第 8 章　用户对象和用户事件

教学提示：PowerBuilder 提供了大量的系统控件和对象，功能强大，使用方便，但有时仍然满足不了用户的需要。因此，用户需要定义自己所需的对象，对原有的控件和对象的功能进行扩展，以提高代码的重用性、可靠性和可维护性。尽管 PowerBuilder 提供了大量的预定义事件，可以解决常见的问题，但仍然满足不了用户的实际需要，定义和使用用户事件是必不可少的。

教学要求：本章主要介绍定义用户对象的一些基本方法和技巧以及用户自定义事件的设计过程。

8.1　用户对象分类

在面向对象程序设计中，有两个重要的概念：对象和类。在 PowerBuilder 中，系统提供了很多标准的对象和类，它们有各自的功能、表现特征、属性、事件和函数，还支持自定义用户对象。

用户对象是拓展 PowerBuilder 功能的最有效方法之一。利用用户对象，我们既可以扩展系统原有对象的功能、增加新的使用方法，又能够创建出高度可重用的自定义部件，在一个或多个应用程序中反复使用，缩减开发和维护的时间，进一步提高应用程序的开发效率。同时，还可以将其他语言，如 C++开发的代码嵌入到 PowerBuilder 应用程序中。

用户对象是封装了一组相关代码和属性、完成特定功能的可重用对象。用户对象一般用于完成通用的功能。例如，应用程序可能经常使用某个【关闭】按钮执行一直组操作，之后关闭窗口；也可能经常使用某个列表框列出所有的部门；还可能对所有的数据窗口控件使用相同的错误类型检查。当应用程序需要某种反复使用的特性时，应该定义用户对象。

用户对象只需定义一次，就能够反复多次使用，并且每修改一次，就能把修改结果反映到所有使用该用户对象的地方。

在 PowerBuilder 中用户对象分为两类：可视用户对象(Visual User Object)和类(非可视)用户对象(Class User Object)。

8.1.1　可视用户对象

可视用户对象是一个可视的控件或一组完成一定功能的控件。比如可以把两个命令按钮组合在一起作为一个用户对象。这样，在应用程序使用这个用户对象时，这两个按钮总是一致的。可视用户对象共分为以下 3 类：

1) 标准可视用户对象(Standard Visual)

标准可视用户对象是对 PowerBuilder 现有控件的扩充，它在现有控件的基本功能的基础上增加应用程序需要的功能。标准可视用户对象继承了原始控件的各种特征，包括属性、

事件和函数。

2) 定制可视用户对象(Custom Visual)

定制可视用户对象是若干标准控件或已经存在的可视用户对象的组合，它把不同的部件集成在一起，每一部分都具有其原有的属性、事件，都可以在用户对象画板中编写脚本。而在使用时又作为一个整体，一旦将该对象放在窗口上之后，就不能再为其中某一组成部件的事件进行编码了。

3) 外部可视用户对象(External Visual)

外部可视用户对象实际上就是在PowerBuilder应用程序中使用其他语言(比如C或C++)编写的控件，这些可视控件存放在 DLL 文件中。通常是为了完成 PowerBuilder 本身难以完成或不支持的功能。

8.1.2　类用户对象

类用户对象是对不可视组件的逻辑和功能的封装。PowerBuilder 中有两种类型的类用户对象：标准类用户对象和定制类用户对象。类用户对象最主要的优点就是真正的不可视。所以它不消耗 GUI 资源，仅仅消耗对象和它的工作结构以及为动态创建它所需的内存。

1) 标准类用户对象(Standard Class)

标准类用户对象是对 PowerBuilder 内部的不可见对象的继承，比如 Message 对象或事务对象，并且允许用户用自己的代码扩充缺省的行为以便使它们适应用户的需要。比如，可以定义一个自己的 Error 对象，它显示的是中文的报错信息等。

2) 定制类用户对象(Custom Class)

定制类用户对象并不继承 PowerBuilder 中的任何基本对象，只通过用户自己定义的实例变量、函数以及事件来完成特定的功能。定制类用户对象只有两个系统预定义事件：Constructor 和 Destructor。

8.2　创建用户对象

创建用户对象和创建其他的 PowerBuilder 对象类似，既可以从头创建一个新的用户对象，也可以继承一个原有的用户对象。下面简要介绍如何创建新的用户对象。在 PowerBuilder 中，单击 PowerBar 工具条栏中的【New】图标，在弹出的对话框中选择【PBObject】标签页，选中要创建的用户对象的类型，单击【OK】按钮即可创建。

8.2.1　创建标准可视用户对象

单击 PowerBar 工具条栏中的【New】图标，在弹出的对话框中选择【PB Object】标签页中的【Standard Visual】项，单击【OK】按钮，如图 8.1 所示。弹出【Select Standard Visual Type】对话框，选择一种标准可视类型对象，单击【OK】按钮即可创建，如图 8.2 所示。

下面以实例来说明标准可视用户对象的创建过程。现在要创建的标准可视用户对象是一个用来关闭窗口的按钮对象 u_close，步骤如下。

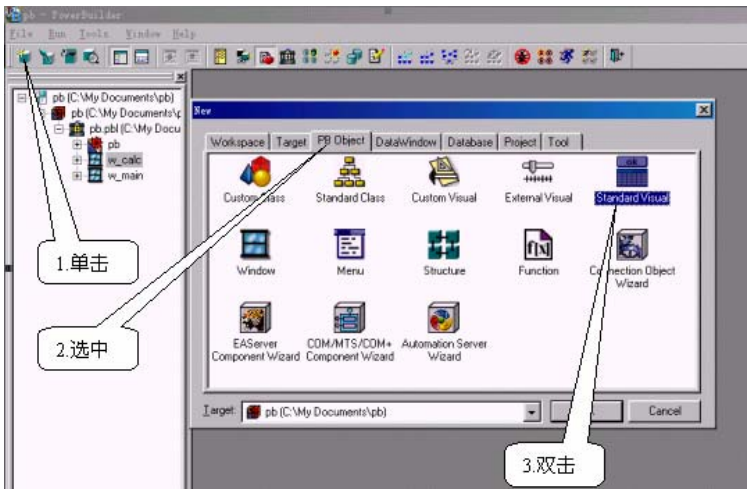

图 8.1　创建可视用户对象

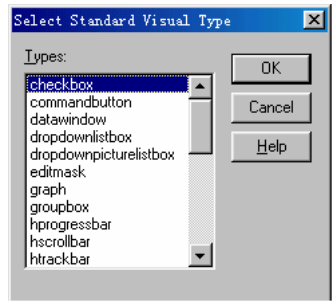

图 8.2　选择用户对象

(1) 在【Select Standard Visual Type】对话框中选择【commandbutton】选项，单击【OK】按钮，弹出 User Object 画板，此画板同窗口的编辑画板类似，只是有一部分工具不可用。

(2) 按钮的 Text 属性初始设置为“none”，在按钮上单击鼠标右键，然后选择【Properties】菜单项，打开属性对话框，将按钮的 Text 属性改为“退出”。

(3) 在按钮上单击鼠标右键，然后选择【Script】菜单项，打开脚本画板。在脚本画板中单击选择事件的下拉列表框的箭头，显示出命令按钮的预定义事件，为 Clicked 事件输入脚本。

(4) 单击 PainterBar 上的【Return】按钮返回。再单击【Save】按钮将新建的用户对象存盘，用户对象名存为 u_ close。这样一个简单的用户对象就创建完成了，如图 8.3 所示。

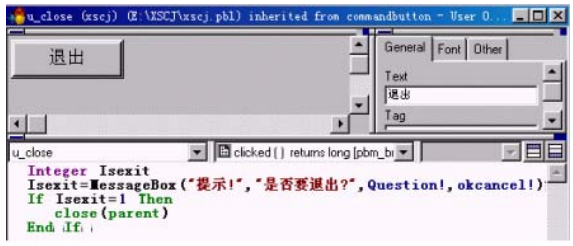

图 8.3　退出标准可视用户对象

8.2.2　创建定制可视用户对象

定制可视用户对象是将多个控件以及可视用户对象组合成一个整体，完成一定的功能和操作。编程时经常会遇到数据的录入和修改，操作都十分类似。我们可以创建一个用户对象 u_dataedit，它包括一个数据窗口控件，包括插入、删除和提交按钮各一个。我们只要给它指定一个数据窗口对象，就可以进行插入、删除和更新操作。具体操作步骤如下。

(1) 单击PowerBar工具条栏中的【New】图标，在弹出的对话框中选择【PBObject】标签页，选中【Custom Visual】项，单击【OK】按钮，打开 User Object 画板，如图 8.4 所示。

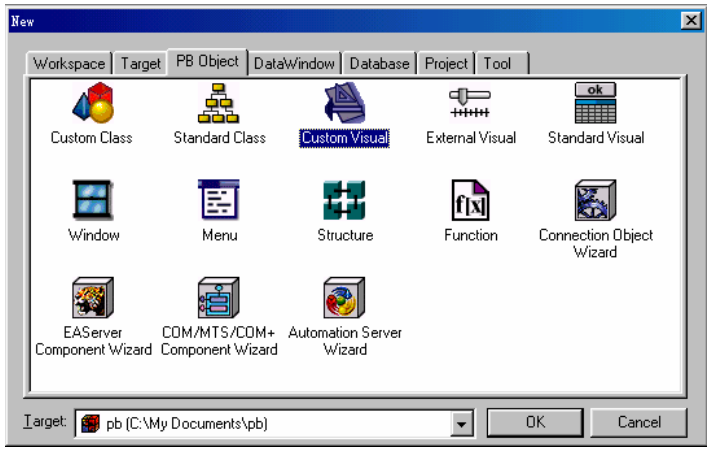

图 8.4　定制可视用户对象

(2) 在定制可视用户对象中放置控件的方法与在窗口中放置控件一样。单击窗口控件下拉按钮，在弹出的工具条中选择所需的控件和用户对象。在用户对象 u_dataedit 中放置 3 个按钮、1 个数据窗口控件，界面如图 8.5 所示。

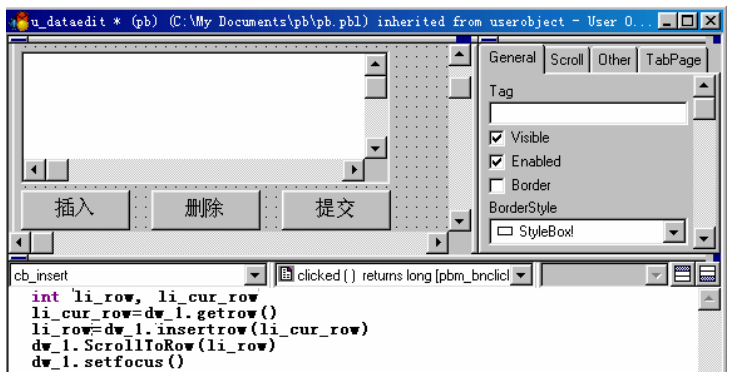

图 8.5　定制数据操作的可视用户对象

(3) 将 3 个按钮的 Text 属性分别改为“插入”、“删除”和“提交”，将 Name 属性分别改为 cb_insert、cb_delete 和 cb_save 。分别为它们的 Clicked 事件编写如下脚本。

在当前行之前插入一条新的记录 cb_insert 的 Clicked 事件的脚本：

```
int li_row, li_cur_row
li_cur_row=dw_l.GetRow( )
li_row=dw_l.InsertRow(li_cur_row)
dw_l.ScrollToRow(li_row)
dw_l.SetFocus( )
```

删除记录 cb_delete 的 Clicked 事件脚本：

```
Long ll_cur_row
ll_cur_row=dw_l.GetRow( )
dw_l.DeleteRow(ll_cur_row)
dw_l.setfocus( )
```

保存修改 cb_save 的 Clicked 事件脚本：

```
String ls_errtext
If dw_l.Update( )=l Then
  Commit;
Else
  Ls_errtext=SQLCA.SQLErrtext
  RollBack;
  MessageBox("存盘失败!",ls_errtext)
End If
```

(4) 为数据窗口 dw_l 的 Rowfocuschanged 事件输入如下脚本：

```
this.SelectRow(0,False)
this.SelectRow(currentrow,true)    //将当前行设置为选中行
```

8.2.3　创建外部可视用户对象

单击 PowerBar 工具条栏中的【New】图标，在弹出的对话框中选择【PB Object】标签页中的【External Visual】项，这将打开 User Object 设计界面。其他步骤与标准与可视用户对象相同。

8.2.4　创建标准类用户对象

类用户对象没有可视部分，它通常用于封装应用逻辑和特定功能，需要在代码中通过编写程序来使用。类用户对象有两种类型：标准类用户对象和定制类用户对象。

创建一个标准类用户对象，应单击 PowerBar 工具条栏中的【New】图标，在弹出的对话框中选择【PB Object】标签页中的【Standard Class】项，如图 8.6 所示。将打开【Select Standard Class Type】对话框，如图 8.7 所示，用户可以选择想要扩充的类用户对象。下面我们定义一个 Transaction 类型的用户对象，用于管理数据库连接。通过以下方法可以定义若干个函数用于不同的数据库事务。

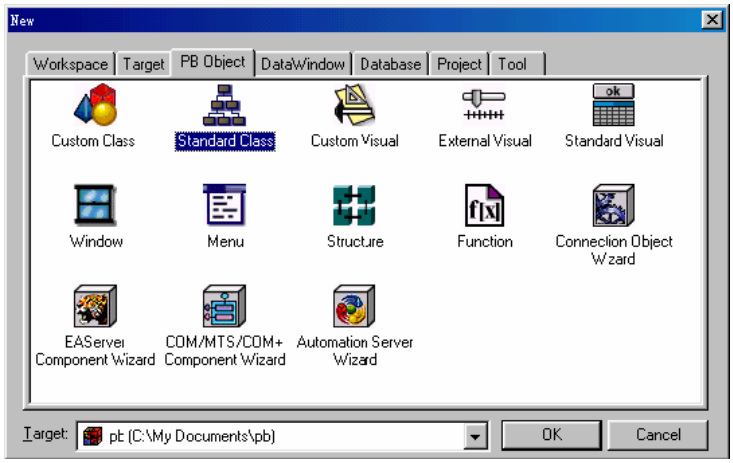

图 8.6　定义标准类用户对象

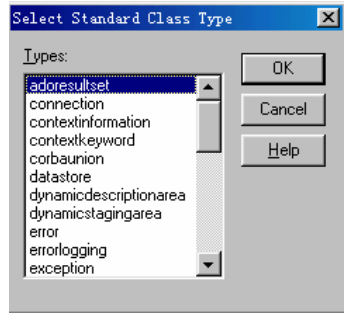

图 8.7　选择类对象类型

(1) 在【Select Standard Class Type】对话框中选择【Transaction】选项，并单击【OK】

按钮进入 User Object 画板。

(2)在函数列表框中选择【Functions】选项，新建一个函数 f_connect_anywhere 用于 Adaptive Server Anywhere 数据库建立连接，程序清单如下：

```
this.DBMS="ODBC"
this.DbParm="Connectstring="+ar_dbfile
Connect Using this;
If this.SQLCODE=-1 Then
MessageBox("连接","不能连接到 Adaptive Server Anywhere 数据库！")
End If
Return this.SQLCODE
```

这样，一个简单的标准类用户对象就定义完成了。

8.2.5　创建定制类用户对象

定制类用户对象是用户自己设计的对象，用于封装不需要可视特性的处理过程。这些对象并不继承某个 PowerBuilder 对象或控制，完全由用户通过定义实例变量、函数、事件来实现。

定制类用户对象只有两个系统预定义事件。

Constructor：发生在窗口的 Open 事件之前或用户对象被动态地放置在窗口上时。

Destructor：发生在窗口的 Close 事件之后或用户对象被动态地从窗口上取消时。

创建一个定制类用户对象，应单击 PowerBar 工具条栏中的【New】图标，在弹出的对话框中选择【PB Object】标签页中的【Custom Class】项，定制类画板与定制窗口画板或定制可视用户对象画板相似。

8.3　使用用户对象

创建了用户对象后，在应用程序中就可以随时使用它们了。可视用户对象在使用前需要放置到窗口或其他定制可视用户对象上，类用户对象则需要在代码中通过编写程序来使用。

8.3.1　使用可视用户对象

可视用户对象的使用非常简单,它的使用和窗口上其他标准控件对象的使用是相同的，通过把可视用户对象放置到窗口或其他定制可视用户对象上来使用这些对象。在学生成绩管理系统中无论哪一个窗口，都用到【退出】按钮，其功能都是退出当前窗口。为避免重复书写代码、提高程序的重用性，我们在创建窗口的时候，用前面介绍的退出标准用可视用户对象 u_close 创建每个窗口的【退出】按钮，以录入窗口为例，介绍可视用户对象的使用过程。

(1) 打开要放置可视用户对象的窗口。

(2) 从窗口画板的控件下拉列表中选择用户对象图标，如图 8.8 所示，系统打开【Select Object】对话框。

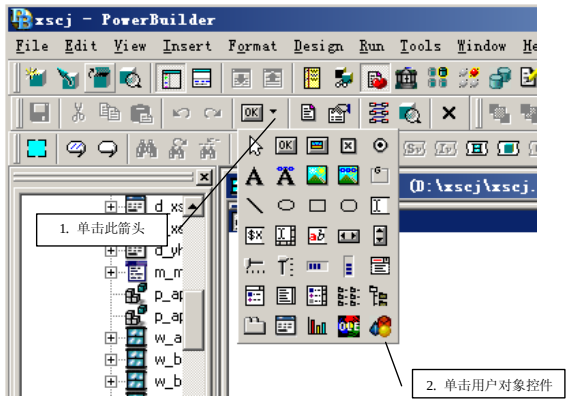

图 8.8　使用用户对象

(3) 选择要使用的用户对象，单击【OK】按钮，关闭对话框，如图 8.9 所示。

(4) 在窗口上要放置用户对象的地方单击，所选用户对象即出现在窗口上，如图 8.10 所示。

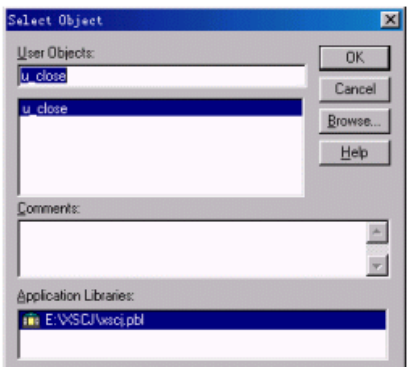

图 8.9　选择要使用的用户对象

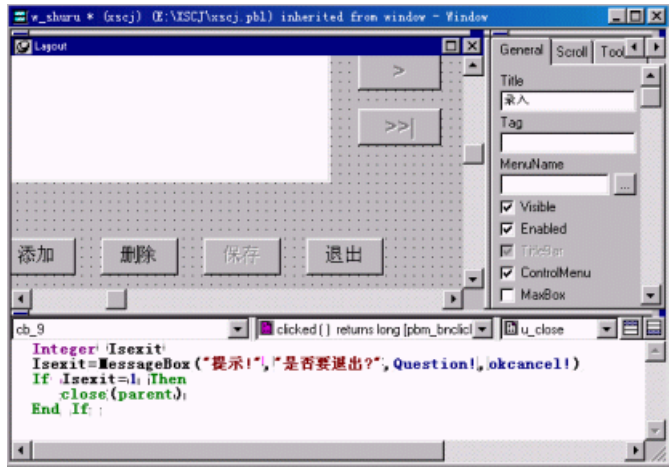

图 8.10　退出用户对象实例

在窗口或定制可视用户对象上放置了可视用户对象后，可以像使用标准控件那样为它命名、调整大小、调整位置、编写事件处理程序等。需要注意的是，在窗口画板中不能直接修改所使用的用户对象。当需要修改时，应关闭窗口画板，然后使用用户对象画板进行修改。在代码中可以使用 PowerScript 函数 OpenUserObject( )或 OpenUserObjectWithParm( )动态地往窗口上添加用户对象，也可以使用 CloseUserObject( )函数将其删除。

8.3.2　使用类用户对象

类用户对象没有可视成分，它们不能像可视用户对象那样放置到窗口上。使用类用户对象时，需要在代码中创建它的一个实例，步骤如下。

(1) 说明类用户对象类型的变量，用 CREATE 语句创建该对象的一个实例。

(2) 在变量的整个作用域中，代码都能访问该对象的属性、事件、函数，引用方法依

然使用点操作符，就像使用系统预定义对象那样(比如事务对象)。

(3) 不再使用该用户对象时，使用 DESTROY 语句删除该对象，以释放它所占用的内存。

例如，创建了一个标准类用户对象 u_ds，它从数据库存储对象继承得到，那么代码中可以这样使用 u_ds：

```
u_ds  u_ds_delete                          //说明 u_ds 类型的变量
u_ds_delete=CREATE u_ds                    //创建用户对象实例
u_ds_delete.DataObject="d_user_delete"     //将数据窗口对象与类用户对象联系起来
u_ds_delete.SetTransObject(SQLCA)          //设置类用户对象使用的事务对象
u_ds_delete.Retrieve()                     //检索数据…应用程序所需的其他处理
DESTROY  u_ds_delete                       //不用后删除该用户对象
```

8.4　用户对象示例

8.4.1　窗口与用户对象间的通讯

用户对象是一个完成特定功能的封闭组件，与结构化程序设计中的子程序类似。

窗口与可视用户对象之间交换信息的方法很多，归纳起来，比较好的通讯方式有 3 种：使用函数；使用用户事件；直接引用用户对象的属性。

例如，用户对象控件 uo_1 中放置了一个单行文本编辑框 sle_1，那么，在窗口的事件处理程序中可以给用户对象中的单行文本编辑框的 Text 属性直接赋值，表述如下。

uo_1.sle_1.text=“测试用户对象通讯”使用函数进行信息交换时，定义一些公共的(Public)用户对象级函数，利用这些函数把内部信息传递给窗口，或把窗口信息传递到用户对象内部。

使用用户事件进行信息交换时，可以为用户对象定义一些用户事件以及相应的参数，这样也可以通过编写事件处理程序、触发事件来完成用户对象与窗口之间的通讯。

8.4.2　用户对象示例

下面通过例子完整地介绍使用用户对象的全过程，以进一步加深理解。

本例使用用户对象实现数据窗口中数据的控制功能，包括数据的增加、修改、删除、存盘等，如图 8.11 所示。

图 8.11　用户对象示例

对数据窗口中数据的控制功能是比较常用的，使用用户对象来实现比较适合。下面介绍实现的方法。

(1) 建立定制的可视用户对象 uo_dwcontrol，其中包含命令按钮 cb_1(提取数据)、cb_2(插入行)、cb_3(删除行)、cb_4(存盘)、cb_5(退出)。

(2) 选择 Declare Instance Variables 标签声明实例变量如下。

```
datawindow  dw_parm    //声明数据窗口变量
```

(3) 为各个按钮添加事件代码。

①【提取数据】按钮的 Clicked 事件代码如下。

```
dw_parm.retrieve()
dw_parm.setfocus()
```

②【插入行】按钮的 Clicked 事件代码如下。

```
Long row,i
Row=dw_1.getrow()
ic=dw_parm.InsertRow(row)
if i>0 then
  dw_parm.scrolltorow(i)
  dw_parm.setrow(i)
end if
```

③【删除】按钮的 Clicked 事件代码如下。

```
dw_parm.deleterow(dw_parm.getrow())
```

④【存盘】按钮的 Clicked 事件代码如下。

```
int i
i=dw_parm.update( )
if i=1 then
  commit;
else
  rollback;
end if
```

⑤【退出】按钮的 Clicked 事件代码如下。

```
close(getparent( ).getparent( ))
```

(4) 保存用户对象 uo_dwcontrol 后退出。

(5) 建立窗口 w_uo_dwcontrol，类型为 main。

(6) 在窗口上加入用户对象 uo_1，原型为 uo_dwcontrol。

(7) 在窗口上放置数据窗口控件 dw_1，且 dw_1 的数据对象为学生基本信息。

(8) 窗口的 Open 事件脚本为：

```
dw_1.settransobject(sqlca)
uo_1.dw_parm=dw_1
```

(9) 保存所创建的窗口对象。

(10) 执行程序，单击按钮后从数据窗口中的数据变化可见按钮的功能。

说明：(1) 可视的用户对象建成后就可以像普通的按钮控件一样，加入到任意窗口界面上。用户对象一旦加入窗口，窗口中的脚本将能够直接识别用户对象的属性变量，但用户对象中的脚本只能识别其自身的实例变量和全局变量以及它所寄宿的窗口，不能识别窗口中的其他对象及其变量。因此，在构造用户对象时应通过实例变量识别窗

口中有必要进行通信的其他控件，如在实例中的 dw_parm 就是用语标志窗口中的数据窗口控件 dw_1 的。

(2) 为了在控件中取得其宿主对象，可用 getparent( )函数。

8.5 用 户 事 件

在 PowerBuilder 中，系统的运行由事件(Event)驱动。事件是指作用在对象上的动作，事件处理程序是指动作发生时相应执行的一段程序。

对于每一个事件，其处理程序可有可无；对于编写了处理程序的事件，当事件被触发时，执行其处理程序，否则什么也不执行。

PowerBuilder 对于窗口对象和窗口上的各种控件都提供了丰富的标准事件。但有时应用程序需要截获这些标准事件之外的事件，这就必须定义用户事件。当然，有时也会因为别的原因使用用户事件。

用户事件是用户给 PowerBuilder 的窗口对象或窗口上的控件对象定义的除标准事件之外的事件。这些事件可以在适当的情况下被触发，也可以由用户编程进行触发。

在下列情况下经常使用用户事件：

(1) 窗口与用户对象之间的通信。窗口和用户对象之间可以通过用户事件进行通信。

(2) 按键处理问题。应用程序需要截获除标准事件之外的操作信息。比如想对用户在一个静态文本域上的鼠标双击操作进行处理，但是一个静态文本域的标准事件中并没有 DoubleClicked 事件。在这种情况下，只能定义用户事件，使该用户事件获得双击的事件标识。接着用户就能在这个新定义的用户事件内编写代码做相应的处理了。

(3) 代码放置的合理性。比如前面介绍菜单时曾经说过，菜单是一个相当脆弱的对象，尽量不要在它上面放置过多的代码。如果要对某些菜单项做一些复杂的操作，可以在拥有这个菜单的窗口上定义一个相应的用户事件。然后只要在该菜单项的 Clicked 事件中触发窗口相应的用户事件就可以了。在这种情况下，对用户事件的定义更加简单。因为它根本不需要自动触发，所以不需要定义它的事件标识。

8.5.1 创建用户事件

事件是从属于某个对象的，因此创建用户事件的工作只能在 Windows 画板中进行。首先在 Windows 画板中选中一个对象，可以是当前打开的窗口对象或窗口上的一个控件对象。然后选择【Insert】|【Event】命令，PowerBuilder 将打开一个名为“Untitled”的窗口，用于创建一个用户事件。如果把这个窗口最大化，就可以看到结果如图 8.12 所示。

如图 8.12 所示，这个窗口和定义函数的窗口非常相似。用户可以在其中修改这个用户事件的名字、返回值类型、参数信息和权限，它们的操作界面和函数画板的窗口完全相同。

定义带参数的用户事件，其事件号必须选 None，若选择其他的事件号，则定义的参数无效。因为其他事件号的参数个数及类型都是由 PowerBuilder 确定的，用户不能修改。

在定义用户事件时，选择的事件标识应与所在对象吻合，否则用户事件不起作用。用户事件定义好后，可以像对象的其他标准事件一样编程和使用。

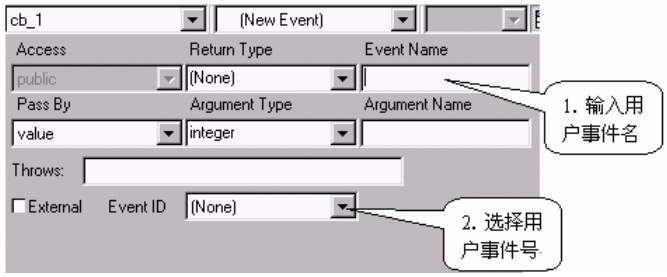

图 8.12　用户事件定义窗口

8.5.2　删除用户事件

定义的用户事件号和名称不能更改，只能删除。在该事件的代码编辑窗口，单击鼠标右键，将弹出一个菜单，选择【Delete Event】菜单，将删除该事件。要注意的是，PowerBuilder 对该删除操作不予提醒，一经选择【Delete Event】菜单，就立即删除。因此若事件已有代码，则应小心，只有用户事件才能删除。

8.5.3　触发用户事件

定义用户事件后，就需要设计事件处理程序。就像其他系统常用事件一样，没有事件处理程序，即使发生了该事件，应用程序也不做任何处理。如果使用的是 PowerBuilder 提供的标准事件标识，那么它会在该标识对应的操作进行时被触发。如果 PowerBuilder 没有使用任何事件标识，那么它永远不会自动触发，只能用户自己编程对它进行触发。

触发用户事件可以用下面两种格式：

(1) `object_name.TriggerEvent(event_name)`

(2) `object_name.PostEvent(event_name)`

其中：object_name 为对象名；event_name 为事件名，对系统事件而言，是枚举类型，如 Clicked!等，对用户事件而言，是一个字符串。

例如：

```
cb_1.TriggerEvent(Clicked!)   //触发命令按钮 cb_1 的鼠标单击事件
cb_1.TriggerEvent("u_key")   //触发命令按钮 cb_1 的用户自定义事件 u_key
```

TriggerEvent 和 PostEvent 的区别在于：TriggerEvent 立即触发该事件，PostEvent 则把事件排在事件队列的末尾。

(3) object_name.[Trigger|Post][Static|Dynamic]EVENT event_name ([para_list])

其中：

object_name 是事件所属对象的对象名。

Trigger 选项和 Post 选项只能选择一个，默认时为 Trigger。Trigger 表示立即执行指定事件的事件处理程序，然后再执行该语句后面的代码；Post 表示将该事件放置到对象的事件队列中，然后继续执行该语句后面的代码，至于发出去的事件的事件处理程序何时执行，由操作系统决定。

Static 选项和 Dynamic 选项只能选择一个，默认时为 Static。Static 表示编译时指定事

件必须存在，系统要进行返回值类型检查；Dynamic 表示编译时指定事件可以不存在，系统把返回值类型检查推迟，到应用程序运行时进行。

EVENT 是关键字，表示后面的 event_name 是个事件名而不是函数。

Para_list 是事件参数列表，有多个参数，参数之间用逗号分隔。

如果用户事件定义了参数，只能使用上述格式触发事件，而不能使用函数 TriggerEvent() 和 PostEvent()。

8.5.4　用户事件编程实例

下面以数据窗口控件为例介绍用户自定义事件的创建过程，为数据窗口定义一个 MyKey 事件，步骤如下。

(1) 在窗口中选中数据窗口控件，单击画板工具条中的【Script】按钮，打开【PowerScript】画板。

(2) 从【Select Event】下拉列表框中选择【New Event】选项。

(3) 在【Event ID】列表框中选择 Windows 消息号 pbm_dwnkey，在【Event Name】编辑框中输入用户自定义事件名，在此输入 MyKey，此事件是在相应对象中按任意键时触发，如图 8.13 所示。

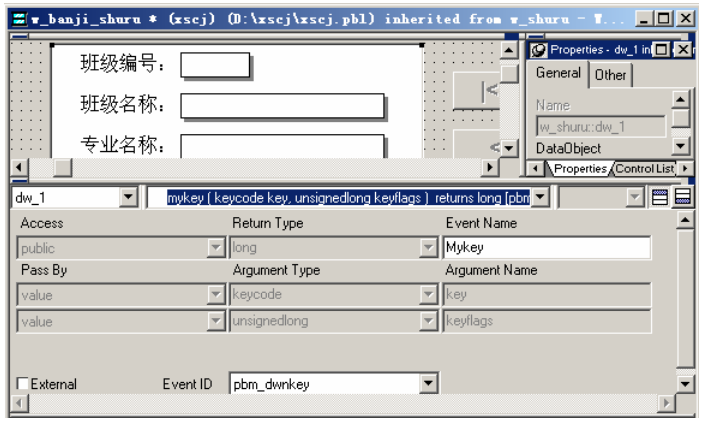

图 8.13　数据窗口的 Mykey 事件定义窗口

(4) 在窗口工作区再次选择数据窗口控件，并进入编程工作区，在事件下拉列表框会出现刚才定义的事件 MyKey。

(5) 编写 MyKey 事件处理程序：

```
sle_1.text="数据窗口的用户自定义事件 MyKey 触发"
```

(6) 运行程序，把焦点调整到数据窗口控件上，按任意键，触发其用户自定义事件，如图 8.14 所示。

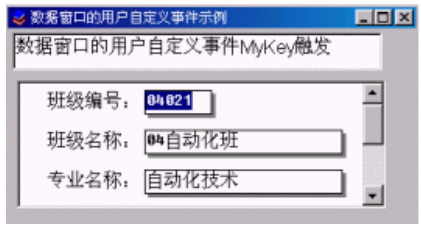

图 8.14　数据窗口的用户事件运行结果

在实际应用中，可以根据需要用此方法来定义合适的自定义事件。

在程序中触发事件：事件除了在其触发时机到来时触发外，还可以在程序中通过语句来触发。比如在窗口的 Timer 事件处理程序中触发某按钮的 Clicked 事件，如：

```
cb_enabled.triggerevent(clicked!)
```

在程序中可以通过 TriggerEvent()函数和 PostEvent()函数来触发指定对象的特定事件，并立即执行其处理程序。

格式为：

```
objectname.triggerevent(event{,word, long})
```

8.6　小　　结

PowerBuilder 提供了大量的系统控件和对象，功能强大，使用方便，但有时仍然满足不了用户的需要。因此，用户需要定义自己所需的对象，对原有的控件和对象的功能进行扩展，以提高代码的重用性、可靠性和可维护性。标准可视用户对象和标准类用户对象在 PowerBuilder 程序设计中应用极为广泛，应熟练掌握其创建过程和使用方法。

通过定义用户事件，能完成许多 PowerBuilder 所不能实现的一些特殊功能。在定义用户事件时，可以使用 PowerBuilder 的消息作为事件消息，也可以不使用任何消息，而是在代码中用函数来触发它。定义完成的用户事件可以像其他的常规事件一样使用，如果它们定义了事件消息，它们也会在达到条件时被触发。灵活地使用用户事件，可以巧妙地实现许多特殊的功能。

8.7　实　　训

实训目的

本节将练习用户对象和用户自定义事件的设计过程，以帮助学员熟练掌握其创建过程和使用方法。

实训内容

(1) 自己制作按钮控件，使之有鼠标移动触发的事件和边界属性。

(2) 创建和应用类用户对象，可以实现在 DataWindow 中检索、更新、插入和删除数据。

(3) 当焦点落在命令按钮上时，按回车键能代替鼠标。在数据窗口中，按回车键可以跳到下一个输入项，而不是下一行。当在最后一行的最后一列按回车键时，将增加一个空行。在最后一行按向下的箭头键时，也增加一个空行。

实训步骤

1. 自己制作按钮控件，使之有鼠标移动触发的事件和边界属性。

分析

按钮控件没有边界属性，也不可以改变文字和背景的颜色，因此这里使用静态文本控件，经过改造而成，可以有更加灵活的表征变化。

实训步骤

(1) 在系统主工具条中单击【User Object】图标，打开【Select User Object】对话框。单击【New】按钮，进入【New User Object】对话框。

(2) 单击选择 Visual 选项组中的【Standard】图标，然后单击【OK】按钮，系统弹出【Select Standard Visual Type】对话框。

(3) 选择【statictext】选项，单击【OK】按钮，可见新的对象原形。

(4) 双击新按钮，在跳出的【Statictext】属性对话框中选中【Enabled】复选框，然后关闭属性对话框。

(5) 在系统菜单中选择【Declare】|【User Events】命令，可见弹出的用户事件设计对话框如图所示。在【Event Name】编辑框中键入事件名称“ue_mousemove”，在消息列表中找到【pbm_mousemove】并双击之，消息名将自动跳到【Event ID】编辑框中。单击【OK】按钮返回。

(6) 右击新按钮对象方框，在弹出的选单中选择【Script】选项，打开脚本输入窗口。拉开【Select Event】下拉列表，可见在事件清单的最后出现了 ue_mousemove 事件。关闭脚本输入窗口，返回对象设计窗口。

(7) 在系统菜单中选择【File】|【Save As】命令，键入“u_commandbutton”，单击【OK】按钮。选择【File】|【Close】命令，关闭对象设计窗口。

2. 用户事件实例

分析

当焦点落在命令按钮上时，按回车键代替鼠标。在数据窗口中，按回车键可以跳到下一个输入项，而不是下一行。当在最后一行的最后一列按回车键时，将增加一个空行。在最后一行按向下的箭头键时，也增加一个空行。为实现上述功能，可用自定义事件。

实训步骤

(1) 创建一个应用程序，创建 Windows 窗口。

(2) 为增加记录的命令按钮的 Clicked 事件编写代码如下。

```
long row
row=dw_1.insertrow(0)
dw_1.setrow(row)
dw_1.scrolltorow(row)
dw_1.setfocus()
```

(3) 为增加记录的命令按钮定义一个用户事件 u_keydown，事件号为 pbm_keydown，当焦点落在该控件上时，按任意键都将触发 u_keydown。给 u_keydown 编写代码：

```
if  keydown(keyenter!) then   // 如果按了回车键，则触发 clicked 事件
this.triggerevent(clicked!)
end if
```

(4) 为插入记录的命令按钮的 Clicked 事件编写代码如下。

```
long row
row=dw_1.insertrow(dw_1.getrow())
dw_1.setrow(row)
dw_1.scrolltorow(row)
dw_1.setfocus()
```

(5) 为插入记录的命令按钮定义一个用户事件 u_keydown，事件号为 pbm_keydown，当焦点落在该控件上时，按任意键都将触发 u_keydown。给 u_keydown 编写代码如下。

```
if  keydown(keyenter!) then   // 如果按了回车键，则触发 clicked 事件
cb_insert.event  clicked()
end if
```

(6) 为删除记录的命令按钮的 Clicked 事件编写代码如下。

```
dw_1.deleterow(dw_1.getrow())
```

(7) 为删除记录的命令按钮定义一个用户事件 u_keydown，事件号为 pbm_keydown，当焦点落在该控件上时，按任意键都将触发 u_keydown。给 u_keydown 编写代码如下。

```
if  keydown(keyenter!) then   // 如果按了回车键，则触发 clicked 事件
this.event  clicked()
end if
```

(8) 为显示记录的命令按钮的 Clicked 事件编写代码如下。

```
dw_1.retrieve()
```

(9) 为显示记录的命令按钮定义一个用户事件 u_keydown，事件号为 pbm_keydown，当焦点落在该控件上时，按任意键都将触发 u_keydown。给 u_keydown 编写代码如下。

```
if  keydown(keyenter!) then   // 如果按了回车键，则触发 clicked 事件
this.triggerevent(clicked!)
end if
```

(10) 为存盘的命令按钮的 Clicked 事件编写代码如下。

```
dw_1.update()
```

(11) 为存盘的命令按钮定义一个用户事件 u_keydown，事件号为 pbm_keydown，当焦

点落在该控件上时，按任意键都将触发 u_keydown。给 u_keydown 编写代码如下。

```
if  keydown(keyenter!) then   // 如果按了回车键，则触发 clicked 事件
this.triggerevent(clicked!)
end if
```

(12) 为返回的命令按钮的 Clicked 事件编写代码如下。

```
close(parent)
```

(13) 为返回的命令按钮定义一个用户事件 u_keydown，事件号为 pbm_keydown，当焦点落在该控件上时，按任意键都将触发 u_keydown。给 u_keydown 编写代码如下。

```
if  keydown(keyenter!) then   // 如果按了回车键，则触发 clicked 事件
this.triggerevent(clicked!)
end if
```

(14) 为数据窗口定义一个用户事件 u_keyenter，事件号为 pbm_dwnprocessenter，当焦点落在该控件上时，按回车键都将触发 u_keyenter。给 u_keyentern 编写代码如下。

```
int col
int row
col=getcolumn()
row=getrow()
if  col<4  then                 // 当前列不是最后一列
     setcolumn(col+1)           //  将下一列变为当前列
else
   if  row<rowcount()  then     //当前列是最后一列，但当前行不是最后一行
       setrow(row+1)            //将下一行的第一列变为当前列
       scrolltorow(row+1)
       setcolumn(1)
   else                         //当前列是最后一列且当前行是最后一行
       row=insertrow(0)         //增加一行
       setrow(row)              //将新行的第一列变为当前列
       scrolltorow(row)
       setcolumn(1)
   end  if
end  if
return  1                       //放弃系统原来的操作
```

(15) 为数据窗口再定义一个用户事件 u_keyarrow，事件号为 pbm_dwntabdownout，当焦点落在该控件上时，按向下的箭头键触发 u_keyarrow。给 u_keyarrow 编写代码如下。

```
long row
row=insertrow(0)                //增加一行
setrow(row)                     //将新行变为当前行
scrolltorow(row)
```

通过了本节的上机实验，学员应该能够掌握创建用户对象和用户事件的过程，并灵活地应用它们。

8.8 习　　题

1. 用户对象有哪几种？如何创建与使用？
2. 为什么要使用用户事件？怎样创建和使用用户事件？