

## 第6章 Windows应用程序开发

- Windows应用程序是基于Microsoft Windows平台上的一种经典的可视化应用程序，特别适合于包含丰富图形用户界面的应用程序。Windows应用程序运行之后，往往显示一个或多个Windows窗体，具有友好的交互功能。使用.NET Framework和C#语言可以开发功能强大的Windows窗体应用程序。

## 6.1 窗体

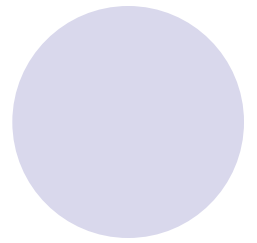
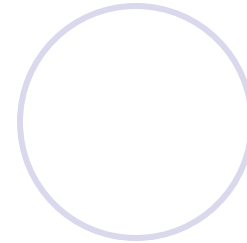
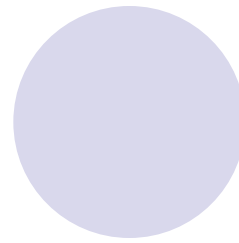
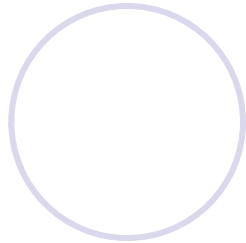
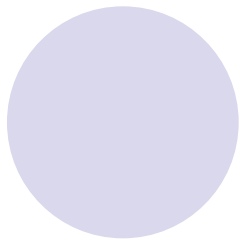
- 在Windows窗体应用程序中，“窗体”是向用户显示信息的可视界面，窗体包含可添加到窗体上的各种控件。而“控件”是显示数据或接收输入数据的相对独立的用户界面(**User Interface, UI**)元素，如文本框、按钮、下拉框和单选按钮等。用户还可以使用**UserControl**类自定义控件实现特殊的功能。
- 每个Windows窗体应用程序都至少拥有一个窗体，而且需要导入**System.Windows.Forms**命名空间。

# 1. 窗体的属性

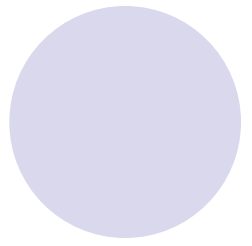
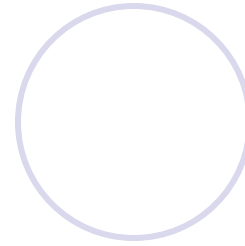
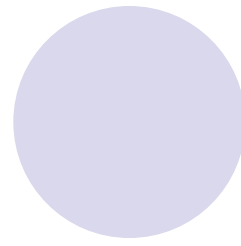
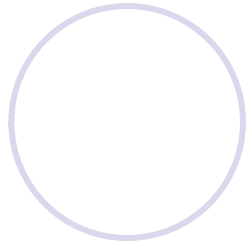
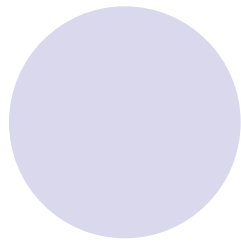
- 窗体的属性决定了窗体的外观和操作。大部分属性既可以在设计阶段通过属性窗口设置，也可以在程序运行时动态设置，少部分属性只能在设计阶段进行设置，或只能在运行期间设置。表6-1为窗体常用属性。

表6-1 窗体常用属性

属 性	描 述
Name	窗体的名称
Text	窗体的标题栏文本
Size Width Height	获取或设置窗体的大小，如this.Size =new Size(30,30) 其中Size.Width等效于窗体的Width属性值 其中Size.Height等效于窗体的Height属性值
StartPositon	窗体运行时在屏幕上显示的初始位置
WindowState	窗体运行时的初始状态，只能在设计阶段通过属性窗口设置。 Normal：正常窗口状态；Minimized：最小化状态；Maxmize 大化状态
Icon	窗体的图标
Location X Y	设置窗体在屏幕上的显示位置，如this.Location = new Point(500,50) 横坐标的值 纵坐标的值



Font	窗体的字体，如 <code>this.Font = new Font("宋体", 9, FontStyle.Bold);</code>
ForeColor BackColor	窗体上显示文字的颜色，如 <code>this.ForeColor = Color.Red;</code> 窗体的背景颜色
BackgroundI mage	窗体的背景图片
BackgroundI mageLayout	窗体的背景图片布局方式， <b>Tile</b> (平铺)、 <b>Center</b> (居中)、 <b>Stretch</b> (拉伸)、 <b>Zoom</b> (缩放)
Cursor	鼠标光标在窗体中的显示样式
FormBorderSt yle	窗体的边框样式，只能在设计阶段通过属性窗口设置
Enable	窗体是否可用



- 在设计阶段设置窗口的属性：首先在窗体设计器中选中窗体，然后在属性窗口中单击“属性”按钮，查看属性窗口的左侧栏，选择要修改的窗体的属性，在对应属性的右侧栏中输入或选择属性值(有的属性是枚举型的)。如修改窗体的**Text**属性为“窗体”，则在**Text**属性对应的右侧栏中输入“窗体”，

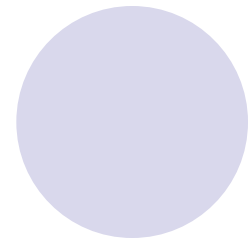
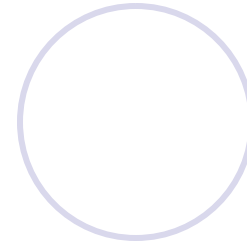
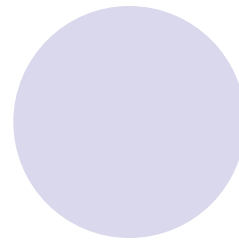
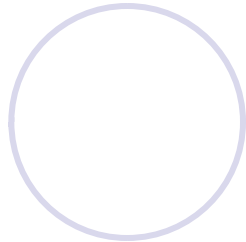
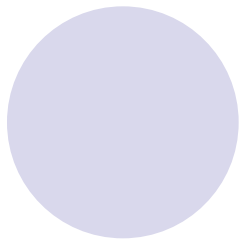
## 2. 窗体的事件

- 当用户通过鼠标或键盘与窗体交互操作时，会产生各种事件。通过创建事件处理程序，用户可以实现各种处理功能。常用的事件如表6-2所示。在属性窗口中，单击“事件”按钮可以查看窗体的所有事件。

## 表6-2 窗体的常用事件

事 件	描 述
Click	鼠标触发事件，在单击窗体时发生
DoubleClick	鼠标触发事件，在双击窗体时发生
MouseDown	鼠标触发事件，按下任一鼠标键时发生
MouseUp	鼠标触发事件，释放任一鼠标键时发生
MouseMove	鼠标触发事件，鼠标移动时发生
KeyPress	键盘触发事件，按下并释放一个会产生ASCII码的键时发生
KeyDown	键盘触发事件，按下任一键时发生
KeyUp	键盘触发事件，释放任一键时发生

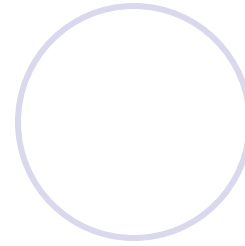
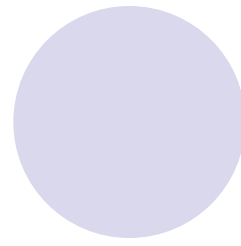
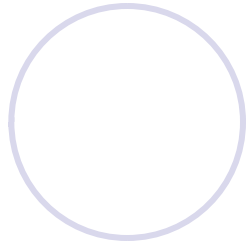
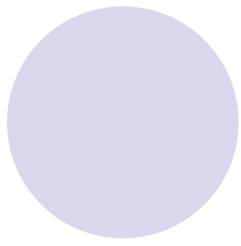




Load	在第一次显示窗体前发生，当应用程序启动时自动执行 <b>Load</b> 事件，所以该事件通常用来在启动应用程序时初始化属性和变量
Activated	当使用代码激活或用户激活窗体时发生
Resize	在调整控件大小时发生
FormClosing	当用户关闭窗体时，在窗体已关闭并指定原因之前发生
FormClosed	当用户关闭窗体时，在窗体已关闭并指定原因之后发生



- 程序启动运行时，首先触发这些事件中的**Load**事件。在编写应用程序时，通常把一些初始化工作放在**Load**事件中。
- 在窗体设计器中，选中窗体(单击窗体的空白处)，单击窗体属性窗口的“事件”按钮，然后双击某一事件选项(如双击窗体的**Click**事件，如图6.2所示)，系统就会在代码编辑窗口自动生成该事件的框架，



- 初学者不要在代码编辑窗口手动输入该框架，因为系统在代码编辑窗口自动生成框架的同时，也在**Form1.Designer.cs**文件中加载了相关代码。程序员只需在**Click**事件的框架基础上进行填写代码。代码加在一对花括号**{}**中间。

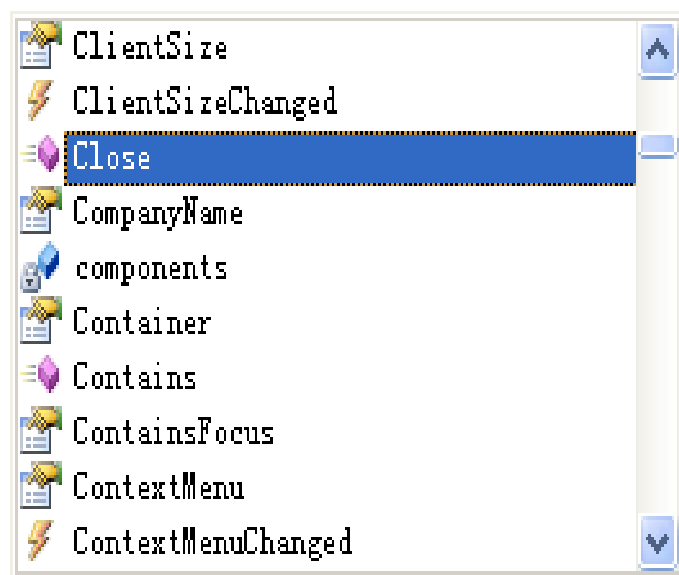
【例6-1】单击窗体时，修改窗体的**Text**属性为“触发窗体的**Click**事件”。触发窗体的单击事件前后标题的变化。事件代码如下

- `private void Form1_Click(object sender, EventArgs e)`
- `{`
- `this.Text = "触发窗体的Click事件";`  
`//运行时设置窗体的Text属性`
- `}`

### 3. 窗体的常用方法

- 在代码编辑器中输入一个对象名及“.”运算符后，会自动弹出一个下拉表框，显示出该对象可以使用的属性、事件、方法，如图6.6所示，图中的**close**方法前面的图标代表方法。在弹出的下拉表框中通过鼠标(或通过键盘)选中某一方法后，按回车键，则该方法自动添加到代码编辑器中。表6-3列举了窗体的常用方法。

```
private void Form1_Click(object sender, EventArgs e)
{
    this.Text = "触发窗体的Click事件";
    this.
}
```



```
void Form.Close()
关闭窗口。

异常:
    System.InvalidOperationException
    System.ObjectDisposedException
```

图6.6 窗体的方法

# 表6-3 窗体的常用方法

方 法	描 述
Activate ( )	激活窗体并赋予它焦点
Close ( )	关闭并卸载窗体
Hide ( )	隐藏窗体
Show ( )	加载并显示非模态窗体
ShowDialog ( )	加载并显示模态窗体
Refresh	通过重绘更新窗体及其子控件的外观
BringToFront	将窗体移动到其他窗体的前面
SendToBack ( )	将窗体移动到其他窗体的后面
SetBounds ( )	将窗体的边界设置为指定的位置和大小

## 6.2 Windows常用控件的使用

- Windows常用控件有标签、按钮、文本框、单选按钮、复选按钮、列表框、组合框、分组框、面板、图片框等。



- 1. 为窗体添加控件
- 通常使用窗体设计器向窗体中添加控件。首先在窗体设计器中打开要添加控件的窗体(在解决方案资源管理器中双击相应窗体的节点即可打开该窗体)，然后打开工具箱窗口(面板)。
- 可使用下列几种方法向窗体中添加控件：
  - 双击工具箱中的控件，将在窗体的默认位置添加默认大小的控件。
  - 在工具箱中选中一个控件，按住鼠标左键不放，这时鼠标指针变成该控件在工具箱中的图标形状，把鼠标指针移到窗体的相应位置，然后松开鼠标左键

## 2. 调整控件

- 控件添加到窗体中之后，可以对控件进行调整，包括其位置、大小、对齐方式等。
- 要调整控件的摆放，首先要选中窗体中的控件(如果要选择多个控件时，可以先按下**CTRL**键或**SHIFT**键，同时用鼠标单击要选择的其他控件；或者按下鼠标左键拖动鼠标，选择一个范围，该范围内的控件均被选中)，然后通过格式菜单或工具栏上的格式按钮进行调整。如图6.8所示为调整窗体中所有的控件左对齐。

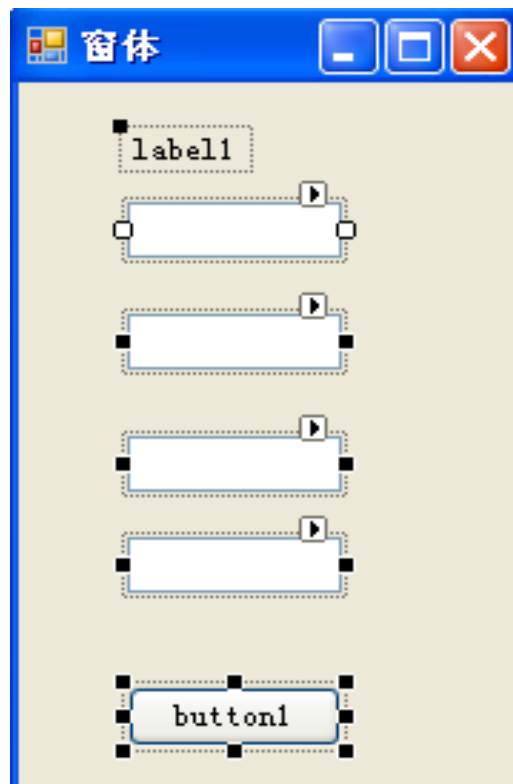
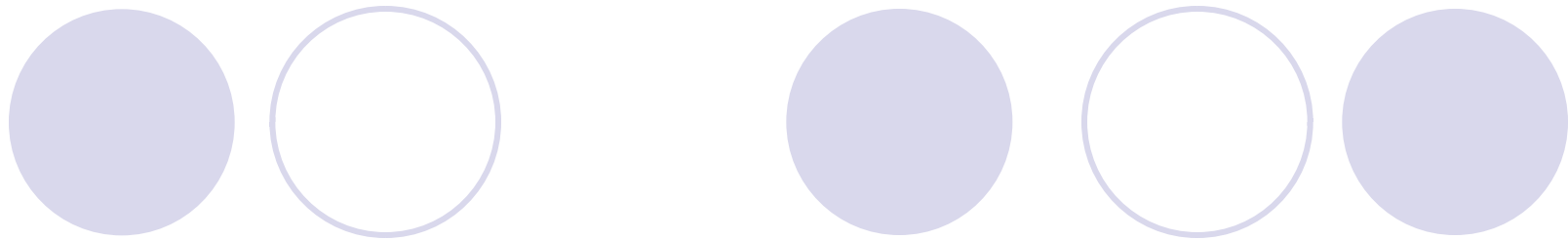


图6.8 调整控件左对齐

### 3. 设置控件的Tab键顺序

- 控件的**Tab**键顺序决定了当用户使用**Tab**键切换焦点时的顺序。默认情况下，控件的**Tab**键顺序就是控件添加到窗体的顺序。可以使用“视图”→“**Tab**键顺序”菜单项把窗体设计器切换到**Tab**键顺序选择模式，再次使用该命令将切换回设计模式，如图6.9所示。另外，也可以通过在属性窗口中设置控件的**TabIndex**属性来改变它们的**Tab**键顺序。

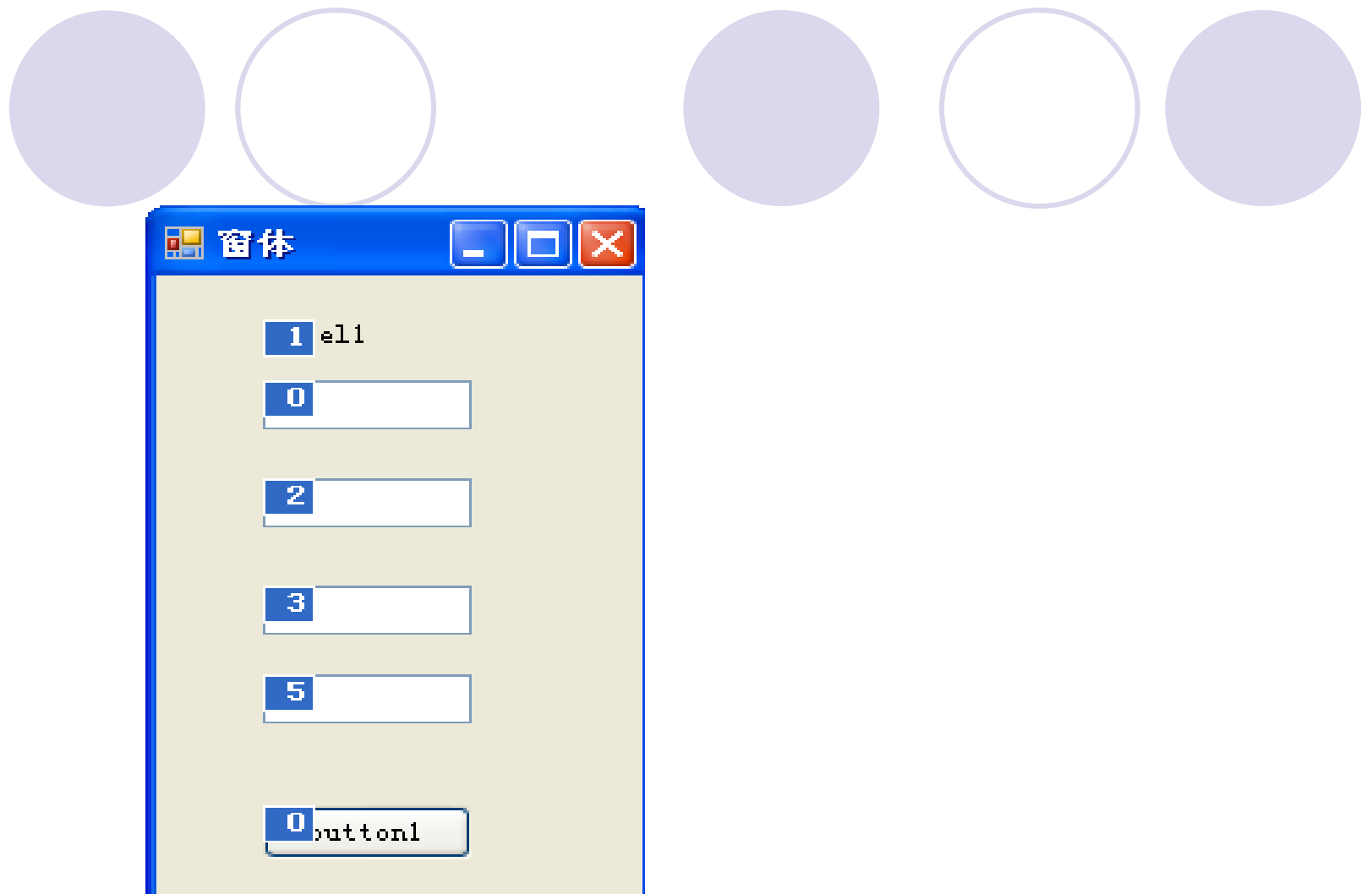
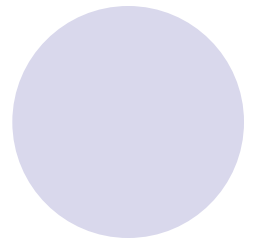
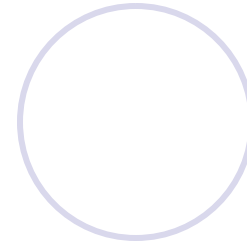
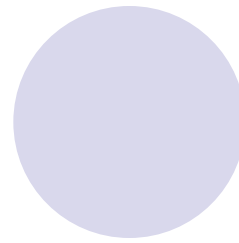
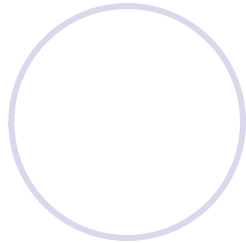
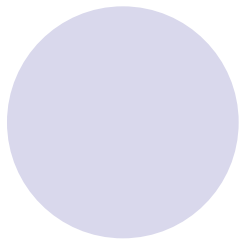


图6.9 显示控件的Tab键顺序

## 4. 常用控件的属性

属 性	描 述
Anchor	设置控件的哪个边缘锚定到其容器边缘
Dock	设置控件停靠到父容器的哪个边缘
Name	设置或获取控件的名称
Text	设置或获取与此控件关联的文本
Size Width Height	设置或获取控件的大小，如 <code>textBox1.Size = new Size(30, 30)</code> 其中 <b>Size.Width</b> 等效于控件的 <b>Width</b> 属性值 其中 <b>Size.Height</b> 等效于控件的 <b>Height</b> 属性值
Parent	设置或获取控件的父容器
Location X Y	设置控件在其容器中的显示位置，如 <code>textBox1.Location = new Point(500, 50)</code> 设置或获取控件的左边界到容器左边界的距离 设置或获取控件的顶部到容器顶部的距离

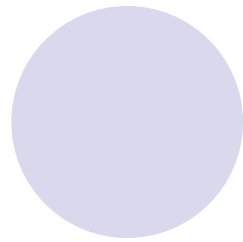
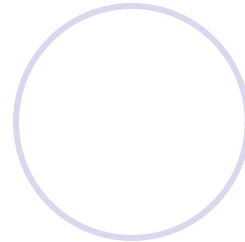
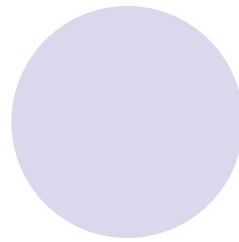
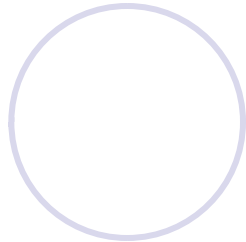
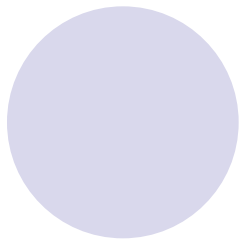


Font	设置或获取控件显示文字的字体，如 <code>textBox1.Font = new Font("宋体", 9, FontStyle.Bold);</code>
ForeColor BackColor	设置或获取控件的前景颜色 设置或获取控件的背景颜色
Cursor	设置或获取当鼠标指针位于控件上时显示的光标样式
TabIndex	设置或获取控件容器上控件的 <b>Tab</b> 键顺序
TabStop	设置用户能否使用 <b>Tab</b> 键将焦点放在该控件上
Tag	设置或获取包括有关控件的数据对象
Visible	设置是否在运行时显示该控件
Enable	设置控件是否可以对用户交互做出响应

## 5. 常用控件的事件

事 件	描 述
Click	鼠标触发事件，在单击控件时发生
DoubleClick	鼠标触发事件，在双击控件时发生
DragDrop	当一个对象被拖放到控件上、用户释放鼠标时发生
DragEnter	当被拖动的对象进入控件的边界时发生
DragLeave	当被拖动的对象离开控件的边界时发生
DragOver	当被拖动的对象在控件的范围时发生
MouseDown	当鼠标指针位于控件上并按下鼠标键时发生
MouseUp	当鼠标指针位于控件上并释放鼠标键时发生
MouseMove	鼠标指针移到控件上时发生
KeyPress	控件有焦点的情况下，按下任一键时发生，在 KeyUp前发生

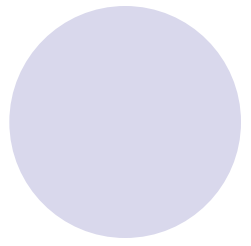
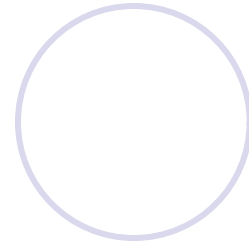
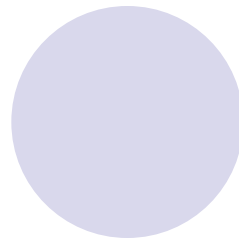
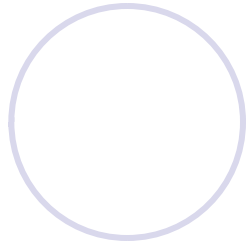
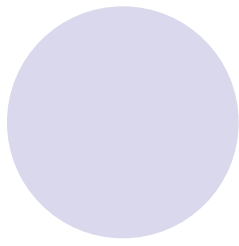




KeyDown	控件有焦点的情况下，按下任一键时发生，在KeyPress前发生
KeyUp	控件有焦点的情况下，释放任一键时发生
GetFocus	在控件获得焦点时发生
LostFocus	当控件失去焦点时发生
Paint	在重绘控件时发生
Resize	在调整控件大小时发生
Validated	在控件完成验证时发生
Validating	在控件正在验证时发生

## 6.2.1 标签控件和超链接标签控件

- 1. 标签控件(Label控件)
- Label控件，在工具箱中的图标是，用于显示(输出)文本或图像信息。
- Label控件也可以用来为其他控件定义访问键。在标签(Label)控件中定义访问键时，用户可以按ALT键和指定字符将焦点移动到Tab键顺序中的下一个控件上。因为标签无法接收焦点，所以焦点自动移动到Tab键顺序中的下一个控件上。为标签设定访问键步骤：
  - (1) 向窗体中添加一标签控件，然后绘制另一个控件。或按任意顺序添加控件，并将该标签的TabIndex属性设置为比另一个控件小1。
  - (2) 将该标签的UseMnemonic属性设置为True(UseMnemonic属性允许用户启用访问键功能)。在该标签的Text属性中使用“and”符(&)为该标签分配访问键。



- 例如，向带标签的控件分配访问键，打开项目和窗体，向窗体中添加标签控件，然后按任意顺序添加**TextBox**控件若干，并将标签的**TabIndex**属性设置为1，另一个控件**textBox1**的**TabIndex**属性设置为2(保证比上述的标签**TabIndex**属性大1即可)。将该标签的**UseMnemonic**属性设置为**True**。该标签的**Text**属性设置为“ab&v”，在字符“v”前使用“and”符(&)为该标签分配访问键。此时标签变为，当程序运行时，同时按下**ALT**键和**v**键，**textBox1**将获得焦点。

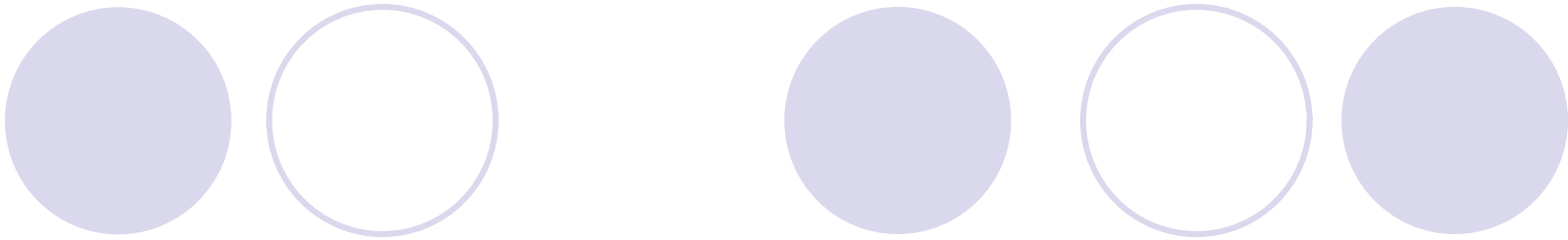
# 标签除了具有常用的属性外，还有如表6-6所示的属性

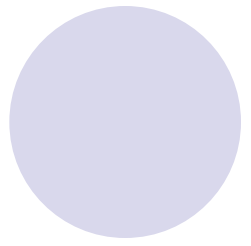
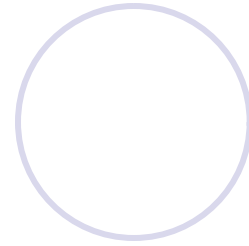
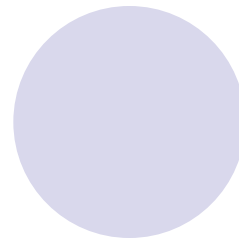
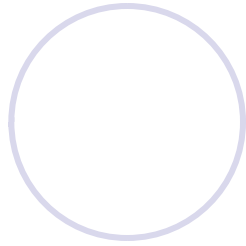
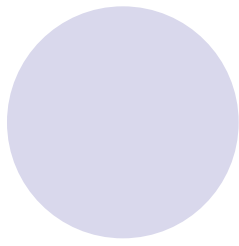
属 性	描 述
Image	设置或获取显示在Label上的图像
ImageList	设置或获取包含要在Label控件上显示的图像的图像列表ImageList
ImageIndex	与ImageList组合使用，ImageList中的图像索引号
TextAlign ImageAlign	设置或获取标签中文本/图像的对齐方式
AutoSize	设置或获取一个值(True或False)，表示是否自动调整控件的大小以完整显示标签中的内容
BorderStyle	设置或获取控件的边框样式。其值为枚举型：None(无边框)、FixedSingle(单行边框)、Fixed3D(三维边框)

## 2. 超链接标签控件(LinkLabel控件)

**LinkLabel** 控件也叫超链接标签控件，在工具箱中的图标是，它除了具有**Label**控件的所有属性、方法和事件外，还有针对超链接和链接颜色的属性及事件，如表6-7所示

属性/事件		描 述
属性	LinkColor	设置或获取显示普通链接时使用的颜色
	VistedLinkColor	设置或获取显示以前访问过的链接时所使用的颜色
	DisabledLinkColor	设置或获取显示禁用链接时所用的颜色
	LinkBehavior	设置或获取一个表示链接的行为的值
事件	LinkClicked	当单击控件内的链接时发生

- 
- **【例6-2】** 创建一个Windows窗体应用程序，其项目名为**ex06-02**。在“资源管理器”中选中该项目，单击鼠标右键，选择“添加”→“Windows窗体”菜单项，进入添加新项窗口，创建新窗体**Form2**。然后在**form1**的**linkLabel1**的**LinkClicked**事件中填写代码。在**LinkClicked**事件处理中，调用**Show**方法打开刚刚建立的窗体，并将**linkLabel1.LinkVisited**属性设置为**True**。观察窗体的变化。



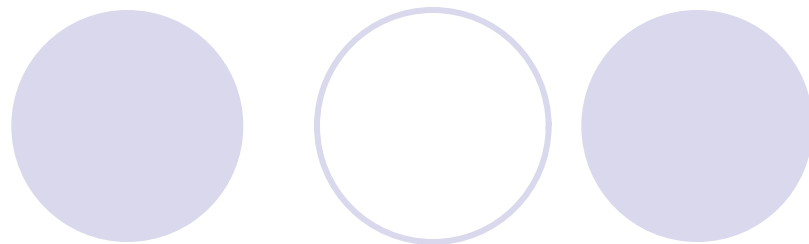
- 代码如下：
- `private void linkLabel1_LinkClicked(object sender, LinkLabelLinkClickedEventArgs e)`
- `{`
- `Form2 f2 = new Form2();`
- `f2.Text = "被链接的窗口";`
- `f2.Show();`
- `linkLabel1.LinkVisited = true;`
- `//表示已被链接过`
- `}`

## 【例6-3】 使用LinkLabel控件启动Internet Explorer，并链接到Web网页

- 在linkLabel1控件的LinkClicked事件中编写如下代码：
- `private void linkLabel1_LinkClicked_1(object sender, LinkLabelLinkClickedEventArgs e)`
- `{`
- `//调用Process.Start方法来通过一个URL打开默认的浏览器`
- `System.Diagnostics.Process.Start("http://www.baidu.com");`
- `}`
- 其中，`System.Diagnostics.Process.Start`方法以某个URL启动默认浏览器。



## 6.2.2 按钮控件



- **Button(按钮)控件**，在工具箱中的图标是，功能是在窗体上创建一个按钮，允许用户通过单击它来完成指定的操作。每当用户单击按钮后，就会触发**Click**事件处理程序。单击**Button**控件后还会触发其他事件，如**MouseEnter**、**MouseDown**等，如果要为这些事件设置相关的事件处理程序，则确保它们之间的操作不会有冲突。另外，按钮控件不支持双击事件。

## 【例6-4】单击Button1，改变Label1的字体、颜色。

- 创建一个Windows窗体应用程序，项目名称为ex06-04，向窗体中添加一个Button控件和一个Label控件。在Button1的Click事件中编写如下代码：
- ```
private void button1_Click(object sender, EventArgs e)
```
- ```
{
```
- ```
    label1.Text = "Button单击事件修改Label属性";
```
- ```
    label1.Font = new Font("宋体", 16,
```
- ```
    FontStyle.Bold);
```
- ```
    label1.ForeColor = Color.Red;
```
- ```
}
```

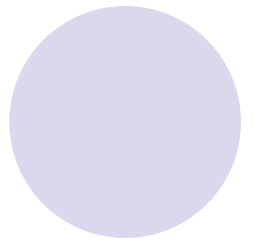
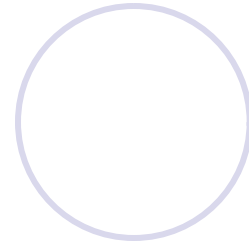
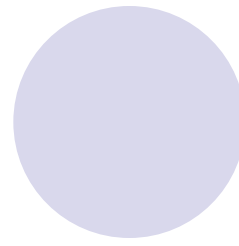
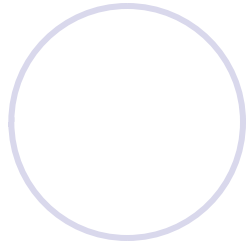
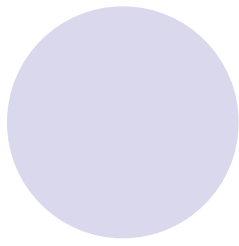
## 【例6-5】 通过本例了解按钮的鼠标按下事件、单击事件、鼠标抬起事件，当用鼠标单击按钮时，这些事件依次发生的顺序

- 代码如下：
- namespace ex06\_05
- {
- public partial class Form1 : Form
- {
- public Form1()
- {             InitializeComponent();             }
- private void button1\_Click(object sender, EventArgs e)
- {             label1.Text += "按钮的单击事件\n";             }
- private void button1\_MouseDown(object sender, MouseEventArgs e)
- {             label1.Text += "按钮的鼠标键按下事件\n";             }
- private void button1\_MouseUp(object sender, MouseEventArgs e)
- {             label1.Text += "按钮的鼠标键抬起事件\n";             }
- }
- }

## 6.2.3 文本框控件、富文本框控件

### 1. TextBox控件

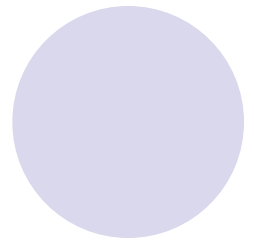
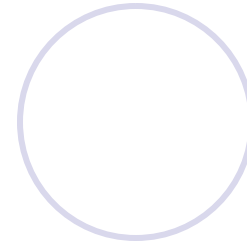
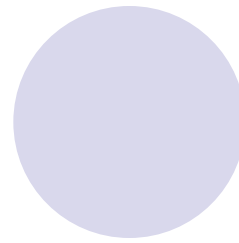
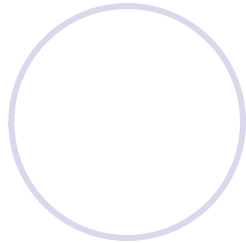
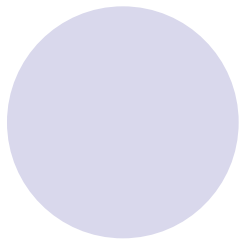
- **TextBox**(文本框)控件，在工具箱中的图标是，功能是获取用户在窗体内输入或显示的文本。文本框控件不仅可以编辑单行文本，还可以编辑多行文本。在单行文本编辑情况下，用户可以使用文本框的**Text**属性访问其中的内容；在多行文本编辑情况下，则要使用**Lines**属性，它是一个字符串数组，其中每个元素对应多行文本中的一行。用户在文本框中所能输入的字符数只受可用内存的限制，但可以设置**MaxLength**来限制所能输入的字符数。
- 在通常情况下，文本框中不能使用**Enter**键进行换行或使用**Tab**键输入制表符，这两个键的默认行为是触发窗体的**AcceptButton**属性的**Click**事件和切换输入焦点，要想在文本框中进行换行操作需**Ctrl+Enter**组合键。



- 如果仍希望在文本框中使用**Enter**键和**Tab**键，则需要把文本框的**AcceptsReturn**属性和**AcceptsTab**属性设置为**True**。**TextBox**控件的常用属性如表6-8所示，**TextBox**控件的常用事件见表6-9所示，**TextBox**控件的常用方法如表6-10所示。

# 表6-8 TextBox控件的常用属性

| 属 性                    | 描 述                                                          |
|------------------------|--------------------------------------------------------------|
| <b>PasswordChar</b>    | 用来替换在单行文本框中输入文本的密码字符(仅在 <b>MultiLine</b> 为 <b>False</b> 时有效) |
| <b>MultiLine</b>       | 为 <b>True</b> ，则允许用户输入多行文本信息；为 <b>False</b> ，则为单行文本          |
| <b>Scrollbars</b>      | 当 <b>MultiLine</b> 为 <b>True</b> 时，指定文本框是否显示滚动条              |
| <b>WordWrap</b>        | 当 <b>MultiLine</b> 为 <b>True</b> 时，一行的文本宽度超过文本框的宽度时，是否允许自动换行 |
| <b>MaxLength</b>       | 允许输入到文本框中的最大字符数                                              |
| <b>SelectedText</b>    | 文本框中被选中的文本(程序运行时设置)                                          |
| <b>SelectionLength</b> | 文本框中被选中文本的字符数(程序运行时设置)                                       |



|                         |                                                                       |
|-------------------------|-----------------------------------------------------------------------|
| <b>SelectionStart</b>   | 文本框中被选中文本的开始位置(程序运行时设置)                                               |
| <b>ReadOnly</b>         | 若为 <b>True</b> ，表示文本框中的文本为只读                                          |
| <b>CharacterCasing</b>  | 是否自动改变输入字母的大小写，默认为 <b>Normal</b> ， <b>Lower</b> 为小写， <b>Upper</b> 为大写 |
| <b>CausesValidation</b> | 若为 <b>True</b> ，控件获得焦点时，将触发 <b>Validating</b> 、 <b>Validated</b> 事件   |

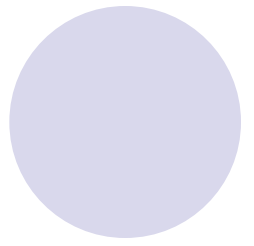
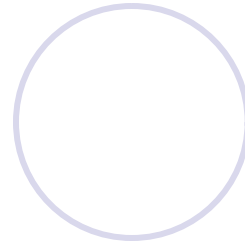
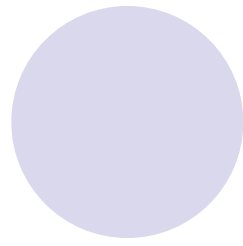
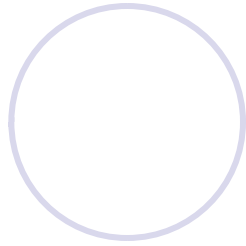
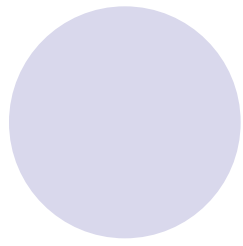
# 表6-9 TextBox控件的常用事件

| 事 件                | 描 述                                          |
|--------------------|----------------------------------------------|
| <b>Enter</b>       | 成为活动控件时发生                                    |
| <b>GetFocus</b>    | 控件获得焦点时发生(在 <b>Enter</b> 事件之后发生)             |
| <b>Leave</b>       | 从活动控件变化为不活动控件时发生                             |
| <b>KeyDown</b>     | 文本框获得焦点，有键按下时发生                              |
| <b>KeyPress</b>    | 文本框获得焦点，有键按下后释放时发生(在 <b>KeyDown</b> 事件之后发生)  |
| <b>KeyUp</b>       | 文本框获得焦点，有键按下后释放时发生(在 <b>KeyPress</b> 事件之后发生) |
| <b>TextChanged</b> | 文本框内的文本信息发生改变时发生                             |



# 表6-10 TextBox控件的常用方法

| 方 法                   | 描 述                  |
|-----------------------|----------------------|
| <b>AppendText( )</b>  | 在文本框当前文本的末尾追加新的文本    |
| <b>Clear( )</b>       | 清除文本框中的所有内容          |
| <b>Copy( )</b>        | 将文本框中被选中的文本复制到剪贴板中   |
| <b>Cut( )</b>         | 将文本框中被选中的文本剪切到剪贴板中   |
| <b>Paste( )</b>       | 将剪贴板中的内容复制到文本框中的当前位置 |
| <b>Focus( )</b>       | 将文本框设置为获得焦点          |
| <b>Select( )</b>      | 在文本框中选择指定起点和长度的文本    |
| <b>SelectAll( )</b>   | 在文本框中选择所有的文本         |
| <b>DeselectAll( )</b> | 取消文本框中的选择            |

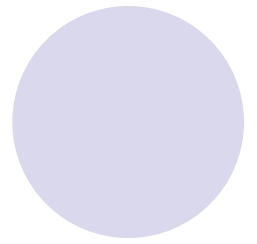
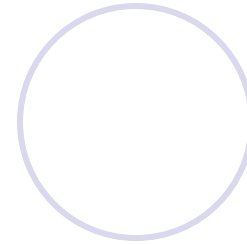
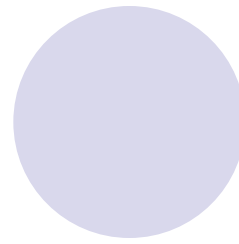
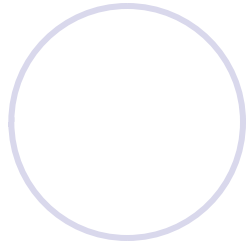
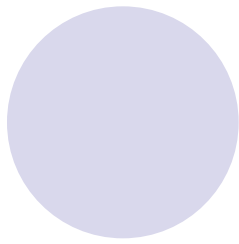


- **【例6-6】** 编一程序，完成登录功能。在类别**textBox3**中，如果输入的是管理员或普通用户，则**Button1**有效，否则**Button1**无效。**textBox2**中输入的数字长度要大于等于6，而且只能输入数字。本例所需控件及属性值如表**6-11**所示

表6-11 各个控件的属性及其值

| 控 件     | 属 性  | 值  | 控 件      | 属 性          | 内 容        |
|---------|------|----|----------|--------------|------------|
| Label1  | Text | 账户 | textBox1 | Text         | 输入的是账户     |
| Label2  | Text | 密码 | textBox2 | Text         | 输入的是密码(数字) |
| Label3  | Text | 类别 | textBox2 | PasswordChar | *          |
| Button1 | Text | 登录 | textBox3 | Text         | 输入的是类别     |

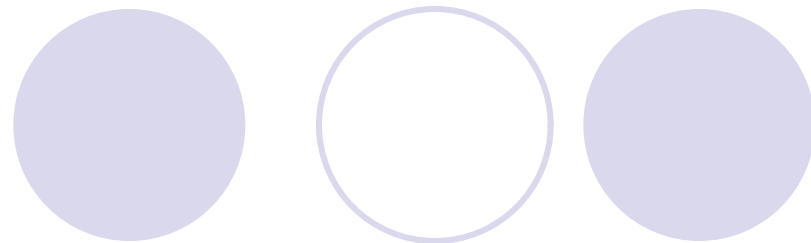
```
namespace ex06_07
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }
        private void textBox3_Validating(object sender,
CancelEventArgs e)
        //在验证控件时触发
        {
            if (textBox3.Text == "管理员" || textBox3.Text == "普通用户")
            {
                MessageBox.Show("请登录");//弹出消息框，调用消息框的
Show方法
                button1.Enabled = true;
            }
            else
            {
                textBox3.Focus(); //textBox3获得焦点
                MessageBox.Show("请重新输入是普通用户还是管理员");
                button1.Enabled = false; //Button1 设为无效
            }
        }
    }
}
```



- private void textBox2\_Validated(object sender, EventArgs e)  
//成功验证后触发
- {
- if (textBox2.Text.Length < 6)
- {
- MessageBox.Show("密码长度大于等于6");
- textBox2.Focus();
- }
- }

- private void textBox2\_KeyPress(object sender, KeyPressEventArgs e)
- //键按下事件, 整个按键动作完成后触发, 对CTRL、ALT等控制键无反应
- { //如果密码栏输入的是非数字则拒绝接受
- if ((e.KeyChar < 48 || e.KeyChar > 57) && e.KeyChar !=
- 8)
  - e.Handled = true; //拒绝接受键入的字符
  - }
- private void button1\_Click(object sender, EventArgs e)
- {
- Form2 tt = new Form2();
- //创建一个Form2类对象tt。需先建立Form2窗体
- tt.Show(); //显示另一窗体
- }
- }
- }

## 2. RichTextBox控件



- RichTextBox控件也叫富文本框控件，它能够处理有格式的文本，还可以显示字体、颜色、链接、嵌入的图像等。

# RichTextBox控件的常用属性和方法

| 属性/方法                 | 描 述                                      |
|-----------------------|------------------------------------------|
| <b>CanRedo</b>        | 若为 <b>True</b> ，则允许恢复上一个被撤销的操作           |
| <b>CanUndo</b>        | 若为 <b>True</b> ，则允许撤销上一个操作               |
| <b>RedoActionName</b> | 通过 <b>Redo()</b> 方法执行的操作名称               |
| <b>UndoActionName</b> | 如果用户选择撤销某个动作，该属性将获得该动作的名称                |
| <b>Rtf</b>            | 包含 <b>Rtf</b> 格式的文本(与 <b>Text</b> 属性相对应) |
| <b>SelectedRtf</b>    | 设置或获取被选中的 <b>RTF</b> 格式的文本               |
| <b>SelectedText</b>   | 设置或获取被选中的文本，丢弃所有格式信息                     |



|                           |                                                                 |
|---------------------------|-----------------------------------------------------------------|
| <b>SelectionAlignment</b> | 被选中文本的对齐方式( <b>Center, Left, Right</b> )                        |
| <b>SelectionBullet</b>    | 被选中文本的项目号                                                       |
| <b>SelectionColor</b>     | 被选中文本的颜色                                                        |
| <b>SelectionFont</b>      | 被选中文本的字体                                                        |
| <b>SelectionProtected</b> | 被选中文本是否允许被修改, 若为 <b>True</b> , 则处于写保护状态                         |
| <b>Redo()</b> 方法          | 恢复上一个被撤销的操作                                                     |
| <b>Undo()</b> 方法          | 撤销上一个操作                                                         |
| <b>Find()</b> 方法          | 查找是否存在特定的字符串, 存在则返回第一个字符串的位置, 否则返回-1                            |
| <b>LoadFile()</b> 方法      | 将指定路径下的 <b>RTF</b> 文件或 <b>TXT</b> 文件内容载入 <b>RichTextBox</b> 并显示 |
| <b>SaveFile()</b> 方法      | 将 <b>RichTextBox</b> 中的内容以 <b>RTF</b> 或其他特定类型文件格式保存到指定路径下       |

## 6.2.4 单选按钮控件

- **RadioButton(单选按钮)控件**，在工具箱中的图标是，该控件可以显示文本、图像或同时显示两者。在一个容器内如果有多个**RadioButton**，那么只允许有一个**RadioButton**处于选中状态。该特性使得**RadioButton**只适合于允许用户选取一个选项的情况。使用**Text**属性可以设置其显示的文本。当单击**RadioButton**控件时，其**Checked**属性设置为**True**，并调用**Click**事件处理程序。当**Checked**属性的值更改时，将引发**CheckedChanged**事件

# RadioButton控件的主要属性和事件

| 属性/事件 |                       | 描 述                                                             |
|-------|-----------------------|-----------------------------------------------------------------|
| 属性    | <b>Checked</b>        | 设置或获取 <b>RadioButton</b> 是否( <b>True</b> 或 <b>False</b> )处于选中状态 |
|       | <b>Appearance</b>     | 设置或获取一个值，该值确定 <b>RadioButton</b> 的外观                            |
| 事件    | <b>CheckedChanged</b> | 当 <b>Checked</b> 属性的值更改时触发                                      |