在PB环境利用面向对象技术改善软件复用性

朱 江 彭容修 (武汉华中科技大学电信系 430074)

摘 要 本文阐述了在PowerBuilder集成开发环境(IDE)下,开发软件系统中用建立系统基础类库的方式来协调和规划软件开发的问题。主要阐述了两个问题:(1) 用户自定义的系统基础类库做为系统开发的基础。(2) 在建立用户自定义的系统基础类库的同时,建立相应成员函数进行功能注册的方式实现对象的多态性。

关键词 面向对象 PowerBuilder 项目管理 中间层

1 问题的提出

在用PowerBuilder开发一些比较大型的软件系统,比如企业级MIS、MPRII系统时候,包含比较多的子系统,需要多名程序开发人员在系统分析员的协调下,不同的人员负责不同的子系统,存在项目的统筹和协调问题,实际开发过程会遇到如下一些情况:

(1) 在相对紧凑的子系统之中,程序员在开发具体模块时,所有的窗口和控件对象直接使用系统的窗口和对象。由于系统的基础类对象不能修改,结果是为保证程序具有风格一致的界面,如按钮控件具有相同的外观、相同的字体,窗口有相同的外观等,需要做许多重复的修改工作,这一点在开发后阶段调整时表现得尤为突出。

(2) 在相对独立的子系统之间,不同系统自己定义自己的对象(比如数据窗口对象、窗口对象),包括外观和在其中扩展了各自子系统需要的功能,这样每个子系统都存在功能和代码的交叉,导致了代码冗余,集成时系统的界面风格最后不能很好的统一,需要调整,也涉及很大的工作量。

(3) 如果同时一团体开发的多个类似项目,多个项目间没有连续性,上一个项目完成之后,在进行类似的下一个项目时,很多与业务无关的代码又开始重新编写,简单的例子就是数据库系统中最常用的对记录进行维护的窗口,需要完成记录的添加、删除、修改、显示上一条记录、显示下一条记录等等,类似的代码一次一次的被重复编写,效率较低。

作者在参与这样一些系统开发过程中时,用系统基础类库(可以看为PowerBuilder基本对象类和应用程序的中间层)作为各子系统开发基础的方式,提高了编程管理的有效性,很好的解决了这些问题。同时,由于系统基础类库具有业务无关性,使得二次开发的效果也很好。

2 程序处理的要点

2.1 建立自定义的基础类库作为系统的基础

在PowerBuilder中建立完整的自定义对象的基础类库,包括窗口对象及系统所有可能用到控件对象。一般而言包括以下几个库:

(1) 标准的可视用户对象库SysBaseObject.pbl:

系统基本的对象控件库,其中包括一些基本的常用的控件对象:DataWindow对象、EditMask对象、ListBox对象、ListView对象、MultiLineEdit对象、Picture对象PictureBotton对象、PictureListBox对象、RadioButton对象、SingleLineEdit对象、StaticText对象、Tab对象、TreeView对象等等,每个子系统使用的对象类必须是其子孙。例如系统所用的数据窗口必须是由这个库中的数据窗口对象继承。

(2) 系统窗口基类库SysBaseWindow.pbl:

系统的基本窗口对象库,包含一些可以完成特殊功能的窗口对象，其中 w_BaseWindow 是系统所有窗口的基类,系统中每个窗口都必须是其子孙,那么系统中不同类型的窗口，比如响应窗口 w_BaseRespondWindow，w_BaseAboutWindow 窗口，完成数据维护功能的窗口 w_BaseWhWindow 等等，都可以在这里建立祖先，它们也由 w_BaseWindows 继承而来。

(3) 自定义的用户可视对象库 SysUserObject.pbl:

里面包含一些比如分隔条、进度条、图形按钮、日历等常用的用户对象。这可以根据具体的项目进行调整。

建立好这些库之后，一个原则就是：把这些基础类库作为各个子系统的基础加到各系统中去,程序员在开发具体的窗口模块时,所用到的对象都由这四个库中的对象继承而来(各子系统特殊对象除外)，同时对象共同的功能和特性，全部都编写到祖先对象中，由各子孙继承。

当系统要做一些或大或小的调整时，比如所有的窗口图标改变、改变系统中按钮的风格时，直接可以在用户自定义的基础类库(中间层)中完成，而不会因为 PowerBuilder 提供的基类不能修改导致大量的逐个修改工作。

但当把许多功能和特性集中到祖先类中时,如何让其子孙有选择的继承这些特性,同时避免复杂紊乱的继承关系，这是用 PowerBuilder 开发项目时一个比较实际问题，与SYSBASE公司提供的系列PFC类库相类似，通过建立成员函数进行功能注册的方式可以很好的解决这个问题。

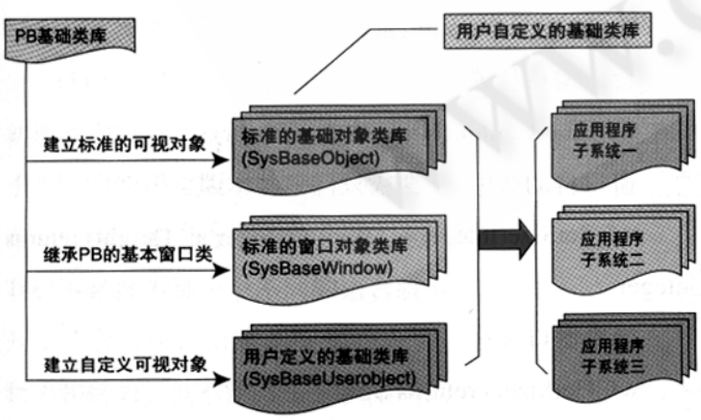

图 1　系统模块构成结构图

2.2 建立成员函数进行功能注册实现对象的多态性

有点PowerBuilder开发经验的人都知道，Freeform格式的数据窗口一般可以用来录入数据,要在其中的各个字段切换默认是按下Tab键，而按下回车键，数据窗口是向后滚动一条记录，这不太符合输入者的习惯。一般而言，开发中我们都需要在数据窗口的Key事件中编写相应的代码，使得输入者按下回车键后进行字段切换，方便输入。这种情况很常见,没有必要给每个数据窗口都编写相同的代码,处理的方式是在建立祖先数据窗口,把代码编写在祖先中。但又存在一个问题,我们只需要完成输入功能的Freeform形式的数据窗口才具备这种功能,其他情况比如浏览数据用的表格形式的数据窗口或图表形式的数据窗口等等则不需要。虽然说给不同用途的数据窗口建立相应的祖先层次,使用的时候有选择的继承可以解决问题,但是系统稍微复杂的话,这种对象的继承有时候会有较多的层次，让程序员摸不着头脑，而且 PB 不允许一个对象同时继承不同的祖先,且不能动态的为对象选择祖先,一旦需要改变祖先，又要重新再设计一次。

结合上面提到的建立用户自定义的基础类库的方式,把相同对象的共同对象尽量写在一个祖先对象中,然后用建立相应成员函数进行注册功能的方式实现对象的多态性，实践证明是一个比较好的处理办法。

下面以建立完成ENTER按键到TAB按键的映射的代码为例说明这种思想。

例：完成 ENTER 按键到 TAB 按键的映射功能

(1) 建立祖先的数据窗口(如果已经有祖先对象,跳过这一步)。打开对象画板，按下 [New] 按钮，在New User Object 对话框中选择 Visual Standard(标准可视对象)，建立一个 uo_dw 用户对象(即祖先数据窗口)，保存到 SysBaseObject.pbl 库中。

在对象画板为祖先的数据窗口中按如下步骤建立代码：

(2) 建立一个布尔型实例变量ib_EnterToTab，初始化值为 False。

在 [Decleae] 菜单中选择 [Instance Variables...]，

PRIVATE：

Boolean ib_EnterToTab=False

(3) 添加用户自定义成员函数 of_SetEnterToTab (Boolean ab_Switch) Return Integer

加入如下代码：

```
/*******************
**<DataWindow Name>.of_SetEnterToTab (ab_EnterToTab)
** [参数] Boolean ab_EnterToTab
** [返回值] Integer -1:失败 1：成功
```

```
**********/
if ISNULL (ab_entertotab) Then Return -1
ib_EnterToTab=ab_EnterToTab
Return 1
```

(4) 添加用户自定义事件 dwKeyEnter，Event ID 为：pbm_dwnprocessenter，在事件中建立下面的代码：

```
integer VK_TAB=09
//如果用户建立成员函数注册了该功能，则执行相应处理，否则不与理会
if ib_EnterToTab then
// 模拟 Tab 按键的按下和弹起
keybd_event(VK_TAB,0,0,0)
keybd_event(VK_TAB,0,2,0)
// 终止数据窗口对回车键的处理
return 1
end if
```

(其中keybd_event是Windows API函数，需要声明，在[Decleae] 菜单中选择 [Local Extenal Funtions...]，输入如下API声明：Subroutine keybd_event(int bVk,int bScan,ulong dwFlags,ulong dwExtraInfo) LIBRARY "user32.dll")

这样在祖先的数据窗口中已经完成了 Enter 按键到 Tab 按键的影射功能，这段代码是否执行，则要看私有实例变量的值，而改变私有实例变量的值则通过成员函数 of_SetEnterToTab()来完成。系统使用数据窗口时由该祖先继承而来，如果需要Enter按键到Tab按键的影射功能，可以在窗口Open 事件或数据窗口对象的Constructor事件或任何时候通过代码 of_SetEnterToTab(TRUE)、of_SetEnterToTab(FALSE)打开、关闭该功能。

当需要把许多常用的功能集中到祖先中去时，这种方式的灵活性就体现出来了。以数据窗口为例，开发过程中通常为数据窗口扩展的一些特性有：

· 是否让DataWindow 自动处理数据库错误信息

· 点击标题栏自动排序

· 完成 ENTER 按键到 TAB 按键的映射

· 设备数据窗口是否屏蔽上下键滚动记录

· 设置数据窗口在Retrieve之后是否滚回到Retrieve之前的记录

在祖先数据窗口中，每添加一个功能，就对应的建立一个注册该功能的成员函数，集成以上功能特性，则需要相应建立如下成员函数：

of_DbError(Boolean ab_dbError) Return Integer

是否让 DataWindow 自动处理数据库错误信息

of_SetClickHeaderOrder(Boolean ab_Switch) Return Integer

是否点击标题栏自动排序

of-SetEnterToTab(Boolean ab_Switch) Return Integer

是否完成 ENTER 按键到 TAB 按键的映射

of_RetrieveScrollBack(BoolEan ab_Switch) Return Integer

是否设置数据窗口在 Retrieve 之后是否滚回到 Retrieve 之前的记录

of_ShieldUpDownArrow (Boolean ab_ShieldUpDownArrow) Return Integer

设置数据窗口是否屏蔽上下键滚动记录

同样对于窗口，以下是PowerBuilder窗口类没有但是在实际开发中又经常用到的特性，包括：设置窗口顶层显示，响应ESCAPE按键关闭窗口，自动调整窗口内的对象位置以适应不同的窗口大小和不同的分辨率，以达到最佳显示效果等等，可以在 SysBaseWindow 库为祖先窗口 w_BaseWindow 中建立相应的功能的成员函数做相应的处理：

wf_SetAlwaysOnTop(boolean ab_alwaysontop) returns integer

设置窗口顶层显示

wf_SetEscCloseWindow(boolean ab_escclosewindow) returns integer

响应 ESCAPE 按键关闭窗口

wf_SetWindowPosition (integer ai_XPosition,integer ai_YPosition) returns integer

设定窗口位置

wf_AutoCenter (boolean ab_Center)

窗口自动居中

wf_SetSize(integer ai_Width,integer ai_Height) returns integer

设置窗口大小

wf_GetSize() returns String

获得当前窗口大小，以像素值返回

当系统中所有的数据窗口和窗口都由该祖先继承过来，需要以上的任何一个特性，便可以通过相应的成员函数进行注册，另外 PowerScript 还支持函数的重载，可以建立多种参数形式的成员函数注册同一个功能。

有了上面这些与业务相关的设计作为基础，系统分析

(下转第 47 页)

(上接第67页)

员就可以根据具体的项目分析和设计好这些中间层的各种各样的子孙类，它们可能与业务相关也可能与业务无关。常见的比如数据维护窗口类，能对窗口内的数据窗口对象容器中的记录自动维护，为其编写大量的数据处理的成员函数进行相应的数据处理。程序员在使用这些类编程的时候，就可以充分利用这些类提供的大量功能进行设计，而把编程的精力尽量放在那些与业务密切相关的问题上来。

3 结束语

上面是作者用PowerBuilder开发一些大项目时在编程控制方面的一些体会。当然，上面只是举了一个简单的例子进行说明了这种处理的方式，不是特别复杂但又是一个很实际的问题。在实际的开发过程中，可以根据项目的具体情况设计好这些基础类库，这样做明显的一个好处是，系统分析员在整个开发阶段可以对基础类库不停的进行修改改进，以丰富和改进程序的功能。通过这些自定义的接口(可以称之为中间层)，使得我们能非常简单而有效地控制项目的编程活动进程和定制自己的界面，提高了工作效率，也减少了错误产生的几率，而且，设计好基础类库，可以保证用户在开发多个项目时保持连续性，这种与业务无关的中间层设计使得二次开发非常的方便。■