第二章 常用控件和类的使用

Visual Studio.Net(简称 VS.NET)使用控件(组件)设计 Windows 应用程序。将 VS.NET 工具箱窗口中的控件放到窗体中，使用属性窗口改变控件的属性，或在程序中用语句修改属性，为控件增加事件函数，完成指定的功能。

3.1 控件通用属性

大部分控件，例如 Label、Button、TextBox 等，都是 Control 类的派生类。Control 类中定义了这些派生类控件通用的一组属性和方法，这些属性是：

- Name：控件的名称，区别控件类不同对象的唯一标志，例如建立一个 Button 控件类对象，可用如下语句，Button button1=new Button()，那么 Name 属性的值为 button1。
- Location：表示控件对象在窗体中的位置。本属性是一个结构，结构中有两个变量，x 和 y，分别代表控件对象左上角顶点的 x 和 y 坐标，该坐标系以窗体左上角为原点，x 轴向左为正方向，y 轴向下为正方向，以像素为单位。修改 Location，可以移动控件的位置，例如：button1.Location=new Point(100,200)语句移动按钮 button1 到新位置。
- Left 和 Top：属性值等效于控件的 Location 属性的 X 和 Y。修改 Left 和 Top，可以移动控件的位置，例如：button1.Left=100 语句水平移动按钮 button1。
- Size：本属性是一个结构，结构中有两个变量，Width 和 Height 分别代表控件对象的宽和高，例如可用语句 button1.Size.Width=100 修改 Button 控件对象 button1 的宽。
- BackColor：控件背景颜色。
- Enabled：布尔变量，为 true 表示控件可以使用，为 false 表示不可用，控件变为灰色。
- Visible：布尔变量，为 true 控件正常显示，为 false 控件不可见。
- Modifier：定义控件的访问权限，可以是 private、public、protected 等。默认值为 private。
- Cursor：鼠标移到控件上方时，鼠标显示的形状。默认值为 Default，表示使用默认鼠标形状，即为箭头形状。

3.2 Form 类

Form 类是.Net 系统中定义的窗体类(WinForm)，它属于 System.Windows.Forms 名字空间。Form 类对象具有 Windows 应用程序窗口的最基本功能。它可以是对话框、单文档或多文档应用程序窗口的基类。Form 类对象还是一个容器，在 Form 窗体中可以放置其它控件，例如菜单控件，工具条控件等等，还可以放置子窗体。

1. Form 类常用属性

- AutoScroll：布尔变量，表示窗口是否在需要时自动添加滚动条。
- FormBorderStyle：窗体边界的风格，如有无边界、单线、3D、是否可调整等。
- Text：字符串类对象，窗体标题栏中显示的标题。
- AcceptButton：记录用户键入回车时，相当于单击窗体中的那个按钮对象。
- CanceButton：记录用户键入 ESC 键时，相当于单击窗体中的那个按钮对象。以上两个

属性多用于对话框，例如打开文件对话框，用户键入回车，相当于单击确定按钮。

- MaxiMizeBox：窗体标题栏右侧最大化按钮是否可用，设置为 false，按钮不可用。
- MiniMizeBox：窗体标题栏右侧最小化按钮是否可用，设置为 false，按钮不可用。如果属性 MaxiMizeBox 和 MiniMizeBox 都设置为 false，将只有关闭按钮。在不希望用户改变窗体大小时，例如对话框，将两者都设置为 false。

2. Form 类常用方法

- Close()：窗体关闭，释放所有资源。如窗体为主窗体，执行此方法，程序结束。
- Hide()：隐藏窗体，但不破坏窗体，也不释放资源，可用方法 Show()重新打开。
- Show()：显示窗体。

3. Form 类常用事件

- Load：在窗体显示之前发生，可以在其事件处理函数中做一些初始化的工作。

3.3 标签(Label)控件

标签控件用来显示一行文本信息，但文本信息不能编辑，常用来输出标题、显示处理结果和标记窗体上的对象。标签一般不用于触发事件。

1. Label 控件常用属性

- Text：显示的字符串
- AutoSize：控件大小是否随字符串大小自动调整，默认值为 false，不调整。
- ForeColor：Label 显示的字符串颜色。
- Font：字符串所使用的字体，包括所使用的字体名，字体的大小，字体的风格等等，具体修改方法见下边的例子。

2. 例子 e3_3：我的第一个程序

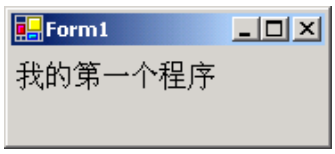
下面的例子在窗口中显示一行文本，该例虽然简单，但包括了用 Visual Studio.Net 建立 C# Windows 应用程序的基本步骤。具体实现步骤如下：

(1) 建立一个新项目，生成一个空白窗体(Form1)，见图 2.4.2A。可以用属性窗口(图 2.4.2B 中图)修改窗体的属性，例如修改 Form1 的属性 Text，可以修改窗体的标题。用鼠标拖动窗体的边界小正方形，可以修改窗体打开时的初始大小。

(2) 双击工具箱窗口(图 2.4.2B 左图)中 Windows 窗体类型下的 Label 条目，在窗体 Form1 放置一个 Label 控件。该控件用来显示一行文本。可以用鼠标拖放 Label 到窗体的任意位置，并可拖动 Label 边界改变控件的大小。

(3) 选中 Label 控件，在属性窗口中找到属性 text，把它的值由“Label1”修改为“我的第一个程序”。接着在属性窗口中选中 Font 属性，单击 Font 属性右侧的标题为…的按钮，打开对话框，在对话框中可以修改 Label 控件显示字符串的字体名称和字号等，也可以单击 Font 属性左边的+号，在出现的子属性中编辑。编辑完成后，单击 Font 属性左边的-号，隐藏 Font 的子属性。修改 ForeColor 属性可以修改 Label 控件显示字符串的颜色。这是在设计阶段修改属性。

(4) 编译，运行，可以看到窗口中按指定字体大小和颜色显示：我的第一个程序。运行效果如右图。



(5) 保存项目。生成一个可执行程序需要多个文件，这些文件组成一个项目。一般把一个项目存到一个子目录中。单击文件/存所有文件菜单项，保存所有文件。

(6) 关掉 VS.NET，再启动。用文件/打开项目菜单项打开刚才关闭的项目文件(扩展名为 sln)。应能看到刚才关闭的设计界面。必须打开项目，才能完成编译工作。

3.4 按钮(Button)控件

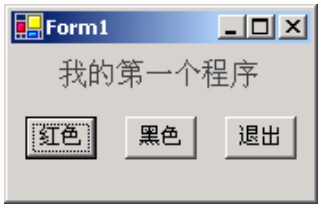
用户单击按钮，触发单击事件，在单击事件处理函数中完成相应的工作。

1. Button 控件的常用属性和事件

- 属性 Text：按钮表面的标题
- 事件 Click：用户单击触发的事件，一般称作单击事件。

2. 例子 e3_4

本例说明如何用程序修改属性，如何使用方法，增加事件函数。该例在窗口中显示一行文字，增加 2 个按钮，单击标题为红色的按钮把显示的文本颜色改为红色，单击标题为黑色的按钮把显示的文本颜色改为黑色。实现步骤如下：



(1)继续上例，放三个 Button 控件到窗体，修改属性 Text，使标题分别为红色，黑色，退出。设计好的界面如右图。

(2)选中标题为红色的按钮，打开事件窗口(见图 2.4.2B 右图)，显示该控件所能响应的所有事件，其中左侧为事件名称，右侧为事件处理函数名称，如果为空白，表示还没有事件处理函数，选中 Click 事件，双击右侧空白处，增加单击(Click)标题为红色的按钮的事件处理函数如下：

```
private void button1_Click(object sender, System.EventArgs e)
{   label1.ForeColor=Color.Red;//运行阶段修改属性
}//注意 label1 是控件的名字(label 的 Name 属性)，用它来区分不同的控件。
```

(3)单击(Click)标题为黑色的按钮的事件处理函数如下：

```
private void button2_Click(object sender, System.EventArgs e)
{   label1.ForeColor=Color.Black;}
```

(4)单击(Click)标题为退出的按钮的事件处理函数如下：

```
private void button3_Click(object sender, System.EventArgs e)
{   Close();}
```

Close()为窗体(Form)的方法，作用是关闭注窗体。由于关闭了主窗体，程序也就结束了。注意，引用窗体的方法和属性时可不用指定对象名，换句话讲，如不指定属性或方法的对象名，默认为窗体的属性或方法。而使用其它组件的属性及方法要指明所属组件对象，例如 label1.ForeColor=Color.Red;

(5)编译，运行，单击标题为红色的按钮，窗体显示字符串颜色变为红色，单击标题为黑色的按钮，窗体显示字符串颜色变为黑色，单击标题为退出的按钮，结束程序。

3.5 事件处理函数的参数

事件处理函数一般有两个参数，第一个参数(object sender)为产生该事件的对象的属性 Name 的值，例如上例单击标题为红色的按钮，第一个参数 sender 的值为 button1。如上例标题为红色的按钮和标题为黑色的按钮使用同一个单击事件处理函数，其事件处理如下：

```
private void button1_Click(object sender,System.EventArgs e)
{   if(sender==button1)
        label1.ForeColor=Color.Red;
    else
        label1.ForeColor=Color.Black;
```

}

事件处理函数第二个参数(System.EventArgs e)代表事件的一些附加信息，事件不同，所代表的信息也不相同，例如在后边的例子中可以看到，按下鼠标的事件处理函数中，e.X 和 e.Y 分别为发生事件时鼠标位置的 x 坐标和 y 坐标，e.Button 表示用户单击了鼠标那个键，如为 MouseButtons.Left，表示单击了鼠标左键。

为了使这两个按钮使用相同的单击事件处理函数，首先为标题为红色的按钮增加单击事件处理函数，即是上边的代码，事件函数名称为：button1_Click。选中标题为黑色的按钮，打开事件窗体(见图 2.4.2B 右图)，选中 Click 事件，从其右侧下拉列表中选择事件处理函数为 button1_Click，这样两个按钮就使用相同的单击事件处理函数了。

3.6 文本框(TextBox)控件

TextBox 控件是用户输入文本的区域，也叫文本框。

1. TextBox 控件属性和事件

- 属性 Text：用户在文本框中键入的字符串
- 属性 MaxLength：单行文本框最大输入字符数。
- 属性 ReadOnly：布尔变量，为 true，文本框不能编辑。
- 属性 PasswordChar：字符串类型，允许输入一个字符，如输入一个字符，用户在文本框中输入的所有字符都显示这个字符。一般用来输入密码。
- 属性 MultiLine：布尔变量，为 true，多行文本框，为 false，单行文本框。
- 属性 ScrollBars：MultiLine=true 时有效，有 4 种选择：=0，无滚动条，=1，有水平滚动条，=2，有垂直滚动条，=3，有水平和垂直滚动条。
- 属性 SelLength：可选中文本框中的部分或全部字符，本属性为所选择的文本的字符数。
- 属性 SelStart：所选中文本的开始位置。
- 属性 SelText：所选中的文本
- 属性 AcceptsReturn：MultiLine=true 时有效，布尔变量，为 true，键入回车，换行，为 false，键入回车键，相当于单击窗体中的默认按钮。
- 事件 TextChanged：文本框中的字符发生变化时，发出的事件。

2. 例子 e3_6

本例要求用户在编辑框中输入两个乘数，单击按钮把相乘的结果在编辑框中显示出来。

(1)建立一个新的项目。放四个 Label 控件到窗体，Text 属性分别为：被乘数，乘数，积，*,=。

(2)放三个 textBox 控件到窗体，属性 Name 从左到右分别为：textBox1、textBox2、textBox3，属性 Text 都为空。

(3)放三个 Button 控件到窗体，Text 属性分别修改为求积，清空，退出。设计的界面如上图。

(4)标题为求积的按钮的单击事件处理函数如下：

```
private void button1_Click(object sender, System.EventArgs e)
{   float ss,ee;
    ss=Convert.ToSingle(textBox1.Text);
    ee=Convert.ToSingle(textBox2.Text);
    textBox3.Text=Convert.ToString(ss*ee);
}
```

(5)标题为清空的按钮的单击事件处理函数如下：

```
private void button2_Click(object sender,System.EventArgs e)
{   textBox1.Text="";
    textBox2.Text="";
    textBox3.Text="";
}
```

(6)标题为退出的按钮的单击事件处理函数如下:

```
private void button3_Click(object sender, System.EventArgs e)
{   Close();}
```

(7) 编译，运行，在文本框 textBox1，textBox2 分别输入 2 和 3，单击标题为求积的按纽，textBox3 中显示 6，单击标题为清空的按钮，三个文本框被清空，单击标题为退出的按纽，结束程序。

3.7 Convert 类

Convert 类中提供了一些静态方法，用来把一种类型数据转换为另一种类型数据。例如，Convert.ToSingle(textBox1.Text) 把字符串 textBox1.Text 转换为单浮点数。Convert.ToString(3.14)把单浮点数 3.14 转换为字符串。其它转换函数还有：ToInt、ToInt16 等等。

3.8 单选按钮(RadioButton)和 GroupBox 控件

RadioButton 是单选按钮控件，多个 RadioButton 控件可以为一组，这一组内的 RadioButton 控件只能有一个被选中。GroupBox 控件是一个容器类控件，在其内部可放其它控件，表示其内部的所有控件为一组，其属性 Text 可用来表示此组控件的标题。例如把 RadioButton 控件放到 GroupBox 控件中，表示这些 RadioButton 控件是一组。有一些特性是互斥的，例如性别，选择这类特性可用 RadioButton 和 GroupBox 控件。

1. GroupBox 控件常用属性

GroupBox 控件常用属性只有一个，属性 Text，指定 GroupBox 控件顶部的标题。

2. RadioButton 控件属性和事件

- 属性 Text：单选按钮控件旁边的标题。
- 属性 Checked：布尔变量，为 true 表示按钮被选中，为 false 表示不被选中。
- 事件 CheckedChanged：单选按钮选中或不被选中状态改变时产生的事件。
- 事件 Click：单击单选按钮控件时产生的事件。

3. 例子 e3_8

该例用 RadioButton 控件修改 Label 控件字符串的字体为：宋体、黑体、楷体。具体实现步骤如下：

(1) 建立一个新的项目。

(2) 放 Label 控件到窗体,属性 Text=“不同的字体”。字体为宋体。

(3) 放 GroupBox 控件到窗体,其属性 Text=“选择字体”。

(4) 放三个 RadioButton 控件到 GroupBox 中,其属性 Text 分别为：宋体、黑体、楷体。宋体 RadioButton 控件的属性 Checked=true。设计好的界面如右图。

(5) 为三个 RadioButton 控件的 CheckedChanged 事件增加事件处理函数如下：

```
private void radioButton1_CheckedChanged(object sender, System.EventArgs e)
{   if(radioButton1.Checked)
    label1.Font=new Font("宋体",label1.Font.Size);
}//label1显示的字体变为宋体，字体大小不变
private void radioButton2_CheckedChanged(object sender, System.EventArgs e)
{   if(radioButton2.Checked)
    label1.Font=new Font("黑体",label1.Font.Size);
}
private void radioButton3_CheckedChanged(object sender, System.EventArgs e)
{   if(radioButton3.Checked)
    label1.Font=new Font("楷体_GB2312",label1.Font.Size);
}
```

(6) 编译，运行，单击 RadioGroup1 中的三个 RadioButton 按钮，可以改变字体。注意三个按钮只能选一个，既只能选一种字体。考虑一下，是否可用 Click 事件。

3.9 Font 类

Font 类有两个构造函数：第一个是 new Font(字体名称,字号)，例如，label1.Font=new Font("黑体",9),用法还可参考例 e3_8。第二个是 new Font(字体名称,字号，字体风格)，其中第三个参数是枚举类型，具体定义如下：

```
enum FontStyle{
    Regular     =0,//正常字体
    Bold        =1,//黑体
    Italic      =2,//斜体
    BoldItalic  =3,//黑斜体
    Underline   =4,//下划线，5=黑体下划线，6=斜体下划线，7=黑斜体下划线
    Strikeout   =8}//删除线，9=黑体删除线，10=斜体删除线，依此类推。
```

例如修改标签控件字体为斜体：

```
label1.Font=new Font("黑体",9,label1.Font.Style|FontStyle.Italic);
```

或者：`label1.Font=new Font("黑体",9,label1.Font.Style|(FontStyle)2);`

修改标签控件字体不为斜体：

```
label1.Font=new Font("黑体",9,label1.Font.Style&~FontStyle.Italic);
```

或者：`label1.Font=new Font("黑体",9,label1.Font.Style&(FontStyle)(~2));`

用法还可参考例 e3_11。

3.10 多选框(CheckBox)控件

CheckBox 是多选框控件，可将多个 CheckBox 控件放到 GroupBox 控件内形成一组，这一组内的 CheckBox 控件可以多选，不选或都选。可用来选择一些可共存的特性，例如一个人的爱好。

1. CheckBox 控件属性和事件

- 属性 Text：多选框控件旁边的标题。
- 属性 Checked：布尔变量，为 true 表示多选框被选中，为 false 不被选中。

- 事件 Click：单击多选框控件时产生的事件。
- 事件 CheckedChanged：多选框选中或不被选中状态改变时产生的事件。

2. 例子 e3_10A

在窗口中增加 2 个 CheckBox 控件，分别用来选择是否爱好音乐和是否爱好文学，用鼠标单击 CheckBox 控件，改变爱好选择，用 Label 控件显示所选择的爱好。实现步骤如下：

(1)建立新项目。放 Label 控件到窗体，属性 Text="你的爱好是："。

(2)放 GroupBox 控件到窗体，属性 Text="爱好"。放两个 CheckBox 控件到 GroupBox 中，属性 Text 分别为：音乐、文学。设计界面如下图。

(3)标题为音乐的多选框控件的 CheckedChanged 事件处理函数如下：

```
private void checkBox1_CheckedChanged(object sender, System.EventArgs e)
{   String text1="你的爱好是：";
    if(checkBox1.Checked)
        text1=text1+checkBox1.Text;
    if(checkBox2.Checked)
        text1+=checkBox2.Text;
    label1.Text=text1;
}
```

(4)将标题为文学的多选框控件的 CheckedChanged 事件处理函数，设置为标题为音乐的多选框控件的 CheckedChanged 事件处理函数，具体步骤见 3.5 节。

(5)编译，运行。选中音乐将在标签控件中显示：你的爱好是：音乐，再选中文学显示：你的爱好是：音乐文学，...。

3. 例子 e3_10B

该例同上例，但按选中音乐和文学的顺序在标签中显示爱好，实现步骤如下：

(1)建立一个新项目。为 Form1 类增加私有变量 String s="你的爱好是："。

(2)放 Label 控件、GroupBox 控件、两个 CheckBox 到窗体，属性设置同上例。

(4)标题为音乐的多选框控件 CheckBox1 的 CheckedChanged 事件处理函数如下：

```
private void checkBox1_CheckedChanged(object sender,System.EventArgs e)
{   int n=s.IndexOf("音乐");//s中有字符串"音乐"吗? n =-1表示没有
    if(n==-1)// n =-1，表示上次没选此项，此次选中，应增加"音乐"
        s+="音乐";
    else//否则，表示上次已选此项，此次不选中，应删除"音乐"
        s=s.Remove(n,2);
    label1.Text=s;
}
```

(5)标题为文学的多选框控件 CheckBox2 的 CheckedChanged 事件处理函数如下：

```
private void checkBox2_CheckedChanged(object sender,System.EventArgs e)
{   int n=s.IndexOf("文学");//s中有字符串"文学"吗?=-1表示没有
    if(n==-1)//=-1，表示上次没选此项，此次选中，应增加"文学"
        s+="文学";
    else//否则，表示上次已选此项，此次不选中，应删除"文学"
        s=s.Remove(n,2);
    label1.Text=s;
}
```

(6)编译，运行。选中音乐在标签中显示：你的爱好是：音乐，再选中文学显示：你的爱好

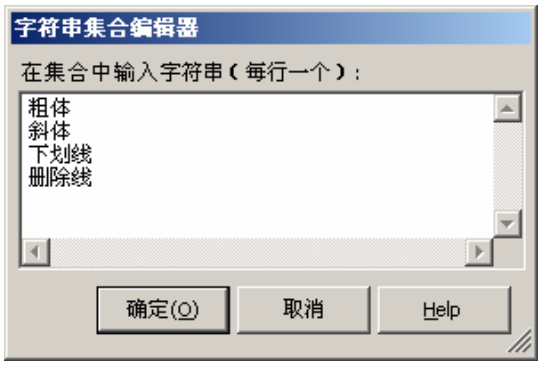
是：音乐文学，不选音乐显示：你的爱好是：文学，再选音乐显示：你的爱好是：文学音乐。

3.11 列表选择控件(ListBox)

列表选择控件列出所有供用户选择的选项，用户可从选项中选择一个或多个选项。

1. 列表选择控件的常用属性、事件和方法

- 属性 Items：存储 ListBox 中的列表内容，是 ArrayList 类对象，元素是字符串。
- 属性 SelectedIndex：所选择的条目的索引号，第一个条目索引号为 0。如允许多选，该属性返回任意一个选择的条目的索引号。如一个也没选，该值为-1。
- 属性 SelectedIndices：返回所有被选条目的索引号集合，是一个数组类对象。
- 属性 SelectedItem：返回所选择的条目的内容，即列表中选中的字符串。如允许多选，该属性返回选择的索引号最小的条目。如一个也没选，该值为空。
- 属性 SelectedItems：返回所有被选条目的内容，是一个字符串数组。
- 属性 SelectionMode：确定可选的条目数，以及选择多个条目的方法。属性值可以使：none(可以不选或选一个)、one(必须而且必选一个)、MultiSimple(多选)或 MultiExtended(用组合键多选)。
- 属性 Sorted：表示条目是否以字母顺序排序，默认值为 false，不允许。
- 方法 GetSelected()：参数是索引号，如该索引号被选中，返回值为 true。
- 事件 SelectedIndexChanged：当索引号(即选项)被改变时发生的事件。



2. 例子 e3_11

根据列表框的选择，为字符串加下划线、删除线、变斜体、变粗体。具体步骤如下：

(1) 建立一个新项目。放 Label 控件到窗体,其属性 Text="字体风格"。

(2) 放置 ListBox 控件到窗体中,属性 Name=listBox1。选中 ListBox 控件，在属性窗口中，单击 Items 属性右侧的三个小点，打开字符串集合编辑器对话框，在其中输入四项：粗体、斜体、下划线、删除线，注意每一项要换行。如上图。

(3) 设置列表选择控件 ListBox1 属性 SelectionMode 为 MultiExtended，允许多选。

(4) 为列表选择控件的事件 SelectedIndexChenged 增加事件处理函数如下：

```
private void listBox1_SelectedIndexChanged(object sender, System.EventArgs e)
{   int Style=0,k=1;//Style=0正常字体，1=黑体，2=斜体，3=黑斜体等，参见3.9节
    for(int i=0;i<listBox1.Items.Count;i++)//此例Count=4，为什么？
    {if(listBox1.GetSelected(i))//例如此例GetSelected(0)=true表示粗体被选中
        Style=Style|k;//增加指定风格
     else
        Style=Style&(~k);//取消指定风格
     k=k*2;
    }
    FontStyle m=new FontStyle();
    m=(FontStyle)Style;
    label1.Font=new Font(label1.Font.Name,9,m);}
```

(5) 编译，运行，单选或用 Ctrl 键多选，看一下效果。运行效果如上图。

3.12 下拉列表组合框(ComboBox)控件

控件 ComboBox 中有一个文本框，可以在文本框输入字符，其右侧有一个向下的箭头，单击此箭头可以打开一个列表框，可以从列表框选择希望输入的内容。现介绍该控件用法。

1. ComboBox 控件的常用属性、事件和方法

- 属性 DropDownStyle：确定下拉列表组合框类型。为 Simple 表示文本框可编辑，列表部分永远可见。为 DropDown 是默认值，表示文本框可编辑，必须单击箭头才能看到列表部分。为 DropDownList 表示文本框不可编辑，必须单击箭头才能看到列表部分。
- 属性 Items：存储 ComboBox 中的列表内容，是 ArrayList 类对象，元素是字符串。
- 属性 MaxDropDownItems：下拉列表能显示的最大条目数(1—100)，如果实际条目数大于此数，将出现滚动条。
- 属性 Sorted：表示下拉列表框中条目是否以字母顺序排序，默认值为 false，不允许。
- 属性 SelectedItem：所选择条目的内容，即下拉列表中选中的字符串。如一个也没选，该值为空。其实，属性 Text 也是所选择的条目的内容。
- 属性 SelectedIndex：编辑框所选列表条目的索引号，列表条目索引号从 0 开始。如果编辑框未从列表中选择条目，该值为-1。
- 事件 SelectedIndexChanged：被选索引号改变时发生的事件。

2. 例子 e3_12 选择 Windows 操作系统提供的所有字体

增加一个 ComboBox 控件，用来选择字符串使用的字体名。本例提供方法使控件 ComboBox 的下拉列表中显示 Windows 操作系统中使用的所有字体名。运行效果如右图。实现步骤如下：

(1) 建立新项目。放 Label 控件到窗体，其属性 Text="选择不同字体"。

(2) 放 ComboBox 控件到窗体中，属性 Name=comboBox1，属性 DropDownStyle=DropDownList，不能在编辑框中输入字体名，只能从下拉列表中选取。

(3) 为窗体 Form1 的事件 Load 增加事件处理函数如下：

```
private void Form1_Load(object sender, System.EventArgs e)
{//Families 是类 FontFamily 的一个静态属性，得到操作系统中所使用的所有字体名
  FontFamily[] families=FontFamily.Families;//静态属性没有类的对象也可使用
  foreach (FontFamily family in families)
        comboBox1.Items.Add(family.Name);//注意 Add 方法的使用
}
```

(4) 为 comboBox1 的事件 SelectedIndexChenged 增加事件处理函数如下：

```
private void comboBox1_SelectedIndexChanged(object sender, System.EventArgs e)
{label1.Font=new Font(comboBox1.Text,9);}
```

(5)编译，运行，在下拉列表中选择不同字体名，标签的字体变为选择的字体。从下拉列表中可以看到操作系统中的所有字体名称已经在列表中。

3.13 ToolTip 控件

在一些 Windows 应用程序中，例如 Word 程序，当鼠标在工具条的按钮上停留一段时间后，会在旁边出现提示，ToolTip 控件就是为实现此功能的。可以用 ToolTip 控件为任何控

件增加提示，本节介绍该控件的使用方法。

例子 e3_13 为 Button 控件增加提示

(1) 建立一个新项目。放 Button 控件到窗体，Name 属性为 Button1。

(2) 把 toolTip 控件放到窗体中，属性 Name=ToolTip1。

(3) 在 Form1 的构造函数中，增加语句如下：

```
toolTip1.SetToolTip(button1,"这是一个按钮");
```

(4) 编译，运行，当鼠标在 Button 上停留一段时间后，会在旁边出现提示：这是一个按钮。

3.14 超级链接(LinkLable)控件

控件 LinkLable 是控件 Label 的派生类，和控件 Label 不同的是显示的字符有下划线，可以为 LinkLable 控件的 LinkClicked 事件增加事件处理函数，当鼠标指向 LinkLable 控件，鼠标形状变为手形，单击该控件，调用这个事件处理函数，可以打开文件或网页。

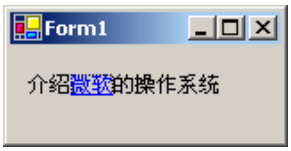
1. 超级链接控件的属性、方法和事件

- 属性 LinkColor：用户未访问过的链接的字符颜色，默认为蓝色。
- 属性 VisitedLinkColor：用户访问链接后的字符颜色。
- 属性 LinkVisited：如果已经访问过该链接，则为 true；否则为 false。
- 属性 LinkArea：是一个结构，变量 LinkArea.Start 表示字符串中开始加下划线的字符位置，LinkArea.Length 表示字符串中加下划线字符的个数。
- 事件 LinkClicked：单击控件 LinkLable 事件。

2. 例子 e3_14:用 LinkLabel 控件超级链接到微软网站。

(1) 建立一个新工程。放 LinkLabel 控件到窗体，属性 Text="介绍微软的操作系统"。

(2) 修改 LinkLabel 控件属性 LinkArea.Length=2，LinkArea.Start=2。也可在构造函数用语句修改：linkLabel1.LinkArea=new LinkArea(2,2);



(3) 为 LinkLabel 控件的事件 LinkClicked 增加事件处理函数：

```
private void linkLabel1_LinkClicked(object sender,
System.Windows.Forms.LinkLabelLinkClickedEventArgs e)
{   linkLabel1.LinkVisited=true;
    System.Diagnostics.Process.Start("http://www.micosoft.com.cn");
}
```

(4) 运行，效果如右图，注意只有字符微软带下划线。单击微软，打开浏览器访问微软主页。

(5) 如果要打开一个窗口，列出 C 盘根目录下的文件及文件夹，LinkLabel 控件事件 LinkClicked 事件处理函数修改如下：

```
linkLabel1.LinkVisited=true;
System.Diagnostics.Process.Start("C:/");
```

(6) 如果要打开指定程序，例如打开记事本程序，修改 LinkClicked 事件处理函数如下：

```
linkLabel1.LinkVisited=true;
System.Diagnostics.Process.Start("notepad");
```

3.15 定时(Timer)控件

定时控件(Timer)也叫定时器或计时器控件，是按一定时间间隔周期性地自动触发事件的控件。在程序运行时，定时控件是不可见的。

3. 定时控件的属性、方法和事件

- 属性 Interval：周期性地自动触发事件的时间间隔，单位为毫秒。
- 属性 Enabled：为 true,启动定时器。调用方法 Start()也可启动定时器。
- 方法 Start()和 Stop()：启动和停止定时器。设置属性 Enabled=false 也可停止定时器。
- 事件 Tick：每间隔属性 Interval 指定的时间，产生事件 Tick。

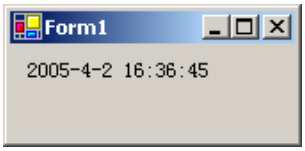
4. 例子 e3_15 用标签控件显示当前日期和时间

(1) 建立一个新项目。放 Timer 组件到窗体，Name 属性为 timer1。

(2) 放 Label 控件到窗体，Name 属性为 label1。

(3) 为窗体 Form1 的事件 Load 增加事件处理函数如下：

```
private void Form1_Load(object sender, System.EventArgs e)
{   this.timer1.Interval=100;
    this.timer1.Enabled=true;
    label1.Text=DateTime.Now.ToString();
}
```



(4) 为 Timer1 的 Tick 事件增加事件处理函数如下：

```
private void timer1_Tick(object sender, System.EventArgs e)
{   label1.Text=DateTime.Now.ToString();
}
```

(5) 编译，运行，标签控件位置显示日期和时间。运行效果如上图。

3.16 DateTime 类

DateTime 类中提供了一些静态方法，可以用来得到日期、星期和时间，下面是一些常用的方法。

- 得到日期和时间，并转换为字符串。

```
String s=DateTime.Now.ToString();//或 DateTime.Today.ToString()
```

- 得到年、月和日期

```
int y=DateTime.Now.Year;//得到年
int m=DateTime.Now.Month;//得到月
int d=DateTime.Now.Day;//得到日期
String s=DateTime.Now.DayOfWeek.ToString();//英文表示的星期
```

- 得到小时、分和秒

```
int h=DateTime.Now.Hour;//得到小时
int m=DateTime.Now.Minute;//得到分
int s=DateTime.Now.Second;//得到秒
```

- 定义一个 DateTime 类对象，表示 1999 年 1 月 13 日 3 时 57 分 32.11 秒

```
System.DateTime moment=new System.DateTime(1999,1,13,3,57,32,11);
```

- 加法和减法(减法请读者自己完成)

```
System.DateTime dTime=new System.DateTime(1980,8,5);//1980 年 8 月 5 日
//时间间隔，17 天 4 小时 2 分 1 秒
System.TimeSpan tSpan=new System.TimeSpan(17,4,2,1);
System.DateTime result=dTime+tSpan;//结果是：1980 年 8 月 22 日 4:2:1 AM.
```

3.17 菜单

Windows 应用程序一般都有一个菜单，通过选择菜单中的不同菜单项，完成指定的功能。使用主菜单控件 MainMenu 可以很容易建立 windows 应用程序的主菜单。

1. 菜单的组成及功能

放主菜单控件 MainMenu 到窗体中，可以为窗体增加一个主菜单。主菜单一般包括若干顶级菜单项，例如，文件、编辑、帮助等。单击顶级菜单项，可以出现弹出菜单，弹出菜单中包含若干菜单项，例如单击文件顶级菜单项，其弹出菜单一般包括打开文件、存文件、另存为等菜单项，用鼠标单击菜单项，可以执行菜单项命令。有的菜单项还包括子菜单。

所有菜单项都可以有快捷键，即菜单项中带有下划线的英文字符，当按住 ALT 键后，再按顶级菜单项的快捷键字符，可以打开该顶级菜单项的弹出菜单。弹出菜单出现后，按菜单项的快捷键字符，可以执行菜单项命令。增加快捷键的方法是在菜单项的标题中，在要设定快捷键英文字符的前边增加一个字符&，例如，菜单项的标题为：打开文件(&O)，菜单项的显示效果为：打开文件(O)。菜单项可以有加速键，一般在菜单项标题的后面显示，例如，菜单项打开文件的加速键一般是 Ctrl+O，不打开菜单，按住 Ctrl 键后，再按 O 键，也可以执行打开文件命令。设定加速键的方法是修改菜单项的 ShortCut 属性。

2. 用程序生成菜单

放主菜单控件 MainMenu 到窗体中，可以为该窗体增加一个主菜单，Visual Studio.Net 自动添加如下语句：

```
MainMenu mainMenu1=new MainMenu();
This.Menu=mainMenu1;//指定主窗口的主菜单是 mainMenu1。
```

可以建立多个 MainMenu 类对象，用第二条语句修改使主窗口使用不同的主菜单。有了主菜单对象，用如下语句为主菜单增加顶级菜单项：

```
MenuItem myFile=mainMenu1.MenuItem.Add(“文件(&F)”);//顶级菜单项：文件
```

有了顶级菜单项对象，用如下语句为顶级菜单项的弹出菜单增加菜单项：

```
myFile.MenuItem.Add(“打开(&O)”);//文件顶级菜单项的弹出菜单的菜单项：打开
```

实际上，这些都可以用 Visual Studio.Net 自动生成。

3. 菜单项的属性和事件

- 属性 Checked：布尔变量，=true，表示菜单项被选中，其后有标记：√。
- 属性 ShortCut：指定的加速键，可以从下拉列表中选择。
- 属性 ShowShortCut：布尔变量，true(默认值)，表示显示加速键，false，不显示。
- 属性 Text：菜单项标题。如为字符-，为分隔线。如指定字符前加&，例如：颜色(&c)，增加快捷键，即用 Alt+c 访问颜色菜单。
- 常用事件 Click：单击菜单项事件。

4. 例子 e3_17 增加菜单

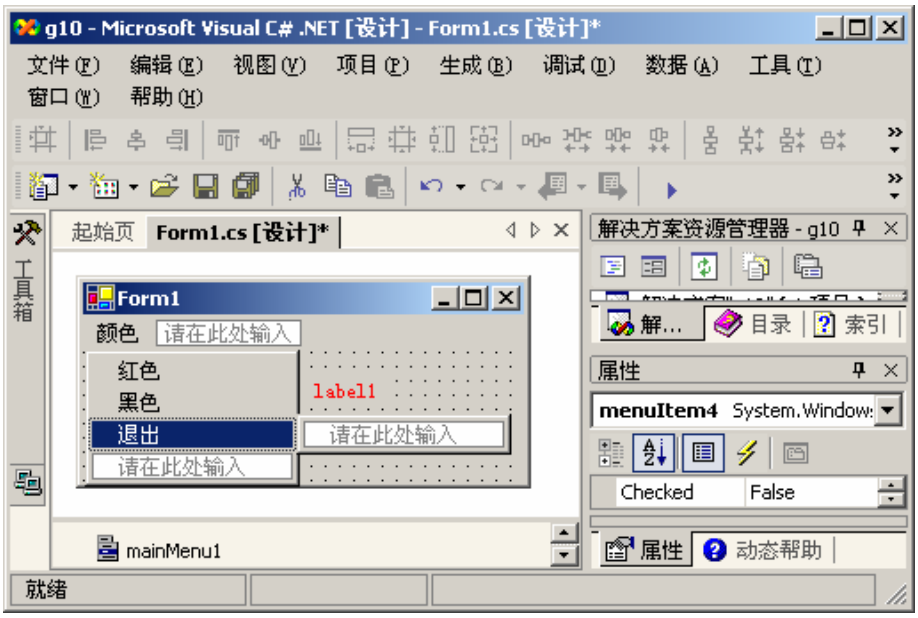
本例在窗体中建立主菜单，主菜单包括一个顶级菜单项：颜色，其弹出菜单包括两个菜单项：红色、黑色，单击标题为红色的菜单项，把窗体中显示的字符串变为红色，单击标题为黑色的菜单项，把窗体中显示的字符串变为黑色。实现步骤如下：

(1)建立一个新项目。放 Label 控件到窗体。

(2)双击工具箱中 Mainmenu 控件，在窗体中增加主菜单。右下角有一主菜单图标，在左上角有一方框，其中有文字：请在此处输入，在此方框中输入菜单标题。

(3)在方框内输入字符”颜色”,在其下部方框内输入字符”红色”为一菜单项，在”红色”下输入字符”黑色”为另一菜单项，再输入”退出”菜单项。如希望在选中某一菜单项后出现下一

级子菜单，可在菜单项右侧方框中输入子菜单项名。如果菜单项属性 Text 的值为-，则菜单项为分隔符。可以用鼠标拖动菜单项移动菜单项的位置。集成环境设计界面如下图。



(4)标题为红色的菜单项的单击(Click)事件处理函数如下：

```
private void menuItem2_Click(object sender,System.EventArgs e)
{label1.ForeColor=Color.Red;}//改变字体颜色为红色
```

(5)标题为黑色的菜单项的单击(Click)事件处理函数如下：

```
private void menuItem3_Click(object sender, System.EventArgs e)
{label1.ForeColor=Color.Black;}//改变字体颜色为黑色
```

(6)标题为退出的菜单项的单击(Click)事件处理函数如下：

```
private void menuItem4_Click(object sender, System.EventArgs e)
{    Close();}//退出程序
```

(7) 编译，运行，单击红色和黑色菜单项，能改变字符串的颜色。效果如上图。

3.18 工具条

一般 Windows 应用程序都有一个工具条，可以认为工具条上的按钮为菜单的某一菜单项的快捷按钮，单击工具条按钮相当于单击相应菜单项，完成同样的功能。

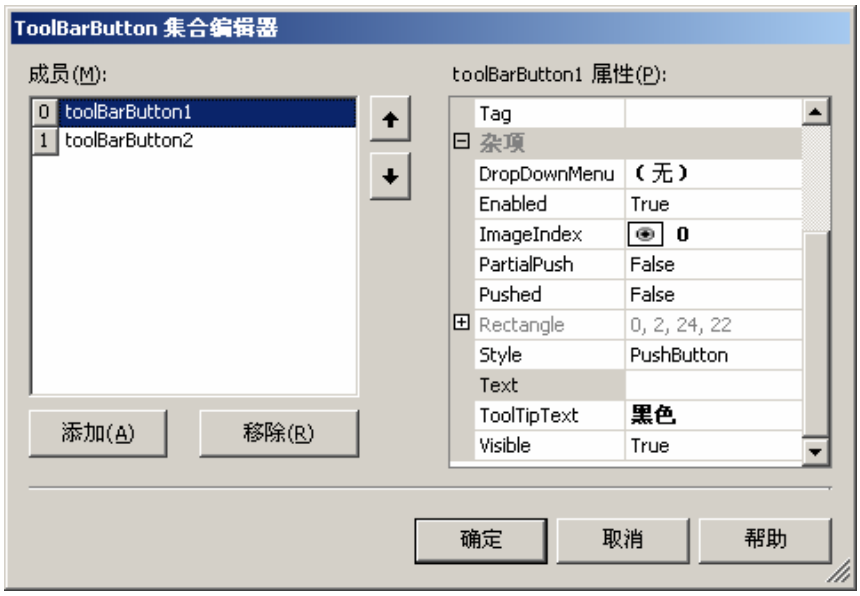
1. 工具条的组成及功能

放工具条控件 ToolBar 到窗体中，可以为该窗体增加一个工具条。在工具条中可以增加 Button 按钮和其它控件，例如象 Word 程序的工具条中用下拉列表控件(ComboBox)选择字号、字体等。一般工具条按钮上都有一个图标，提示用户该按钮的使用功能。按钮的所有图标存放到 ImageList 类对象中。单击任何一个按钮，都产生工具条控件的 ButtonClick 事件，在这个事件处理事件函数中，要用语句区分用户单击了那一个按钮，以完成相应的功能。

2. 控件 ToolBar 的属性、事件和方法

- 属性 BorderStyle：边界风格，=None(默认值)，无边界；=FixedSingle，单线边界；=Fixed3D，立体风格边界。
- 属性 Button：集合属性，存储 ToolBar 的按钮对象。单击其后的按钮，可以打开

ToolBarButton 集合编辑器对话框(见下图)，增加或删除按钮，修改按钮属性。



- 属性 ImageList：指定一个 ImageList 类对象，该对象中可以存储若干图标，这些图标作为 ToolBar 控件按钮的图标。
- 属性 Wrappable：布尔变量，=true(默认值)，当窗体 Form 水平尺寸小于工具条的水平尺寸时，一行不能显示所有按钮，允许下一行显示；=false，不允许。
- 事件 ButtonClick：ToolBar 控件的单击事件。在 ButtonClick 事件处理事件函数中，要用语句区分用户单击了那一个按钮，以完成相应的功能。
- 属性 ShowToolTips：布尔变量，=true，允许显示提示信息。
- 方法 IndexOF()：参数为 ToolBar 控件中按钮的属性 Name，返回其索引值。

3. ToolBar 控件中 ToolBarButton 按钮的属性

ToolBar 控件中 ToolBarButton 按钮可以看作独立的控件，它有自己独立的属性。下面介绍 ToolBar 控件中 ToolBarButton 按钮的属性。

- 属性 ImageIndex：ToolBar 控件属性 ImageList 指定一个 ImageList 类对象，该对象中的图标作为 ToolBar 控件按钮的图标。这个属性指定本按钮使用 ImageList 类对象中存储的第几个图标。
- 属性 Style：有 4 个值，=PushButton，为普通按钮；=Separator，为一分割符，再左边和右边的两个按钮中间增加一个间隙；=ToggleButton，开关按钮，单击该按钮，按钮被按下，不抬起，再单击，抬起。=DropDownButton，下拉按钮，按钮右侧有一个下拉箭头，单击下拉箭头，可以弹出下拉列表。
- 属性 Text：ToolBar 控件中按钮除了有图标外，还可以有属性 Text 指定的文字。
- 属性 ToolTipText：当鼠标在工具条按钮上停留一段时间后，将在工具条按钮旁边出现此属性指定的提示。

4. 例子 e3_18

现为上例的菜单增加工具条，有两个按钮，单击按钮分别使字体变红、变黑。步骤如下：

(1) 继续菜单的例子，放 ImageList 控件到窗体。

(2) 放 ToolBar 控件到窗体。修改属性 ImageList=ImageList1。

(3) 单击 ImageList 属性 Images 后按钮，打开 Image 集合编辑器，单击添加按钮，打开选择文件对话框。按指定路径选择图标的文件后，单击确定按钮，增加图标到 ImageList

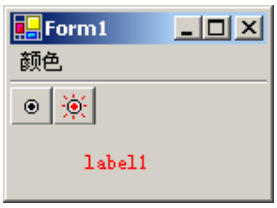
对象中。在 C:\Program Files\Microsoft Office\Office\forms\2052 文件夹和 C:\program files\Microsoft Visual Studio.Net\Common7\Graphics\Icon\Misc 文件夹中有若干图标。也可用画笔程序自己设计图标，图标的宽和高应比工具条按钮的宽和高略小，存为.ico 文件。也可以用抓图软件抓其它程序的图标。任选以上方法，为 ImageList 对象增加两个图标。

(4) 单击 ToolBar 控件属性 Buttons 后按钮，打开 ToolBarButton 集合编辑器(见上图)，单击添加按钮，增加一个按钮，从其属性 ImageIndex 后的下拉列表中选择按钮使用的图标，设置按钮的 ToolTipText 属性为：改变字体为红色，为工具按钮增加提示。同样方法增加第二个按钮，按钮的 ToolTipText 属性为：改变字体为黑色。

(5) 设定 ToolBar 控件属性 ShowToolTips 为 true。

(6) 为 ToolBar 控件的 ButtonClick 事件增加事件函数如下：

```
private void toolBar1_ButtonClick(object sender,
                    System.Windows.Forms.ToolBarButtonClickEventArgs e)
{   int n=toolBar1.Buttons.IndexOf(e.Button);//n为工具条中被单击按钮的序号
    switch(n)
    {   case 0://第一个按钮，调用相应的菜单项的事件处理函数。
            this.menuItem3_Click(sender,e);
            break;
        case 1://第二个按钮
            this.menuItem2_Click(sender,e);
            break;
    }
 }
```



(7) 编译，运行，单击两个工具条按钮，可以分别使字体变为红色或黑色。见上图。

3.19 状态栏(StatusBar)控件

Windows 应用程序的状态栏一般用来显示一些信息，如时间，鼠标位置等。

1. 状态栏控件的属性

- 属性 Panels：集合属性，存储状态栏中的各个分栏对象。单击其后标题为…的按钮，可以打开 StatusBarPanels 集合编辑器对话框，增加或删除分栏，修改分栏属性。
- 属性 ShowPanel：布尔变量，=true，允许显示多栏；=false，不允许。

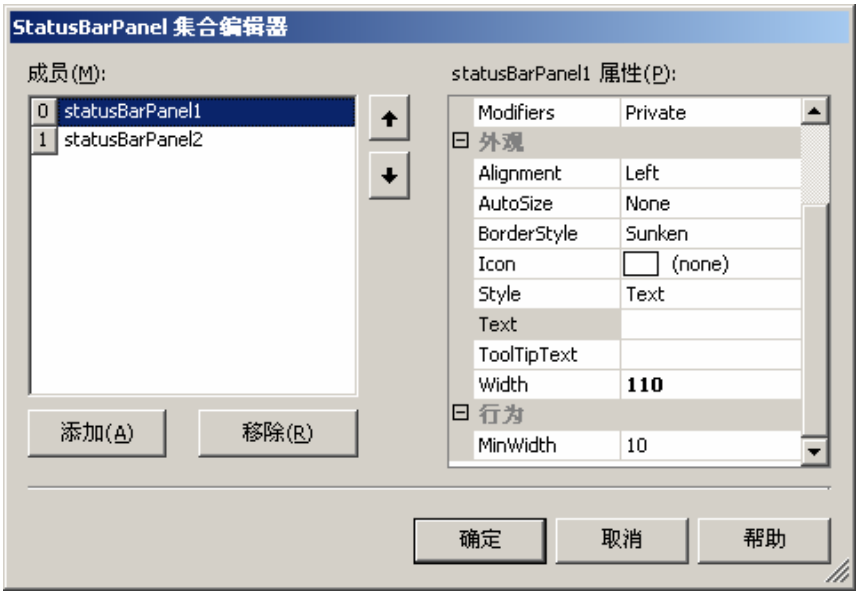
2. 状态栏(StatusBar)控件分栏的属性

状态条可以为单栏，也可以为多栏。属性 Text，表示在状态栏中显示的内容。如为单栏，在单栏中显示字符串的语句是：statusBar1.Text="在单栏中显示的文本"，如为多栏，在第 2 栏中显示字符串的语句是：statusBar1.Panels[1].Text="在第 2 栏中显示的文本"。

- 属性 Alignment：对齐方式，可以为左对齐、右对齐和中间对齐。
- 属性 Text：表示在状态栏中显示的内容。
- 属性 Width：栏的宽度。
- 属性 BorderStyle：指定状态栏控件上 每个分栏的边框外观。边界风格，=None(默认值)，不显示边框；=**Raised**，三维凸起边框；=**Sunken**，三维凹陷边框显示。

3. **例子 e3_19 为窗体增加状态条，在状态条内显示时间和鼠标位置。**

(1) 建立新项目。放 StatusBar 控件到窗体。单击 StatusBar 控件属性 Panels 后按钮，打开 StatusBarPanels 集合编辑器(如下图)，单击添加按钮，增加若 2 栏。其序号为 0、1。



(2) 修改 StatusBar 控件属性 ShowPanel=true。

(3) 放 Timer 组件到窗体，Name=Timer1，属性 Interval=1000，Enabled=true。

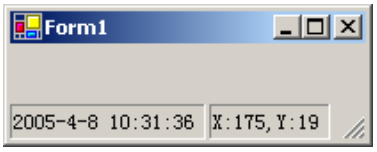
(4) 为 Timer1 的 Tick 事件增加事件处理函数如下：

```
private void timer1_Tick(object sender, System.EventArgs e)
{   statusBar1.Panels[0].Text=DateTime.Now.ToString();
}
```

(5) 为 Form1 的 MouseMove 事件增加事件处理函数如下：

```
private void Form1_MouseMove(object sender,System.Windows.Forms.MouseEventArgs e)
{statusBar1.Panels[1].Text="X:"+e.X.ToString()+",Y:"+e.Y.ToString();
}
```

(6) 编译，运行，如右图，在第 1 栏中可以看到当前时间，在窗口中移动鼠标，在第 2 栏中可以看到鼠标的位置不断变化。



3.20 鼠标事件

从类 System.Windows.Forms.Control 派生的控件都有鼠标事件，控件的 Click 事件本质上也是鼠标事件。一些控件还有单独的鼠标事件，例如 Form。鼠标事件有：

- MouseDown：如果鼠标位于控件区域，按下鼠标按键时产生该事件。
- MouseUp：如果鼠标位于控件区域，抬起鼠标按键时产生该事件。
- MouseMove：如果鼠标在控件区域移动，产生该事件。
- MouseEnter：鼠标进入控件区域，产生该事件。
- MouseLeave：鼠标离开控件区域，产生该事件。

鼠标事件处理函数一般有两个参数，第一个参数(object sender)是产生该事件的对象的属性 Name 的值，例如，为 Form1 的 MouseDown 事件增加事件函数，单击 Form1，第一个参数 sender 代表 Form1 对象。(System.Windows.Forms.MouseEventArgs e)是事件处理函数第二个参数，代表事件的一些信息，事件不同，所代表的信息也不相同，鼠标按下事件处理函数中，e.X 为发生事件时鼠标位置的 x 坐标，e.Y 为发生事件时鼠标位置的 y 坐标，e.Button

为 MouseButtons.Left，表示单击了鼠标左键等等，Right 和 Middle 则分别代表右键和中间键。e.Clicks 为鼠标单击的次数，如果大于 2 次，则为双击。

例子 e3_20：在窗体中的指定区域，双击鼠标左键，用 Label 控件显示双击鼠标的位置。指定区域的左上角坐标为(20,20)，宽为 200，高为 200。

(1) 建立一个新项目。放 Label 控件到窗体。属性 Name=label1。

(2) Panel 控件可以将窗体分为多个区域。放 Panel 控件到窗体，属性 Location.X=20，Location.Y=20，属性 Width=200，Height=200，属性 Name=p1。

(3) 为 Panel 的 MouseDown 事件增加事件函数如下：

```
private void p1_MouseDown(object sender,System.Windows.Forms.MouseEventArgs e)
{   if(e.Button==MouseButtons.Left&&e.Clicks>1)//如果是双击左键
        label1.Text="X:"+e.X.ToString()+",Y:"+e.Y.ToString();
}
```

(4) 编译，运行，分别在指定区域和区域外双击鼠标左键，看一下效果。分别在指定区域和区域外双击鼠标右键，看一下效果。

3.21 快捷菜单(ContextMenu)

使用过 Word 程序的人都知道，在其程序窗口的不同位置单击右键，会出现不同弹出菜单，这个弹出菜单叫快捷菜单，这节介绍如何在应用程序中增加快捷菜单。快捷菜单和主菜单的属性、事件和方法基本一致，只是快捷菜单没有顶级菜单项，因此这里就不多介绍了。

例子 e3.21

例子在窗口中显示一行字符串，加入两个按纽，单击按纽 button1 把字符串变为红色，单击按纽 button2 把字符串变为黑色。为两个按钮建立快捷菜单，快捷菜单中有 2 个菜单项，单击菜单项把字符串变为红色或黑色。为窗体建立快捷菜单，菜单中仅有 1 个退出菜单项，单击退出菜单项，退出程序。具体实现步骤如下：

(1) 建立一个新项目。放 Label 控件到窗体。

(2) 放 2 个 Button 控件到窗体，标题（属性 Text）分别为红色，黑色。

(3) 标题为红色的按钮的单击事件处理函数如下：

```
private void button1_Click(object sender, System.EventArgs e)
{   label1.ForeColor=Color.Red;}
```

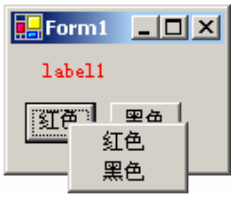
(4) 标题为黑色的按钮的单击事件处理函数如下：

```
private void button2_Click(object sender, System.EventArgs e)
{   label1.ForeColor=Color.Black;}
```

(5) 放 2 个 ContextMenu 控件到窗体，属性 Name 分别为 contextMenu1，contextMenu2。

(6) 选中 contextMenu1 控件，在菜单编辑器中增加两个标题分别为红色和黑色的菜单项，它们的单击事件处理函数分别是单击红色按钮和单击黑色按钮的事件处理函数。

(7) 选中 contextMenu2 控件，在菜单编辑器中增加标题为退出的菜单项，并为其增加单击事件处理函数，为事件处理函数增加语句：Close();



(8) 将红色按钮和黑色按钮的属性 ContextMenu 指定为 contextMenu1。Form 的属性 ContextMenu 指定为 contextMenu2。

(9) 编译，运行，右击标题为红色的按钮，快捷菜单 contextMenu1 打开，单击快捷菜单中标题为红色的菜单项，将使窗体显示的字符串颜色变为红色，右击标题为黑色的按钮，快捷菜单 contextMenu1 打开，单击快捷菜单中标题为黑色的菜

单项，将使窗体显示的字符串颜色变为黑色，右击窗体，快捷菜单 contextMenu2 打开，单击快捷菜单中标题为退出的菜单项，将退出应用程序。运行效果如上图。

3.22 综合例子：计算器

具体步骤如下：

(1) 建立一个新项目。Form 属性 MaxiMizeBox=false，属性 MiniMizeBox=false。属性 FormBorderStyle=FixedDialog，窗口不能修改大小。

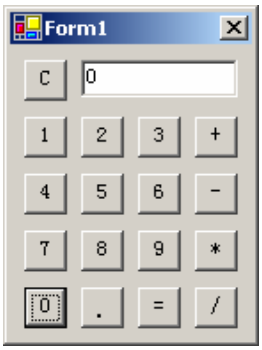
(2) 放 textBox 控件到窗体，属性 Name=textBox1，属性 Text="0"，属性 ReadOnly=true。

(3) 增加 10 个 Button 控件，前 9 个按钮属性 Name 分别为：Button1-Button9，最后一个为 Button0，属性 Text 分别为：1、2、3、4、5、6、7、8、9、0。

(4) 增加 7 个 Button 控件，属性 Name 分别为：btn_dot、btn_equ、btn_add、btn_sub、btn_mul、btn_div、btn_C，属性 Text 分别为：.、=、+、-、*、/、C。设计界面如下图。

(5) 控件 Button0 单击事件处理函数如下：

```
private void button0_Click(object sender, System.EventArgs e)
{   if(sender==button0) append_num(0);
    if(sender==button1) append_num(1);
    if(sender==button2) append_num(2);
    if(sender==button3) append_num(3);
    if(sender==button4) append_num(4);
    if(sender==button5) append_num(5);
    if(sender==button6) append_num(6);
    if(sender==button7) append_num(7);
    if(sender==button8) append_num(8);
    if(sender==button9) append_num(9);
}
```



(6) 为 Form1 类增加方法如下：

```
public void append_num(int i)
{   if(textBox1.Text!="0")
        textBox1.Text+=Convert.ToString(i);
    else
        textBox1.Text=Convert.ToString(i);
}
```

(7) 将 Button1-Button9 的单击事件处理函数设定为 Button0 单击事件处理函数。

(8) 为标题为.按钮增加事件处理函数如下：

```
private void btn_dot_Click(object sender, System.EventArgs e)
{   int n=textBox1.Text.IndexOf(".");
    if(n==-1)//如果没有小数点，增加小数点，否则不增加
        textBox1.Text=textBox1.Text+".";
}
```

(9) 编译，单击数字按钮，在 textBox1 可以看到输入的数字，也可以输入小数。

(10) 先实现加法，必须定义一个浮点类型变量 sum，初始值为 0，记录部分和。

(11) 输入了第一个加数，然后输入任一运算符(+、-、*、\或=)，应首先清除编辑框中显示的第一个加数，才能输入第二个加数。为实现此功能，必须定义一个布尔变量 blnClear，

初始值为 false，表示输入数字或小数点前不清除编辑框中显示，输入运算符(+、-、*、\或=)后，blnClear=true，表示再输入数字或小数点先清除编辑框中显示。修改前边程序，输入数字或小数点前，要判断变量 blnClear，如为 true，清除编辑框中显示的内容后，再显示新输入的数字或小数点，同时修改 blnClear=false。为此修改 append_num 方法如下：

```
public void append_num(int i)
{   if(blnClear)//如果准备输入下一个加数，应先清除textBox1显示内容
    {   textBox1.Text="0";//阴影部分为新增语句
        blnClear=false;
    }
    if(textBox1.Text!="0")
        textBox1.Text+=Convert.ToString(i);
    else
        textBox1.Text=Convert.ToString(i);
}
```

(12) 修改 btn_dot_Click 方法如下：

```
private void btn_dot_Click(object sender, System.EventArgs e)
{   if(blnClear) //如果准备输入下一个数，应先清除textBox1显示内容
    {   textBox1.Text="0";//阴影部分为新增语句
        blnClear=false;
    }
    int n=textBox1.Text.IndexOf(".");
    if(n==-1)//如果没有小数点，增加小数点，防止多次输入小数点
        textBox1.Text=textBox1.Text+".";
}
```

(13) 如果计算 1+2-3 的运算结果，先单击按钮 1，编辑框中显示 1，再单击按钮+,执行运算 sum=sum+1(注意此时 sum=0)，显示 sum 到编辑框中(实际显示不变)，记住此次输入的运算符，这里为+号。单击按钮 2，编辑框中显示 2，再单击按钮-,按记录的运算符(这里是+)计算 sum=sum+2，显示 sum 到编辑框中，记住此次输入的运算符，这里为-号，依此类推。为实现此功能，必须定义一个字符串变量 strOper，记录输入的运算符，初始值为"+"，保证输入第一个运算符后，执行运算 sum=sum+第一个加数，由于初始 sum=0，也就是 sum=第一个加数。标题为+的按钮的单击事件处理函数如下：

```
private void btn_add_Click(object sender, System.EventArgs e)
{   double dbSecond=Convert.ToDouble(textBox1.Text);
    if(!blnClear)//如果未输入第二个操作数，不运算
        switch(strOper)//按记录的运算符号运算
        {   case "+":
            sum+=dbSecond;
                break;
        //在此增加其它运算符-、*、\代码
        }
    if(sender==btn_add)
        strOper="+";
    //在此增加运算符-、*、\、=代码
```

```
    textBox1.Text=Convert.ToString(sum);
    blnClear=true;
}
```

(14) =号处理语句和+号处理基本一致，修改标题为+按钮的事件函数如下：

```
private void btn_add_Click(object sender, System.EventArgs e)
{   double dbSecond=Convert.ToDouble(textBox1.Text);
    if(!blnClear)//如果未输入第二个操作数，不运算
        switch(strOper)//按记录的运算符号运算
        {   case "+":
            sum+=dbSecond;
                break;
        //在此增加运算符-、*、\代码
        }
    if(sender==btn_add)
        strOper="+";
    if(sender==btn_equ)//为=号处理增加的语句
        strOper="=";
    textBox1.Text=Convert.ToString(sum);
    blnClear=true;
}
```

将 btn_equ 按钮的单击事件函数设定为+按钮的单击事件函数。

(15) 为标题为 C 按钮增加事件函数如下：

```
private void btn_C_Click(object sender, System.EventArgs e)
{   textBox1.Text="0";
    sum=0;
    blnClear=false;
    strOper="+";
}
```

(16) 请读者自己补上减法，乘法，除法运算的语句。

习题：

(1) 在窗口中显示一行字符串，加入两个按纽，单击按纽 1 把字符串改为红色，单击按纽 2 把字符串改为黑色。使字符串为红色时红色按纽不能使用，字符串为黑色时黑色按纽不能使用。(提示：可以修改按钮的属性 Enabled 为 false 使其不能使用。)

(2) 将上题改为用按扭修改字体的大小，分别为大字体和小字体。(参见 3.9 节)

(3) 加一文本框控件和一按纽，单击按纽将文本框控件输入内容显示标签控件上。(提示：单击按钮事件处理函数中加语句 label1.Text=textBox1.Text)。

(4) 修改上题，使文本框控件和标签控件文本同步显示(提示：文本框控件的 TextChanged 事件处理函数中加语句 label1.Text=textBox1.Text)。

(5) 加一文本框控件和一按纽，单击按纽将文本框控件输入的文本中选中的内容显示在标签控件上（提示：单击按钮事件处理函数中加语句 label1.Text=textBox1.SelText。）

(6) 加一文本框控件和一按纽，单击按纽将文本框控件输入的文本的字符、选中的内容的字符数和选中的内容的开始位置显示在标签控件上。

(7) 用控件 RadioButton 选择性别，把选择的结果用 Label 控件显示出来。

(8) 例子 e3_8 中如改为响应单击事件 Click，可能出现什么问题？

(9) 用控件 ComboBox 修改标签控件字体的大小。(用属性 Item 在下拉列表中输入大小)。

(10) 放 ListBox 控件到窗体中,属性 Name=listBox1。列表框有三项分别为：苹果，梨子，香蕉。允许多选。标签控件同步显示 ListBox 控件所做的选择。提示：为 ListBox 控件的 SelectedIndexChenged 事件增加事件函数，

```
label1.Text="所选择的是：";
for(int i=0;i<listBox1.SelectedIndices.Count;i++)
    label1.Text+=listBox1.SelectedItems[i].ToString()+", ";
```

(11) 放 ListBox、TextBox 和 3 个 Button 控件到窗体中,属性 Name 分别为 listBox1、textBox1、Button1、Button2、Button3。Button 控件属性 Text 分别为：增加、删除、清空。单击增加按钮，把 textBox 中输入的内容作为一个条目增加到 listBox1 中，单击删除按钮，删除 listBox1 中所选择的条目，单击清空按钮，清除 listBox1 所有条目。提示：增加用语句：listBox1.Items.Add(textBooooox1.Text)。删除所选择的条目用语句：listBox1.Items.RemoveAt(listBox1.SelectedIndex)。清除 listBox1 所有条目用语句：listBox1.Items.Clear()。

(12) 在窗体中显示字符，每隔 1 秒字体变大些，变到一定尺寸后，每隔 1 秒字体变小些，如此循环。增加一个按钮，可以启动和停止字符串字体大小变化，按钮标题给出正确提示。

(13) 在窗体中显示字符，每隔 1 秒字符移动一定距离，先右移，移到右边界，再左移，移到左边界，又一次右移，如此循环。(提示：修改 Label 的 Left 属性值。)

(14) 修改例子 e3_17，使显示字符串为红色时，标题为红色的菜单项无效；使显示字符串黑色时，标题为黑色的菜单项无效。

(15) 修改例子 e3_17，使显示字符串为红色时，标题为红色的菜单项前增加选中标志；使显示字符串黑色时，标题为黑色的菜单项前增加选中标志。

(16) 为例 e3_17 的菜单项增加加速键，键入 Alt+c 打开顶级菜单项颜色的弹出菜单，弹出菜单打开后,键入 B 执行标题为黑色的菜单项命令,键入 R 执行标题为红色的菜单项命令。

(17) 为例子 e3_17 的菜单项定义加速键（属性 Shortcut），键入 ctrl+r 使显示字符串为红色，键入 ctrl+b 使显示字符串黑色。

(18) 为例子 e3_17 顶级菜单项颜色增加单击事件处理函数,在事件处理函数中判断显示的字符串的颜色，决定是否为相应的菜单项增加选中标志。

(19) 拖动鼠标左键时，在状态栏中显示鼠标的位置。

(20) 模拟画笔程序，在左侧增加工具按钮，在下部增加颜色按钮。

(21) 在工具栏中加三个按钮，单击按钮时，按钮保持按下状态，再单击按钮，按钮抬起。在按下状态，使标签控件中字符串加下画线、斜体或加粗，抬起则取消。(提示：工具栏中按钮的属性 Style 设置为 ToggleButton。属性 Pushed 是一个布尔变量，表示工具栏按钮当前是否处于按下状态)

(22) 用工具栏中按钮的下拉菜单实现使标签控件字符的颜色变为红色、黑色。(提示：如工具栏中按钮的属性 Style 设置为 DropDownButton，按钮可有一个下拉菜单。首先创建一个 ContextMenu 菜单，指定工具栏中按钮的属性 DropDownMenu 的值为创建的 ContextMenu 菜单对象，将在按下按钮时显示这个菜单。)

(23) 完成计算器的减法和乘除程序。增加求平方，对数等功能。(例如 Math.Sqrt())

带格式的：项目符号和编号

删除的内容:

工具栏中按钮的属性 Style 设置为 ToolBarButtonStyle.DropDownButton，按钮可有一个下拉菜单。首先创建一个 ContextMenu 菜单，指定工具栏中按钮的属性 DropDownMenu 的值为创建的 ContextMenu 菜单对象，将在按下按钮时显示这个菜单。请用工具栏中按钮的下拉菜单实现使标签控件字符的颜色变为红色、黑色。