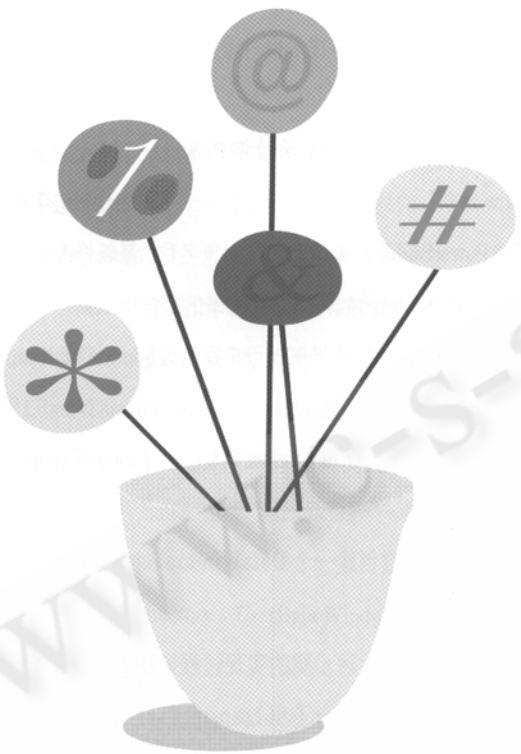
DataWindow 的动态保存及调用技术

Techniques for Saving and Using DataWindow Dynamically



高延红 （济南山东师范大学物理系 250014）

董祥军 孙吉红

（济南山东轻工业学院计算机科学与技术系 250100）

摘要：已经有许多文章讨论了DataWindow的动态创建方法，但是其中论述的保存及调用方法却不是很有效。本文讨论了一种遵循PowerBuilder程序员编程习惯的方法，它不仅使DataWindow的共享变得非常容易，而且为用户自定义报表的保存和调用提供了一条简便途径，文中给出了操作实例。

关键词：PowerBuilder 数据窗口 动态保存

1 引言

已经有许多文章讨论了DataWindow的动态建立方法，但是其中论述的保存及调用方法却不是很有效。例如有一种方法是将DataWindow的语法保存在一个用户自定义的数据表中，调用时再通过相应的函数动态生成DataWindow，该方法固然能够实现基本要求，但缺点也是非常明显的：使用者较难看懂那些苦涩的语法，只有在动态生成DataWindow之后才能看到它的样子，这在开发过程中难以共享。本文讨论了一种遵循PowerBuilder编程习惯的方法，即利用PowerBuilder自身提供的相关函数和功能实现DataWindow的动态保存及调用技术。该方法不仅使DataWindow的共享变得非常容易，而且还为用户自定义报表的保存和调用提供了一条简便途径。

2 实现原理

该方法的主要原理是将DataWindow（动态创建或者静态创建的）保存在PowerBuilder的库文件中，即以.pbl文件的形式存在，这样做的优点是不言而喻的：在开发程序时，你可以在PowerBuilder开发环境中非常方便地浏览和使用这些DataWindow。这些都非常简单，难点是如何动态（即通过程序代码）来实现这些功能，下面将详细讨论。

其实PowerBuilder为我们提供了几个有效的库操作函数：

LibraryCreate：创建一个PBL库文件；

LibraryImport：将一个PowerBuilder对象的语法导入PBL库中；

LibraryExport：从PBL库中导出PowerBuilder对象的语法；

LibraryDirectory：返回一个给定类型的对象列表；

另外还有一个函数ImportString：将一串以tab为定界符的字符串数据插入到一个DataWindow控件中。

本文并不准备向说明书那样详细介绍这些函数的语法和用法，具体使用方法请读者参照PowerBuilder的Help文件，但为了便于下面代码的阅读，笔者在相关位置标注了适当的注释。

3 实现方法

3.1 DataWindow的动态保存技术

我们用图1来说明该方法的实现过程，相关控件名称已在图中标出，该图可以嵌入到你的系统中。为了简化说明过程，这里假定你已经建立了用

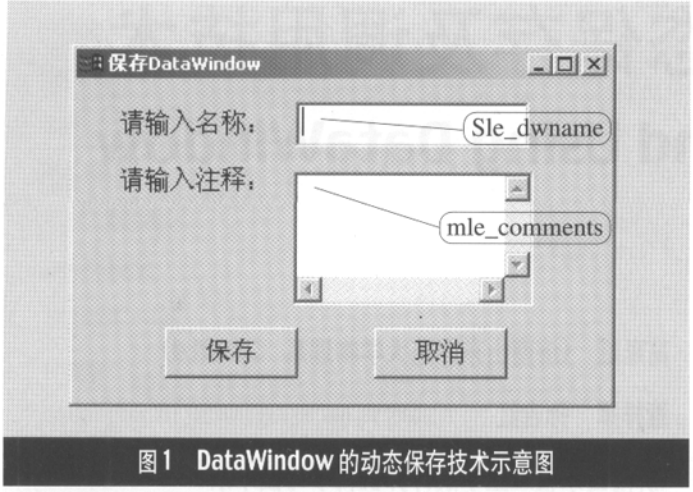


图1 DataWindow的动态保存技术示意图

于保存DataWindow的库文件UserDataWindow.PBL，而且假定你已经生成了DataWindow的语法并保存在变量ls_dwsyntax中且没有错误。下面是图1中保存按钮的clicked事件脚本，为了便于排版，用&将较长的语句进行了分割。

```
/* 保存按钮的clicked事件脚本 */
string  ls_importerrorbuffer
integer   li_returncode
if sle_dwname.text = "" then
   MessageBox("没有DataWindow名称","请输入DataWindow名称")
   setFocus(sle_dwName)
   return
end if
/* 检查名称中是否包含空格 */
if pos(sle_dwname.text, " ") > 0 then
   messagebox("DataWindow名称中不能有空格",&
      "DataWindow名称中不能有空格，请重新输入！")
   return
end if
/* 下面用LibraryImport函数将DataWindow语法保存到库文件UserDataWindow.pbl中。其参数依次为库文件名称、DataWindow名称，对象类型（此处为：ImportDataWindow!）、DataWindow语法变量（或常量）、可能的出错信息、DataWindow注释。保存成功时返回值1，出错时返回值-1 */
li_returncode = ibraryImport("UserDataWindow.pbl",&
      sle_dwName.text, ImportDataWindow!,&
      ls_dwsyntax, &
      ls_importerrorbuffer, mle_comments.text)
If li_returncode < 1 then
   messagebox("DataWindow保存失败", ls_importerrorbuffer)
end if
close(Parent)
```

3.2 DataWindow的动态调用技术

DataWindow的动态调用技术相对保存技术有点复杂，但是不难理解。大概步骤是：

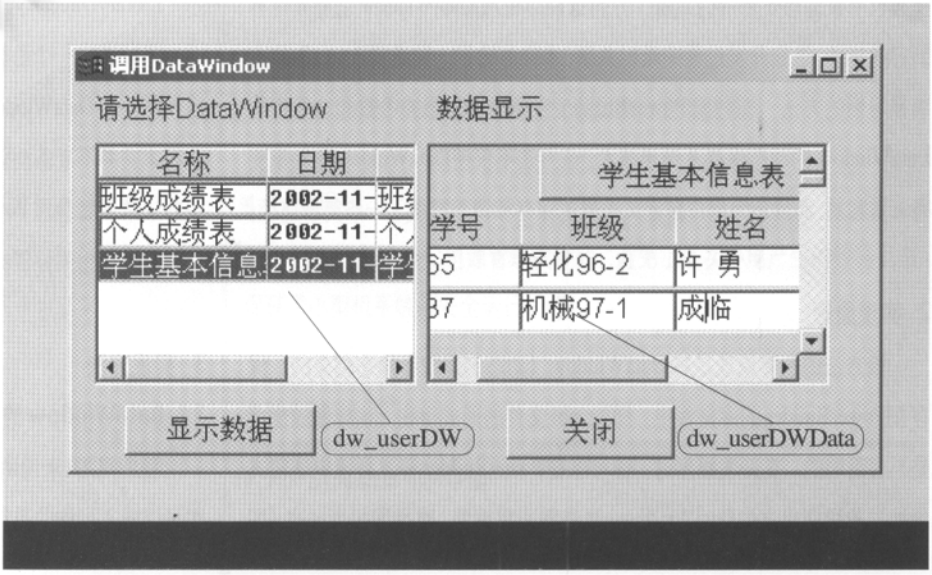
(1) 用LibraryDirectory函数将PowerBuilder对象（本文使用DataWindow对象）的列表从库中导出到一个字符串中，该字符串是以tab为定界符的PowerBuilde对象信息包括对象名称、最新修改时间以及注释。

(2) 用ImportString函数将上述字符串的数据插入到一个DataWindow控件中，该DataWindow控件用来向用户提供信息，如DataWindow名称等内容。

(3) 根据用户选择的DataWindow名称，用LibraryExport函数从库中导出对象的语法。

(4) 用Create函数创建一个DataWindow对象并将该对象放到一个DataWindow控件中。

下面用图2来说明上述步骤的实现过程，相关控件的名称已经在图中标出。特别说明的是，为了用ImportString函数将用户自定义DataWindow的相应数据导入到一个DataWindow控件中(如图2中的dw_userDW)，必须建立一个DataWindow对象（本文中的名字为d_lib_string)，该DataWindow对象是基于一个包含三列的表，名称和类型分别为：name(varchar(200))，date(varchar(20))，comments(varchar(200))，该表必须在你的数据库中（你必须自己创建）。注意，该表并不存放任何数据，只是用这个结构来向用户显示自定义DataWindow的名称等相关信息，这是



ImportString 函数要求的，其含义已如列名所示；列的类型随数据库（DBMS）的不同而有所差别，但必须是字符型（列date不要定义为日期型），列的宽度只需满足要求即可。

图 2 中实现的基本操作是：在窗口的 open 事件中将用户自定义 DataWindow 的信息显示在 dw_userDW 中，用户单击 dw_userDW 时加亮所选择的行，用户单击“显示数据”按钮时相应的数据显示在 dw_userDWData 中。相关事件的脚本如下，在脚本中对相关函数的使用作了简要解释。

窗口的 open 事件脚本：

```
/* 用户调用窗口的 open 事件脚本 */
    string ls_objects
    dw_userdw.reset()
    /* 为 dw_userdw 控件指定 DataWindow 对象 d_lib_string */
    dw_userdw.dataobject = "d_lib_string"
    /* 用 LibraryDirectory 将 UserDataWindow.pbl 中的 PowerBuilder 对象输出到字符串 ls_objects 中，该字符串是以 tab 为定界符的 PowerBuilder 对象的相关信息，顺序是：name、date、comments，它将作为 ImportString 函数的参数。*/
ls_objects = LibraryDirectory ("UserData Window.pbl", dirdatawindow!)
if ls_objects <> "" THEN
    /* 如果输出正常，用 ImportString 函数将 ls_objects 中的数据导入到 dw_userdw 控件中 */
    dw_userdw.importstring(ls_objects)
    dw_userdw.sort()
    /* 将 DataWindow 置为只读 */
    dw_userdw.Object.DataWindow.ReadOnly="Yes"
else
    messagebox(" 错误 "," 输出 DataWindow 出错！ ")
end if
```

DataWindow 控件 dw_userDW 的单击事件脚本如下，其功能是加亮用户所选择的行。

```
if this.getrow()>0 then
    this.selectrow(0,false)
    this.selectRow(row,true)
end if
```

“显示数据”按钮的脚本如下：

```
String ls_dwsyn, ls_errors,d_name
if dw_userdw.getrow()>0 then
    /* 取得用户选择的 DataWindow 的名称 */
```


```
    d_name=dw_userdw.getItemString(dw_userdw.getRow(),1)
    /* 用 LibraryExport 函数得到 DataWindow 的语法 */
    ls_dwsyn = LibraryExport("userdatawindow.pbl", d_name, ExportDataWindow!)
    /* 用 Create 函数动态创建 DataWindow*/
    dw_userdwdata.Create(ls_dwsyn, ls_errors)
    dw_userdwdata.settransobject(SQLCA)
    dw_userdwdata.retrieve()
end if
```

特别注意的是，在编译应用程序时不要把 UserDataWindow.PBL 编译进去，即在发布应用程序时要将 UserDataWindow.PBL 同其他应用程序一同发布，否则用户在保存与调用 DataWindow 时将因为找不到文件而出错。当然，也可以根据实际应用情况在程序中加一个判断，在发现 UserDataWindow.PBL 文件不存在时用 LibraryCreate 函数建立。

4 进一步的讨论

该方法不仅适用于 DataWindow 的动态保存与调用，也适用于其他所有的 PowerBuilder 对象。另外，我们还可以在自己的应用程序中编写一个类似于 PowerBuilder 中的对象浏览器。

参考文献

1 陈明，杨劲松，PowerBuilder 8.0 高级编程技术 [M]，北京希望电子出版社。

2 温为民等，Power Builder 7.0实例应用进阶 [M]，机械工业出版社，2000。

3 樊金生等，Power Builder 6.5 使用教程 [M]，科学出版社，2000.6。

4 PowerBuilder Code Examples.