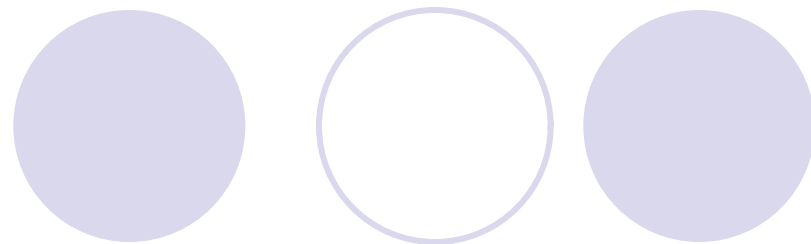


6.2.5 复选框控件

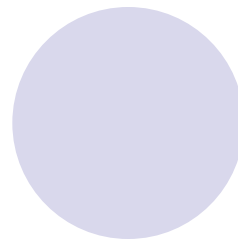
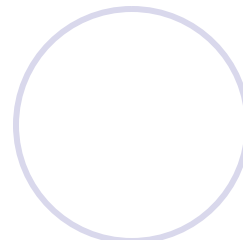
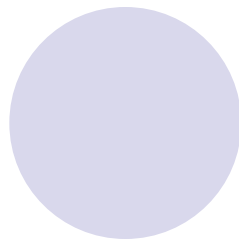
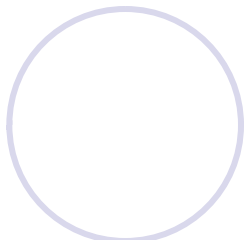
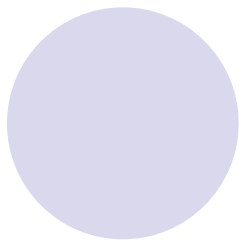
- **CheckBox(复选框)**控件，在工具箱中的图标是，它的属性和事件与单选按钮非常相似。下面几个特性与单选按钮控件有所不同。
- **CheckState**属性：用来设置复选框的状态，复选框有三种状态。**Checked(选中)**图标为；**Unchecked(未选中)**图标为；**Indeterminate(无效)**图标为。复选框在一个组内被选中与否，不影响同组的其他复选框的状态。
- **ThreeState**属性：用来设置是否具有第三状态(即无效状态)。当其值为**True**时，**CheckState**属性有三种状态；当其值为**False**时，**CheckState**属性具有两种状态(只有选中和未选中状态)。
- **CheckedChanged**事件：当复选框**Checked**属性改变时，就会触发该事件。

6.2.6 列表框控件



- **ListBox**(列表框)控件，在工具箱的图标是，通常用于供用户选择的选项。列表框中显示的项目保存在**Items**属性中。用户可以通过该属性访问列表框中的数据项或向列表框添加数据项。列表框的常用属性、方法、事件如表**6-14**所示。

属性/方法/事件		描 述
属性	Items	列表框中所有数据项的集合，利用这个集合可以增加或删除数据项
	SelectedIndex	列表框中被选中项的索引(从 0 开始)，当多项被选中时，表示第一个被选中的项
	SelectedIndices	列表框中所有被选中项的索引的集合(从 0 开始)
	SelectedItem	列表框中当前被选中的选项。当多个项被选中时，表示第一个被选中的项
	SelectedItems	列表框中所有被选中的项的集合(不是索引号，而是数据项)
	SelectionMode	列表框的选择模式： None (无法选择项)， One (只能选择一项)， MultiSimple (可以选择多项)， MultiExtended (可以选择多项，并且用户可使用 SHIFT 键、 CTRL 键及方向键来选择)
	MultiColumn	是否允许列表框以多列的形式显示
	ColumnWidth	在列表框允许多列显示的情况下，指定列的宽度
	Sorted	若为 True ，则将列表框的所有选项按字母顺序排序；若为 False ，则按加入的顺序排列

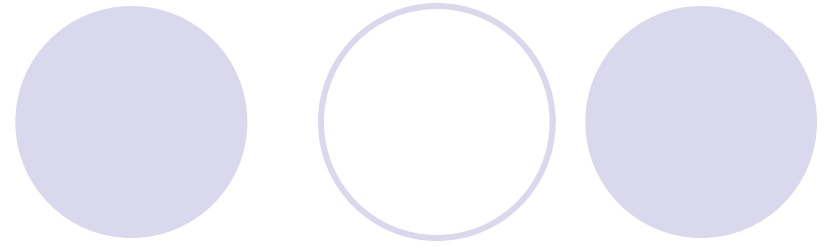


方法	Add ()	向列表框中添加数据项
	Insert ()	将数据项插入到列表框的指定索引处
	Remove ()	从数据项集合中移除指定的属性项
	Clear ()	清除列表框中所有的数据项
事件	SelectedIndexChanged	当 SelectedIndex 属性值发生改变后触发
	SelectedValueChanged	当 SelectedValue 属性更改时触发

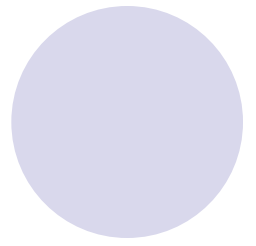
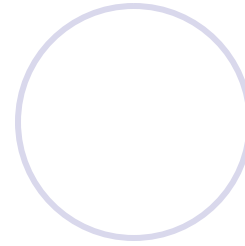
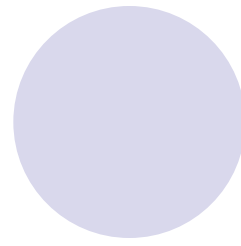
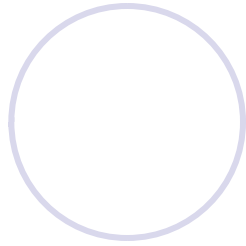
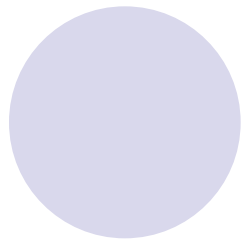
6.2.7 组合框控件

- **ComboBox(组合框)**控件，在工具箱中的图标是，用于在下拉组合框中显示数据。**ComboBox**控件分两部分：上部是一个允许用户输入的文本框；下部是允许用户选择一个项的列表框。除具有与**TextBox**控件及**ListBox**控件相同的属性、方法、事件外，还有**DropDownStyle**属性等常用属性，其值为：**DropDown**(标准组合框，可以在文本框中进行编辑，同时可以单击箭头按钮显示下拉列表框，并从中选择某一数据项)；**DropDownList**(不能在文本框中进行编辑，只能单击箭头按钮显示下拉列表框进行选择)；**Simple**(可以在文本框中进行编辑或从列表框中选择某一数据项，下拉列表框一直处于显示状态)。

6.2.8 分组框控件



- **GropBox(分组框)**控件，或叫群组框控件，具有“容器”的特性，能够把其他控件装入其中。在窗体上添加一个分组框后，再把其他控件添加到分组框的边框线以内，这样就把这些控件装入到分组框中。如果拖动分组框，则它内部的其他控件也跟着一起移动。如果删除该分组框，则它内部的其他控件也同时被删除。



- **【例6-7】** 利用TextBox、ComboBox、GroupBox、RadioButton、CheckBox、RichTextBox、ListBox等控件完成个人信息的输入，并将信息提取出来(载入)，或保存到文件中去，以及复制、粘贴等功能。设计界面如图6.19所示，控件属性及属性值如表6-15所示。

个人信息

姓名:

性别:

学位:

- ☐ 本科
- ☐ 硕士
- ☐ 博士

爱好:

- ☐ 钓鱼
- ☐ 旅游
- ☐ 打球

简介:

专业:

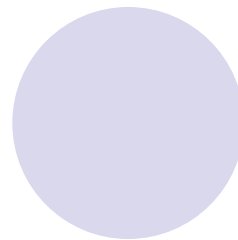
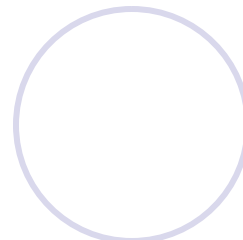
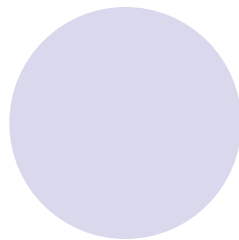
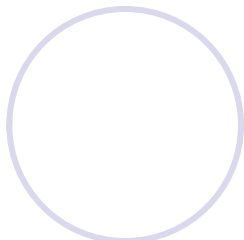
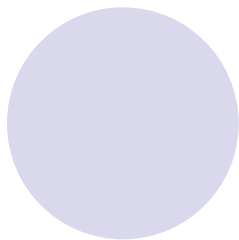
- 网络工程
- 计算机应用
- 电子信息
- 电子信息科学与技术
- 通信工程
- 机械制造
- 经济管理
- 应用数学

清除 保存 载入 复制 粘贴

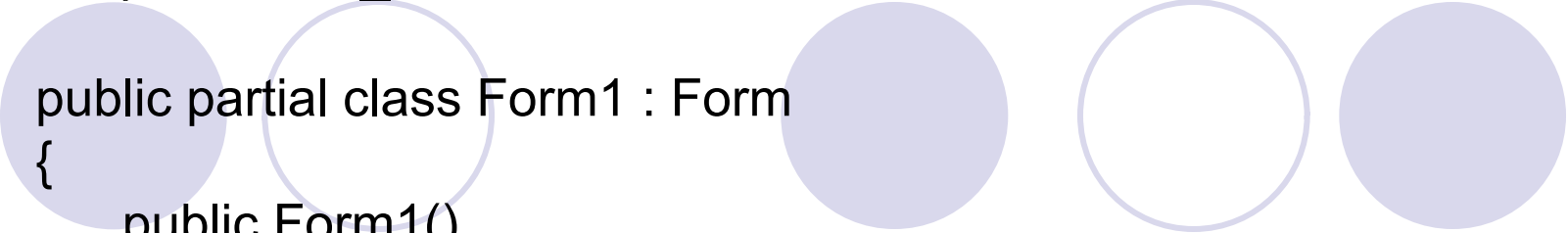
图6.19 设计界面

表6-15 控件的属性及属性值

控 件	属 性	值/含义	控 件	属 性	值
Label1	Text	姓名:	listBox1	Items	网络工程 计算机应用 电子信息 电子信息科学与技术 通信技术 机械制造经济管 理应用数学
Label2	Text	性别:			
Label3	Text	简介:			
Label4	Text	专业:			
textBox1	Text	含义: 输入 姓名	Button1	Text	清除
comboBox1	Text	男或女	Button2	Text	保存
	Items	男 女	Button3	Text	载入
groupBox1	Text	学位	Button4	Text	复制



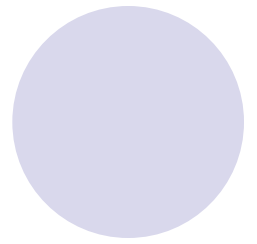
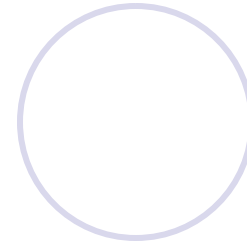
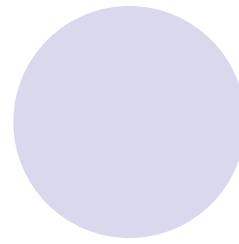
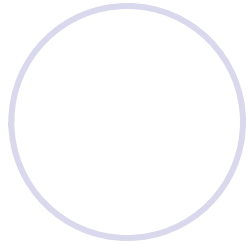
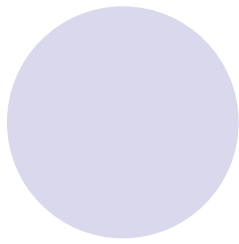
groupBox2	Text	爱好	Button5	Text	粘贴
radioButton1	Text	本科	Form1	Text	个人信息
	AutoCheck	False	radioButton2	Text	硕士
radioButton3	Text	博士		AutoCheck	False
	AutoCheck	False	checkBox1	Text	钓鱼
richTextBox1	Text	含义：显示个人信息内容	checkBox2	Text	旅游
			checkBox3	Text	打球



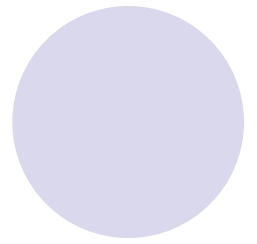
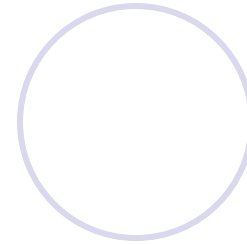
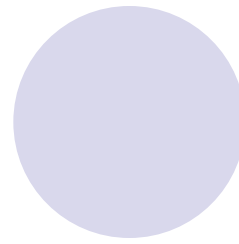
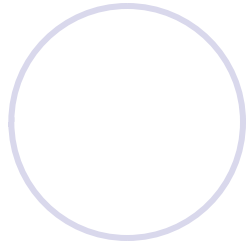
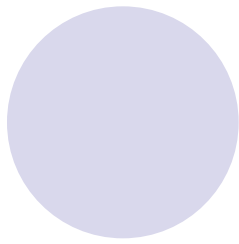
```
namespace ex06_08
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }

        //radioButton1的Checked属性值改变时，触发
        radioButton1_CheckedChanged事件
        private void radioButton1_CheckedChanged(object sender,
            EventArgs e)
        {
            if (radioButton1.Checked == true)
                richTextBox1.Text += "学位: " + radioButton1.Text + "\n";
        }

        private void radioButton2_CheckedChanged(object sender,
            EventArgs e)
        {
            if (radioButton2.Checked == true)
                richTextBox1.Text += "学位: " + radioButton2.Text + "\n";
        }
    }
}
```



```
private void radioButton3_CheckedChanged(object sender, EventArgs e)
{
    if (radioButton3.Checked == true) // 如果选中
        richTextBox1.Text += "学位: " + radioButton3.Text + "\n";
}
//textBox1中有键按下时，触发textBox1_KeyPress事件
private void textBox1_KeyPress(object sender, KeyPressEventArgs e)
{
    if(e.KeyChar==13) //如果键入回车键，则向richTextBox1中添加姓名
        richTextBox1.Text += "姓名: " + textBox1.Text + "\n";
}
//通常初始化在Form1_Load事件中完成
private void Form1_Load(object sender, EventArgs e)
{
    richTextBox1.Clear();
}
```



- //button1的单击事件：清除功能
- private void button1_Click(object sender, EventArgs e)
- {
- richTextBox1.Clear();
- }
- //comboBox1中有键按下时，触发comboBox1_KeyPress事件
- private void comboBox1_KeyPress(object sender,
- KeyPressEventArgs e)
- {
- if ((e.KeyChar==13)&&comboBox1.Text!="")
- richTextBox1.Text+="性别: "+comboBox1.Text+"\n";
- }

● //checkBox1的Checked属性改变会触发它的CheckedChanged事件

● private void checkBox1_CheckedChanged(object sender, EventArgs e)
● {

● if (checkBox1.Checked == true)
● richTextBox1.Text += "爱好" + checkBox1.Text + "\n";

● }
● private void checkBox2_CheckedChanged(object sender, EventArgs e)
● {

● if (checkBox2.Checked == true)
● richTextBox1.Text += "爱好" + checkBox2.Text + "\n";

● }
● private void checkBox3_CheckedChanged(object sender, EventArgs e)
● {

● if (checkBox3.Checked == true)
● richTextBox1.Text += "爱好" + checkBox3.Text + "\n";

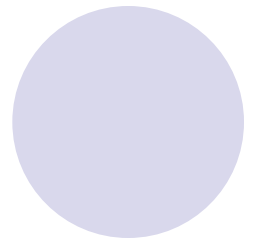
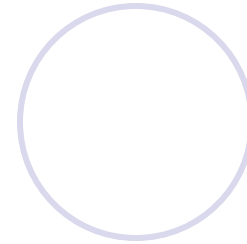
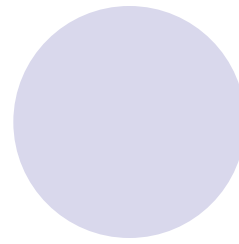
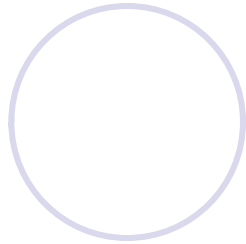
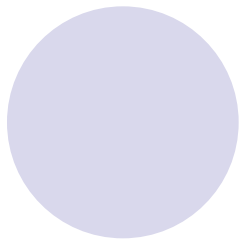
● }
● //listBox1中的选项发生改变时触发它的SelectedIndexChanged事件

● private void listBox1_SelectedIndexChanged(object sender, EventArgs e)
● {

● richTextBox1.Text+="专业: "+listBox1.SelectedItem+"\n";

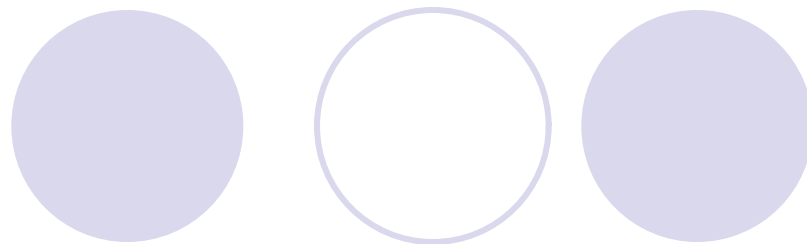
● }


- private void comboBox1_SelectedIndexChanged(object sender, EventArgs e)
- {
- richTextBox1.Text += "性别: " + comboBox1.Text + "\n";
- }
- private void button2_Click(object sender, EventArgs e)//实现保存功能
- {
- //将文件以RTF格式保存到指定路径下
- richTextBox1.SaveFile(@"e:\zm\c#example\xinxi.rtf",RichText
- BoxStr
- eamType.RichText);
- }
- private void button3_Click(object sender, EventArgs e)//实现载入功能
- {
- //将指定文件加载到RichTextBox中
- richTextBox1.LoadFile(@"e:\zm\c#example\xinxi.rtf",RichText
- BoxStr
- eamType.RichText);
- }



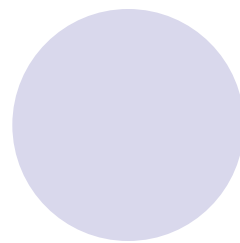
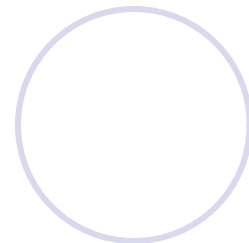
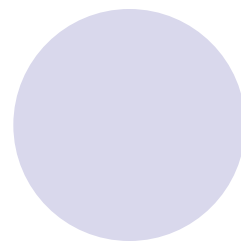
- `private void button4_Click(object sender, EventArgs e)//`
实现复制功能
- `{`
- `richTextBox1.Copy();`
- `}`
- `private void button5_Click(object sender,`
`EventArgs e)//实现粘贴功能`
- `{`
- `richTextBox1.Paste();`
- `}`
- `}`

6.2.9 面板控件



- **Panel(面板)**控件，在工具箱中的图标  Panel
- 是，可以包含其他控件，使用面板控件来组合控件的集合。与分组框相比，共同点是它们都是容器控件，区别是，面板没有标题属性

6.2.10 图片框控件



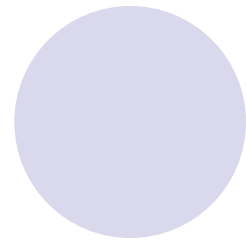
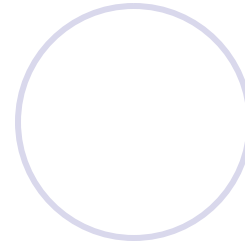
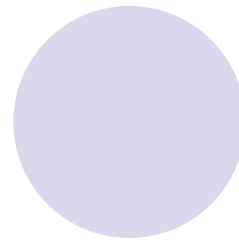
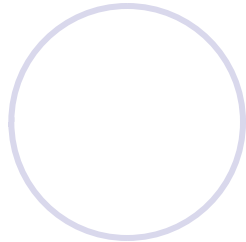
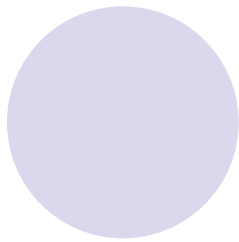
- PictureBox(图片框)控件，在工具箱中的图标是



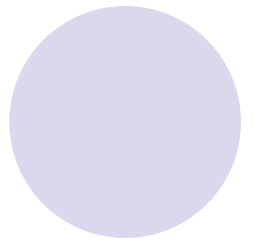
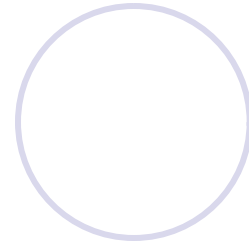
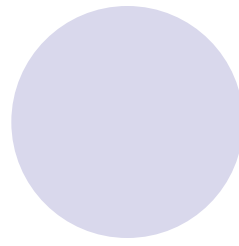
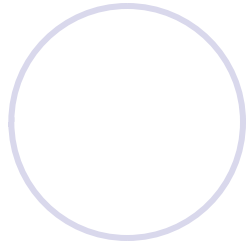
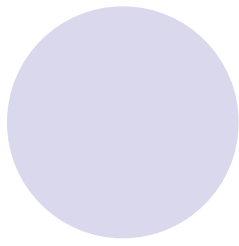
PictureBox，可以用来在窗体或某容器中的指定位置上显示图片。**PictureBox**的常用属性和方法如表6-16所示。

表6-16 PictureBox的常用属性和方法

属性/方法	描 述
Image	图片框中显示的图片
ImageLocation	图片加载的磁盘路径或Web位置
BackgroundImage	图片框的背景图片
BackgroundImageLayout	图片框的背景图片布局方式

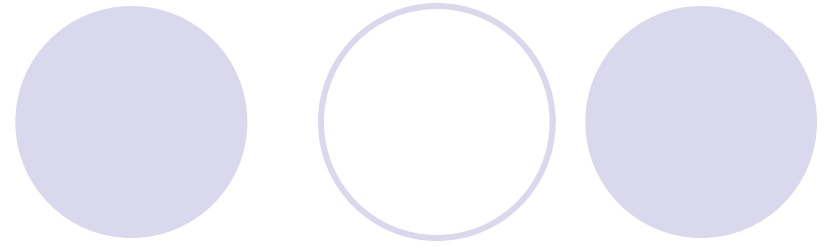


SizeMode	图片框中显示图片的方式： AutoSize （自动调整图片框的大小）、 CenterImage （图片居中显示）、 Normal （图片从图片框的左上角开始显示）、 StretchImage （图片拉伸或收缩，使之占满图片框无法保持图片的宽高比）、 Zoom （保持宽高比不变前提下将图片缩放，使之占满图片框）
Load（ ）	程序运行期间将指定路径的图片加载到图片框中



- 程序员可以在设计阶段通过属性窗口直接设置**Image**属性，将图片导入到图片框内显示，也可以在程序运行阶段将图片文件的路径赋予**ImageLocation**属性，或者调用**Load()**方法，将图片加载到图片框中

6.2.11 定时器控件

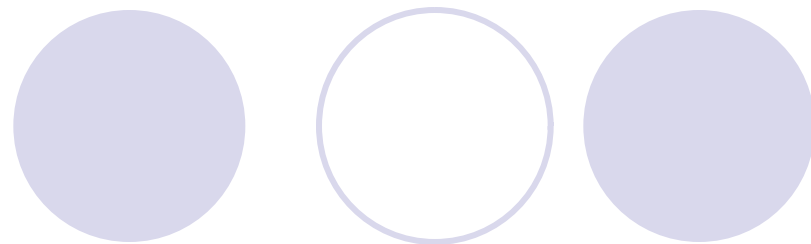


- **Timer(定时器)**控件，按照规定的时间间隔(**Interval**属性)，重复地触发事件(**Tick**事件)。定时器的常用属性、方法、事件如表**6-17**所示。

定时器的常用属性、方法、事件

属性/方法 /事件	描 述
Interval属性	设定时间间隔，单位为毫秒
Enable属性	是否有效
Start()方法	启动定时器
Stop()方法	停止定时器
Tick事件	当时钟处于运行(有效)状态时，每当到达指定时间间隔，就会触发这个事件

6.2.12 滚动条控件

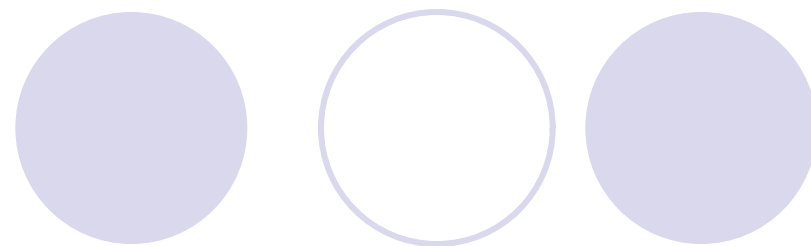


- 滚动条控件包括水平滚动条(**HScrollBar**)和垂直滚动条(**VScrollBar**)。当一个页面上的内容不能完全显示时，可以拖动滚动条上的滑块，或者单击滚动条两端的箭头来移动页面，从而浏览页面上的不同部分。滚动条的属性、事件如表**6-18**所示。

滚动条的属性、事件

属性/事件	描 述
Value	滚动条滑块的当前位置
Maximum	滚动条滑块滚动范围的最大值
Minmum	滚动条滑块滚动范围的最小值
LargeChange	鼠标单击滚动箭头与滑块之间的区域时，滑块的改变值
SmallChange	鼠标单击滚动箭头时，滑块的改变值
Dock	滚动条在所在容器中的停靠位置
RighttoLeft	对HScrollBar，滚动条的最小值是在左边还是右边
Scroll事件	通过鼠标或键盘操作移动滚动条的滑块后，将触发该事件
ValueChanged事件	当控件的值被改变时触发

6.2.13 月历控件



- **MonthCalendar**(月历)控件，为用户查看和设置日期信息提供了一个直观的图形界面。该控件以网格形式显示日历，网格包含选择的年份月份对应的日期，这些日期排列在周一到周日下的7列中，并突出显示选定的日期范围。可以单击月份标题任何一侧的箭头按钮来选择不同的月份。与 **DateTimePicker** 控件不同，月历控件可以选择多个日期，但其选择范围仅限一周(按住**SHIFT**键用鼠标单击范围)。**MonthCalendar**控件通常用于选择日期示。其属性如表6-19所示。

表6-19 MonthCalendar控件的常用属性

属 性	描 述
TitleBackColor	月历标题背景色
TitleForeColor	月历标题前景色
SelectionRange	在月历中显示的时间范围， Start 为开始， End 为截止
MinimumSize	最小值
ShowToday	是否显示今天日期
ShowTodayCircle	是否在今天日期上加上红圈
ShowWeekNumbers	是否在月历左侧显示周数

- **【例6-8】** 通过触发monthCalendar1的DateChanged事件，在该事件中显示当前日期、选择的起始日期及结束日期。代码如下：
- private void monthCalendar1_DateChanged(object sender, DateRangeEventArgs e)
- {
- this.label1.Text="今天日期：
"+this.monthCalendar1.TodayDate.ToString("yyyy年MM月dd日");
- this.label2.Text="起始日期：
"+this.monthCalendar1.SelectionStart.ToString ("yyyy年MM月dd日");
- this.label3.Text="结束日期：
"+this.monthCalendar1.SelectionEnd.ToString ("yyyy年MM月dd日");
- }

6.2.14 工具栏控件

- **ToolStrip(工具栏)**控件，双击工具箱中的工具栏控件，它的图标会显示在窗体设计窗口的脚标区域。同时在窗体的工作区顶部出现如图6.23所示的工具栏。单击工具栏按钮右端的下拉箭头，从下拉列表中选择所需要的类型，把它们添加到工具栏中。或者在**ToolStrip**控件的属性窗口中单击**Items**属性右侧的按钮，显示“项集合编辑器”窗口，如图6.24所示。选择某个项以后，单击“添加”按钮，即可向工具栏中添加项。

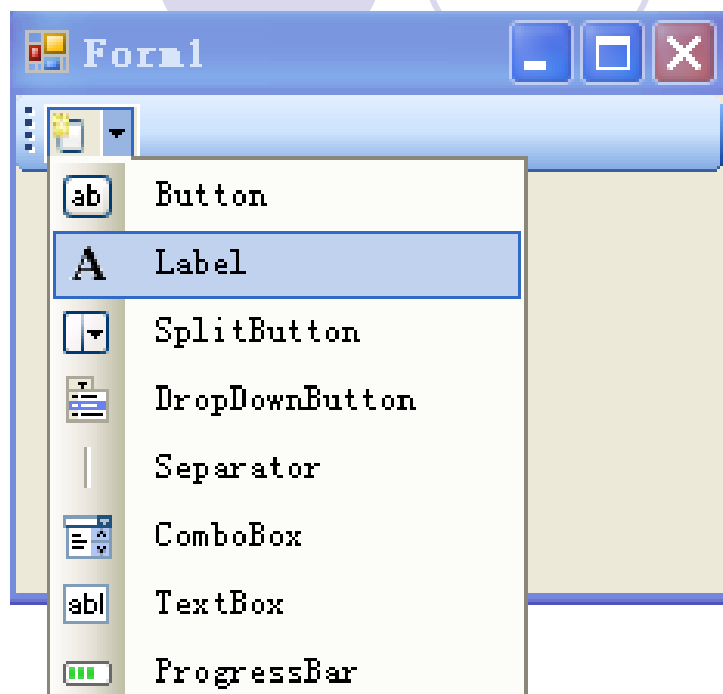


图6.23 在窗体中的工具栏

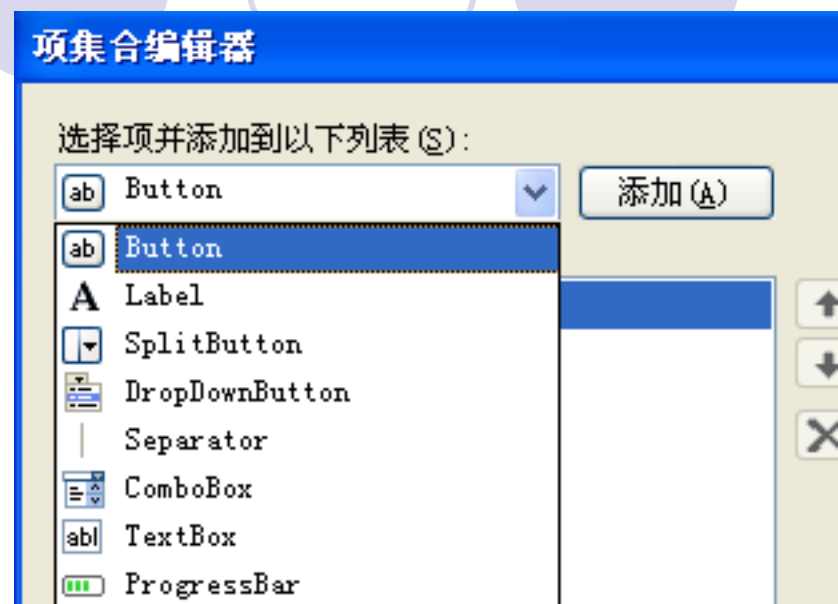
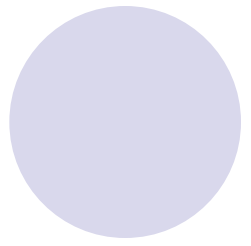
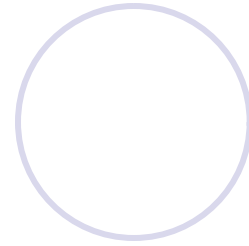
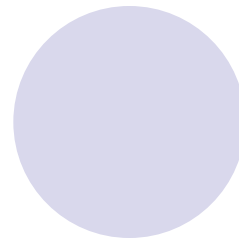
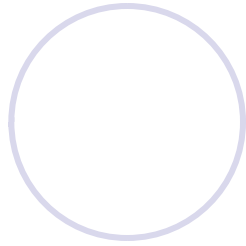
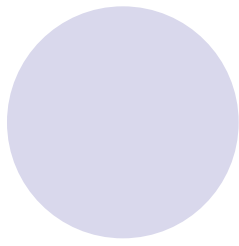
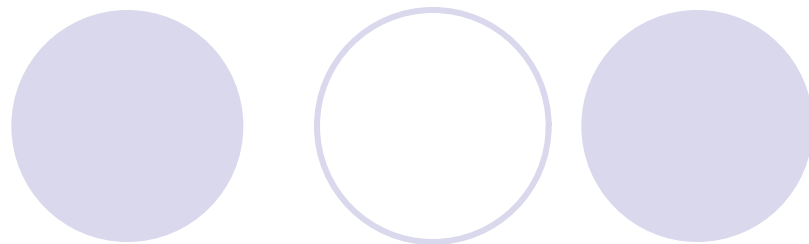


图6.24 通过属性向工具栏中添加项



- 若向工具栏中添加按钮项后，在项集合编辑器的右侧会出现**ToolStripButton** 常用属性。其常用属性解释如下：
 - **DisplayStyle**: 按钮标题的显示方式；
 - **Image**: 设置工具按钮上的显示图标；
 - **Text**: 指定显示在按钮上的文本内容；
 - **ToolTipText**: 指定按钮的提示内容。
- 工具栏按钮一旦添加至工具栏，单击某一按钮即可在属性窗口设置其属性，不必再用“项集合编辑器”窗口来设置。

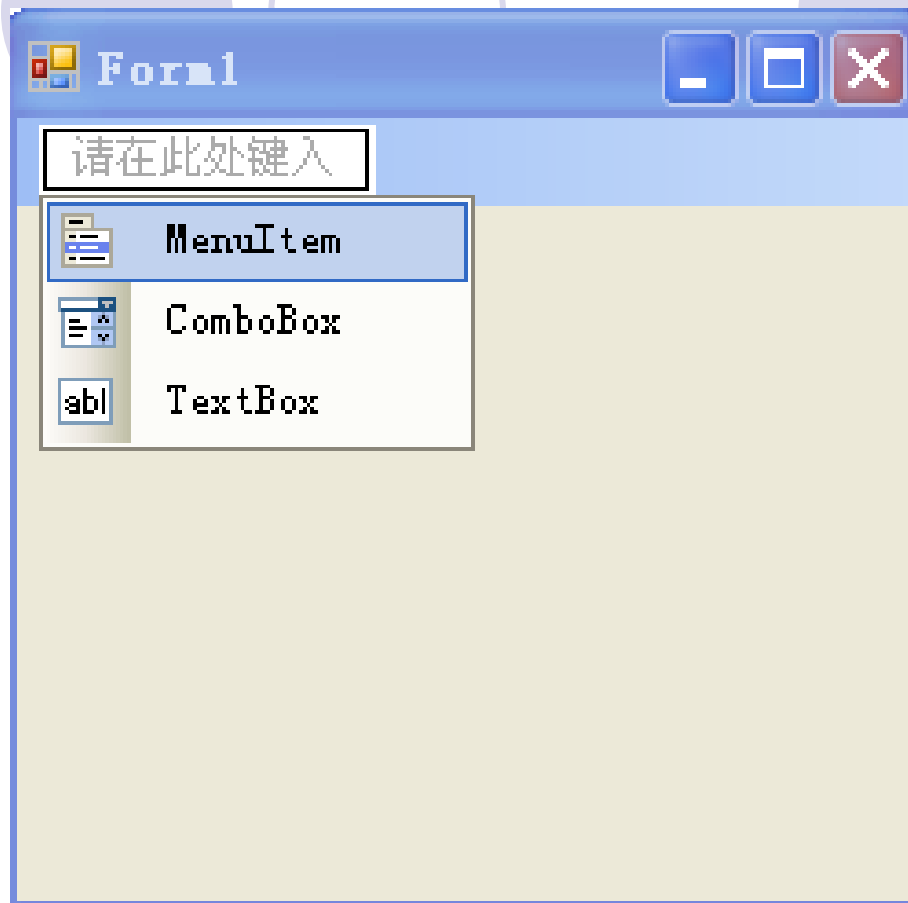
6.2.15 状态栏控件



- **StatusStrip(状态栏)**控件，添加状态栏后，在窗体设计器中单击窗体上的状态栏控件下拉箭头，在弹出的下拉列表中选择所需项的类型(**StatusLabel**、**ProgressBar**、**Drop DownButton**、**SplitButton**)，将其添加到状态栏中。
- 同样也可以在状态栏的属性窗口中通过**Items**属性添加状态栏中的项。最常用的**StatusLabel**面板(标签面板)的常用属性如下：
 - **AutoSize**：决定是否根据内容调整面板的大小。
 - **BorderSiders**：指定面板边框的显示。

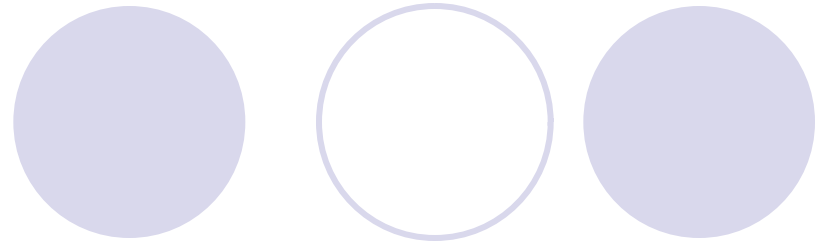
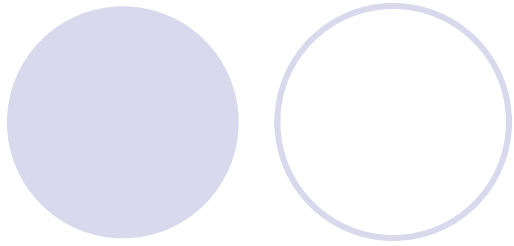
6.3 菜单

- 6.3.1 菜单控件
- **MenuStrip(菜单)控件**。**Windows**应用程序通常提供菜单，菜单包括各种基本命令，并按照主题分组。
- 将**MenuStrip**控件添加到窗体后，它出现在**Windows**窗体设计器底部的栏中，同时在窗体的顶部出现主菜单设计器。
- 1. 菜单项的设计
- 当菜单控件添加到窗体中以后，可以直接在标题栏下的菜单中添加各个菜单项，如图6.26所示。

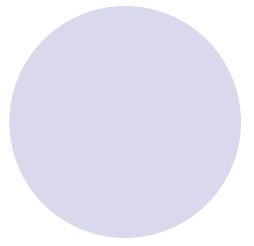
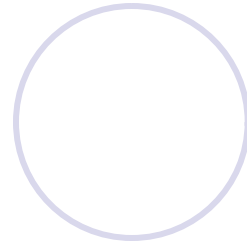
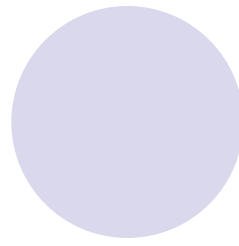
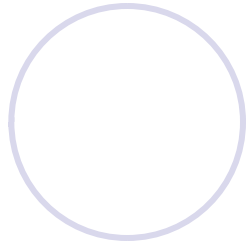
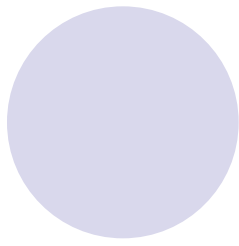


也可以在属性窗口中单击菜单的**Items**属性右侧的按钮，打开项集合编辑器，在该编辑器中添加各个菜单项。

图6.26 直接在菜单中添加各个菜单项



- 2. 分隔条
- 当二级菜单中选项较多时，可以使用分隔条将其分组，使得菜单结构更加清晰。
- 在已输入的菜单项上单击鼠标右键，就会弹出一个菜单，从中选择“插入”→“**Separator**”菜单项，可以在选项的上面添加一个分隔条。



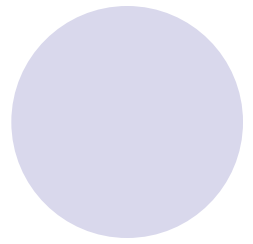
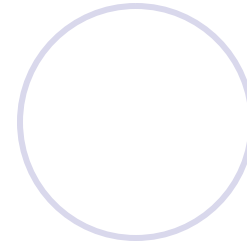
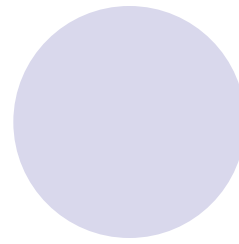
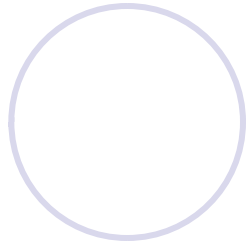
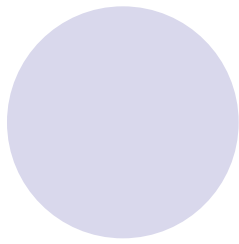
- 3. 访问键和快捷键
- 访问键(**Access key**): 访问键是菜单项文字后面的括号中带下画线的字符, 它是在菜单设计时定义的。定义访问键后, 运行时可按下“**ALT+访问键**”。在菜单项的标题文字后面加上一个由“&”引导的字母, 即可完成访问键的设定。
- 快捷键(**ShortCut key**): 设置每个菜单项时, 在属性窗口**ShortCutKeys**属性右端, 单击下拉箭头, 就可在列表选择一个合适的快捷键。

【例6-9】 创建一个菜单驱动，实现在富文本框内的一个简单文字编辑功能的程序，用户界面菜单如图6.27所示。

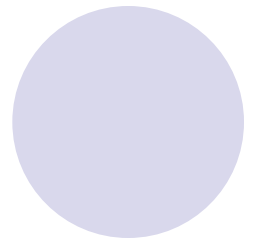
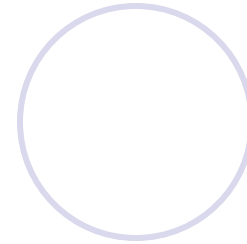
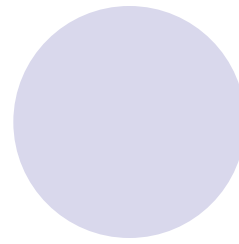
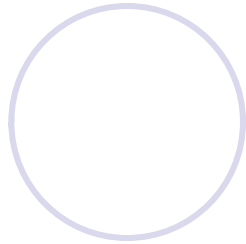
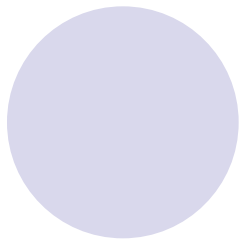


图6.27 例6-10用户界面的菜单及各个菜单项

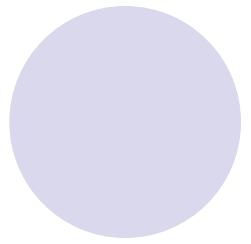
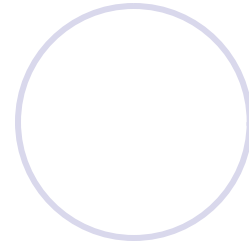
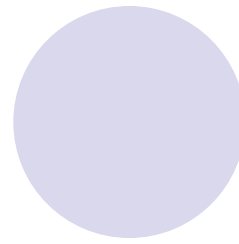
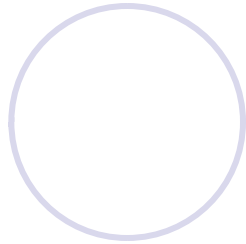
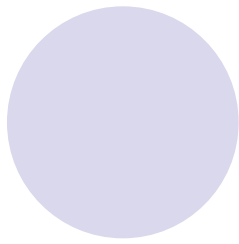
- private void 新建ToolStripMenuItem_Click(object sender, EventArgs e)
- {
- File.Create(@"E:\zm\c#example\xinjian.rtf");
- //引用命名空间: using System.IO;
- MessageBox.Show("名为xinjian.rtf文件已经创建成功","创建文件");
- }
- private void 打开ToolStripMenuItem_Click(object sender, EventArgs e)
- {
- richTextBox1.LoadFile(@"E:\zm\c#example\xinxi.rtf", RichText
- reamType.RichText);
- }
- private void 保存ToolStripMenuItem_Click(object sender, EventArgs e)
- {
- richTextBox1.SaveFile(@"E:\zm\c#example\baocun.rtf");
- MessageBox.Show("名为baocun.rtf文件已存成", "存文件");
- }



- private void 退出ToolStripMenuItem_Click(object sender, EventArgs e)
- {
- this.Close();
- }
- private void 剪切ToolStripMenuItem_Click(object sender, EventArgs e)
- {
- richTextBox1.Cut();
- }
- private void 复制ToolStripMenuItem_Click(object sender, EventArgs e)
- {
- richTextBox1.Copy();
- }



```
private void 撤销ToolStripMenuItem_Click(object sender, EventArgs e)
{
    if (richTextBox1.CanUndo == true)
        richTextBox1.Undo();
    else
        MessageBox.Show("无法撤销");
}
private void 查找ToolStripMenuItem_Click(object sender, EventArgs e)
{
    int position = richTextBox1.Text.IndexOf("好好学习");
    if (position >= 0)
        richTextBox1.Select(position, 2);
    //从当前的位置选取前两个字符
}
```

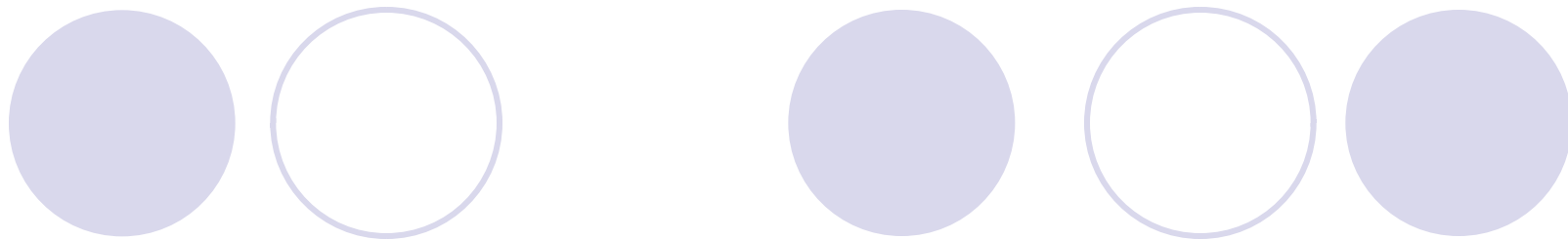
- `private void 替换ToolStripMenuItem_Click(object sender, EventArgs e)`
- `{`
- `string str1 = richTextBox1.SelectedText; //选中的文本给str1`
- `richTextBox1.SelectedText = str1.Replace("好好学习", "勤奋努力");`
- `//在str1中查找“好好学习”，如果存在，则替换之`
- `}`
- `private void 粘贴ToolStripMenuItem_Click(object sender, EventArgs e)`
- `{ richTextBox1.Paste(); }`

6.3.2 上下文菜单

- **ContextMenuStrip**控件也叫上下文菜单。要显示弹出菜单，或在用户右击鼠标时显示一个菜单，就应该使用**ContextMenuStrip**类。菜单项的设计与**MenuStrip**相同，也是添加**ToolStripMenuItems**，定义每项的**Click**事件，执行某项任务。
- 上下文菜单应赋予特定的控件，并设置该控件的**ContextMenuStrip**属性，这样就把该特定控件和上下文菜单联系起来，当用户在右击该控件时，就可显示该上下文菜单的各菜单项。选中其中一个菜单项，即可执行该菜单项的**Click**事件。

6.4 对话框

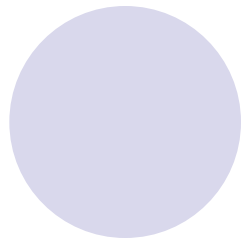
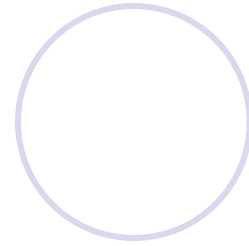
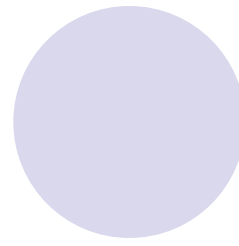
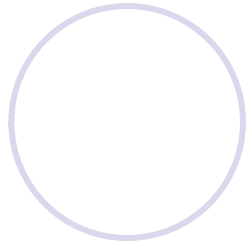
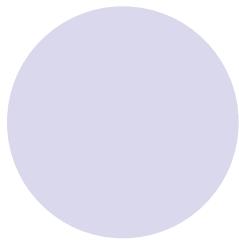
- 对话框主要用于显示信息或接收用户的输入。对话框是窗体的一种特殊形式，只要把窗体的边框风格(**FormBorderStyle**)设置为**FixedDialog**，就可以创建对话框。对话框不可以改变大小，通常需要把它的**ControlBox**、**MaximizeBox**、**MinimizeBox**属性设置为**False**，用来从标题栏中去掉最大化、最小化按钮。
- 对话框可以分为模式对话框和非模式对话框两种。模式对话框是指用户只能对当前的对话框窗体进行操作，在该窗体关闭之前不能切换到程序的其他窗口；非模式对话框是指当前所操作的对话框窗体可以与程序的其他窗体进行切换。



- 使用**ShowDialog()**方法显示对话框为模式对话框。使用**Show()**方法显示对话框为非模式对话框。使用**ShowDialog()**方法可以接收用户的输入，另外，还可以使用**ShowDialog()**方法的返回值或对话框的**DialogResult**属性。对话框的**DialogResult**属性的可能值在枚举**System.Windows.Forms.DialogResult**中定义。枚举**System.Windows.Forms.DialogResult**的成员及含义如表6-20所示。

表6-20 枚举System.Windows.Forms.DialogResult的成员及其含义

枚 举 成 员	描 述
Abort	对话框的返回值为Abort(终止)
Cancel	对话框的返回值为Cancel(取消)
Ignore	对话框的返回值为Ignore(忽略)
No	对话框的返回值为No(否)
None	对话框没有返回值，该情况表示模式对话框仍在运行
OK	对话框的返回值为OK(确定)
Retry	对话框的返回值为Retry(重试)
Yes	对话框的返回值为Yes(是)



- 以下代码表示对话框**Dialog1**的返回值是否为**Ignore**，如果是，则执行相应操作。
- `if`
`(Dialog1.ShowDialog()==System.Windows.Forms.DialogResult.Ignore)`
- `{ }`
- 或 `if`
`(Dialog1.DialogResult==System.Windows.Forms.DialogResult.Ignore)`
- `{ }`