

3. C#控件查询手册

1、窗体（Form）

1、常用属性

（1）Name 属性：用来获取或设置窗体的名称，在应用程序中可通过 Name 属性来引用窗体。

（2）WindowState 属性：用来获取或设置窗体的窗口状态。取值有三种：Normal（窗体正常显示）、Minimized（窗体以最小化形式显示）和 Maximized（窗体以最大化形式显示）。

（3）StartPosition 属性：用来获取或设置运行时窗体的起始位置。

（4）Text 属性：该属性是一个字符串属性，用来设置或返回在窗口标题栏中显示的文字。

（5）Width 属性：用来获取或设置窗体的宽度。

（6）Height 属性：用来获取或设置窗体的高度。

（7）Left 属性：用来获取或设置窗体的左边缘的 x 坐标（以像素为单位）。

（8）Top 属性：用来获取或设置窗体的上边缘的 y 坐标（以像素为单位）。

（9）ControlBox 属性：用来获取或设置一个值，该值指示在该窗体的标题栏中是否显示控制框。值为 true 时将显示控制框，值为 false 时不显示控制框。

（10）MaximizeBox 属性：用来获取或设置一个值，该值指示是否在窗体的标题栏中显示最大化按钮。值为 true 时显示最大化按钮，值为 false 时不显示最大化按钮。

（11）MinimizeBox 属性：用来获取或设置一个值，该值指示是否在窗体的标题栏中显示最小化按钮。值为 true 时显示最小化按钮，值为 false 时不显示最小化按钮。

（12）AcceptButton 属性：该属性用来获取或设置一个值，该值是一个按钮的名称，当按 Enter 键时就相当于单击了窗体上的该按钮。

（13）CancelButton 属性：该属性用来获取或设置一个值，该值是一个按钮的名称，当按 Esc 键时就相当于单击了窗体上的该按钮。

（14）Modal 属性：该属性用来设置窗体是否为有模式显示窗体。如果有模式地显示该窗体，该属性值为 true；否则为 false。当有模式地显示窗体时，只能对模式窗体上的对象进行输入。必须隐藏或关闭模式窗体（通常是响应某个用户操作），然后才能对另一窗体进行输入。有模式显示的窗体通常用做应用程序中的对话框。

（15）ActiveControl 属性：用来获取或设置容器控件中的活动控件。窗体也是一种容器控件。

（16）ActiveMdiChild 属性：用来获取多文档界面（MDI）的当前活动子窗口。

（17）AutoScroll 属性：用来获取或设置一个值，该值指示窗体是否实现自动滚动。如果此属性值设置为 true，则当任何控件位于窗体工作区之外时，会在该窗体上显示滚动条。另外，当自动滚动打开时，窗体的工作区自动滚动，以使具有输入焦点的控件可见。

（18）BackColor 属性：用来获取或设置窗体的背景色。

（19）BackgroundImage 属性：用来获取或设置窗体的背景图像。

（20）Enabled 属性：用来获取或设置一个值，该值指示控件是否可以对用户交互作出响应。如果控件可以对用户交互作出响应，则为 true；否则为 false。默认值为 true。

（21）Font 属性：用来获取或设置控件显示的文本的字体。

（22）ForeColor 属性：用来获取或设置控件的前景色。

（23）IsMdiChild 属性：获取一个值，该值指示该窗体是否为多文档界面（MDI）子

窗体。值为 `true` 时，是子窗体，值为 `false` 时，不是子窗体。

(24) `IsMdiContainer` 属性：获取或设置一个值，该值指示窗体是否为多文档界面 (MDI) 中的子窗体的容器。值为 `true` 时，是子窗体的容器，值为 `false` 时，不是子窗体的容器。

(25) `KeyPreview` 属性：用来获取或设置一个值，该值指示在将按键事件传递到具有焦点的控件前，窗体是否将接收该事件。值为 `true` 时，窗体将接收按键事件，值为 `false` 时，窗体不接收按键事件。

(26) `MdiChildren` 属性：数组属性。数组中的每个元素表示以此窗体作为父级的多文档界面 (MDI) 子窗体。

(27) `MdiParent` 属性：用来获取或设置此窗体的当前多文档界面 (MDI) 父窗体。

(28) `ShowInTaskbar` 属性：用来获取或设置一个值，该值指示是否在 Windows 任务栏中显示窗体。

(29) `Visible` 属性：用于获取或设置一个值，该值指示是否显示该窗体或控件。值为 `true` 时显示窗体或控件，为 `false` 时不显示。

(30) `Capture` 属性：如果该属性值为 `true`，则鼠标就会被限定只由此控件响应，不管鼠标是否在此控件的范围内。

2、常用方法

下面介绍一些窗体的最常用方法。

(1) `Show` 方法：该方法的作用是让窗体显示出来，其调用格式为：`窗体名.Show()`；其中窗体名是要显示的窗体名称。

(2) `Hide` 方法：该方法的作用是把窗体隐藏出来，其调用格式为：`窗体名.Hide()`；其中窗体名是要隐藏的窗体名称。

(3) `Refresh` 方法：该方法的作用是刷新并重画窗体，其调用格式为：`窗体名.Refresh()`；其中窗体名是要刷新的窗体名称。

(4) `Activate` 方法：该方法的作用是激活窗体并给予它焦点。其调用格式为：`窗体名.Activate()`；

其中窗体名是要激活的窗体名称。

(5) `Close` 方法：该方法的作用是关闭窗体。其调用格式为：`窗体名.Close()`；

其中窗体名是要关闭的窗体名称。

(6) `ShowDialog` 方法：该方法的作用是将窗体显示为模式对话框。其调用格式为：`窗体名.ShowDialog()`；

3. 常用事件

(1) `Load` 事件：该事件在窗体加载到内存时发生，即在第一次显示窗体前发生。

(2) `Activated` 事件：该事件在窗体激活时发生。

(3) `Deactivate` 事件：该事件在窗体失去焦点成为不活动窗体时发生。

(4) `Resize` 事件：该事件在改变窗体大小时发生。

(5) `Paint` 事件：该事件在重绘窗体时发生。

(6) `Click` 事件：该事件在用户单击窗体时发生。

(7) `DoubleClick` 事件：该事件在用户双击窗体时发生。

(8) `Closed` 事件：该事件在关闭窗体时发生。

文本框类控件

2、Label (标签)控件

1、常用属性:

(1) **Text** 属性: 用来设置或返回标签控件中显示的文本信息。

(2) **AutoSize** 属性: 用来获取或设置一个值, 该值指示是否自动调整控件的大小以完整显示其内容。取值为 **true** 时, 控件将自动调整到刚好能容纳文本时的大小, 取值为 **false** 时, 控件的大小为设计时的大小。默认值为 **false**。

(3) **Anchor** 属性: 用来确定此控件与其容器控件的固定关系的。所谓容器控件指的是这样一种情况: 往往在控件之中还有一个控件, 例如最典型的就是窗体控件中会包含很多的控件, 像标签控件、文本框等。这时称包含控件的控件为容器控件或父控件, 而父控件称为子控件。

这样将遇到一个问题, 即子控件与父控件的位置关系问题, 即当父控件的位置、大小变化时, 子控件按照什么样的原则改变其位置、大小。**Anchor** 属性就规定了这个原则。对于 **Anchor** 属性, 可以设定 **Top**、**Bottom**、**Right**、**Left** 中的任意几种, 设置的方法是在属性窗口中单击 **Anchor** 属性右边的箭头,

(4) **BackColor** 属性: 用来获取或设置控件的背景色。当该属性值设置为 **Color.Transparent** 时, 标签将透明显示, 即背景色不再显示出来。

(5) **BorderStyle** 属性: 用来设置或返回边框。有三种选择:

BorderStyle.None 为无边框 (默认), **BorderStyle.FixedSingle** 为固定单边框, **BorderStyle.Fixed3D** 为三维边框。

(6) **TabIndex** 属性: 用来设置或返回对象的 **Tab** 键顺序。

(7) **Enabled** 属性: 用来设置或返回控件的状态。值为 **true** 时允许使用控件, 值为 **false** 时禁止使用控件, 此时标签呈暗淡色, 一般在代码中设置。

3、TextBox (文本框) 控件

1、主要属性:

(1) **Text** 属性: **Text** 属性是文本框最重要的属性, 因为要显示的文本就包含在 **Text** 属性中。默认情况下, 最多可在一个文本框中输入 2048 个字符。如果将 **MultiLine** 属性设置为 **true**, 则最多可输入 32KB 的文本。**Text** 属性可以在设计时使用【属性】窗口设置, 也可以在运行时用代码设置或者通过用户输入来设置。可以在运行时通过读取 **Text** 属性来获得文本框的当前内容。

(2) **MaxLength** 属性: 用来设置文本框允许输入字符的最大长度, 该属性值为 0 时, 不限制输入的字符数。

(3) **MultiLine** 属性: 用来设置文本框中的文本是否可以输入多行并以多行显示。值为 **true** 时, 允许多行显示。值为 **false** 时不允许多行显示, 一旦文本超过文本框宽度时, 超过部分不显示。

(4) **HideSelection** 属性: 用来决定当焦点离开文本框后, 选中的文本是否还以选中的方式显示, 值为 **true**, 则不以选中的方式显示, 值为 **false** 将依旧以选中的方式显示。

(5) **ReadOnly** 属性: 用来获取或设置一个值, 该值指示文本框中的文本是否为只读。值为 **true** 时为只读, 值为 **false** 时可读可写。

(6) **PasswordChar** 属性: 是一个字符串类型, 允许设置一个字符, 运行程序时, 将输入到 **Text** 的内容全部显示为该属性值, 从而起到保密作用, 通常用来输入口令或密码。

(7) **ScrollBars** 属性: 用来设置滚动条模式, 有四种选择: **ScrollBars.None** (无

滚动条), `ScrollBars.Horizontal` (水平滚动条), `ScrollBars.Vertical` (垂直滚动条), `ScrollBars.Both` (水平和垂直滚动条)。

注意: 只有当 `MultiLine` 属性为 `true` 时, 该属性值才有效。在 `WordWrap` 属性值为 `true` 时, 水平滚动条将不起作用

(8) `SelectionLength` 属性: 用来获取或设置文本框中选定的字符数。只能在代码中使用, 值为 0 时, 表示未选中任何字符。

(9) `SelectionStart` 属性: 用来获取或设置文本框中选定的文本起始点。只能在代码中使用, 第一个字符的位置为 0, 第二个字符的位置为 1, 依此类推。

(10) `SelectedText` 属性: 用来获取或设置一个字符串, 该字符串指示控件中当前选定的文本。只能在代码中使用。

(11) `Lines`: 该属性是一个数组属性, 用来获取或设置文本框控件中的文本行。即文本框中的每一行存放在 `Lines` 数组的一个元素中。

(12) `Modified`: 用来获取或设置一个值, 该值指示自创建文本框控件或上次设置该控件的内容后, 用户是否修改了该控件的内容。值为 `true` 表示修改过, 值为 `false` 表示没有修改过。

(13) `TextLength` 属性: 用来获取控件中文本的长度。

(14) `WordWrap`: 用来指示多行文本框控件在输入的字符超过一行宽度时是否自动换行到下一行的开始, 值为 `true`, 表示自动换到下一行的开始, 值为 `false` 表示不自动换到下一行的开始。

2、常用方法:

(1) `AppendText` 方法: 把一个字符串添加到文件框中文本的后面, 调用的一般格式如下:

文本框对象.`AppendText(str)`, 参数 `str` 是要添加的字符串。

(2) `Clear` 方法: 从文本框控件中清除所有文本。调用的一般格式如下: 文本框对象.`Clear()`该方法无参数。

(3) `Focus` 方法: 是为文本框设置焦点。如果焦点设置成功, 值为 `true`, 否则为 `false`。调用的一般格式如下:

文本框对象.`Focus()`该方法无参数。

(4) `Copy` 方法: 将文本框中的当前选定内容复制到剪贴板上。调用的一般格式如下: 文本框对象.`Copy()`该方法无参数。

(5) `Cut` 方法: 将文本框中的当前选定内容移动到剪贴板上。调用的一般格式如下: 文本框对象.`Cut()`该方法无参数。

(6) `Paste` 方法: 用剪贴板的内容替换文本框中的当前选定内容。调用的一般格式如下:

文本框对象.`Paste()`该方法无参数。

(7) `Undo` 方法: 撤销文本框中的上一个编辑操作。调用的一般格式如下: 文本框对象.`Undo()`该方法无参数。

(8) `ClearUndo` 方法: 从该文本框的撤销缓冲区中清除关于最近操作的信息, 根据应用程序的状态, 可以使用此方法防止重复执行撤销操作。调用的一般格式如下:

文本框对象.`ClearUndo()`该方法无参数。

(9) `Select` 方法: 用来在文本框中设置选定文本。调用的一般格式如下: 文本框对象.`Select(start,length)`

该方法有两个参数, 第一个参数 `start` 用来设定文本框中当前选定文本的第一个字符的位置, 第二个参数 `length` 用来设定要选择的字符数。

(10) **SelectAll** 方法：用来选定文本框中的所有文本。调用的一般格式如下： 文本框对象.**SelectAll()**该方法无参数。

3、常用事件：

(1) **GotFocus** 事件：该事件在文本框接收焦点时发生。

(2) **LostFocus** 事件：该事件在文本框失去焦点时发生。

(3) **TextChanged** 事件：该事件在 **Text** 属性值更改时发生。无论是通过编程修改还是用户交互更改文本框的 **Text** 属性值，均会引发此事件。

4、RichTextBox 控件

RichTextBox 是一种既可以输入文本、又可以编辑文本的文字处理控件，与 **TextBox** 控件相比，**RichTextBox** 控件的文字处理功能更加丰富，不仅可以设定文字的颜色、字体，还具有字符串检索功能。另外，**RichTextBox** 控件还可以打开、编辑和存储.rtf 格式文件、ASCII 文本格式文件及 Unicode 编码格式的文件。

1、常用属性

上面介绍的 **TextBox** 控件所具有的属性，**RichTextBox** 控件基本上都具有，除此之外，该控件还具有一些其他属性。

(1) **RightMargin** 属性：用来设置或获取右侧空白的大小，单位是像素。通过该属性可以设置右侧空白，如希望右侧空白为 50 像素，可使用如下语句：
RichTextBox1.RightMargin=RichTextBox1.Width-50;

(2) **Rtf** 属性：用来获取或设置 **RichTextBox** 控件中的文本，包括所有 RTF 格式代码。可以使用此属性将 RTF 格式文本放到控件中以进行显示，或提取控件中的 RTF 格式文本。此属性通常用于在 **RichTextBox** 控件和其他 RTF 源(如 **MicrosoftWord** 或 **Windows** 写字板)之间交换信息。

(3) **SelectedRtf** 属性：用来获取或设置控件中当前选定的 RTF 格式的格式文本。此属性使用户得以获取控件中的选定文本，包括 RTF 格式代码。如果当前未选定任何文本，给该属性赋值将把所赋的文本插入到插入点处。如果选定了文本，则给该属性所赋的文本值将替换掉选定文本。

(4) **SelectionColor** 属性：用来获取或设置当前选定文本或插入点处的文本颜色。

(5) **SelectionFont** 属性：用来获取或设置当前选定文本或插入点处的字体。

2、常用方法

前面介绍的 **TextBox** 控件所具有的方法，**RichTextBox** 控件基本上都具有，除此之外，该控件还具有一些其他方法。

(1) **Redo** 方法：用来重做上次被撤销的操作。调用的一般格式如下：**RichTextBox** 对象.**Redo()**
该方法无参数。

(2) **Find** 方法：用来从 **RichTextBox** 控件中查找指定的字符串。经常使用的调用格式如下：

[格式 1]: **RichTextBox** 对象.**Find(str)**

[功能]: 在指定的“**RichTextBox**”控件中查找文本，并返回搜索文本的第一个字符在控件内的位置。如果未找到搜索字符串或者 **str** 参数指定的搜索字符串为空，则返回值为 1。

[格式 2]: **RichTextBox** 对象.**Find(str,RichTextBoxFinds)**

[功能]: 在“**RichTextBox** 对象”指定的文本框中搜索 **str** 参数中指定的文本，并返回文本的第一个字符在控件内的位置。如果返回负值，则未找到所搜索的文本字符串。还可以

使用此方法搜索特定格式的文本。参数 `RichTextBoxFinds` 指定如何在控件中执行文本搜索。

[格式 3]: `RichTextBox` 对象.`Find(str,start,RichTextBoxFinds)`

[功能]: 这里 `Find` 方法与前面的格式 2 基本类似, 不同的只是通过设置控件文本内的搜索起始位置来缩小文本搜索范围, `start` 参数表示开始搜索的位置。此功能使用户得以避开可能已搜索过的文本或已经知道不包含要搜索的特定文本的文本。如果在 `options` 参数中指定了 `RichTextBoxFinds.Reverse` 值, 则 `start` 参数的值将指示反向搜索结束的位置, 因为搜索是从文档底部开始的。

(3) `SaveFile` 方法: 用来把 `RichTextBox` 中的信息保存到指定的文件中, 调用格式有以下三种。

[格式 1]: `RichTextBox` 对象名.`SaveFile(文件名);`

[功能]: 将 `RichTextBox` 控件中的内容保存为 RTF 格式文件中。

[格式 2]: `RichTextBox` 对象名.`SaveFile(文件名,文件类型);`

[功能]: 将 `RichTextBox` 控件中的内容保存为“文件类型”指定的格式文件中。

[格式 3]: `RichTextBox` 对象名.`SaveFile(数据流,数据流类型);`

[功能]: 将 `RichTextBox` 控件中的内容保存为“数据流类型”指定的数据流类型文件中。

(4) `LoadFile` 方法: 使用 `LoadFile` 方法可以将文本文件、RTF 文件装入 `RichTextBox` 控件。

主要的调用格式有以下三种。

[格式 1]: `RichTextBox` 对象名.`LoadFile(文件名);`

[功能]: 将 RTF 格式文件或标准 ASCII 文本文件加载到 `RichTextBox` 控件中。

[格式 2]: `RichTextBox` 对象名.`LoadFile(数据流,数据流类型);`

[功能]: 将现有数据流的内容加载到 `RichTextBox` 控件中。

[格式 3]: `RichTextBox` 对象名.`LoadFile(文件名,文件类型);`

[功能]: 将特定类型的文件加载到 `RichTextBox` 控件中。

5、NumericUpDown 控件

【Windows 窗体】控件组中的 `NumericUpDown` 控件看起来像是一个文本框与一对用户可单击以调整值的箭头的组合。可以通过单击向上和向下按钮、按向上和向下箭头键来增大和减小数字, 也可以直接输入数字。单击向上箭头键时, 值向最大值方向增加; 单击向下箭头键时, 值向最小值方向减少。该控件在工具箱中的图标为。

1、常用属性:

(1) `DecimalPlaces`: 获取或设置该控件中显示的小数位。

(2) `Hexadecimal`: 获取或设置一个值, 该值指示该控件是否以十六进制格式显示所包含的值。

(3) `Increment`: 获取或设置单击向上或向下按钮时, 该控件递增或递减的值。

(4) `Maximum`: 获取或设置该控件的最大值。

(5) `Minimum`: 获取或设置该控件的最小值。

(6) `Value`: 获取或设置该控件的当前值。

与 `TextBox` 控件一样, `NumericUpDown` 控件的常用事件有: `ValueChanged`、`GotFocus`、`LostFocus` 等。

按钮类控件

6、Button（按钮）控件

Button 控件又称按钮控件，是 Windows 应用程序中最常用的控件之一，通常用它来执行命令。如果按钮具有焦点，就可以使用鼠标左键、Enter 键或空格键触发该按钮的 Click 事件。通过设置窗体的 AcceptButton 或 CancelButton 属性，无论该按钮是否有焦点，都可以使用户通过按 Enter 或 Esc 键来触发按钮的 Click 事件。一般不使用 Button 控件的方法。Button 控件也具有许多如 Text、ForeColor 等的常规属性，此处不再介绍，只介绍该控件有特色的属性。以后介绍的控件也采用同样的方法来处理。

1、常用属性

（1）DialogResult 属性：当使用 ShowDialog 方法显示窗体时，可以使用该属性设置当用户按了该按钮后，ShowDialog 方法的返回值。值有：OK、Cancel、Abort、Retry、Ignore、Yes、No 等。

（2）Image 属性：用来设置显示在按钮上的图像。

（3）FlatStyle 属性：用来设置按钮的外观。

2、常用事件：

（1）Click 事件：当用户用鼠标左键单击按钮控件时，将发生该事件。

（2）MouseDown 事件：当用户在按钮控件上按下鼠标按钮时，将发生该事件。

（3）MouseUp 事件：当用户在按钮控件上释放鼠标按钮时，将发生该事件。

7、GroupBox（分组框）控件

GroupBox 控件又称为分组框，它在工具箱中的图标是。该控件常用于为其他控件提供可识别的分组，其典型的用法之一就是给 RadioButton 控件分组。可以通过分组框的 Text 属性为分组框中的控件向用户提供提示信息。

设计时，向 GroupBox 控件中添加控件的方法有两种：一是直接在分组框中绘制控件；二是把某一个已存在的控件复制到剪贴板上，然后选中分组框，再执行粘贴操作即可。位于分组框中的所有控件随着分组框的移动而一起移动，随着分组框的删除而全部删除，分组框的 Visible 属性和 Enabled 属性也会影响到分组框中的所有控件。分组框的最常用的属性是 Text，一般用来给出分组提示。

8、RadioButton（单选按钮）控件

RadioButton 又称单选按钮，其在工具箱中的图标为，单选按钮通常成组出现，用于提供两个或多个互斥选项，即在一组单选钮中只能选择一个，如图 9-14 所示。

1、常用属性：

（1）Checked 属性：用来设置或返回单选按钮是否被选中，选中时值为 true，没有选中时值为 false。

（2）AutoCheck 属性：如果 AutoCheck 属性被设置为 true（默认），那么当选择该单选按钮时，将自动清除该组中所有其他单选按钮。对一般用户来说，不需改变该属性，采用默认值（true）即可。

（3）Appearance 属性：用来获取或设置单选按钮控件的外观。当其取值为 Appearance.Button 时，将使单选按钮的外观像命令按钮一样；当选定它时，它看似已被按下。当取值为 Appearance.Normal 时，就是默认的单选按钮的外观。

（4）Text 属性：用来设置或返回单选按钮控件内显示的文本，该属性也可以包含访问

键，即前面带有“&”符号的字母，这样用户就可以通过同时按 Alt 键和访问键来选中控件。

2、常用事件：

(1) Click 事件：当单击单选按钮时，将把单选按钮的 Checked 属性值设置为 true，同时发生 Click 事件。

(2) CheckStateChanged 事件：当 Checked 属性值更改时，将触发 CheckStateChanged 事件。

9、CheckBox（复选框）控件

CheckBox 控件的常用属性如下。

(1) TextAlign 属性：用来设置控件中文字的对齐方式，有 9 种选择，如图 9-16 所示。从上到下、从左至右分别是：ContentAlignment.TopLeft、ContentAlignment.TopCenter、ContentAlignment.TopRight、ContentAlignment.MiddleLeft、ContentAlignment.MiddleCenter、ContentAlignment.MiddleRight、ContentAlignment.BottomLeft、ContentAlignment.BottomCenter 和 ContentAlignment.BottomRight。该属性的默认值为 ContentAlignment.MiddleLeft，即文字左对齐、居控件垂直方向中央。

(2) ThreeState 属性：用来返回或设置复选框是否能表示三种状态，如果属性值为 true 时，表示可以表示三种状态——选中、没选中 和 中间态（CheckState.Checked、CheckState.Unchecked 和 CheckState.Indeterminate），属性值为 false 时，只能表示两种状态——选中 和 没选中。

(3) Checked 属性：用来设置或返回复选框是否被选中，值为 true 时，表示复选框被选中，值为 false 时，表示复选框没被选中。当 ThreeState 属性值为 true 时，中间态也表示选中。

(4) CheckState 属性：用来设置或返回复选框的状态。在 ThreeState 属性值为 false 时，取值有 CheckState.Checked 或 CheckState.Unchecked。在 ThreeState 属性值被设置为 True 时，CheckState 还可以取值 CheckState.Indeterminate，在此时，复选框显示为浅灰色选中状态，该状态通常表示该选项下的多个子选项未完全选中。

CheckBox 控件的常用事件有 Click 和 CheckStateChanged 等，其含义及触发时机与单选按钮完成一致。

列表类控件

10、ListBox（列表框）控件

ListBox 控件又称列表框，它在工具箱中的图标为，它显示一个项目列表供用户选择。在列表框中，用户一次可以选择一项，也可以选择多项。

1、常用属性：

(1) Items 属性：用于存放列表框中的列表项，是一个集合。通过该属性，可以添加列表项、移除列表项和获得列表项的数目。

(2) MultiColumn 属性：用来获取或设置一个值，该值指示 ListBox 是否支持多列。值为 true 时表示支持多列，值为 false 时不支持多列。当使用多列模式时，可以使控件得以显示更多可见项。

(3) ColumnWidth 属性：用来获取或设置多列 ListBox 控件中列的宽度。

(4) SelectionMode 属性：用来获取或设置在 ListBox 控件中选择列表项的方法。当

SelectionMode 属性设置为 SelectionMode.MultiExtended 时，按下 Shift 键的同时单击鼠标或者同时按 Shift 键和箭头键之一（上箭头键、下箭头键、左箭头键和右箭头键），会将选定内容从前一选定项扩展到当前项。按 Ctrl 键的同时单击鼠标将选择或撤销选择列表中的某项；当该属性设置为 SelectionMode.MultiSimple 时，鼠标单击或按空格键将选择或撤销选择列表中的某项；该属性的默认值为 SelectionMode.One，则只能选择一项。

（5）SelectedIndex 属性：用来获取或设置 ListBox 控件中当前选定项的从零开始的索引。如果未选定任何项，则返回值为 1。对于只能选择一项的 ListBox 控件，可使用此属性确定 ListBox 中选定的项的索引。如果 ListBox 控件的 SelectionMode 属性设置为 SelectionMode.MultiSimple 或 SelectionMode.MultiExtended，并在该列表选定多个项，此时应用 SelectedIndices 来获取选定项的索引。

（6）SelectedIndices。该属性用来获取一个集合，该集合包含 ListBox 控件中所有选定项的从零开始的索引。

（7）SelectedItem 属性：获取或设置 ListBox 中的当前选定项。

（8）SelectedItems 属性：获取 ListBox 控件中选定项的集合，通常在 ListBox 控件的 SelectionMode 属性值设置为 SelectionMode.MultiSimple 或 SelectionMode.MultiExtended（它指示多重选择 ListBox）时使用。

（9）Sorted 属性：获取或设置一个值，该值指示 ListBox 控件中的列表项是否按字母顺序排序。如果列表项按字母排序，该属性值为 true；如果列表项不按字母排序，该属性值为 false。默认值为 false。在向已排序的 ListBox 控件中添加项时，这些项会移动到排序列表中适当的位置。

（10）Text 属性：该属性用来获取或搜索 ListBox 控件中当前选定项的文本。当把此属性值设置为字符串值时，ListBox 控件将在列表内搜索与指定文本匹配的项并选择该项。若在列表中选择了一项或多项，该属性将返回第一个选定项的文本。

（11）ItemsCount 属性：该属性用来返回列表项的数目。

2、常用方法：

（1）FindString 方法：用来查找列表项中以指定字符串开始的第一个项，有两种调用格式。

[格式 1]：ListBox 对象.FindString(s);

[功能]：在“ListBox 对象”指定的列表框中查找字符串 s，如果找到则返回该项从零开始的索引；如果找不到匹配项，则返回 ListBox.NoMatches。

[格式 2]：ListBox 对象.FindString(s,n);

[功能]：在 ListBox 对象指定的列表框中查找字符串 s，查找的起始项为 n+1，即 n 为开始查找的前一项的索引。如果找到则返回该项从零开始的索引；如果找不到匹配项，则返回 ListBox.NoMatches。

注意：FindString 方式只是词语部分匹配，即要查找的字符串在列表项的开头，便认为是匹配的，如果要精确匹配，即只有在列表项与查找字符串完全一致时才认为匹配，可使用 FindStringExact 方法，调用格式与功能与 FindString 基本一致。

（2）SetSelected 方法：用来选中某一项或取消对某一项的选择，调用格式及功能如下。

[格式]：ListBox 对象.SetSelected(n,l);

[功能]：如果参数 l 的值是 true，则在 ListBox 对象指定的列表框中选中索引为 n 的列表项，如果参数 l 的值是 false，则索引为 n 的列表项未被选中。

（3）Items.Add 方法：用来向列表框中增添一个列表项，调用格式及功能如下。

[格式]：ListBox 对象.Items.Add(s);

[功能]：把参数 s 添加到“listBox 对象”指定的列表框的列表项中。

(4) **Items.Insert** 方法：用来在列表框中指定位置插入一个列表项，调用格式及功能如下。

[格式]：ListBox 对象.Items.Insert(n,s);

[功能]：参数 n 代表要插入的项的位置索引，参数 s 代表要插入的项，其功能是把 s 插入到“listBox 对象”指定的列表框的索引为 n 的位置处。

(5) **Items.Remove** 方法：用来从列表框中删除一个列表项，调用格式及功能如下。

[格式]：ListBox 对象.Items.Remove(k);

[功能]：从 ListBox 对象指定的列表框中删除列表项 s。

(6) **Items.Clear** 方法：用来清除列表框中的所有项。其调用格式如下：ListBox 对象.Items.Clear();该方法无参数。

(7) **BeginUpdate** 方法和 **EndUpdate** 方法：这两个方法均无参数，调用格式分别如下：

ListBox 对象.BeginUpdate();


ListBox 对象.EndUpdate();

这两个方法的作用是保证使用 **Items.Add** 方法向列表框中添加列表项时，不重绘列表框。即在向列表框添加项之前，调用 **BeginUpdate** 方法，以防止每次向列表框中添加项时都重新绘制 ListBox 控件。完成向列表框中添加项的任务后，再调用 **EndUpdate** 方法使 ListBox 控件重新绘制。当向列表框中添加大量的列表项时，使用这种方法添加项可以防止在绘制 ListBox 时的闪烁现象。一个例子程序如下：

```
Public void AddToMyListBox()
{
    listBox1.BeginUpdate();
    for(intx=1;x<5000;x++)
    {
        listBox1.Items.Add("Item"+x.ToString());
    }
    listBox1.EndUpdate();
}
```

ListBox 控件常用事件有 Click 和 SelectedIndexChanged，SelectedIndexChanged 事件在列表框中改变选中项时发生。

11、ComboBox（组合框）控件

ComboBox 控件又称组合框，在工具箱中的图标为。默认情况下，组合框分两个部分显示：顶部是一个允许输入文本的文本框，下面的列表框则显示列表项。可以认为 ComboBox 就是文本框与列表框的组合，与文本框和列表框的功能基本一致。与列表框相比，组合框不能多选，它无 SelectionMode 属性。但组合框有一个名为 DropDownStyle 的属性，该属性用来设置或获取组合框的样式。

组合框的样式是左边的组合框能够通过文本框输入文本，中间的组合框则不能输入文本，只能选择列表项。

12、CheckedListBox（复选列表框） 控件

CheckedListBox 控件又称复选列表框，它扩展了 ListBox 控件，它几乎能完成列表框可以完成的所有任务，并且还可以在列表项旁边显示复选标记。两种控件间的其他差异在于，

复选列表框只支持 **DrawMode.Normal**，并且复选列表框只能有一项选定或没有任何选定。

需要注意的是：选定的项是指窗体上突出显示的项，已选中的项是指左边的复选框被选中的项。除具有列表框的全部属性外，它还具有以下属性。

(1) **CheckOnClick** 属性：获取或设置一个值，该值指示当某项被选定时是否应切换左侧的复选框。如果立即切换选中标记，则该属性值为 **true**；否则为 **false**。默认值为 **false**。

(2) **CheckedItems** 属性：该属性是复选列表框中选中项的集合，只代表处于 **CheckState.Checked** 或 **CheckState.Indeterminate** 状态的那些项。该集合中的索引按升序排列。

(3) **CheckedIndices** 属性：该属性代表选中项（处于选中状态或中间状态的那些项）索引的集合。

13、PictureBox（图片框）控件的使用

PictureBox 控件又称图片框，常用于图形设计和图像处理应用程序，在该控件中可以加载的图像文件格式有：位图文件（.Bmp）、图标文件（.ICO）、图元文件（.wmf）、.JPEG 和 .GIF 文件。下而仅介绍该控件的常用属性和事件。

1、常用属性：

(1) **Image** 属性：用来设置控件要显示的图像。把文件中的图像加载到图片框通常采用以下三种方式。

设计时单击 **Image** 属性，在其后将出现【..】按钮，单击该按钮将出现一个【打开】对话框，在该对话框中找到相应的图形文件后单击【确定】按钮。产生一个 **Bitmap** 类的实例并赋值给 **Image** 属性。

形式如下：

Bitmap=new **Bitmap**(图像文件名);

pictureBox 对象名.**Image**=**p**;

通过 **Image.FromFile** 方法直接从文件中加载。形式如下：

pictureBox 对象名.**Image**=**Image.FromFile**(图像文件名);

(2) **SizeMode** 属性：用来决定图像的显示模式。

14、Timer（定时器控件或计时器控件）控件

Timer 控件又称定时器控件或计时器控件，在工具箱中的图标是，该控件的主要作用是按一定的时间间隔周期性地触发一个名为 **Tick** 的事件，因此在该事件的代码中可以放置一些需要每隔一段时间重复执行的程序段。在程序运行时，定时器控件是不可见的。

1、常用属性：

(1) **Enabled** 属性：用来设置定时器是否正在运行。值为 **true** 时，定时器正在运行，值为 **false** 时，定时器不在运行。

(2) **Interval** 属性：用来设置定时器两次 **Tick** 事件发生的时间间隔，以毫秒为单位。如它的值设置为 500，则将每隔 0.5 秒发生一个 **Tick** 事件。

2、常用方法：

(1) **Start** 方法：用来启动定时器。调用的一般格式如下：

Timer 控件名.**start**();

该方法无参数。

(2) **Stop** 方法：用来停止定时器。调用的一般格式如下：

Timer 控件名.**stop**();


该方法无参数。

3、常用事件：

定义器控件响应的事件只有 Tick，每隔 Interval 时间后将触发一次该事件。

15、ProgressBar（进度条控件）控件

1、ProgressBar 控件

ProgressBar 控件又称进度条控件，它在工具栏中的图标为，该控件在水平栏中显示适当长度的矩形来指示进程的进度。当执行进程时，进度条用系统突出显示颜色在水平栏中从左向右进行填充。进程完成时，进度栏被填满。当某进程运行时间较长时，如果没有视觉提示，用户可能会认为应用程序不响应，通过在应用程序中使用进度条，就可以告诉用户应用程序正在执行冗长的任务且应用程序仍在响应。

ProgressBar 控件的常用属性如下。

(1) Maximum 属性：用来设置或返回进度条能够显示的最大值，默认值为 100。

(2) Minimum 属性：用来设置或返回进度条能够显示的最小值，默认值为 0。

(3) Value 属性：用来设置或返回进度条的当前位置。

(4) Step 属性：用来设置或返回一个值，该值用来决定每次调用 PerformStep 方法时，Value 属性增加的幅度。例如，如果要复制一组文件，则可将 Step 属性的值设置为 1，并将 Maximum 属性的值设置为要复制的文件总数。在复制每个文件时，可以调用 PerformStep 方法按 Step 属性的值增加进度栏。

ProgressBar 控件的常用方法如下。

(1) Increment 方法：用来按指定的数量增加进度条的值，调用的一般格式如下：

progressBar 对象.Increment(n);

其功能是把“progressBar 对象”指定的进度条对象的 Value 属性值增加 n，n 为整数。调用该方法之后，若 Value 属性大于 Maximum 属性的值，则 Value 属性值就是 Maximum 值，若 Value 属性小于 Minimum 属性值，则 Value 属性值就是 Minimum 值。

(2) PerformStep 方法：用来按 step 属性值来增加进度条的 Value 属性值，调用的一般格式如下：progressBar 对象.PerformStep();

该方法无参数。例如，下列程序段是一个显示复制多个文件的进度的进度条使用方法。

```
Private void CopyWithProgress(string[]filenames)
```

```
{
pBar1.Visible=true;
pBar1.Minimum=1;
pBar1.Maximum=filenames.Length;
pBar1.Value=1;
pBar1.Step=1;
for(intx=1;x<=filenames.Length;x++)
{
if(CopyFile(filenames[x-1])==true)
{
pBar1.PerformStep();
}
}
}
```

ProgressBar 控件能响应很多事件，但一般很少使用。

2. TrackBar 控件（滑块控件、跟踪条控件）

TrackBar 控件又称滑块控件、跟踪条控件，它在工具箱中的图标是“ ”。该控件主要用于在大量信息中进行浏览，或用于以可视形式调整数字设置。TrackBar 控件有两部分：缩略图（也称为滑块）和刻度线。缩略图是可以调整的部分，其位置与 Value 属性相对应。刻度线是按规则间隔分隔的可视化指示符。跟踪条控件可以按指定的增量移动，并且可以水平或者垂直排列。

TrackBar 控件的常用属性如下。

- （1）Maximum 属性：用来获取或设置 TrackBar 控件可表示的范围上限，即最大值。
 - （2）Minimum 属性：用来获取或设置 TrackBar 控件可表示的范围下限，即最小值。
 - （3）Orientation 属性：用来获取或设置一个值，该值指示跟踪条是在水平方向还是在垂直方向。
 - （4）LargeChange 属性：用来获取或设置一个值，该值指示当滑块长距离移动时应为 Value 属性中加上或减去的值。
 - （5）SmallChange 属性：用来获取或设置当滑块短距离移动时对 Value 属性进行增减的值。
 - （6）Value 属性：用来获取或设置滑块在跟踪条控件上的当前位置的值。
 - （7）TickFrequency 属性：用来获取或设置一个值，该值指定控件上绘制的刻度之间的增量。
 - （8）TickStyle 属性：用来获取或设置一个值，该值指示如何显示跟踪条上的刻度线。
- TrackBar 控件的常用事件是 ValueChanged，该事件在 TrackBar 控件的 Value 属性值改变时发生。

16、HScrollBar 控件和 VScrollBar 控件的使用

滚动条（ScrollBar）是大部分 Windows 应用程序都具有的控件，是 Windows 界面的一种常见元素，通常分为水平滚动条（HScrollBar）和垂直滚动条（VScrollBar）。HScrollBar 在工具箱中的图标是，VScrollBar 控件在工具箱中的图标是。这两个控件主要用于在应用程序或控件中水平或垂直滚动，以方便在较长的列表中或大量信息中转移。

1、常用属性：

- （1）Minimum 和 Maximum 属性：与 TrackBar 控件的同名属性基本相同。
- （2）Value 属性：用于设置或返回滑块在滚动条中所处的位置，其默认值为 0。当滑块的位置值为最小值时，滑块移到水平滚动条的最左端位置，或移到垂直滚动条的顶端位置。当滑块的位置值为最大值时，滑块移到水平滚动条的最右端位置或垂直滚动条的底端位置。
- （3）SmallChange 和 LargeChange 属性：这两个属性主要用于调整滑块移动的距离。其中 SmallChange 属性用于控制当鼠标单击滚动条两边的箭头时，滑块滚动的值，即 Value 属性增加或减小的值。而 LargeChange 属性则控制当用鼠标直接单击滚动条时滑块滚动的值。当用户按下 PageUp 键或 PageDown 键或者在滑块的任何一边单击滚动条轨迹时，Value 属性将按照 LargeChange 属性中设置的值进行增加或减小。

2、常用事件：

- （1）Scroll 事件：该事件在用户通过鼠标或键盘移动滑块后发生。
- （2）ValueChanged 事件：该事件在滚动条控件的 Value 属性值改变时发生。滚动条的

使用方法与 `TrackBar` 控件基本一致，此处不再赘述。

对话框类控件

17、`OpenFileDialog` 控件

`OpenFileDialog` 控件又称打开文件对话框，主要用来弹出 Windows 中标准的【打开文件】对话框。该控件在工具箱中的图标为。

`OpenFileDialog` 控件的常用属性如下。

(1) `Title` 属性：用来获取或设置对话框标题，默认值为空字符串（""）。如果标题为空字符串，则系统将使用默认标题：“打开”。

(2) `Filter` 属性：用来获取或设置当前文件名筛选器字符串，该字符串决定对话框的【另存为文件类型】或【文件类型】框中出现的选项内容。对于每个筛选选项，筛选器字符串都包含筛选器说明、垂直线条（|）和筛选器模式。不同筛选选项的字符串由垂直线条隔开，例如：“文本文件(*.txt)|*.txt|所有文件(*.*)|*.*”。还可以通过用分号来分隔各种文件类型，可以将多个筛选器模式添加到筛选器中，例如：“图像文件(*.BMP;*.JPG;*.GIF)|*.BMP;*.JPG;*.GIF|所有文件(*.*)|*.*”。

(3) `FilterIndex` 属性：用来获取或设置文件对话框中当前选定筛选器的索引。第一个筛选器的索引为 1，默认值为 1。

(4) `FileName` 属性：用来获取在打开文件对话框中选定的文件名的字符串。文件名既包含文件路径也包含扩展名。如果未选定文件，该属性将返回空字符串（""）。

(5) `InitialDirectory` 属性：用来获取或设置文件对话框显示的初始目录，默认值为空字符串（""）。

(6) `ShowReadOnly` 属性：用来获取或设置一个值，该值指示对话框是否包含只读复选框。如果对话框包含只读复选框，则属性值为 `true`，否则属性值为 `false`。默认值为 `false`。

(7) `ReadOnlyChecked` 属性：用来获取或设置一个值，该值指示是否选定只读复选框。如果选中了只读复选框，则属性值为 `true`，反之，属性值为 `false`。默认值为 `false`。

(8) `Multiselect` 属性：用来获取或设置一个值，该值指示对话框是否允许选择多个文件。如果对话框允许同时选定多个文件，则该属性值为 `true`，反之，属性值为 `false`。默认值为 `false`。

(9) `FileNames` 属性：用来获取对话框中所有选定文件的文件名。每个文件名都既包含文件路径又包含文件扩展名。如果未选定文件，该方法将返回空数组。

(10) `RestoreDirectory` 属性：用来获取或设置一个值，该值指示对话框在关闭前是否还原当前目录。假设用户在搜索文件的过程中更改了目录，且该属性值为 `true`，那么，对话框会将当前目录还原为初始值，若该属性值为 `false`，则不还原成初始值。默认值为 `false`。

`OpenFileDialog` 控件的常用方法有两个：`OpenFile` 和 `ShowDialog` 方法，本节只介绍 `ShowDialog` 方法，该方法的作用是显示通用对话框，其一般调用形式如下：

通用对话框对象名.`ShowDialog()`;

通用对话框运行时，如果单击对话框中的【确定】按钮，则返回值为 `DialogResult.OK`；否则返回值为 `DialogResult.Cancel`。其他对话框控件均具有 `ShowDialog` 方法，以后不再重复介绍。

18、`SaveFileDialog` 控件

SaveFileDialog 控件又称保存文件对话框，主要用来弹出 Windows 中标准的【保存文件】对话框。该控件在工具箱中的图标为。

SaveFileDialog 控件也具有 FileName、Filter、FilterIndex、InitialDirectory、Title 等属性，这些属性的作用与 OpenFileDialog 对话框控件基本一致，此处不再赘述。

需要注意的是：上述两个对话框只返回要打开或保存的文件名，并没有真正提供打开或保存文件的功能，程序员必须自己编写文件打开或保存程序，才能真正实现文件的打开和保存功能。

19、FontDialog 控件

FontDialog 控件又称字体对话框，主要用来弹出 Windows 中标准的【字体】对话框。该控件在工具箱中的图标为。字体对话框的作用是显示当前安装在系统中的字体列表，供用户进行选择。下面介绍字体对话框的主要属性。

- (1) Font 属性：该属性是字体对话框的最重要属性，通过它可以设定或获取字体信息。
- (2) Color 属性：用来设定或获取字符的颜色。
- (3) MaxSize 属性：用来获取或设置用户可选择的最大磅值。
- (4) MinSize 属性：用来获取或设置用户可选择的最小磅值。
- (5) ShowColor 属性：用来获取或设置一个值，该值指示对话框是否显示颜色选择框。如果对话框显示颜色选择框，属性值为 true，反之，属性值为 false。默认值为 false。
- (6) ShowEffects 属性：用来获取或设置一个值，该值指示对话框是否包含允许用户指定删除线、下划线和文本颜色选项的控件。如果对话框包含设置删除线、下划线和文本颜色选项的控件，属性值为 true，反之，属性值为 false。默认值为 true。

20、ColorDialog 控件

ColorDialog 控件又称颜色对话框，主要用来弹出 Windows 中标准的【颜色】对话框。该控件在工具箱中的图标为。颜色对话框的作用是供用户选择一种颜色，并用 Color 属性记录用户选择的颜色值。下面介绍颜色对话框的主要属性。

- (1) AllowFullOpen 属性：用来获取或设置一个值，该值指示用户是否可以使用该对话框定义自定义颜色。如果允许用户自定义颜色，属性值为 true，否则属性值为 false。默认值为 true。
- (2) FullOpen 属性：用来获取或设置一个值，该值指示用于创建自定义颜色的控件在对话框打开时是否可见。值为 true 时可见，值为 false 时不可见。
- (3) AnyColor 属性：用来获取或设置一个值，该值指示对话框是否显示基本颜色集中可用的所有颜色。值为 true 时，显示所有颜色，否则不显示所有颜色。
- (4) Color 属性：用来获取或设置用户选定的颜色。

21、PrintDialog 控件和 PrintDocument 控件

PrintDialog 控件在工具箱中的图标是，PrintDocument 控件在工具箱中的图标是。使用 PrintDialog 控件可以显示 Windows 标准的【打印】对话框，在该对话框中用户可以选择打印机、选择要打印的页及页码范围等。

需要注意的是：该对话框并不负责具体的打印任务，要想在应用程序中控制打印内容必须使用 PrintDocument 控件。关于这两个控件的详细使用方法读者可参阅相关资料或

VisualC#的帮助文件。

22、用户自定义对话框

除了可以使用 Windows 自带的标准对话框外，用户还可以把自己设计的窗体定义成对话框。使用自定义对话框有以下几个要点。

- (1) 将窗体的 **FormBorderStyle** 属性值设置为 **FixedDialog**。
- (2) 根据需要向窗体上添加控件。
- (3) 使用窗体的 **ShowDialog** 方法显示窗体，即显示出对话框。

23、菜单控件

Windows 的菜单系统是图形用户界面（GUI）的重要组成部分之一，在 VisualC#中使用 **MainMenu** 控件可以很方便地实现 Windows 的菜单，**MainMenu** 控件在工具箱中的图标为。

1. 菜单的结构

典型的菜单结构是顶层菜单项横着排列，单击某个菜单项后弹出的称为菜单或子菜单，它们均包含若干个菜单项，菜单项其实是 **MenuItem** 类的一个对象。菜单项有的是变灰显示的，表示该菜单项当前是被禁止使用的。有的菜单项的提示文字中有带下划线的字母，该字母称为热键（或访问键），若是顶层菜单，可通过按“ALT+热键”打开该菜单，若是某个子菜单中的一个选项，则在打开子菜单后直接按热键就会执行相应的菜单命令。有的菜单项后面有一个按键或组合键，称快捷键，在不打开菜单的情况下按快捷键，将执行相应的命令。

2. 菜单项的常用属性

(1) **Text** 属性：用来获取或设置一个值，通过该值指示菜单项标题。当使用 **Text** 属性为菜单项指定标题时，还可以在字符前加一个“&”号来指定热键（访问键，即加下划线的字母）。例如，若要将“File”中的“F”指定为访问键，应将菜单项的标题指定为“&File”。

(2) **Checked** 属性：用来获取或设置一个值，通过该值指示选中标记是否出现在菜单项文本的旁边。如果要放置选中标记在菜单项文本的旁边，属性值为 **true**，否则属性值为 **false**。默认值为 **false**。

(3) **DefaultItem** 属性：用来获取或设置一个值，通过该值指示菜单项是否为默认菜单项。值为 **true** 时，是默认菜单项，值为 **false** 时，不是默认菜单项。菜单的默认菜单项以粗体的形式显示。当用户双击包含默认项的子菜单后，默认项被选定，然后子菜单关闭。

(4) **Enabled** 属性：用来获取或设置一个值，通过该值指示菜单项是否可用。值为 **true** 时表示可用，值为 **false** 表示当前禁止使用。

(5) **RadioCheck** 属性：用来获取或设置一个值，通过该值指示选中的菜单项的左边是显示单选按钮还是选中标记。值为 **true** 时将显示单选按钮标记，值为 **false** 时显示选中标记。

(6) **Shortcut** 属性：用来获取或设置一个值，该值指示与菜单项相关联的快捷键。

(7) **ShowShortcut** 属性：用来获取或设置一个值，该值指示与菜单项关联的快捷键是否在菜单项标题的旁边显示。如果快捷组合键在菜单项标题的旁边显示，该属性值为 **true**，如果不显示快捷键，该属性值为 **false**。默认值为 **true**。

(8) **MdiList** 属性：用来获取或设置一个值，通过该值指示是否用在关联窗体内显示的多文档界面（MDI）子窗口列表来填充菜单项。若要在该菜单项中显示 MDI 子窗口列表，则设置该属性值为 **true**，否则设置该属性的值为 **false**。默认值为 **false**。

3. 菜单项的常用事件

菜单项的常用事件主要有 **Click** 事件，该事件在用户单击菜单项时发生。

24、多窗体程序设计

Windows 应用程序很少只由一个窗体组成，一般情况下一个应用程序均拥有很多个窗体。C#项目刚建立时只有一个名为 Form1 的窗体，要建立多窗体应用程序应首先为项目添加窗体，添加窗体的方法如下。

单击工具栏上的按钮或执行【项目】→【添加 Windows 窗体】命令，将会出现【添加新项】对话框。在该对话框的【模板】下面的列表框中选中【Windows 窗体】图标，在【名称】文本框中输入窗体名，然后单击【打开】按钮，即为应用程序添加了一个窗体。

25、MDI 应用程序设计

1、MDI 应用程序的概念

在前面的章节中，所创建的都是单文档界面（SDI）应用程序。这样的程序（如记事本和画图程序）仅支持一次打开一个窗口或文档。如果需要编辑多个文档，必须创建 SDI 应用程序的多个实例。而使用多文档界面（MDI）程序（如 Word 和 AdobePhotoshop）时，用户可以同时编辑多个文档。

MDI 程序中的应用程序窗口称为父窗口，应用程序内部的窗口称为子窗口。虽然 MDI 应用程序可以具有多个子窗口，但是每个子窗口却只能有一个父窗口。此外，处于活动状态的子窗口最大数目是 1。子窗口本身不能再成为父窗口，而且不能移动到它们的父窗口区域之外。除此以外，子窗口的行为与任何其他窗口一样（如可以关闭、最小化和调整大小等）。一个子窗口在功能上可能与父窗口的其他子窗口不同，例如，一个子窗口可能用于编辑图像，另一个子窗口可能用于编辑文本，第 3 个子窗口可以使用图形来显示数据，但是所有的窗口都属于相同的 MDI 父窗口。

2、与 MDI 应用程序设计有关的属性、方法和事件

常用的 MDI 父窗体属性如下：

（1）ActiveMdiChild 属性：该属性用来表示当前活动的 MDI 子窗口，如果当前没有子窗口，则返回 null。

（2）IsMdiContainer 属性：该属性用来获取或设置一个值，该值指示窗体是否为多文档界面（MDI）子窗体的容器，即 MDI 父窗体。值为 true 时，表示是父窗体，值为 false 时，表示不是父窗体。

（3）MdiChildren 属性：该属性以窗体数组形式返回 MDI 子窗体，每个数组元素对应一个 MDI 子窗体。

常用的 MDI 子窗体的属性有：

（1）IsMdiChild 属性：该属性用来获取一个值，该值指示该窗体是否为多文档界面（MDI）的子窗体。值为 true 时，表示是子窗体，值为 false 时，表示不是子窗体。

（2）MdiParent 属性：该属性用来指定该子窗体的 MDI 父窗体。与 MDI 应用程序设计有关的方法中，一般只使用父窗体的 LayoutMdi 方法，该方法的调用格式如下：

MDI 父窗体名.LayoutMdi(Value);

该方法用来在 MDI 父窗体中排列 MDI 子窗体，以便导航和操作 MDI 子窗体。参数 Value 决定排列方式，取值有：MdiLayout.ArrangeIcons（所有 MDI 子窗体以图标的形式排列在 MDI 父窗体的工作区内）、MdiLayout.TileHorizontal（所有 MDI 子窗口均水平平铺在 MDI 父窗体的工作区内）、MdiLayout.TileVertical（所有 MDI 子窗口均垂直平铺在 MDI 父窗体的工作区内）和 MdiLayout.Cascade（所有 MDI 子窗口均层叠在 MDI 父窗体的工作区内）。常用的 MDI 父窗体的事件是 MdiChildActivate，当激活或关闭一个 MDI 子窗体时将发

生该事件。

3、菜单合并

父窗体和子窗体可以使用不同的菜单，这些菜单会在选择子窗体的时候合并。如果需要指定菜单的合并方式，程序员可以设置每个菜单项的 `MergeOrder` 属性和 `MergeType` 属性。

(1) `MergeOrder` 属性：用来确定当两个菜单合并时菜单项出现的顺序，具有较低 `MergeOrder` 的菜单项会首先出现。

(2) `MergeType` 属性：当合并的两个菜单的某些菜单项的 `MergeOrder` 属性值相等时，使用该属性可以控制这些菜单项的显示方式。

26、键盘事件处理

键盘事件在用户按下键盘上的键时发生，可分为两类。第一类是 `KeyPress` 事件，当按下的键表示的是一个 ASCII 字符时就会触发这类事件，可通过它的 `KeyPressEventArgs` 类型参数的属性 `KeyChar` 来确定按下键的 ASCII 码。使用 `KeyPress` 事件无法判断是否按下了修改键（例如 `Shift`、`Alt` 和 `Ctrl` 键），为了判断这些动作，就要处理 `KeyUp` 或 `KeyDown` 事件，这些事件组成了第二类键盘事件。该类事件有一个 `KeyEventArgs` 类型的参数，通过该参数可以测试是否按下了一些修改键、功能键等特殊按键信息。

1. `KeyPressEventArgs` 类的主要属性（`KeyPress` 事件的一个参数类型）

(1) `Handled` 属性：用来获取或设置一个值，该值指示是否处理过 `KeyPress` 事件。

(2) `KeyChar` 属性：用来获取按下的键对应的字符，通常是该键的 ASCII 码。

2. `KeyEventArgs` 类的主要属性（`KeyUp` 和 `KeyDown` 事件的一个参数）

(1) `Alt` 属性：用来获取一个值，该值指示是否曾按下 `Alt` 键。

(2) `Control` 属性：用来获取一个值，该值指示是否曾按下 `Ctrl` 键。

(3) `Shift` 属性：用来获取一个值，该值指示是否曾按下 `Shift` 键。

(4) `Handled` 属性：用来获取或设置一个值，该值指示是否处理过此事件。

(5) `KeyCode` 属性：以 `Keys` 枚举型值返回键盘键的键码，该属性不包含修改键（`Alt`、`Control` 和 `Shift` 键）信息，用于测试指定的键盘键。

(6) `KeyData` 属性：以 `Keys` 枚举类型值返回键盘键的键码，并包含修改键信息，用于判断关于按下键盘键的所有信息。

(7) `KeyValue` 属性：以整数形式返回键码，而不是 `Keys` 枚举类型值。用于获得所按下键盘键的数字表示。

(8) `Modifiers` 属性：以 `Keys` 枚举类型值返回所有按下的修改键（`Alt`、`Control` 和 `Shift` 键），仅用于判断修改键信息。

27、鼠标事件处理

对鼠标操作的处理是应用程序的重要功能之一，在 `VisualC#` 中有一些与鼠标操作相关的事件，利用它们可以方便地进行与鼠标有关的编程。

(1) `MouseEnter` 事件：在鼠标指针进入控件时发生。

(2) `MouseMove` 事件：在鼠标指针移到控件上时发生。事件处理程序接收一个 `MouseEventArgs` 类型的参数，该参数包含与此事件相关的数据。该参数的主要属性及其含义如下。

`Button` 属性：用来获取曾按下的是哪个鼠标按钮。该属性是 `MouseButtons` 枚举型的值，取值及含义如下：`Left`（按下鼠标左按钮）、`Middle`（按下鼠标中按钮）、`Right`（鼠标右按钮）、

None（没有按下鼠标按钮）、XButton1（按下了第一个 XButton 按钮，仅用于 Microsoft 智能鼠标浏览器）和 XButton2（按下了第二个 XButton 按钮，仅用于 Microsoft 智能鼠标浏览器）

Clicks 属性：用来获取按下并释放鼠标按钮的次数。

Delta 属性：用来获取鼠标轮已转动的制动器数的有符号计数。制动器是鼠标轮的一个凹口。

X 属性：用来获取鼠标所在位置的 x 坐标。

Y 属性：用来获取鼠标所在位置的 y 坐标。

（3）MouseHover 事件：当鼠标指针悬停在控件上将发生该事件。

（4）MouseDown 事件：当鼠标指针位于控件上并按下鼠标键时将发生该事件。事件处理程序也接收一个 MouseEventArgs 类型的参数。

（5）MouseWheel 事件：在移动鼠标轮并且控件有焦点时将发生该事件。该事件的事件处理程序接收一个 MouseEventArgs 类型的参数

（6）MouseUp 事件：当鼠标指针在控件上并释放鼠标键时将发生该事件。事件处理程序也接收一个 MouseEventArgs 类型的参数。

（7）MouseLeave 事件：在鼠标指针离开控件时将发生该事件。