应用API函数提高PowerBuilder程序的友好性

朱 霞 赵永强 (长沙 中南大学工商管理学院 410083)

摘要：本文提出应用API函数实现图形按钮、图形菜单、通用窗口状态栏等制作技术，用来提高PowerBuilder应用程序的友好和美观性，使PowerBuilder应用程序更专业化。

关键词：PB API 友好性

PowerBuilder是当今流行的数据库前端开发工具，具有灵活、简捷、高效等特点。但与VB、Delphi、C++Builder等相比，它开发出的应用程序在友好、美观方面明显逊色。因此，本文提出应用API函数实现图形按钮、图形菜单、通用窗口状态栏等制作技术。

1 图形按钮的制作

PB提供的图形按钮功能很不完善，例如图形占据整个按钮，不是显示在一侧，程序运行时没有提示信息等。为了使PB编出的程序更具专业化，有必要对其进行功能扩充。这里设计的图形按钮除了具有普通按钮的功能外，还具有显示提示信息(Tip)、按钮局部区域显示图标、动态改变按钮上的图形等功能。如图1所示图形按钮样式。

图 1 图形按钮

1.1 设计思路

为了方便使用图形按钮，将其封装成用户自定义对象，可以在窗口中随意调用。在普通按钮上放置一个位图于一侧，设计难点在于当按钮上的位图被按动时，需要触发按钮被按动的操作，这需要利用WindowsAPI函数SetCapture()来捕获鼠标，并指定到按钮上，利用ReleaseCapture()来释放鼠标捕获，同时利用一个静态文本框来实现浮动提示(Tip)。

1.2 设计过程

首先创建一个custom可视用户对象，在其中放置一个CommandButton控件(cb_1，text不设)和一个Picture控件(p_1，图片不设)，并且将图片控件放在按钮控件一侧。

(1) 在此用户对象中声明一个实例变量：Boolean capturemouse = false

(2) 在local External Functions…中声明两个WindowsAPI函数：

```
Function ulong SetCapture(ulong hWnd) Library "USER32.DLL" // 功能：捕获鼠标
Function BOOLEAN ReleaseCapture() Library "USER32.DLL" // 功能：释放鼠标
```

(3) 在此用户对象的User Event中加入mousedown事件(pbm_lbuttondown)，mouseup事件(pbm_lbuttonup)，mousemove事件(pbm_mousemove)，clicked事件(pbm_bnclicked)。在constructor事件中写脚本如下：

```
cb_1.x = 0    //保证cb_1与窗口上的用户对象大小相等
cb_1.y = 0
cb_1.width = This.width
cb_1.height = This.height
p_1.x = This.width/14
p_1.y = This.height/ 6
If This.Enabled = False then
  cb_1.Enabled = False
  p_1.Enabled = False
End if
```

(4)在cb_1按钮控件自定义事件中加入mousedown、mouseup、mousemove事件。mousedown事件中脚本为：parent.postevent ('mousedown') //激活用户对象的相应事件并在mouseup事件、clicked事件中写入类似脚本。mousemove事件中脚本为：

```
if xpos < p_1.x or xpos > (p_1.x + p_1.width) or ypos < p_1.y &
or ypos > (p_1.y + p_1.height) then  //当鼠标离开图片控件时
```

```
        if capturemouse = true then    // 当捕获鼠标标志为true时
            ReleaseCapture()    // 调用WindowAPI函数释放鼠标捕获
            capturemouse = false   // 设置捕获鼠标标志为false
        end if
end if
        parent.postevent ('mousemove') // 激活用户对象的mousemove事件
```

(5)在图片控件中加入mousemove事件。在其中写入如下脚本：

```
        capturemouse = true // 当鼠标移入图片控件，设置捕获鼠标标志为true
        SetCapture(handle(cb_1))// 当鼠标移入图片控件，调用函数捕获鼠标并指定到按钮
```

1.3 使用图形按钮控件并添加提示信息

新建一窗口，在窗口中放置一个图形按钮用户对象(uo_1，tag=' 修改 '，前面留几个空格为了不让位图遮住按钮上的文字)和一个静态文本框(st_1，visible=false)

(1)在此窗口中声明一个实例变量:string floatmessage // 存放浮动提示信息

(2)在窗口函数 Windows Functions…中建立一个showmessage(integer currentx, integer currenty)函数，返回None。在函数中写入如下脚本：

```
    if st_1.visible=false then
      st_1.x=currentx  // 确定浮动提示的显示位置
      st_1.y=currenty
      st_1.width=len(floatmessage)*38  // 确定浮动提示的长度
      st_1.text=floatmessage // 确定浮动提示的内容
      st_1.visible=true  // 显示浮动提示信息
end if
```

(3)在窗口的 mousemove 事件中写入如下脚本：

```
st_1.visible=false
```

(4)在 uo_1 的 constructor 事件中写入如下脚本：

```
    this.cb_1.text=this.tag
    this.p_1.picturename="edit1.bmp"
```

在 uo_1 的 mousedown、mouseup 事件中分别设置不同的图片，在 uo_1 的 mousemove 事件中确定 floatmessage 的内容、显示位置，调用 showmessage()函数即可。

至此完成图形按钮制作。用户可以灵活采用上述方法设计出各类图形按钮，如动态改变图形按钮底色、增强按钮的立体感等，这大大美化了 PB 程序界面。

2 图形菜单的制作

用过Office的用户一定熟悉其菜单形式，一些常用或重要的菜单项前有一个图标，使得用户操作界面美观、明了。其实在 PB 中利用 API 函数也能制作图形菜单，如图 2 所示。

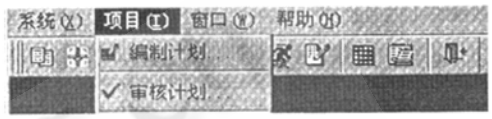

图 2 图形菜单

设计方法如下： 新建一个 MDI 窗口和一个菜单，并将菜单挂在窗口上。在窗口的local External Functions…中声明五个 WindowsAPI 函数：

```
    FUNCTION ulong LoadImageA(ulong hintance, string filename,uint utype,int x,int y,uint fload) LIBRARY "USER32.DLL" // 加载图片
    FUNCTION Boolean SetMenuItemBitmaps(ulong hmenu,uint upos,uint flags,ulong handle_bm1,ulong handle_bm2) LIBRARY "USER32.DLL" // 放置图片
    FUNCTION int GetSystemMetrics( int nIndex ) LIBRARY "USER32.DLL"
    FUNCTION int GetSubMenu(ulong hMenu,int pos) LIBRARY "USER32.DLL" // 取子菜单句柄
    FUNCTION ulong GetMenu(ulong hWindow) LIBRARY "USER32.DLL"// 取菜单句柄
```

在窗口的 open 事件中调用函数，关键脚本如下：

```
    Long ll_MainHandle,ll_SubMenuHandle // 菜单的句柄
    long ll_X,ll_Y,ll_BitmapHandleA,ll_BitmapHandleB // Win32 常量
    Integer IMAGE_BITMAP= 0, LR_LOADFROMFILE = 16, SM_CXMENUCHECK = 71
        Integer SM_CYMENUCHECK = 71, MF_BYPOSITION = 1024
    ll_MainHandle = GetMenu(Handle(this)) //取得菜单的句柄
    ll_SubMenuHandle = GetSubMenu(ll_MainHandle,1) // 取得第二项下级菜单的句柄
    ll_x = GetSystemMetrics(SM_CXMENUCHECK) //图片位置
```

```
ll_y = GetSystemMetrics(SM_CYMENUCHECK)
ll_BitmapHandleA=LoadImageA(0,'jh1.bmp',IMAGE_BITMAP,ll_x,ll_y,LR_LOADFROMFILE)
ll_BitmapHandleB=LoadImageA(0,'jh2.bmp',IMAGE_BITMAP,ll_x,ll_y,LR_LOADFROMFILE)
SetMenuItemBitmaps(ll_SubMenuHandle,0,MF_BYPOSITION,ll_BitmapHandleA,ll_BitmapHandleB)
SetMenuItemBitmaps(ll_SubMenuHandle,2,MF_BYPOSITION,ll_BitmapHandleB,ll_BitmapHandleA)
```

3 通用窗口状态栏

程序中使用状态栏能及时为用户提供有用的帮助信息，而PB中只有MDI Frame with Microhelp风格的窗口才有状态栏，但不能分栏显示信息，并且其他类型的窗口没有提供状态栏。这里提供一种任意类型窗口都可以采用的状态栏设置方法，效果如图3所示：

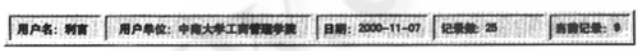

图 3 通用窗口状态栏

为了方便调用通用窗口状态栏，将其封装成用户自定义对象。此状态栏可以实现打开窗口后，动态实时更新多条帮助信息，随窗口大小的变化自适应调整长度，这就提高了程序的人机交互性。

3.1 创建过程

创建一个Visual External类型的用户对象。采用comctl32.dll动态连接库和msctls_statusbar32类，在local External Functions…中声明两个WindowsAPI函数：

```
function long SendMessageLong( long hWindow, uint Msg, ulong wParam, Ref long Parts[] ) Library 'user32' alias for 'SendMessageA'
function long SendMessageString( long hWindow, uint Msg, ulong wParam, Ref string szText ) Library 'user32' alias for 'SendMessageA'
```

定义两个User Object Function：

(1)of_setparts(long alparts []) returns Boolean，设置状态栏的分栏数目及每栏范围，脚本如下：

```
long  llCountParts
boolean    lbSetParts = FALSE
llCountParts = UpperBound( alParts [] )
IF llCountParts > 0 THEN
lbSetParts=(SendMessageLong(Handle(This),SB_SETPARTS,llCountParts,&
alParts [])<>0)
END IF
RETURN lbSetParts
```

(2) of_settext(readonly integer aipart,readonly integer aistyle,string astext) returns Boolean，设置每栏的帮助信息，脚本如下：

```
RETURN(SendMessageString(Handle(This),SB_SETTEXT,aiPart + aiStyle,asText )<> 0)
```

3.2 应用技巧

新建一窗口，放置一个状态栏用户对象(uo_1)于窗口底部。在窗口的open事件中设置帮助信息条数、长度、内容等参数，关键脚本如下：

```
long  llParts []   //用来设置状态栏每栏的长度
llParts [1] = 400
llParts [2] =-1   // -1表示分栏结束，用户可以自定义设置分栏数目及长度
uo_1.of_SetParts( llParts )
// 从0开始设置帮助信息内容
uo_1.of_SetText( 0, 0, " 记录数:"+string(dw_1.rowcount() ))
uo_1.of_SetText( 1, 0, " 当前记录:"+string(dw_1.getrow()))
```

在窗口的resize事件中设置状态栏能随窗口大小的变化自适应调整长度，脚本如下：

```
uo_1.resize(-1,-1)
return 0
```

在其他想要更新帮助信息的事件中，重新设定参数即可实现动态更新多条帮助信息。

4 结束语

以上应用技术在Windows9x/NT、PowerBuilder6.5环境下运行通过，我们在MIS开发中应用了本文中的技术后，用户反映很好。用户界面对整个系统的应用是至关重要的，因为它是人机对话的重要组成部分。应用程序的友好性不佳，软件系统就难以发挥应有的效益。尤其随着软件日益广泛的应用，完善软件功能的同时追求界面友好、美观，这已是不可避免的趋势。API函数的应用还有许多，这里仅抛砖引玉，有待进一步探讨。■