第3章

常用组件

Delphi XE8 的组件板上有很多组件，总共包括五百多个组件，如图 3-1 所示。本章将对 Delphi XE8 中常见的组件做详细的介绍。

图 3-1　组件板

3.1　窗　　体

用户界面是应用系统中直接面对用户的窗体，包括主窗体、子窗体、弹出对话框窗体等，它是对空白窗体加工后的系统运行结果，在任何工程中，都是由窗体的不断开发和累加完成的集合。

3.1.1　Form 组件

一个应用系统至少有一个窗体，一个大型软件工程甚至包含几个、几十个到成百上千个窗体或程序模块，如图 3-2 所示。

Windows 操作系统中，人机交互的界面主要是通过一些窗口和对话框实现的。Delphi XE8 的可视化设计工作就是在窗体中进行的。通常，窗体中会有一些组件，通过这些组件可以实现多种多样的功能。在 Delphi XE8 中，把这些运行期间出现在窗口和对话框中的组件称为可视组件。在窗体中，不仅可以放置组件，还可以放置一些运行期间不可视的组件，这些不可视的组件集中地实现了一些特殊的功能。

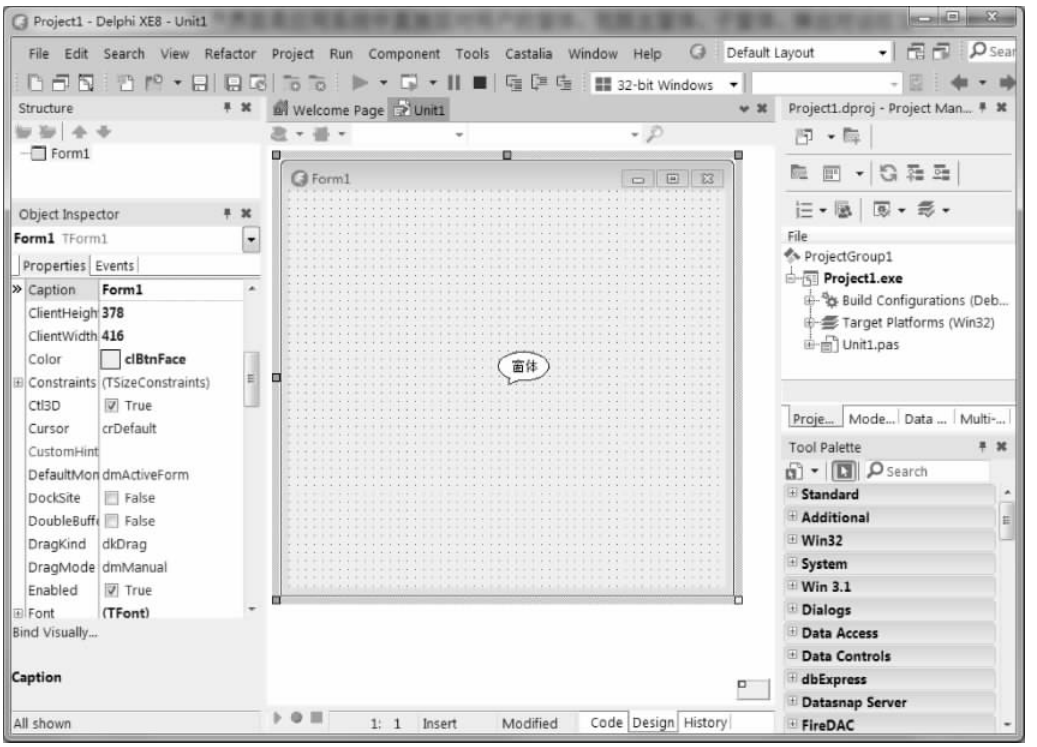

图 3-2　窗体

窗体是应用程序的操作界面,是放置组件的基础。在设计时,窗体就像一个桌面,可以放置各种组件;在运行时,窗体就像一个窗口,是程序与用户交流的地方。可以说,没有窗体,应用程序的框架就很难建立起来。窗体由标题栏、工作区和边界组成,通常标题栏中还有控制菜单、“最大化/复原”按钮、“最小化”按钮、“关闭”按钮等。用鼠标左键按住标题栏可以拖动窗体,用鼠标指向窗体的边界,如果出现双向的箭头,拖动鼠标可以改变窗体的大小。

1. Form 的主要属性

窗体组件(TForm)是一种特殊的组件,它在运行时表现为一个窗体,窗体是一个容器构件,它可以包含其他种类的构件,并协同完成应用程序的整体功能。窗体和其他组件一样,由属性、事件和方法组成。

1) BorderIcons 属性

BorderIcons 属性用来制定窗体标题栏上的图标,可以设置为下列取值。

(1) biSystemMenu:可以通过单击标题栏左边的图标或在标题栏上单击右键来显示控制菜单。控制菜单有时也称为系统菜单。

(2) biMinimize:在标题栏右边显示“最小化”按钮。

(3) biMaximize:在标题栏右边显示“最大化”按钮。

(4) biHelp:在标题栏右边显示“帮助”按钮。只有窗体的 BorderStyle 属性设置为 bsDialog 或者窗体属性 BorderIcons 中不包括 biMinimize 和 biMaximizes 时,biHelp 设置才有效。

2) BorderStyle 属性

BorderStyle 属性用来设置窗体的外观和边框,可以设置为下面的取值。

(1) bsDialog:窗体为标准的对话框,边框大小不可以改变。

(2) bsSingle：窗体具有单线边框，大小不可以改变。

(3) bsNone：窗体没有边框，也没有标题栏，边界的大小不可以改变。

(4) bsSizeable：边框大小可变的窗体。

(5) bsToolwindow：风格与 bsSingle 相同，只是标题栏比较小。另外，对于这种风格的窗体，属性 borderIcons 中设置的 biMinimize 和 biMaximize 并不起作用。

(6) bsSizeToolWin：风格与 bsSizeable 相同，只是标题栏比较小。对于这种风格的窗体，属性 BorderIcons 中设置的 biMinimize 和 biMaximize 也不起作用。

3) Name 属性

Name 属性是对象的名称，它用来唯一地标识对象，Name 属性的取值不能为空，如果工程中有多个窗体，名称不能相同。

通常，应该在系统开发的设计阶段就将整个工程所有窗体的名称确定，然后在编程阶段根据设计文档修改窗体 Name 属性。一般情况下，不要在程序运行期间通过代码修改 Name 属性。

4) Caption 属性

Caption 属性用来指定窗体标题栏中的说明文字，可以为空，省略时，Caption 属性与 Name 属性相同。

5) Font 属性

Font 属性用来设置窗体中文字的字体、颜色和字号等，其中，Font. style 属性为集合型。

6) FormStyle 属性

FormStyle 属性用来指定窗体的类型。

从窗体类型的角度来看，Windows 环境中的应用程序可以分为以下三类。

第一类：多文档界面(MDI)应用程序。一般这种应用程序具有一个父级窗口和多个子窗口，可以同时打开多个文档，分别在多个子窗口中显示。例如，常用的文字处理软件 Word 等，可以同时编辑多个文档。

第二类：单文档界面(SDI)应用程序。这种应用程序同时只能打开一个文档。例如，Windows 操作系统附件中自带的“记事本”，只能同时编辑一个文本文件。

第三类：对话框应用程序。这种应用程序的主界面基于一个对话框类型的窗体。例如，Windows 系统中自带的“扫雷”游戏程序。

FormStyle 属性的取值如下。

(1) fsNormal：普通类型的窗体，既不为 MDI 应用程序的父级窗口，也不为 MDI 应用程序的子窗口。

(2) fsMDIChildMDI：应用程序中的子窗体。

(3) fsMDIFormMDI：应用程序中的父窗体。

(4) FsStayOnTop：在桌面最前端显示窗体。

7) Icon 属性

Icon 属性用来指定标题栏中显示的图标。

单击对象编辑器 Icon 属性右边的省略号按钮 

，在弹出的 PictureEditor 对话框中单击 Load 按钮，就可以装入一个制作好的图标。

8) Position 属性

Position 属性用来描述窗体的大小和显示的位置。可以是下列取值。

(1) poDesigned：窗体显示的位置和大小与设计期间的一致。

(2) poDefault：窗体每次显示时，与上次比较，往右下角移动了一些位置；窗体的高度和宽度由 Windows 决定。

(3) poDefaulPosOnly：窗体以设计期间的大小显示，窗体显示的位置较上次向右下角移动了一些。如果窗体设计时的大小不可以在屏幕上完全显示，就移动到屏幕的左上角显示。

(4) poDefaultSizeOnly：窗体以设计期间的位置显示，窗体的大小由 Windows 决定。

(5) poScreenCenter：窗体以设计期间的大小显示，窗体显示的位置总在屏幕的中间。考虑多个监视器时位置的调整。

(6) poDesktopCenter：窗体以设计期间的大小显示，窗体显示的位置总在屏幕的中间。不考虑多个监视器时的调整。

9) WindowsState 属性

WindowsState 属性用来描述窗体显示的状态，可以是下列取值。

(1) wsNormal：窗体以普通状态显示(既不是最大化状态，也不是最小化状态)。

(2) wsMinimized：窗体以最小化状态显示。

(3) WsMaximized：窗体以最大化状态显示。

2. TForm 的事件

窗体是一个可视化的组件，它几乎可以响应和处理所有的事件，包括外部事件和内部事件。窗体常用的响应事件如表 3-1 所示。

表 3-1 窗体常用的事件响应

事件	含 义
OnActive	当窗体对象被激活时产生此事件
OnClose	当窗体对象被关闭时产生此事件
OnCloseQuery	当窗体对象被关闭时或者调用系统菜单的“关闭”菜单项产生此事件，其中包含 cancel 参数用于决定是否关闭窗体
OnCreate	当窗体对象创建时产生此事件
OnDeactivate	当窗体对象变为非激活时产生此事件
OnDestroy	当窗体对象被销毁前产生此事件
OnHide	当窗体对象被隐藏前产生此事件
OnPaint	当窗体对象客户刷新时产生此事件
OnResize	当窗体对象位置移动时产生此事件
OnShow	当窗体对象显示时产生此事件
OnKeyDown	键按下时产生此事件
OnKeyPress	键按下时产生此事件
OnKeyUp	键放开时产生此事件
OnClick	鼠标单击事件
OnDblClick	鼠标双击事件
OnDragDrop	鼠标拖放事件
OnDragOver	鼠标拖过事件
OnMouseDown	鼠标键按下事件
OnMouseMove	鼠标移过事件
OnMouseUp	鼠标键释放事件

3. 窗体的方法

窗体对象从其父类 TcustomForm 中继承了许多方法，如表 3-2 所示列出了其中一些常用的方法(过程或函数)。

表 3-2 窗体常用的方法

方法名称	含 义
Create	用来创建一个窗体并进行初始化，同时引发一个 OnCreate 事件。用该方法创建的窗体需要调用 Show 方法使之可见
Close	用来关闭一个显示中的窗体，同时调用 CloseQuery 方法来判断是否可以关闭，若可以，则引发一个 OnClose 事件并关闭窗体
CloseQuery	用来判断窗体是否可以被关闭，返回一个逻辑值
Release	用于将窗体对象从内存中彻底清除
Show	用于显示窗体，同时引发一个 OnShow 事件
ShowModal	用于显示一个模式窗体，同时引发一个 OnShow 事件
Print	用于打印窗体

4. 窗体的创建

创建一个新的应用程序(Application 工程)，Delphi XE8 将自动建立一个新的窗体，它代表应用程序的主窗口。应用程序也可以拥有多个窗体作为子窗口、对话框和数据录入窗口等，这就需要创建和显示新的窗体。创建窗体的方法分为两种：静态创建和动态创建。所谓静态创建窗体是指再工程的编辑、设计时创建新窗体；而动态创建窗体是指在工程的运行时通过代码生成窗体。

1) 静态创建新窗体

【例 3.1】 静态创建窗体。

(1) 界面设计

通过集成开发环境中的 File|New|Application 菜单，创建一个应用程序，此时应用程序自动生成一个窗体 Form1，再打开 File|New|Form 菜单生成一个窗体 Form2。在 Form1 中添加两个 Button、一个 Label 组件，Form2 中添加一个 Label 组件，即可完成界面设计，各组件的属性如图 3-3 所示。

图 3-3 静态窗体的创建

(2) 程序设计

```
procedure TForm1.Button1Click(Sender: TObject);                    //创建按钮事件
begin
  //调用 Show 方法显示 Form2 窗体
  form2.show;                                                      //关键分析
end;
```

```
procedure TForm1.Button2Click(Sender: TObject);
begin
  form1.Close;
end;
```

(3) 程序分析

关键分析：编译上述工程时，系统会弹出如图 3-4 所示的提示信息。

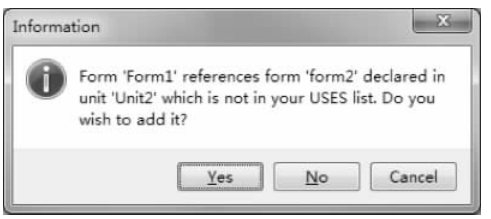

图 3-4　提示信息

因为在调用 form2.show 显示 form2 窗体时需要引用 Unit2 单元，但是 Unit1 中又没有声明要使用 Unit2 单元，所以弹出此提示信息问是否在 Unit1 单元中添加 Unit2 单元，单击 Yes 按钮，Delphi XE8 将自动在 Unit1 单元中添加对 Unit2 单元的引用。

也可以手动添加，单击 Cancel 按钮，在 Unit1 中的实现部分(implementation)添加如下代码：

```
implementation
uses unit2
```

2) 动态创建新窗体

用静态方法创建的多窗体的工程，在运行时将把所有的窗体装入内存。如果工程非常复杂，系统资源将变得非常紧张。这种情况使用下面的方法：在需要某个窗体的时候，临时创建它，使用后将其立即释放。这种方法称为窗体的动态创建。

【例 3.2】 动态创建窗体。

(1) 界面设计

通过集成开发环境中的 File|New|Application 菜单，创建一个应用程序，此时应用程序自动生成一个窗体 Form1，再打开 File|New|Form 菜单生成一个窗体 Form2。在 Form1 中添加两个 Button、一个 Label 组件，Form2 中添加一个 Label 组件，即可完成界面设计，如图 3-3 所示。

(2) 属性设置

界面上各组件的属性如图 3-3 所示。

打开集成开发环境中的 Project|Options 菜单，打开 Project Options 对话框。在 Forms 选项卡中，所有工程中的窗体出现在 Auto-create forms 列表中，表示这些窗体在运行时都是在内存中自动创建的。选中 Form2，将其移至 Available forms 列表框中，如图 3-5 所示。

(3) 程序设计

```
procedure TForm1.Button1Click(Sender: TObject);
begin
  form2: = Tform2.Create(nil);                                    //关键分析 1
  form2.Show;
end;
```

```
procedure TForm1.Button2Click(Sender: TObject);
begin
  form1.Close;
end;
procedure TForm2.FormClose(Sender: TObject; var Action: TCloseAction);
begin
  form2.Release;                                          //关键分析 2
end;
```

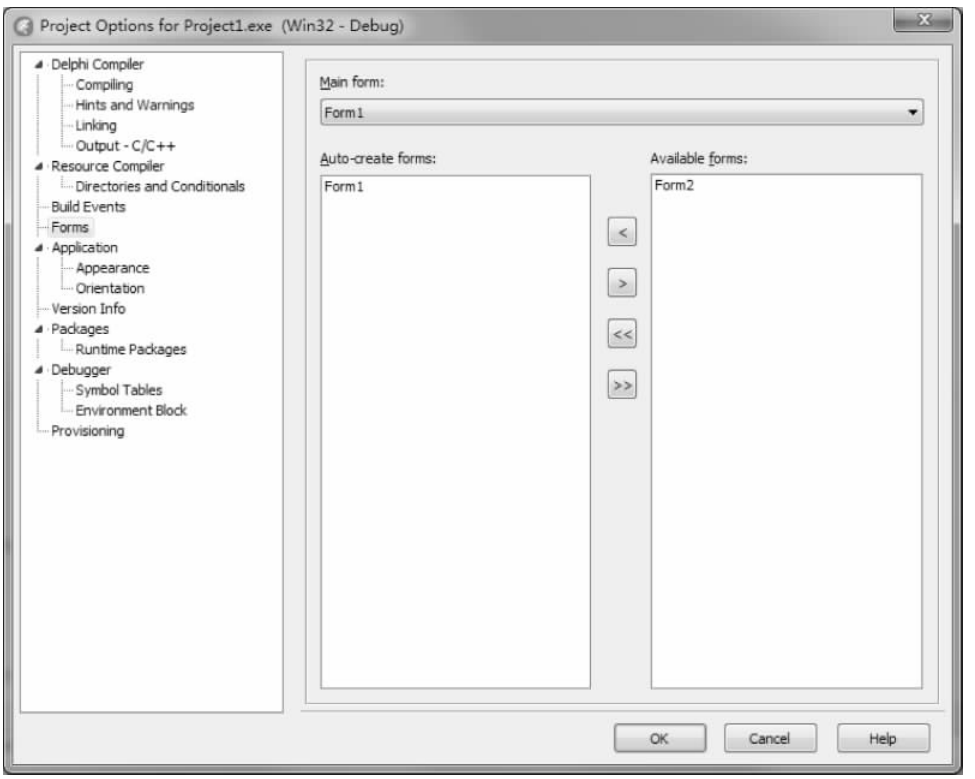

图 3-5 Project Options for Project1.exe 对话框

(4) 程序分析

关键分析 1：使用 Create 方法来创建并完成 Form2 的初始化。

关键分析 2：释放 Form2 所占有的内存。

3.1.2 弹出对话框

对话框是用户与应用程序交换信息的最佳途径之一。使用对话框函数或过程可以调用 Delphi XE8 的内部对话框，这种方法具有操作简单及快速的特点。Delphi XE8 提供以下两种内部对话框。

第一种：信息输出对话框 Showmessage 过程、ShowMessageFmt 过程、MessageDlg 函数、MessageDlgPos 函数、CreateMessageDialog 函数。

第二种：信息输入对话框 InputBox 函数、InputQuery 函数。

1. ShowMessage 过程

ShowMessage 过程显示一个最简单的对话框，其语法格式为：

```
ShowMessage(<信息内容>);
```

说明：

(1) ShowMessage 过程显示的对话框以应用程序的执行文件名作为标题，对话框中只含有一个 OK 按钮，单击该按钮对话框即关闭并返回

(2) ＜信息内容＞指定在对话框中出现的文本。在＜信息内容＞中使用硬回车(＃13)可以使文本换行。对话框的高度和宽度随着＜信息内容＞的增加而增加。

2. ShowMessageFmt 过程的语法格式为：

```
ShowMessageFmt(<信息内容>,<参数组>);
```

说明：ShowMessageFmt 过程与 ShowMessage 过程的功能基本相同，只是参数＜信息内容＞为格式化了的字符串，与＜参数组＞配合，形成显示在对话框中的信息。例如，下述代码将得到如图 3-6 所示的对话框：

```
ShowMessageFmt('张宇同学％s 考了％d 分!',['外语',100]);
```

图 3-6　信息对话框

3. MessageDlg 函数

调用 MessageDlgPos 函数，可以在屏幕的中心处显示信息对话框，其语法格式为：

```
<变量> = MessageDlg(<信息内容>,<类型>,<按钮组>,HelpCtx);
```

说明：

(1) ＜信息内容＞是显示在对话框中的信息

(2) ＜类型＞是对话框的类型，其取值如表 3-3 所示。

表 3-3　对话框类型的取值

取值	说　明
mtWarning	弹出含有感叹号符号的警告话框
mtError	弹出含有红色叉符号的错误对话框
mtInformation	弹出含有蓝色 i 符号的信息对话框
mtConfirmation	弹出含有绿色 ？ 号的确认对话框
mtCustom	弹出不含图标的一般对话框，对话框的标题是程序的名称

(3) ＜按钮组＞指定对话框中出现的按钮，其中出现的按钮与参数的取值如表 3-4 所示。

表 3-4　对话框按钮类型的取值

取　值	说　明
mbYes	单击 Yes 按钮，函数返回 mrYes 或 6
mbNo	单击 No 按钮，函数返回 mrNo 或 7

续表

取　值	说　明
mbOK	单击 OK 按钮，函数返回 mrOk 或 1
mbCancel	单击 Cancel 按钮，函数返回 mrCancel 或 2
mbHelp	Help 按钮
mbAbort	单击 Abort 按钮，函数返回 mrAbort 或 3
mbRetry	单击 Retry 按钮，函数返回 mrRetry 或 4
mbIgnore	单击 Ignore 按钮，函数返回 mrlgnore 或 5
mbAll	单击 all 按钮，函数返回 mrall 或 8
mbNoToAll	单击 Nottoall 按钮，函数返回 9
mbYesToAll	单击 Yestoall 按钮，函数返回 10

＜按钮组＞可以组的形式，如[mbYes，mbNo]表示对话框中出现两个按钮 Yes 和 No；也可以常量的形式，如 mbOKCancel 表示对话框中出现两个按钮 OK 和 Cancel。按钮常量的含义如表 3-5 所示。

表 3-5　按钮常量的取值

取　值	说　明
mbYesNoCancel	三个按钮：Yes、NO、Cancel
mbOKCancel	两个按钮：OK、Cancel
mbAortRetryIgnore	三个按钮：Abort、Retry、Ignore

(4) HelpCtx 指定当用户单击 Help 按钮或按 F1 键时，显示的帮助主题。

(5) MessageDlg 函数将根据用户所单击的按钮，返回相应的值(Word)类型。

4. MessageDlgPos 函数

调用 MessageDlgPos 函数，可以在屏幕的指定位置显示信息对话框，其语法格式为：

```
<变量> = MessageDlgPos(<信息内容>,<类型>,<按钮组>,HelpCtx,X,Y);
```

说明：MessageDlgPos 函数只比 MessageDlg 函数多一项功能，即它可以指定对话框的显示位置坐标：X，Y。

5. CreatMessageDialog 函数

与上述函数与过程不同，CreatMessageDialog 函数生成一个信息框窗体，可以在程序中多次使用方法调用。其语法格式为：

```
<变量> = MessageDiaPos(<信息内容>,<类型>,<按钮组>);
```

说明：函数的参数与 MessageDlg 函数相似，只是返回一个 TForm 类型的对话框，而且并没有把它显示出来。在需要显示该对话框的时候，可以使用窗体 ShowModal 的方法把它弹出。

6. InputBox 函数

InputBox 函数显示一个能接收用户输入的对话框，并返回用户输入的信息。其语法格为：

```
<变量> = InputBox (<对话框标题>,<信息内容>,<默认内容>);
```

说明：

(1) ＜对话框标题＞指定对话框的标题。

(2) ＜信息内容＞指定在对话框中出现的文本。在＜信息内容＞中使用硬回车符(＃13)可以使文本换行。对话框的高度和宽度随着＜信息内容＞的增加而增加。

(3) ＜默认内容＞指定对话框的输入框中显示的文本，可以修改。如果用户单击“确定”按钮，输入框中的文本将返回到＜变量＞中，若用户单击“取消”按钮，将＜默认内容＞返回到＜变量＞中。

7. InputQuery 函数

如果希望对单击 Cancel 按钮(退出事件)另做处理，可以使用 InputQuery 函数。该函数与 InputBox 函数相似，只是返回值是一个布尔值。其语法格式为：

```
<变量> = InputQuery (<对话框标题>,<信息内容>,<字符串变量>);
```

说明：InputQuery 函数与 InputBox 函数的参数相似，默认内容存放在＜字符串变量＞中。可以修改。如果用户单击 OK 按钮，输入框中的文本将赋值到＜字符串变量＞中，并且函数返回 True；若用户单击 Cancel 按钮，＜字符串变量＞中的值保持不变，并返回 False。

3.2 输入显示类组件

3.2.1 Edit 组件

编辑框(Edit)是一种通用组件，既可以输入文本，又可以显示文本，是应用程序中最常用的组件之一。编辑框组件位于 Standard 组件板中，如图 3-7 所示。

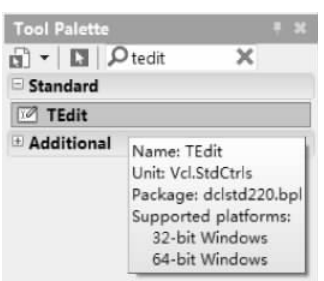

图 3-7 编辑框

Edit 的主要属性如下。

1. AutoSelect 属性

AutoSelect 属性设置编辑框得到焦点时，文本是否自动被选中。

2. AutoSize 属性

AutoSize 属性决定编辑框是否自动随字体的变化而改变大小。

3. Enable 属性

Enable 属性用来设置编辑框是否能用。

4. BorderSytle 属性

BorderSytle 属性用来设置编辑框的边框类型，取 bsSigle 为单线，取 bsNone 为无框。

5. MaxLength 属性

MaxLength 属性设置所能接受的最大字符数。

6. PasswordChar 属性

该属性设置非＃0 字符时，将代替用户输入的字符被显示。

7. ReadOnly 属性

决定编辑框中的文本是否可以编辑。

8. SelStart 属性

被选中文本的开始位置，或光标在文本中的位置。

9. SelText 属性

被选中的文本。

10. SelLength 属性

被选中文本的长度。

11. Text 属性

编辑框中的文本内容。

12. CharCase 属性

控制编辑框中文本的大小写，取值 ecLowerCase，Text 中的文本转换为小写；取值 ecNormal 不改变；取值 ecUpperCase，Text 中的文本转换为大写。

3.2.2 Label 组件

标签(Label)是 Delphi XE8 中最常用的输出文本信息的工具。标签组件位于 Standard 组件板中，如图 3-8 所示。

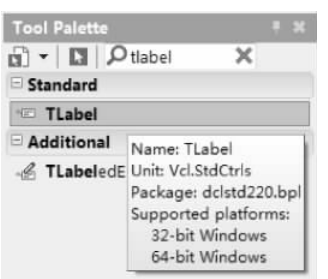

图 3-8 标签

1. Label 的主要属性

1) Caption 属性

属性 Caption 用来显示标签的文本。

2) ShowAccelChar 属性

属性 ShowAccelChar 决定是否将 & 作为热键字符的标记。

3) AutoSize 属性

属性 AutoSize 决定标签是否自动随文本的变化而改变大小。

4) Alignment 属性

属性 Alignment 决定对齐方式：左对齐、居中对齐、右对齐。

5) Layout 属性

属性 Layout 控制文本显示在标签的顶部、中部、底部。

6) WordWrap 属性

属性 WordWrap 控制是否折行显示。

7) Transparent 属性

属性 Transparent 决定背景是否透明。

8) FocusControl 属性

属性 FocusControl 用来设置按下热键时，获得焦点的组件名。

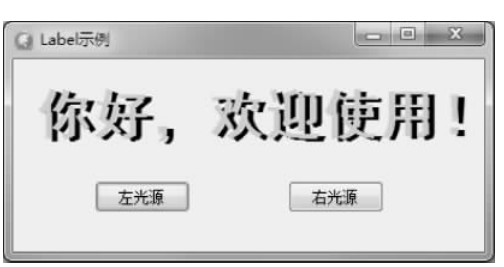

图 3-9 Label 示例

2. Label 的使用

【例 3.3】 利用标签设计一个投影效果。

1) 界面设计

创建一个新的工程，在窗体中添加两个按钮 Button1 和 Button2 组件，再添加两个标签

Label1 和 Label2 组件,如图 3-9 所示。

2) 属性设置

其组件和窗体的属性设置如表 3-6 所示。

表 3-6 窗体和组件属性设置

对象	属性	属性值	说明
Button1	Caption	左光源	按钮标题
Button2	Caption	右光源	按钮标题
Label1	Font. name	黑体	字体
	Caption	你好,欢迎使用!	按钮标题
	Font. Size	32	字体大小
	Font. Style. fsBold	true	加粗
	Font. color	cllime	绿色
	TransParent	True	透明
Label2	Font. name	黑体	字体
	Caption	你好,欢迎使用!	按钮标题
	Font. Size	32	字体大小
	Font. Style. fsBold	true	加粗

3) 程序设计

```
procedure TForm1.Button1Click(Sender: TObject);
begin
  Label2.Left: = label1.Left + 5;
end;
procedure TForm1.Button2Click(Sender: TObject);
begin
  Label2.Left: = label1.Left - 5;
end;
```

3.2.3 Memo 组件

备注框(Memo)是 Delphi XE8 中最常用的输出多行文本信息的工具。备注框组件位于 Standard 组件板中,如图 3-10 所示。

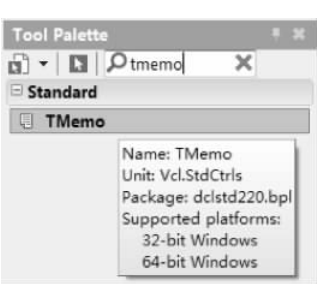

图 3-10 备注框

Edit 组件只能操作单行文本,如果文本长度超出可用空间,则只能显示部分文本。要处理多行文本就要用到备注框 Memo 组件。

1. Memo 的主要属性

备注框在 Delphi XE8 中用 Tmemo 类处理,Tmemo 类是 Tedit 类的衍生类,因此,Tedit 类所具有的属性、事件和方法在 Tmemo 类中都有。另外,为了处理多行文本,Tmemo 类还增加了一些新的属性。

1) CaretPos 属性

CaretPos 属性用来得到光标在编辑区中的位置,如 Memo1. CaretPos. y 表示光标所在行数。

2) Lines 属性

Lines 属性是按行处理文本。Lines 属性实际上是一个 Tstring 类型的对象,用来存放

Memo 对象的文本。Tstrings 类型有一个默认的属性 Strings，定义为：

```
Property Strings[Index: integer]: string;
```

其中，Index 表示字符串组的索引，从 0 开始。因此，Memo1. Lines[0]表示 Memo 组件中第一行的字符串。如下面的代码可以将第 5 行的文本赋值给变量 S：

```
S: = Memo1.lines[4];
```

单击属性窗口 Lines 属性值右端的 ··· 按钮，可以打开 String List Editor 对话框。在其中可以直接输入备注中各行 Lines 的内容，如图 3-11 所示。

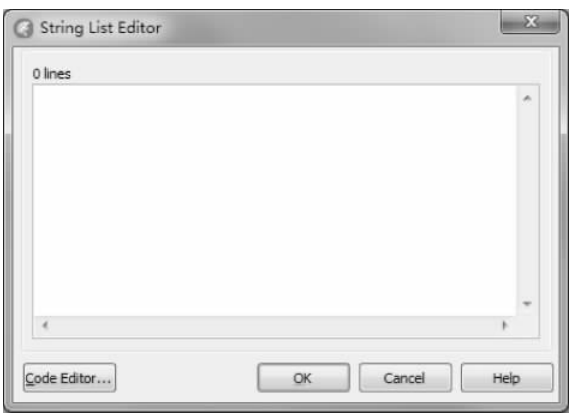

图 3-11 String List Editor 对话框

在编程时可以调用 Memo 的一些方法来处理 Lines 中的数据。

```
Memo1.Lines.Add(s);                 //在 Memo 最后增加一行字符串 s,并返回改行的行值
Memo1.Lines.Delete(x);              //删除 Memo 中的 x + 1 行字符串
Memo1.Lines.Insert(x,s);            //在第 x + 1 行处插入一个新行
Memo1.Lines.Move(x,y);              //将第 x + 1 行移至第 y + 1 行处
Memo1.Lines.LoadFile(path);         //将 path 指定路径的文件加载到 Memo 中
```

3）Modified 属性

Modified 属性用于确定文本是否被改动过。

4）ScrollBars 属性

ScrollBars 属性决定备注框是否具有滚动条，ScrollBars 可以有下列取值。

（1）ssNone：无滚动条。

（2）ssHorizontal：只有水平滚动条。

（3）ssVertical：只有垂直滚动条。

（4）ssBoth：同时具有垂直和水平滚动条。

5）WordWrap 属性

WordWrap 属性设置文本是否能够换行，如果没有水平方向的滚动条，WordWrap 属性值为 True 时，备注框中的文本会自动按字换行，即当一行文本已超过所能显示的长度时，备注框自动将文本折回到下一行显示。自动按字换行可省去用户在行尾插入换行符的麻烦。

6）WantReturns 属性

WantReturns 属性用来设置备注框是否能插入回车键。

7) WantTabs 属性

WantTabs 属性用来设置备注框是否能插入 Tab 键。

2. Memo 的使用

【例 3.4】 利用编辑框,把编辑框中的文本输入到 Memo 中。

1) 界面设计

创建一个新的工程,在窗体中添加一个按钮 Button 组件、一个编辑框 Edit 组件和一个备注框 Memo 组件,各组件的属性设置如图 3-12 所示。

图 3-12 "Memo 示例"界面

2) 程序设计

```
procedure TForm1.Button1Click(Sender: TObject);
begin
  Memo1.Lines.Add(Edit1.Text);
  edit1.Text: = '';
  edit1.SetFocus;
end;

procedure TForm1.FormActivate(Sender: TObject);
begin
  edit1.SetFocus;
end;
```

3.2.4 MaskEdit 组件

MaskEdit 组件是格式化的编辑框,它限制用户在所定义的位置输入要求输入的符号。掩码编辑框(MaskEdit)组件位于 Additional 附加组件板中,如图 3-13 所示。

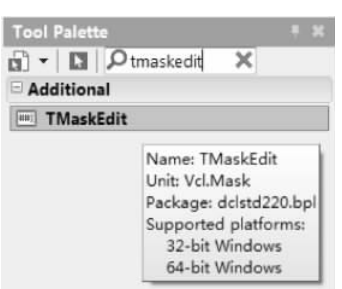

图 3-13 TMaskEdit 组件

1. MaskEdit 常用属性

MaskEdit 组件在 Delphi XE8 中用 TMaskEdit 类处理,TMaskEdit 类也是 TEdit 类的子类,因此,TEdit 类所具有的属性、事件和方法在 TMaskEdit 类中都有。另外,TMaskEdit 类还增加了一些新的属性。

1) EditMask 属性

EditMask 属性用来控制用户输入数据格式的掩码字符串,掩码字符串 EditMask 属性分为三个部分,用分号分隔。第一部分是掩码字符串的主要部分,它确定了数据的格式,其中所用到的特殊字符及基意义参见表 3-7;第二部分决定是否将掩码中的字符串作为数据的一部分,0 表示不作为数据的一部分,1 表示作为数据的一部分,它将影响属性;第三部分指出在掩码中用来代表未输入数据的字符。

表 3-7 EditMask 中的特殊字符

字符	意　义
!	如果"!"出现在掩码中,则输入字符串的首空格不会存成数据,如果"!"没有出现在掩码中,则输入字符串的末空格不会存成数据
>	如果>出现在掩码中,则它后面的所有字符都会变为大写,除非遇到<
<	如果<出现在掩码中,则它后面的所有字符都会变为小写,除非遇到>

续表

字符	意　义
<>	如果<>出现在掩码中，则它后面的所有字符的大小写状态都将不会改变
\	\后面的字符作为原字符显示，使用\可以将任何字符原样显示
L	L字符表示该字符位置处只能输入一个字母，在英文中为：A～Z、a～z
I	I字符表示该字符位置处只能输入一个字母，在英文中为：A～Z、a～z，但不一定输入
A	A字符表示该字符位置处只能输入一个字母或数字字符，在英文中为：A～Z、a～z及0～9
a	a字符表示该字符位置处只能输入一个字母，但一定输入
C	C字符表示该字符位置处可以输入任意一个字符
c	c字符表示该字符位置处可以输入任意一个字符，但一定输入
0	0字符表示该字符位置处只能输入一个数字字符：0～9
9	9字符表示该字符位置处只能输入一个数字字符：0～9，但不一定输入
#	#字符表示该字符位置处只能输入一个数字字符或正负号，但不一定输入
:	“：”字符用来分隔时间中的时、分、秒
/	“/”字符用来分隔日期中的年、月、日
;	“；”字符用来分隔掩码字符串的三个部分
—	“—”字符将自动在text中插入一个空格。当用户输入时，光标会自动跳过该字符

说明：

(1) 由于汉字占两个字符，所以用于控制输入中文的掩码字符也必须有两个，如LL或cc等。

(2) 表中没有列出的一般字符都会被显示出来，输入时这些字符会自动跳过。

(3) EditText与Text属性都可以用来读取用户输入的数据。当掩码字符串中第二部分为1时，EditText与Text的值是相同的；当掩码字符串中第二部分为0时，Text的值不包含掩码字符串自动显示的字符，而只包含用户输入的字符。

2) EditText属性

EditText属性用来返回用户输入的数据。

2. MaskEdit的使用

【例3.5】 个人档案编辑器。

1) 界面设计

创建一个新的工程，在窗体中添加两个Button组件、两个Edit组件、一个Memo组件和两个MaskEdit组件，如图3-14所示。

图3-14 个人档案编辑器

2) 属性设置

各个组件的属性如图3-14所示，其中，MaskEdit1组件的EditMask属性通过单击属性值右端的按钮打开Input Mask Editor对话框来设置，如图3-15所示。

用同样的方法可以设置MaskEdit2组件的EditMask属性，输入掩码字符串：“! 0000年09月09日;1;_”。

3) 程序设计

```
procedure TForm1.Button1Click(Sender: TObject);
begin
```

```
  memo1.Lines.Add('姓     名: ' + edit1.Text);
  memo1.Lines.Add('联系电话: ' + maskedit1.Text);
  memo1.Lines.Add('联系单位: ' + maskedit2.Text);
  memo1.Lines.Add('工作单位: ' + edit2.Text);
end;

procedure TForm1.Button2Click(Sender: TObject);
begin
  edit1.Text: = '';
  maskedit1.Text: = '';
  maskedit2.Text: = '';
  edit2.Text: = '';
end;
```

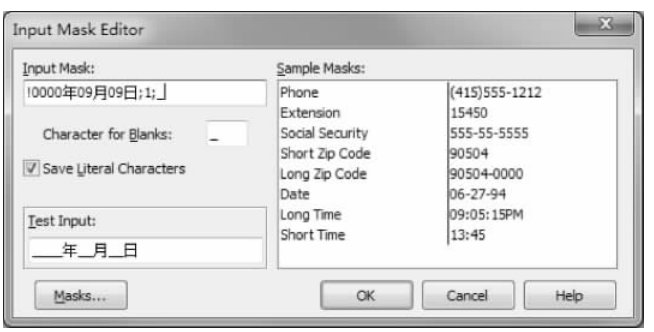

图 3-15　Input Mask Editor 对话框

3.3　按钮类组件

3.3.1　Button 组件

Button 按钮往往作为一个窗体中某些行为的执行工具,Button 按钮位于 Delphi XE8 组件板 Standard 选项卡中,如图 3-16 所示。

1. Button 的主要属性

基本按钮(Button)是程序中最常用的组件之一,下面介绍它的主要属性。

1) Caption 属性

Caption 属性用来指定按钮所显示的文字。

2) Cancel 属性

Cancel 属性用来决定该按钮是否为"取消"按钮,默认值为 False。当 Cancel 属性设为 True 时,单击 Button 与按 Esc 键等价。

3) Default 属性

Default 属性用来决定该按钮是否为默认按钮,默认值为 False。当 Default 属性设为 True 时,单击 Button 与按 Enter 键等价。

4) ModalResult 属性

ModalResult 属性用来决定模式窗体如何被关闭。

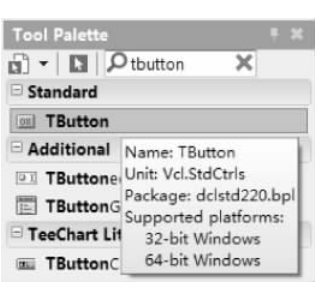

图 3-16　Button 按钮

ModalResult 属性的取值有：mrNone、mrOK、mrCancel、mrAbort、mrRetry、mrIgnore、mrYes、mrNo、mrAll、mrNoToAll、mrYesToAll。当 Button 被置于一个模式窗体(Modal Form)时，ModalResult 属性才有意义。此时，若属性值不是 0(mrNone)，Button 被单击后，所在的模式窗体会自动关闭，并且 ModalResult 属性值会作为 ShowModal 函数的结果返回，因此无须再为其 OnClick 事件编写代码。

2. Button 的事件

Button 组件的绝大部分事件是响应鼠标动作的。Button 组件常用的事件如表 3-8 所示。

表 3-8 Button 按钮常用的事件响应

事　件	含　义
OnClick	鼠标单击事件
OnMouseDown	鼠标键按下事件
OnMouseMove	鼠标移过事件
OnMouseUp	鼠标键释放事件

在下述两种情况下，OnClick 事件将被激发。

(1) 用鼠标单击按钮。

(2) 当按钮获得焦点时，按 Enter 键或空格键。

3.3.2 BitBtn 组件

位图按钮(BitBtn)组件的工作方式与 Button 组件相似，但可以显示一个彩色的位图(Bitmap)。使用位图可以比看到一些专业化的术语更容易让人理解其功能和作用。BitBtn 按钮位于 Delphi XE8 组件板 Additional 选项卡中，如图 3-17 所示。

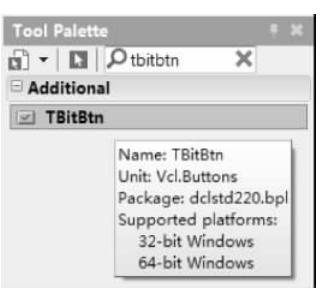

图 3-17 BitBtn 按钮

1. BitBtn 的主要属性

由于 TbitBtn 类是 Tbutton 的子类，它具有 Button 的所有属性，下面介绍一些有别于 Button 的主要属性。

1) Glyph 属性

Glyph 属性为 BitBtn 指定一个. bmp 文件，显示在按钮的表面。

2) Kind 属性

Kind 属性决定 BitBtn 按钮的种类。Kind 属性的取值有：bkCustom、bkOK、bkCancel、bkHelp、bkYes、bkNo、bkClose、bkAbort、bkRetry、bkIgnore、baAll。默认值为 bkCustom，表示其种类为自定义类型，位图由 Glyph 属性指定。Kind 属性的其他取值，使 BitBtn 按钮对应一个特殊的位图，并将其 ModalResult 属性自动设为相应的值。如 bkOK、bkCancel、bkYes 和 bkNo 分别对应于 mrOK、mrCancel、mrYes 和 mrNo 等。

3) Layout 属性

Layout 属性用来控制 BitBtn 按钮中位图与文本的相对位置。Layout 属性的取值有：blGlyphLeft、blGlyphRight、blGlyphTop、blGlyphBottom。分别设置位图出现在文本的左侧、右侧、上方、下方。默认值为 blGlyphLeft。

4) Margin 属性

Margin 属性用来控制 BitBtn 按钮中位图与边界之间的像素个数。Margin 属性的默认值为－1,表示位图位于按钮的正中。当 Layout 属性的值为 blGlyphLeft 时,Margin 属性的值表示按钮左边界与位图的距离;当 Layout 属性的值为 blGlyphRight 时,Margin 属性的值表示按钮右边界与位图的距离;以此类推。

5) Spacing 属性

Spacing 属性用来控制 BitBtn 按钮中位图与文本之间的(距离)像素个数。Spacing 属性的值为－1 时,文本、位图与按钮边界成等距离。当 Spacing 属性的值为非负整数时,表示文本与位图之间的距离。Spacing 属性的默认值为 4。

2. BitBtn 的事件

BitBtn 组件的绝大部分事件和 Button 组件一样是响应鼠标动作的。BitBtn 组件常用的事件如表 3-9 所示。

表 3-9 BitBtn 按钮常用的事件响应

事　件	含　义
OnClick	鼠标单击事件
OnMouseDown	鼠标键按下事件
OnMouseMove	鼠标移过事件
OnMouseUp	鼠标键释放事件

在下述两种情况下,OnClick 事件将被激发。

(1) 用鼠标单击按钮。

(2) 当按钮获得焦点时,按 Enter 键或空格键。

【例 3.6】 含有各种类型位图按钮的窗体。

1. 界面设计

创建一个新的工程,在窗体中添加 11 个位图按钮 BitBtn1～BitBtn11 组件,如图 3-18 所示。

图 3-18 位图按钮示例

2. 属性设置

BitBtn1 的属性设置如表 3-10 所示。其他按钮的属性设置与之类似。

表 3-10 窗体及组件属性设置

对象	属性	属性值	说明
BitBtn1	Caption	是(&Y)	位图按钮的标题
	Kind	bkYes	位图按钮的种类
	Hint	Kind＝bkYes	作为工具提示出现的文本
	Showhint	True	是否出现工具提示

说明:BitBtn11 按钮的 Kind 属性值为 bkClose 时,无须为 BitBtn11 按钮编写事件代码,按钮自动具备关闭窗体的功能。

3.3.3 SpeedButton 组件

快速按钮(SpeedButton)是一种可以成组工作的按钮,具有与位图按钮一样将位图显示在按钮表面的功能;还具有与单选按钮一样允许其中一个按钮被选中(按下)的功能;当它单独使用的时候可以像复选框一样具有开关的功能。快速按钮位于 Additonal 组件板中,如图 3-19 所示。

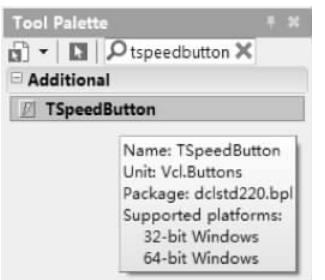

图 3-19 快速按钮

快速按钮的大部分属性与 Button 和 BitBtn 相同,也有一些独具的特殊属性。

1. AllowAllUp 属性

AllowAllUp 属性控制是否允许单击处于按下状态的按钮,使之恢复到松开状态。默认值为 False。

2. Down 属性

Down 属性设置按钮是否处于按下状态。默认值为 False。

3. Flat 属性

当属性 Flat 取值为 True 时,按钮具有 Office 2010 工具栏的风格。默认值为 False。

4. GroupIdex 属性

该属性默认值为 0,表示不与其他 SpeedButton 成组。若取值大于 0,则相同 GroupIndex 值的 SpeedButton 将成组协同工作,像单选按钮组一样。要使 SpeedButton 像复选框那样工作,只需将 GroupIndex 属性设为大于 0,且不与任何其他 SpeedButton 的 GroupIndex 属性值相同,再将 AllowAllUp 属性设为 True 即可。

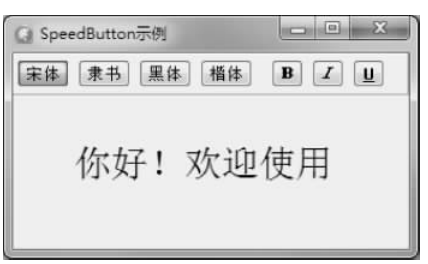

图 3-20 SpeedButton 示例

【例 3.7】 使用 SpeedButton 控制文本的字体与字体风格。

1) 界面设计

创建一个新的工程,在窗体中添

加一个 Label 组件、一个 Panel 组件,在 Panel 组件上添加 7 个 SpeedButton 组件,如图 3-20 所示。

2) 属性设置

各组件的属性设置如表 3-11 示。

表 3-11 窗体及组件属性设置

对象	属性	属性值	说明
SpeedButton1	Caption	宋体	按钮标题
	Down	True	按下状态
	GroupIndex	1	组 1
SpeedButton2	Caption	隶书	按钮标题
	GroupIndex	1	组 1
SpeedButton3	Caption	黑体	按钮标题
	GroupIndex	1	组 1

续表

对象	属性	属性值	说明
SpeedButton4	Caption	楷体	按钮标题
	GroupIndex	1	组 1
SpeedButton5	Caption	B	按钮标题
	AllowAllUp	True	允许按键松开
	Font. Style. fsBold	True	粗体
	GroupIndex	2	组 2
SpeedButton6	Caption	I	按钮标题
	AllowAllUp	True	允许按键松开
	Font. Style. fsItalic	True	斜体
	GroupIndex	3	组 3
SpeedButton7	Caption	U	按钮标题
	AllowAllUp	True	允许按键松开
	Font. Style. fsUnderline	True	下划线
	GroupIndex	4	组 4
Label1	Caption	你好！欢迎使用	标题

3）程序设计

```
procedure TForm1.SpeedButton1Click(Sender: TObject);
begin
  Label1.Font.Name: = '宋体';
end;

procedure TForm1.SpeedButton2Click(Sender: TObject);
begin
  Label1.Font.Name: = '隶书';
end;

procedure TForm1.SpeedButton3Click(Sender: TObject);
begin
  Label1.Font.Name: = '黑体';
end;
procedure TForm1.SpeedButton4Click(Sender: TObject);
begin
  Label1.Font.Name: = '楷体_GB2312';
end;

procedure TForm1.SpeedButton5Click(Sender: TObject);
begin
  if SpeedButton5.Down then
    label1.Font.Style: = label1.Font.Style + [fsBold]      //关键分析
  else
    label1.Font.Style: = label1.Font.Style - [fsBold];
end;

procedure TForm1.SpeedButton6Click(Sender: TObject);
```

```
begin
  if SpeedButton6.Down then
    label1.Font.Style: = label1.Font.Style + [fsItalic]
  else
    label1.Font.Style: = label1.Font.Style - [fsItalic];
end;

procedure TForm1.SpeedButton7Click(Sender: TObject);
  begin
  if SpeedButton7.Down then
    label1.Font.Style: = label1.Font.Style + [fsunderline]
  else
    label1.Font.Style: = label1.Font.Style - [fsunderline];
end;
```

4）程序分析

关键分析：Label1 对象的字体风格 Label1.Font.Style 属性为集合类型，代码中的＋、－运算属于集合的“并”与“差”运算。

3.4 复选框、单选按钮和单选按钮组

3.4.1 CheckBox 组件

复选框(CheckBox)是一个旁边带有文本说明的小方框。CheckBox 位于 Delphi XE8 组件板 Standard 选项卡中，如图 3-21 所示。

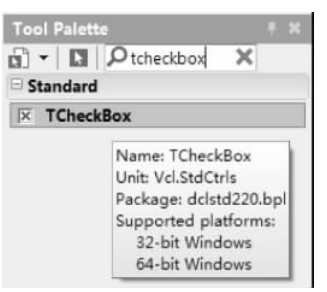

图 3-21 复选框

复选框 CheckBox 具有未选中状态 ☐ 和选中状态 ☑。在运行时，使用鼠标单击复选框可以改变它的状态。复选框还有一种不确定状态，表示既非选中，又非未选中。

1. CheckBox 的主要属性

1）Checked 属性

属性 Checked 用于表明 CheckBox 是否被选中。

2）State 属性

属性 State 进一步确定 CheckBox 的状态。有三种取值：cbChecked、cbUnchecked 和 cbGrayed，分别表示选中、未选中和不确定。

3）AllowGrayed 属性

属性 AllowGrayed 为 True 时，复选框有三种选择；为 False 时，复选框只有选中和未选中两种状态。

2. CheckBox 的使用

【例 3.8】 使用 CheckBox 组件完成简单的选课系统。

1）界面设计

创建一个新的工程，在窗体中添加两个 GroupBox 组件，在一个 GroupBox 组件上添加一个 Memo 组件，在另一个 GroupBox 组件上添加 4 个 CheckBox 组件、添加一个 Panel 组件，在 Panel 组件上添加三个 Label 组件和三个 Edit 组件，再添加两个 Button 组件，各组件的属性设置如图 3-22 所示。

图 3-22　选课系统

2) 程序设计

```
procedure TForm1.Button1Click(Sender: TObject);
var
  str: string;
begin
  memo1.Lines.Add('学号: ' + edit1.Text);
  memo1.Lines.Add('姓名: ' + edit2.Text);
  memo1.Lines.Add('班级: ' + edit3.Text);
  str: = '';
  if checkbox1.Checked then
    str: = str + 'Delphi XE8 程序设计';
  if checkbox2.Checked then
    str: = str + 'VC++程序设计';
  if checkbox3.Checked then
    str: = str + 'C 语言程序设计';
  if checkbox4.Checked then
    str: = str + 'VS.Net 程序设计';
  memo1.Lines.Add('您选的课程: ' + copy(str,0,length(str) - 2) + '.');  //关键分析
end;

procedure TForm1.Button2Click(Sender: TObject);
begin
  edit1.Text: = '';
  edit2.Text: = '';
  edit3.Text: = '';
  checkbox1.Checked: = false;
  checkbox2.Checked: = false;
  checkbox3.Checked: = false;
  checkbox4.Checked: = false;
end;
```

3) 程序分析

关键分析：调用字符串截取函数 Copy 将字符串 Str 末尾的顿号“、”去掉，并在末尾添加一个句号“。”。

3.4.2　RadioButton 组件

单选按钮(RadioButton)又称选项按钮，一般总是作为一个组(单选按钮组)的组成部分工

作的。单选按钮组中只能单击一个选项，即单选按钮组只允许用户从菜单中选择一个选项。RadioButton 位于 Delphi XE8 组件板 Standard 选项卡中，如图 3-23 所示。

1. RadioButton 的主要属性

Checked 属性：用于表明 CheckBox 是否被选中。RadioButton 有两种状态，如果当 Checked 属性为 True 时，表示选中状态，如果当 Checked 属性为 False 时，表示未选中状态。

2. RadioButton 的使用

【例 3.9】 通过 RadioButton 组件来控制 Label1 的字体。

1）界面设计

创建一个新的工程，在窗体中添加一个 Label 组件、4 个 RadioButton 组件，各个组件的属性设置如图 3-24 所示。

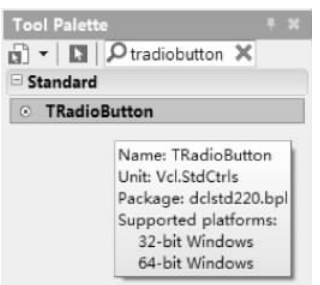

图 3-23 单选按钮

图 3-24 RadioButton 示例

2）程序设计

```
procedure TForm1.RadioButton1Click(Sender: TObject);
begin
  Label1.Font.Name: = '宋体';
end;

procedure TForm1.RadioButton2Click(Sender: TObject);
begin
  Label1.Font.Name: = '隶书';
end;

procedure TForm1.RadioButton3Click(Sender: TObject);
begin
  Label1.Font.Name: = '黑体';
end;

procedure TForm1.RadioButton4Click(Sender: TObject);
begin
  Label1.Font.Name: = '楷体_GB2312';
end;
```

3.4.3 RadioGroup 组件

单选按钮组（RadioGroup）组件巧妙地将一个 GroupBox 与一组 RadioButton 组合在一起，可以使用统一的索引号（ItemIndex），为编程提供了方便。RadioGroup 位于 Delphi XE8

组件板 Standard 选项卡中,如图 3-25 所示。

1. RadioGroup 的主要属性

1) Columns 属性

属性 Columns 用于设置单选按钮组中按钮的列数。范围 1～16,默认值为 1。

2) Items 属性

属性 Items 用于设置各种单选按钮的标题。

3) ItemIndex 属性

单选按钮组中被选中按钮(从 0 开始)的序号。默认值为－1,表示组中按钮均未被选中。

2. RadioGroup 的使用

【例 3.10】 通过 RadioButton 组件来控制 Label1 的字体。

1) 界面设计

创建一个新的工程,在窗体中添加一个 Label 组件、一个 RadioGroup 组件,如图 3-26 所示。

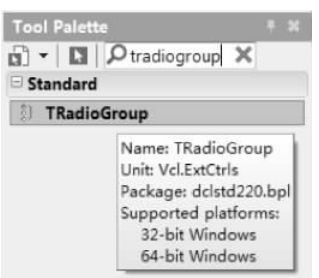

图 3-25 单选按钮组

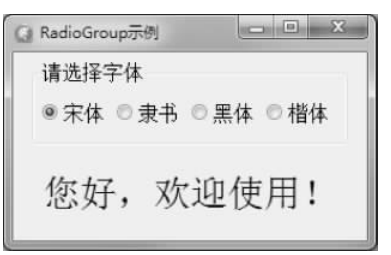

图 3-26 RadioGroup 示例

2) 属性设置

各组件的属性设置如表 3-12 所示。

表 3-12 窗体及组件属性设置

对象	属性	属性值	说　明
RadioGroup1	ItemIndex	0	默认字体“宋体”按钮被选中
	Columns	4	RadioGroup1 按钮按 4 列显示
	Items	见下说明	设置各单选按钮的标题
Label1	Caption	您好,欢迎使用!	标题
	Font. Name	宋体	默认字体为宋体

说明:RadioGroup1. Items 的设置。

单击 RadioGroup1 组件 Items 属性的右端按钮,打开 String List Editor 对话框,在其中依次输入 4 个单选按钮的标题,每行以回车键结束,如图 3-27 所示。

3) 程序设计

```
procedure TForm1.RadioGroup1Click(Sender: TObject);
begin
  case radiogroup1.ItemIndex of
    0: label1.Font.Name: = '宋体';                    //关键分析
    1: Label1.Font.Name: = '隶书';
```

```
    2: label1.Font.Name: ='黑体';
    3: label1.Font.Name: ='楷体_GB2312';
  end;
end;
```

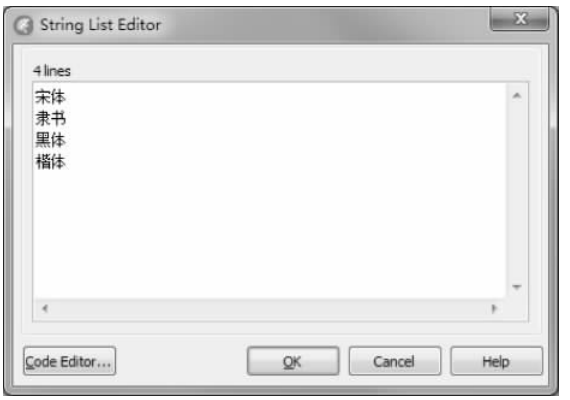

图 3-27 字符串列表编辑器

4）程序分析

关键分析：当 radiogroup1.ItemIndex 为 0 时，表示第一个单选按钮“宋体”被选中，其余各值以此类推。

3.5 列表框、组合框

3.5.1 ListBox 组件

当列表框不能同时显示所有选择项时，将自动加上一个垂直滚动条，使用户可以上下滚动列表框，以查阅所有的选项。列表框位于组件板 Standard 选项卡中，如图 3-28 所示。

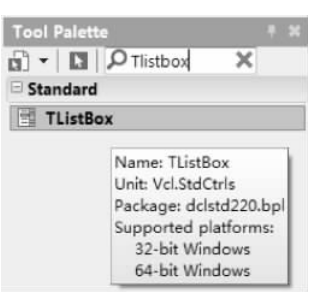

图 3-28 列表框

ListBox 的主要属性如下。

1. Items 属性

Items 属性是以行作为单位，列表框中选项的集合。

2. ItemsIndex 属性

ItemsIndex 属性为选项的索引值。

3. Stored 属性

Stored 属性决定选项是否排序。

4. Columns 属性

Columns 属性决定列表框的列数。

5. MultiSelect 属性

MultiSelect 属性决定是否可以选择多项。

6. SelCount 属性

SelCount 属性表示被选中的项的数目，只读。

7. Selected 属性

Selected 属性用来设置或返回是否被选中。

8. IntegralHeight 属性

IntegralHeight 属性可以有以下取值。

(1) True：自动调整框的高度使每行的高度(IntemHeight)可以完整地被显示。

(2) False：不自动调整框的高度,非完整高度行被显示在框的底部。

9. ItemHeight 属性

ItemHeight 属性用来控制列表框中行的高度,Style 属性为 lbStandard 时不能改变。

10. Style 属性

Style 属性可以有以下取值。

(1) lbStandard：固定 Font. Size 属性与 ItemHeight 属性之比。

(2) lbOwnerDrawFixed：可以调整 ItemHeight,并将自动调整框的高度以适应行高。

(3) LbOwnerDrawVariable：可以调整 ItemHeight 属性,需手动调整框的高度以适应行高。

Style 属性的后两种取值将受到 IntegralHeight 属性的影响,如果 ItegralHeight 属性值为 False,将不会自动调整框的高度。

3.5.2 ComboBox 组件

组合框 ComboBox 兼有 EditBox 和 ListBox 两者的功能,用户可以通过输入文本或选择列表中的项目来进行选择。组合框位于组件板 Standard 选项卡中,如图 3-29 所示。

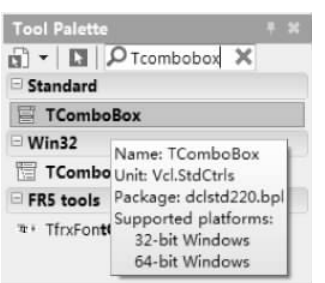

图 3-29 组合框

1. 组合框的主要属性

1) Items 属性

Items 属性是以行作为单位,列表框中选项的集合。

2) ItemsIndex 属性

ItemsIndex 属性为选项的索引值。

3) Stored 属性

Stored 属性决定选项是否排序。

4) DropDownCount 属性

DropDownCount 属性控制组合框下拉列表所能显示选项的最大个数。

5) SelText 属性

SelText 属性用来存储显示于编辑区中被选中项的内容。

6) Style 属性

Style 属性决定组合框的风格。

2. 组合框的使用

【例 3.11】 DIY 计算机的配件选择。

1) 界面设计

创建一个新的应用程序,在窗体上添加 5 个标签、4 个组合框、两个按钮及一个 GroupBox 组件,如图 3-30 所示。

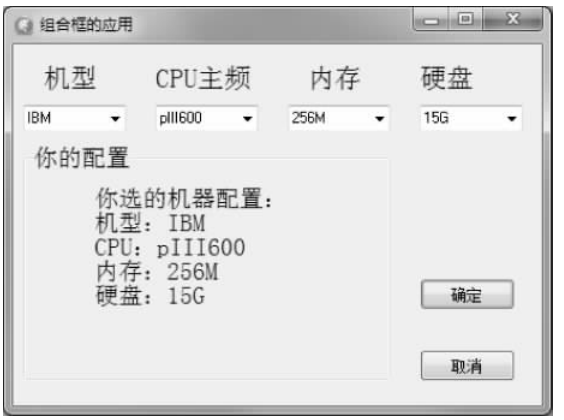

图 3-30 组合框组件应用示例

2）属性设置

设置窗体及窗体中组件的属性，具体如表 3-13 所示。

表 3-13 窗体中组件属性设置

对象	属性	属性值	说明
Form1	Caption	组合框的应用	窗口的标题
Label1	Caption	机型	标签的标题
Label2	Caption	CPU 主频	
Label3	Caption	内存	
Label4	Caption	硬盘	
Label5	Caption	（空）	
Button1	caption	确定	
Button2	caption	取消	
GroupBox1	Caption	你的配置	
ComboBox1	items	（空）	
ComboBox2	items	（空）	
ComboBox3	items	（空）	
ComboBox4	items	（空）	

3）程序设计

```
procedure TForm1.FormCreate(Sender: TObject);
begin
  combobox1.Items.Add('IBM');
  combobox1.Items.Add('AST');
  combobox1.Items.Add('Compaq');
  combobox1.Items.Add('长城');
  combobox1.Items.Add('联想');
  combobox1.Items.Add('清华同方');
  combobox2.Items.Add('586/133');
  combobox2.Items.Add('586/200');
  combobox2.Items.Add('PⅡ/233');
  combobox2.Items.Add('PⅡ/400');
  combobox2.Items.Add('PⅢ/450');
```

```
  combobox2.Items.Add('PⅢ/600');
  combobox3.Items.Add('16MB');
  combobox3.Items.Add('32MB');
  combobox3.Items.Add('64MB');
  combobox3.Items.Add('128MB');
  combobox3.Items.Add('256MB');
  combobox4.Items.Add('2.5GB');
  combobox4.Items.Add('3.2GB');
  combobox4.Items.Add('4.3GB');
  combobox4.Items.Add('9GB');
  combobox4.Items.Add('15GB');
end;

procedure TForm1.Button1Click(Sender: TObject);
var
  i: integer;p: string;
begin
  p: = '你选的机器配置: ' + #13;
  i: = combobox1.ItemIndex;
  p: = p + '机型: ' + combobox1.Items[i] + #13;
  i: = combobox2.ItemIndex;
  p: = p + 'CPU: ' + combobox2.Items[i] + #13;
  i: = combobox3.ItemIndex;
  p: = p + '内存: ' + combobox3.Items[i] + #13;
  i: = combobox4.ItemIndex;
  p: = p + '硬盘: ' + combobox4.Items[i];
  label5.caption: = p;
end;
procedure TForm1.Button2Click(Sender: TObject);
begin
  label5.Caption: = '';
  combobox1.ItemIndex: = -1;
  combobox2.ItemIndex: = -1;
  combobox3.ItemIndex: = -1;
  combobox4.ItemIndex: = -1;
end;
```

3.6 滚 动 条

在很多情况下,滚动条是自动加入的,例如前面讲的列表框,当项目显示不下时列表框将自动加上滚动条,当用户操作滚动条时,列表自动滚动。这一切都不需要编程。如果想自己操纵窗口的滚动,就要用到TScrollBar组件。当用户在滚动条上操作时,将触发OnScroll事件,这样就可以操纵滚动条。TScrollBar组件直接继承于TwinControl中,它位于Standard选项卡中,如图3-31所示

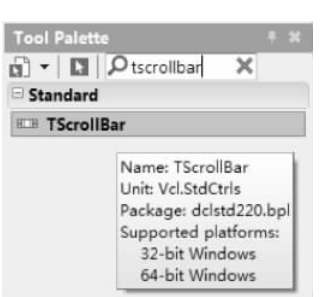

图3-31 滚动条

1. ScrollBar主要属性、方法与事件

1) LargeChange属性

当用户单击滚动条(不是滚动条两端的箭头)时,滚动条滚

动的距离是由 LargeChange 属性设置的，默认是 1。这是一个相对数。假设 LargeChange 属性设为 10，如果 Max 属性减去 Min 属性为 80，则用户只要单击 8 次就能从滚动条的一端到另一端。

2）Max、Min 属性

Max、Min 这两个属性用于设置滚动条可滚动的范围，与 LargeChange 属性和 SmallChange 属性配合来使用，可以设置用户操作滚动条时需要滚多少次才能从一端到另一端。

3）PageSize 属性

当用户按 Page Up 或 Page Down 键时，滚动条滚动的距离是由 PageSize 属性设置的，默认是 1。

4）Position 属性

Position 属性用于设置或返回滚动条中小方块的位置。可以在设计期设置 Position 属性指定小方块的初始位置，也可以在运行期修改 Position 属性使滚动条滚动。

5）SmallChange 属性

SmallChange 属性与 LargeChange 属性相似，不同的是，它是用户单击滚动条两端的箭头时滚动条的距离，默认值是 1。

6）SetPaxams 方法

该过程相当于分别设置 Position、Max 和 Min 属性。

7）OnScroll 事件

当用户操作滚动条时将发生这个事件。其中，第三个参数返回滚动条小方块的位置，第二个参数返回滚动条的状态。

2. 滚动条的使用

【例 3.12】 利用递归的方法计算 N!。

1）窗体设计

创建一个新的应用程序，在窗体上添加两个标签、一个滚动条、两个按钮及一个编辑框组件，如图 3-32 所示

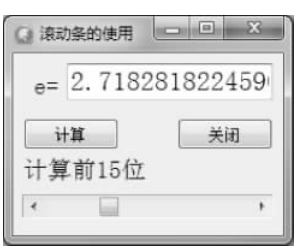

图 3-32 滚动条

2）属性设计

设置窗体及窗体中组件的属性，具体如表 3-14 所示。

表 3-14 窗体中组件属性设置

对象	属性	属性值	说明
Form1	Caption	滚动条的使用	窗口的标题
Label1	Caption	e=	
Label2	Caption	计算前 1 项	
Edit1	Text	（空）	
Button1	Caption	计算	
Button2	Caption	关闭	
Scrollbar1	Min	1	最小值
	Max	100	最大值

3）程序设计

```
function fac(n: integer): real;
```

```
begin
  if n = 0 then
    fac: = 1
  else
    fac: = n * fac(n - 1);
end;
//"计算"按钮触发的事件
procedure TForm1.Button1Click(Sender: TObject);
var
  e: real;
  n,i: integer;
begin
  e: = 0;
  n: = scrollbar1.Position;
  for i: = 0 to n - 1 do
    e: = e + 1/fac(i);
  edit1.Text: = floattostr(e);
end;
//应用滚动条,来获得当前需计算的关系式的项数
procedure TForm1.ScrollBar1Change(Sender: TObject);
begin
  label2.Caption: = format('计算前 %d 项',[scrollbar1.position]);
end;
//"关闭"按钮触发的事件
procedure TForm1.Button2Click(Sender: TObject);
begin
  close;
end;
```

3.7 计 时 器

计时器(Timer)组件可以在应用中以重复的时间间隔产生一个事件。这对不需要用户交互的代码的执行非常有用。

Timer 组件位于 System 组件板中,如图 3-33 所示,属于非可视化组件,在设计时显示为一个小时钟图标,而在运行时则不可见了,通常用来做一些后台处理。

1. Timer 组件的主要属性与事件

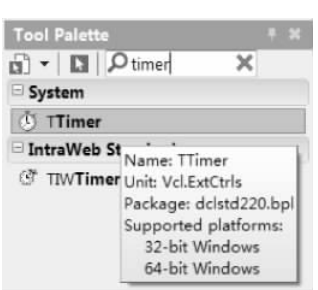

图 3-33 计时器

1) Enabled 属性

Enabled 属性为 True 时,定时器开始工作,为 False 时定时器暂停工作。

2) Interval 属性

该属性用来设置定时器触发的周期(以 ms 计)取值范围为 0～64 767。

3) OnTimer 事件

Timer 组件只提供一个事件,即 OnTimer。该事件以 Interval 属性设置的频率被触发。

2. Timer 组件的使用

【例 3.13】 利用 Timer 组件模拟数字时钟程序。

图 3-34 计时器示例

1）窗体设计

创建一个新的应用程序，在窗体上添加一个 Label 组件、一个 Timer 组件、一个 GroupBox 组件，在 GroupBox 组件上添加两个 RadioButton 组件，如图 3-34 所示。图 3-34 计时器示例。

2）属性设置

设置窗体及窗体中组件的属性，具体如表 3-15 所示。

表 3-15 窗体中组件属性设置

对象	属性	属性值	说明
Form1	Caption	模拟数字时钟	窗口的标题
Label1	Caption	（空）	未设置
	Color	clAqua	底色为绿色
GroupBox1	Caption	请选择显示格式	分组框标题
RadioButton1	Caption	12 小时制	单选按钮标题
RadioButton2	Caption	24 小时制	单选按钮标题
Timer1	Enabled	True	可以使用
	Interval	1000	1000ms 触发一次

3）程序设计

```
procedure TForm1.Timer1Timer(Sender: TObject);              //关键分析
begin
  if radiobutton1.Checked then
    label1.Caption: = formatdatetime('ampmhh: nn: ss',time)
  else
    label1.Caption: = formatdatetime('hh: nn: ss',time);
end;
procedure TForm1.RadioButton1Click(Sender: TObject);
begin
  timer1timer(sender);
end;
procedure TForm1.RadioButton2Click(Sender: TObject);
begin
  timer1timer(sender);
end;
```

4）程序分析

关键分析：本例中 OnTimer 事件根据 Timer1.Interval 的值 1000，每 1000ms 触发一次。

3.8 对话框组件

在 Delphi XE8 组件栏中的 Dialogs 页中提供了许多对话框组件，如图 3-35 所示，为用户提供了一系列标准的 Windows 对话框组件，可以使用它进行设置字体、设置颜色、设置打印机和打开或者保存文件等操作。

3.8.1 OpenDialog 组件

OpenDialog 组件用于打开一个已经存在的文件,用户选择某一文件,其所在的驱动器、文件夹、文件名以及扩展名将被赋予 OpenDialog 的 FileName 属性。OpenDialog 组件位于 Dialogs 组件板,如图 3-35 所示。

OpenDialog 组件的主要属性如下。

1. DefaultExt 属性

DefaultExt 属性用于设置系统自动附加的扩展文件名,即在用户没有设置文件类型时系统会自动附加该文件类型。

2. Filter 属性

Filter 属性用于设置可打开的文件类型。Filter 属性的设置可单击右端 ··· 按钮,打开如图 3-36 所示的 Filter Editor 对话框进行设置。

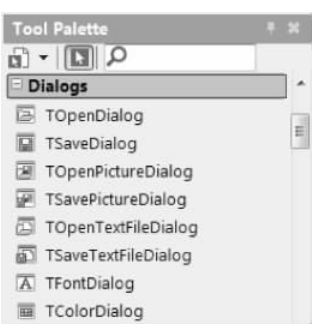

图 3-35 Dialogs 组件板

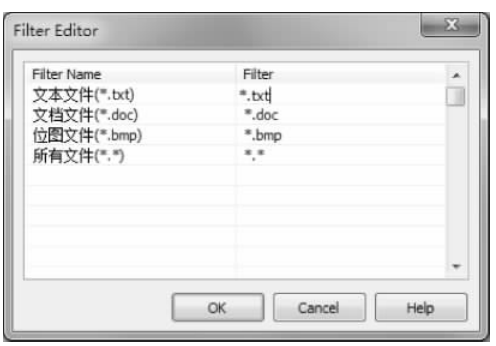

图 3-36 Filter Editor 对话框

3. FilterIndex 属性

FilterIndex 属性设置默认的 Filter 值,为 1 时则默认的文件类型为 Filter 属性中列举的第一个文件类型。

4. InitialDir 属性

InitialDir 属性设置对话框打开的初始化路径。

5. Options 属性

Options 属性用于设置对话框的作用及表现形式。包括是否可选择多个文件、是否允许长文件名、是否可以调节对话框的大小等。

3.8.2 SaveDialog 组件

SaveDialog 组件用于提供一个另存为对话框,用户输入某一文件,其所在的驱动器、文件夹、文件名以及文件扩展名将被赋予 SaveDialog 的 FileName 属性。其属性及使用同 OpenDialog 组件类似。SaveDialog 组件位于 Dialogs 组件板,如图 3-35 所示。

3.8.3 FontDialog 组件

FontDialog 组件用于提供一个字体对话框,用户可以选择需要的字体名称、样式、大小、效果及字体颜色等,这些选择将被赋予 FontDialog 的 Font 属性。FontDialog 组件位于 Dialogs

组件板，如图 3-35 所示。

3.8.4 ColorDialog 组件

ColorDialog 组件用于提供一个颜色对话框，用户可以选择需要的颜色等属性，这些选择将被赋予 ColorDialog 的 Color 属性。ColorDialog 组件位于 Dialogs 组件板，如图 3-35 所示。

3.8.5 公共对话框的使用

【例 3.14】 使用公共对话框，在标签组件中显示打开文件或保存文件的路径名和文件名，然后设置标签的字体以及背景颜色。

1）窗体设计

创建一个新的应用程序，在窗体上添加一个 Label 组件、一个 OpenDialog 组件、一个 SaveDialog 组件，一个 FontDialog 组件和一个 ColorDialog 组件，再添加 4 个 Button 组件，OpenDialog 组件和 SaveDialog 组件的 Filter 属性如图 3-36 所示，其余各个组件的属性设置如图 3-37 所示。

图 3-37 公共对话框示例

2）程序设计

```
procedure TForm1.Button1Click(Sender: TObject);
begin
  if opendialog1.Execute then                          //关键分析
    label1.Caption: = opendialog1.FileName;
end;

procedure TForm1.Button2Click(Sender: TObject);
begin
  if savedialog1.Execute then
    label1.Caption: = savedialog1.FileName;
end;

procedure TForm1.Button3Click(Sender: TObject);
begin
  if fontdialog1.Execute then
    label1.font: = fontdialog1.Font;
end;

procedure TForm1.Button4Click(Sender: TObject);
begin
  if colordialog1.Execute then
    label1.Color: = colordialog1.Color;
end;
```

3）程序分析

程序运行后运行界面如图 3-37 所示。

关键分析：Execute 为打开对话框的方法，Opendialog1 组件通过调用 Execute 方法实现打开对话框。如果打开成功返回一个 True，否则返回一个 False；其余各对话框的 Execute 方法的使用与之相同。

3.9 Win 3.1 组件

在 Delphi XE8 中,与文件有关的组件有选择驱动器、查看目录以及列举文件组件等,下面将介绍在 Win 3.1 组件板中的有关组件,如图 3-38 所示。

3.9.1 FileListBox 组件

FileListBox 组件主要用于显示指定目录的文件名滚动列表,该组件位于如图 3-38 所示 Win 3.1 组件板中。

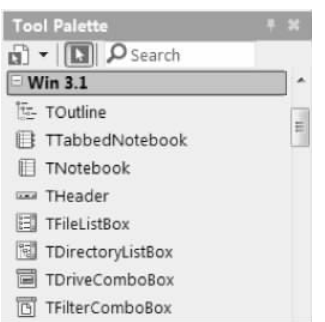

图 3-38 Win 3.1 组件板

FileListBox 组件的主要属性如下。

1. Directory 属性

Directory 属性用于设置当前文件目录,显示的文件列及表将自动更新显示文件目录的文件。

2. Drive 属性

Drive 属性用于设置当前驱动器盘的号,当前属性值改变时,Directory 属性值自动改变为新的驱动器下的当前目录。

3. ExtenderdSelect 属性

ExtenderdSelect 属性若为 True 则可按住 Ctrl 键然后用鼠标选择多个文件。若为 False 则不可以。

4. FileEdit 属性

FileEdit 属性用于将文件列表链接至一个编辑组件,显示列表中当前被选中的文件。

5. FileName 属性

FileName 属性存放了列表中当前被选中的文件的文件名及路径名。

6. FileType 属性

FileType 属性决定了文件列表中显示的文件的属性类型,如只读文件、隐藏文件、系统文件及是否显示目录等选择项。

7. Mask 属性

Mask 属性用于设置文件列表中显示的文件类型。

8. ShowGlyphs 属性

ShowGlyphs 属性用于设置文件是否在文件旁边显示文件图标。

9. MultiSelect 属性

MultiSelect 属性用于设置用户是否可以一次选中多个文件。

3.9.2 DirectoryListBox 组件

DirectoryListBox 组件用于显示指定驱动器下的目录列表,该组件位于如图 3-38 所示 Win 3.1 组件板中。

DirectoryListBox 组件的主要属性如下。

1. Directory 属性

Directory 属性用于设置当前的文件目录。

2. DirLabel 属性

DirLabel 属性用于将目录列表链接至一个 Lable 组件，显示列表中当前被选中目录。

3. Drive 属性

Drive 属性用于设置当前的驱动器盘号，当该属性值改变时，Drive 属性值将自动改变为新的驱动器下的当前目录。

4. FileList 属性

FileList 属性用于将目录列表链接至文件列表，当目录列表中的目录改变时，文件列表会自动进行更新。

3.9.3 DriveComboBox 组件

DriveComboBox 组件用于显示一个可选驱动器下拉列表，该组件位于如图 3-38 所示 Win 3.1 组件板中。

DriveComboBox 组件主要的属性如下。

1. Dirlist 属性

Dirlist 属性用于将本组件链接至目录列表，如驱动器改变，则目录列表会自动更新。

2. Drive 属性

Drive 属性用于存放当前的驱动器盘号。

3. TextCase 属性

TextCase 属性用于决定驱动器盘号使用大写字母还是小写字母。

3.9.4 FilterComboBox 组件

FilterComboBox 组件用于显示一个可选过滤器下拉列表，供用户选择，该组件位于如图 3-38 所示 Win 3.1 组件板中。FilterComboBox 组件的主要属性如下。

1. FileList 属性

FileList 属性用于将本组件链接至文件列表，如当前的文件类型改变，文件列表会自动进行更新。

2. Filer 属性

Filer 属性用于设置各种过滤文件的类型。

3. Mask 属性

Mask 属性用于存放所选的过滤类型的对应。

3.9.5 Win 3.1 组件的应用

【例 3.15】 与文件相关组件的应用示例。

1）窗体设计

创建一个新的应用程序，在窗体上添加两个 Label 组件、一个 Edit 组件、一个 FileListBox 组件，一个 DirectoryListBox 组件、一个 DriveComboBox 组件和一个 FilterComboBox 组件，如图 3-39 所示。

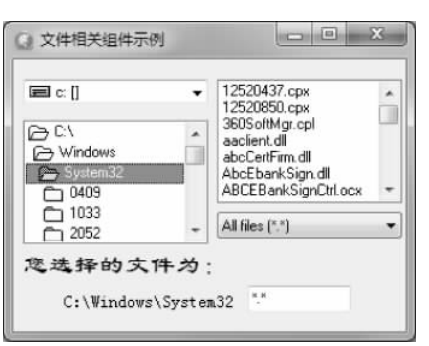

图 3-39 文件相关组件示例

2）属性设置

设置窗体及窗体中组件的属性，具体如表 3-16 所示。

表 3-16 窗体及组件属性设置表

对象	属性	属性值	说 明
Form1	Caption	文件相关组件示例	窗口的标题
FileListBox1	FileEdit	Edit1	在 Edit1 中显示所选文件
	Multimselect	True	可以多选
DirectoryListBox1	FileList	FileListBox1	使 DirectoryListBox1 和 FileListBox1 相关联
	DirLabel	Label2	Label2 中显示所选目录
DriveComboBox1	DirList	DirectoryListBox1	使 DirectoryListBox 和 DriveComboBox1 相关联
FilterComboBox	FileList	FileListBox1	使 FilterComboBox1 和 FileListBox1 相关联
Lable1	Caption	您选择的文件为:	标签标题

3.10 菜 单

一个 Windows 应用程序,往往需要制作标准的菜单界面,包括主菜单 MainMenu、弹出式菜单 PopMenu 两种,它既能体现一个系统的专业化水平,同时也为用户使用系统和操作提供了方便。因此在应用程序中制作菜单是一个基本的任务。

3.10.1 MainMenu 组件

主菜单也称为菜单栏,其中包括一个或多个选择项称为菜单项。当单击一个菜单项时,包含子菜单项的列表即被打开。主菜单位于组件板 Standard 选项卡中,如图 3-40 所示。

1. 菜单编辑器

打开一个新的窗体,在其中添加一个 MainMenu 组件,即可产生主菜单项,菜单项的设置可以通过双击 MainMenu 组件或右键单击 MainMenu 组件,在弹出的快捷菜单中选取 MenuDesigner 项,再或者选择 MainMenu 组件的 Items 属性,单击右端的 ··· 按钮,可打开菜单项编辑器,并产生一个空菜单项,如图 3-41 所示。

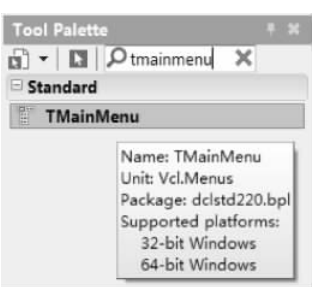

图 3-40 主菜单

图 3-41 菜单项编辑器

在对象编辑器中,Caption 属性输入“&S 设置”,表示可按 Ctrl+S 或 Alt+S 键来选择此菜单项,其中“&”符号后的第一个字符为加速字符。若输入“-”则表示建立菜单分割线将菜单项分组,如图 3-41 所示文字内容菜单项下即是。

用户可能会在已建好的菜单项中插入或删除一个选项,这时就可用鼠标选定位置,按 Insert 键则插入一个新的空白菜单项,若按 Delete 键则删除一个菜单项。

Windows 系统操作界面的菜单大都是多层的,在 Delphi XE8 中建立子菜单时,可选择要

产生子菜单的菜单项，然后按 Ctrl＋→键，便产生下一级子菜单项，如图 3-41 所示，以此类推可建立多重子菜单。

2. MenuItem 的主要属性

1）Name 属性

Name 属性用于设置菜单组件的名称。

2）Caption 属性

Caption 属性显示菜单组件的标题。

3）Checked 属性

Checked 属性设置为 True 时，相应地在菜单项边上加上选择标志“√”。属性设置为 False 时，则无显示，默认值为 False。

4）Enabled 属性

Enabled 属性默认值为 True，表示可以响应用户事件，若设置为 False，则无法响应用户事件，并且相应的菜单项会变灰。

5）Visible 属性

Visible 属性确定菜单项是否显示，True 则显示，False 则隐藏。

6）ShortCut 属性

ShortCut 属性设置该菜单项的热键。

3. MainMenu 组件的使用

【例 3.16】 使用菜单实现控制标签的显示。

1）窗体设计

创建一个新的应用程序，在窗体上添加一个 Label 组件、一个 MainMenu 组件，各个组件的属性设置如图 3-42 所示。

图 3-42 MainMenu 示例

2）程序设计

```
procedure TForm1.N1Click(Sender: TObject);
begin
  label1.Caption: = inputbox('文字输入','请输入文字内容: ','');
end;

procedure TForm1.N5Click(Sender: TObject);
begin
  label1.Font.Name: = '宋体';
end;

procedure TForm1.N6Click(Sender: TObject);
```

```
begin
  label1.Font.Name := '隶书';
end;

procedure TForm1.N7Click(Sender: TObject);
begin
  label1.Font.Name := '黑体';
end;

procedure TForm1.N8Click(Sender: TObject);
begin
  label1.Font.Name := '楷体_GB2312';
end;

procedure TForm1.N9Click(Sender: TObject);
begin
  if N9.Checked then
  begin
    N9.Checked := false;
    label1.Font.Style := label1.Font.Style - [fsbold];
end
else
begin
    N9.Checked := true;
    label1.Font.Style := label1.Font.Style + [fsbold];
end
end;

procedure TForm1.N10Click(Sender: TObject);
begin
  if N10.Checked then
  begin
    N9.Checked := false;
    label1.Font.Style := label1.Font.Style - [fsitalic];
  end
  else
  begin
    N10.Checked := true;
    label1.Font.Style := label1.Font.Style + [fsitalic];
  end
end;

procedure TForm1.N11Click(Sender: TObject);
begin
  if N11.Checked then
  begin
    N11.Checked := false;
    label1.Font.Style := label1.Font.Style - [fsunderline];
  end
  else
  begin
    N11.Checked := true;
    label1.Font.Style := label1.Font.Style + [fsunderline];
  end
end;
```

3.10.2 PopupMenu 组件

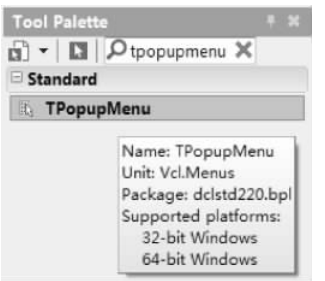

图 3-43 弹出式菜单

应用系统中，对弹出式菜单的支持是一种流行的方式，就是通常使用的右键菜单，当用户在不同的地方单击鼠标右键，就弹出不同的菜单项。主菜单位于组件板 Standard 选项卡中，如图 3-43 所示。

弹出式菜单的设计方法基本和主菜单设计方法相同，如果需要在某个组件上鼠标右键单击打开一个弹出式菜单，那么只需要将此组件的 PopupMenu 属性设置成需要打开的弹出式菜单的名字即可。

小 结

本章中介绍了 Delphi XE8 中的主要组件的属性和基本使用方法，通过一些简明的示例对它们进行了更直观的介绍。通过本章的介绍，读者对 Delphi XE8 组件有了一个初步了解，基本掌握了用一些常用组件来设置窗体并完成需要的功能。

Delphi XE8 中还有一些组件本章中没有介绍，读者可以自己查看组件面板中的组件，参照帮助或对象查看器以及有关资料学习。

习 题

3-1 简述组件的缩放、移动、复制与删除的操作步骤。

3-2 列举出可用于字符输入的各类编辑框的名称以及它们各自不同的特点。

3-3 列举出各类按钮组件的名称以及它们各自不同的特点和用途。

3-4 列举出各类列表组件的名称以及它们各自不同的特点和用途。

3-5 列举出各类布局组件的名称以及它们各自不同的特点和用途。

图 3-44 单选按钮与复选框的使用

3-6 单选按钮与复选按钮的使用。编写如图 3-44 所示界面，选择颜色可改变“示例文本”的颜色并且可以改变字体风格。

3-7 设计一个窗体，利用 ComboBox 编辑框，对客户名单进行管理，要求在编辑框中输入新的值，一旦要加入，应与下拉列表中原有值不重复。

3-8 设计一个窗体，编写文本处理程序，要求有“复制”、“剪贴”、“字体”、“字型”等按钮。

3-9 建立一个股票收益运算器，如图 3-45 所示。要求：默认类型为“基金”，在“买入股数”、“买入价”和“卖出价”中输入相应内容后，单击“计算”按钮，可在“结果”区中显示资金余额状况。

说明：基金交易费 5 元。资金余额状况公式：余额＝买入股数×买入价－((买入股数×买入价＋5)－(买入股数×卖出价－5))。

3-10 继续 3-9 题，当在“类型”框中选择不同的交易类型时，系统可进行分类计算：当买入大于卖出成交后亏损，在“结果”框中以不同的文字、背景(盈利红色、亏损绿色)明确标出，如

图 3-46 所示。

图 3-45　证券收益运算器一

图 3-46　证券收益运算器二

(1) 股票交易费用为股价的 0.35%,基金交易费为 5 元。股票印花税为股价的 0.4%。

(2) 过户费为 1 元。附加费用为 6 元。“股票”的资金余额计算公式：余额＝买入股数×买入价＋买入股数×买入价×0.35%＋过户费＋买入印花税＋附加费－(买入股数×卖出价－买入股数×卖出价×0.35%－过户费－卖出印花税－附加费)。

(3) 基金的资金余额计算公式：余额＝(买入股数×买入价＋5)－(买入股数×卖出价－5)。

3-11　菜单的快捷键和热键的区别是什么？如何设置快捷键和热键？

3-12　如何控制快捷菜单的弹出位置？

3-13　如何创建动态菜单？

3-14　MaskEdit 组件设置掩码格式的属性是哪个？如何设置类似 XH111111－X 的格式？

说明：其中 XH 为固定字母,不用输入,111111 为任意输入的 6 位数字,－为固定分隔符,X 为任意一个字母。