杭州应用工程技术学院学报,第 11 卷第 3 期,1999 年 9 月
Journal of Hangzhou Institute Applied Engineering
Vol.11 No.3, September 1999

PowerBuilder 6.0 共享对象的应用

潘俊强　　历天晔
(杭州应用工程技术学院信电系　杭州 310012)　(浙江大学)

摘　要　介绍了在分布式应用中如何使用 PowerBuilder 提供的共享对象实现数据共享.

关键词　PowerBuilder　分布式应用　共享对象　远程对象
中图分类号　TP311.133.1

PowerBuilder 通过支持共享对象以允许用户在分布式应用中使用共享数据;共享对象是一个能被多个客户连接共享的用户自定义对象.

1　使用方法

为了使多个客户应用共享一个对象,服务程序必须执行以下操作:

(1)执行 SharedObjectRegister 函数注册一个对象的命名实例;

(2)执行 SharedObjectGet 函数以获得一个共享对象的实例引用.

服务程序可以在主线程或一个客户会话内部执行这些操作,并且服务程序不必使用 Create 语句来创建共享对象实例,因为当服务程序在调用 SharedObjectRigister 函数时,PowerBuilder 已自动建立了共享对象实例.

只有在服务程序的主线程和为每个客户连接创建的会话中才能访问共享对象,且客户应用不能直接访问共享对象,而必须先要与能够委托共享对象工作的远程对象建立通信联系后才能访问,在这些远程对象中往往包含一个共享对象的实例引用. 一旦服务程序注册了共享对象实例,并且为该对象检索了公共数据,客户应用就可以通过远程对象提供的接口来调用共享对象中定义的函数,从而将请求传送给共享对象以得到结果数据.

远程对象中定义了间接调用共享对象函数的方法,最典型的处理就是远程对象和共享对象的函数具有相同的函数名,同时参数类型和返回值都相同(见举例).

2　相关函数

(1)**SharedObjectDirectory** 检索已注册的共享对象

收稿日期:1999 - 03 - 01

句法:SharedObjectDirectory(instancenames{,classnames})

参数说明:instancenames 是一个不说明下标界限的字符串数组,用来存放已注册的共享对象的实例名;classnames(此参数可选)也是一个不说明下标界限的字符串数组,用来存放已注册的共享对象的名字.

返回说明:Success! 函数执行成功;FeatureNotSupportedError! 此函数在 Win 3.X 不支持.

(2)**SharedObjectGet** 取得一个共享对象的实例引用

句法:SharedObjectGet(instancename,objectinstance)

参数说明:intancename 你想取得引用的共享对象的实例名,这个名字必须与用 SharedObjectRegister 函数注册时给出的名字相匹配;objectinstance 存放共享对象的实例引用的变量.

返回说明:Success! 函数执行成功;FeatureNotSupportedError! 此函数在 Win 3.X 不支持;SharedObjectCreateInstanceError! 指向共享对象的实例引用没有被创建; SharedObjectNotExistError! 给出的共享对象实例名没有注册.

用法说明:此函数检索一个由 SharedObjectRegister 函数创建的共享对象的实例引用;一旦你得到一个实例引用,你就能象使用其他对象一样访问该对象中的函数和属性.

(3)**SharedObjectRegister** 注册一个用户对象使之能够被共享

句法:SharedObjectRegister(classname,instancename)

参数说明:classname 想要使之共享的对象的名字;instancename 共享对象的实例名,用户自己定义.

返回说明:Success! 函数执行成功; FeatureNotSupportedError! 此函数在 Win 3.X 不支持;SharedObjectCreateInstanceError! 对象没有被创建;SharedObjectExistError! 给出的共享对象实例名已经被使用;SharedObjectCreatPBSessionError! 共享对象的会话没有被创建.

用法说明:使用该函数时,PowerBuilder 将为共享对象打开一个单独的运行时刻会话,并创建一个共享对象实例;在这里为对象实例给出的名字,为用 SharedObjectGet 函数来访问对象实例提供了途径.

(4)**SharedObjectUnregister** 注销一个以前注册过的用户对象,撤销其共享

句法:SharedObjectUnregister(instancename)

参数说明:instancename 共享对象的实例名,以前注册时使用过.

返回说明:Success! 函数执行成功; FeatureNotSupportedError! 此函数在 Win 3.X 不支持;SharedObjectNotExistError! 给出的共享对象实例名不存在.

用法说明:使用此函数给共享对象打上释放标志,直到应用中不存在对此对象的引用时,该对象才被真正释放.

3 实例说明

下面这个例子说明如何使用共享对象检索数据库信息. 服务程序使用一个定时器时从数据库中检索信息,检索结果存放在一个共享对象的实例变量中,这样客户应用就不必通过访问数据库而直接就能取得数据,客户应用通过调用用户对象的函数以获得存储在共享对象的实例变量中的数据.

3.1 结构类型的定义

此例中需定义一结构类型来传递数据,定义如下:

```
s_customers {
```

```
row          integer;//数据检索结果的行数
column       integer;//数据检索结果的列数
custdata[]   any     ;//检索得到的数据
 }
```

3.2 服务程序中共享对象的定义

服务程序中定义一个名为 uo _ customers 的对象，它将被当作共享对象来使用.

以下为该对象的定义：

实例变量

uo _ customers 对象包含下面两个实例变量：

datastore ids _ datastore

uo _ timer　iuo _ mytimer

在 ids _ datastore 变量中保存有从数据库中检索出来的用户数据.

uo _ timer 对象在以下定义.

Constructor 事件　uo _ customers 的 Constructor 事件首先与数据库建立连接并创建用来检索数据库的 datastore 对象实例；然后创建一个用户对象 uo _ timer(从标准类 Timing object 继承)的实例，调用其 Start 函数，该 Start 函数用 300 秒的时间间隔激活定时器，这样就能每 5 分钟检索一次数据库；最后将 uo _ customers 的实例引用传递给 uo _ timer 对象实例，以便在定时器中调用 uo _ customers 的检索函数.

脚本如下：

```
SQLCA.DBMS = "ODBC"
SQLCA.Database = "Powersoft Demo DB V6"//使用 PB 例程的数据库
SQLCA.AutoCommit = False
SQLCA.DBParm = "ConnectString = 'DSN = Powersoft Demo DB V6; UID = dba; PWD = sql'"
Connect using sqlca;

If sqlca.sqlcode = 0  then
  MessageBox("Connect to Database Successful",sqlca.sqlerrtext)
End if

Ids _ datastore = create datastore
Ids _ datastore.dataobject = "d _ customer"
  //数据窗口 d _ customer 是 PB 例程中自带的一个数据窗口
Ids _ datastore.SetTransObject(SQLCA)
Ids _ datastore.retrieve()

Iuo _ mytimer = create uo _ timer
Iuo _ mytimer.Start(300)
Iuo _ mytimer.PassObject(this)
```

Refresh _ custlist()函数　该函数检索数据库，并返回检索到的行数.

脚本如下：

```
long 11 _ rowcount
```

11 _ rowcount = ids _ datastore.retrieve()

returen 11 _ rowcount

Retrieve _ custlist()函数 该函数返回存储在实例变量 ids _ datastore 中的客户数据.

脚本如下:

s _ customers 1 _ customers

1 _ customers.row = ids _ datastore.rowcount()//得到数据的行数

1 _ customers.column = 9//数据窗口 d _ customer 的列数为 9

1 _ customers.Custdata = ids _ datastore.object.data//检索得到的数据

Return 1 _ customers

Destructor 事件 该事件首先停止定时器,然后取消与数据库的连接.

脚本如下:

iuo _ mytimer.Stop()

disconnect using sq1ca;

3.3 定时器对象的定义

uo _ timer 是一个从 Timing Object 继承过来的标准类用户对象[1],在它的 Timer 事件中,将通过调用共享对象的 refresh _ custlist 函数来更新存放在 ids _ datastore 中的数据.

实例变量 该对象只有一个实例变量,用以存放对共享对象(uo _ customers)实例的引用.

Uo _ customers iuo _ shared _ object

PassObject()函数 该函数将对共享对象实例的引用存放在实例变量 iuo _ shared _ object 中,函数带有一个参数 auo _ customers,类型为 uo _ customers.

脚本如下:

iuo _ shared _ object = auo _ customers

return 1

Timer 事件 在此事件中,通过共享对象的实例引用来调用共享对象的 refresh _ custlist 函灵敏,以更新用户数据.

脚本如下:

long rowcount

rowcount = iuo _ shared _ object.refresh _ custlist()

3.4 服务程序中远程对象的定义

服务程序提供一个名为 uo _ custdata 的远程对象,它为客户应用访问共享对象提供了一个接口.

实例变量 该对象也只有一个实例变量,用以存放对共享对象(uo _ customers)实例的引用.

uo _ customers iuo _ shared _ object

Constructor 事件 该事件取得共享对象的实例引用.

脚本如下:

SharedObjectGet("share1",iuo _ shared _ object)

Retrieve _ custlist()函数 该函数调用共享对象中的同名函数,以获得存放在 ids _ datastore 中的客户数据.

脚本如下:

s _ customers 1 _ customers

1 _ customers = iuo _ shared _ object.retrieve _ custlist()

return 1 _ customers

3.5 服务程序的主窗口

服务程序开始运行时，先创建侦听线程(由 Transport 对象的 Listen 函数启动)，然后出现具有注册和注销两个按钮的主窗口.

实例变量　服务程序中只有一个实例变量 transport mytransport，用以创建侦听线程.

服务程序主窗口的 Open 事件　在服务程序主窗口的 Open 事件中，创建一侦听线程.

脚本如下：

```
mytransport = create transport
mytransport.driver = "WinSock"
mytransport.application = "60000"
mytransport.Listen()
```

注册按钮　注册按钮的 Clicked 事件将 uo _ customers 注册为共享对象，并命名为"share1".

脚本如下：

```
Share ObjectRegister("uo _ customers","share 1")
```

注销按钮　注册按钮的 Clicked 事件将注销共享对象.

脚本如下：

```
ShareObjectUnregister("share1")
```

3.6 客户应用的窗口

客户应用的窗口包含一个 DataWindow 控件，用来显示用户数据；当用户按下检索按钮时，客户应用连接至服务器以取得数据，一旦检索到数据，就显示在 DataWindow 控件中.

检索按钮　检索按钮的 Clicked 事件中，将通过远程对象(uo _ custdata)的实例引用，来执行其 retrieve _ custlist 函数以得到用户数据.

脚本如下：

```
s _ customers 1 _ customers
connection myconnect
myconnect = create connection
myconnect.drive = "winsock"
myconnect.application = "60000"//与服务程序创建的侦听线程中的 application 名一致
                               //指定的是服务端口名
myconnect.location = "Local Host"//运行服务程序的服务器名，可以直接用 IP 地址；
                                 //对于本机，此值为"Local Host"
myconnect.ConnectToServer()
uo _ custdata iuo _ custdata
myconnect.CreateInstrance(iuo _ custdata)
1 _ customers = iuo _ custdata.retrieve _ custlist()
dw _ 1.object.data[1,1,1 _ customers.row,1 _ customers.column] = 1 _ customers.custdata
```

4 结束语

共享对象为分布式 PowerBuilder 应用带来了显而易见的好处，它允许你

●能够便利地访问公共数据(统一由共享对象检索数据)，如果没有共享对象的话，则每一个客

户连接必须单独检索这些公共数据.

●减少对数据库的访问次数,从而使数据库服务器能有更多的时间去处理其他事务.

●提高数据安全性,由服务程序统一访问数据库,而不是由客户端应用来访问.

参 考 文 献

1 Hatfield. Bill PowerBuilder 应用程序开发指南 . 史森,史磊,夏丽丽译 . 北京:清华大学出版社,1997.6

The application of shared object in PowerBuilder 6.0

Pan Junqiang (Hangzhou Institute of Applied Engineering Hangzhou 310012)

Li Tianye (Zhejiang Universty)

Abstract This paper introduces how to use shared object provided by PowerBuilder to share data in a distributed application.

Key words PowerBuilder distributed application shared object remote object