
     {  toolTip1.SetToolTip(this.button1, “ 点击打开文件。。。。。”);  }
```

WebBrowser

常用方法：

✓ Navigate(string urlString)
✓GoBack()
✓GoForward()
✓ Refresh()
✓Stop()

常用属性：

✓ Url：设置或者获取该控件的 url
✓StatusText ：在网页点击了一个链接时，WebBrower 的 StatusText 会是新链接的 url

常用事件：

✓DocumentCompleted ：页面加载结束时
✓NewWindow ：在浏览器新窗口打开之前发生。可以处理此事件，在不满足某些条件时防止新窗口打开。

作业 3 ：

点击提交按钮，显示用户此前所做的选择。
点选日期时，在其旁边显示所选的日期。