PowerBuilder 8.0

高级应用技术

崔　巍　林小茶　杨晏文　编著

清华大学出版社
http://www.tup.tsinghua.edu.cn

PowerBuilder 8.0
高级应用技术

崔　巍　林小茶　杨晏文 编著

清华大学出版社

(京)新登字 158 号

内 容 简 介

本书从多个角度介绍了 PowerBuilder 8.0 的应用技术和高级使用方法。绪论和第 1 章是序篇，简单介绍了面向对象的程序设计方法和快速学习 PowerBuilder 8.0 的捷径；第 2—12 章则分别介绍了一些 PowerBuilder 8.0 的高级用法，主要内容包括：MDI 应用、多窗口实例、数据管道、统计图、RichText、OLE 和 DDE 等程序设计技术，及一些高级控件的使用和编程方法；第 13—17 章则重点介绍了 PowerBuilder 8.0 的分布式开发方法，主要包括 EAServer（Jaguar）组件和客户端的开发、以及 COM/MTS 组件和客户端的开发方法等。

本书适合于具有一定 PowerBuilder 的使用和开发基础、而又想进一步提高的读者或开发人员阅读。

书　　名：PowerBuilder 8.0 高级应用技术
作　　者：崔　巍　林小茶　杨晏文
出 版 者：清华大学出版社(北京清华大学学研大厦，邮编 100084)
http：//www. tup. tsinghua. edu. cn
责任编辑：章忆文
印 刷 者：北京密云胶印厂
发 行 者：新华书店总店北京发行所
开　　本：787 × 1092 1/16　**印张**：21　**字数**：505 千字
版　　次：2002 年 5 月第 1 版　2002 年 5 月第 1 次印刷
书　　号：ISBN 7-302-05397-9/TP · 3176
印　　数：0001 ~ 5000
定　　价：30.00 元

目　录

绪论　面向对象程序设计

PowerBuilder是面向对象的开发工具，它支持面向对象的程序设计技术，支持类(Class)、属性(Property)、方法(Method)、继承(Inheritance)、封装(Encapsulation)和多态性(Polymorphism)等面向对象程序设计的概念。在本书的开篇首先向读者介绍一下相关的面向对象概念及在PowerBuilder中所对应的技术。

0.1　面向对象的术语

在面向对象程序设计中可以通过建立可重用的类(执行应用处理逻辑的模块)来提高软件的开发效率。类由属性和方法组成，通过建立类的实例来执行应用处理逻辑。PowerBuilder支持类的概念：

- 类(Class)：在PowerBuilder中称为对象(如窗口、菜单、窗口上的控件和用户对象等)。
- 属性(Property)：在PowerBuilder中包括描述对象自身性质的属性和实例变量。
- 方法(Method)：在PowerBuilder中分为事件和函数。

在本书的其他地方均使用PowerBuilder的术语。

面向对象的程序设计工具应该支持继承、封装和多态性这三个面向对象程序设计的基本原则：

- 继承(Inheritance)：可以从已有的对象派生出新的对象，新的对象继承了祖先对象的所有特征，这样可以节省编程的时间，最大限度地重用代码，并且可以增强一致性。通过继承建立的子孙对象也称作子类(Subclass)。
- 封装(Encapsulation)：对象含有自己的数据和程序代码，允许来自对象外的某些存取或访问。封装也称作信息隐藏。PowerBuilder通过存取权限(access)和限定变量的使用范围(如全局变量、实例变量等)实现封装。
- 多态性(Polymorphism)：同一名称函数根据引用对象不同可以有不同的功能，多态性可以为整个应用和所有对象内部提供一个一致的接口。

PowerBuilder的对象可以分为可视对象(Visual object)和非可视对象(Nonvisual object)，可视对象是那些在界面上看得到的对象，如窗口、菜单等；非可视对象则在界面上看不到，它们用于程序内部的处理，如Transaction(事务对象)、Message(消息对象)和Error(错误对象)等。

0.2 面向对象技术在 PowerBuilder 中的实现

PowerBuilder 在可视和非可视对象中对继承、封装和多态性都提供了全面的支持。

0.2.1 继承

PowerBuilder 可以很方便地建立子孙对象，利用 Inherit 画板可以从指定的祖先对象继承建立新的对象。同时建立的所有对象都是 PowerBuilder 内置系统对象的子孙对象。

例如在第一章将要介绍的样本程序中，w_employee 窗口和 u_employee_object 用户对象都是通过继承建立的可视对象。

在非可视对象中使用继承的一个例子是：建立一个执行基本任务的祖先对象，然后再建立几个完成特殊任务的子孙对象，这些子孙对象可以执行祖先对象中的任务，如图 0-1 所示。

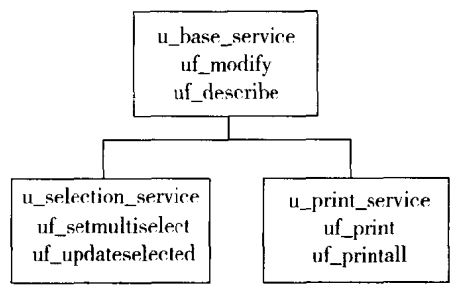

图 0-1 继承建立非可视对象

0.2.2 封装

通过限定实例变量的 Private 或 Protected 存取权限，可以将对象的数据隔离；同样可以对对象的函数限定存取权限，从而达到封装的目的。例如可以按如下方法实现对处理逻辑和数据的封装：

- 根据不同的封装程度将实例变量定义为 Public、Private 或 Protected。
- 定义 Private 或 Protected 类型的对象函数来执行处理逻辑和访问对象的数据。
- 定义一个 Public 函数，通过参数指定要完成的处理。

0.2.3 多态性

多态性的意思是：同一名称的函数根据引用对象的不同，具有不同的功能。这种多态性的函数分为如下两种情况。

一种是独立的、不相关的对象定义了同名的函数，每个函数在各自的对象内执行相应的处理。

另一种情况是，在一个继承链中的各个对象定义了同名的函数，但是这些函数可以具有不同的参数、完成不同的任务。PowerBuilder 根据当前引用的对象执行相应的函数。

0.3　PowerScript 语言使用的几个专题

这一节介绍一些使用 PowerScript 语言方面的专题，这些专题是进一步提高 PowerBuilder 应用技术和应用水平的基础。

0.3.1　常量说明

为了说明常量，只需要在标准的变量说明前添加关键字 CONSTANT：

```
CONSTANT { access } <数据类型> <常量名> = <值>
```

任何可以直接赋值的数据类型都可以说明常量，因此没有 Blob 类型的常量。

PowerScript 中的标识符是不区分大小写的，但习惯上用大写命名常量名，例如：

```
CONSTANT integer GI_CENTURY_YEARS = 100
CONSTANT string IS_ASCENDING = "a"
```

由于常量的值在程序中是不可以改变的，所以在说明之外的任何地方对常量赋值都会产生编译错误。

常量与使用常数相比可以提高程序的可读性和可维护性。

常量与使用变量相比可以提高程序的效率(因为常量是直接使用值，而变量是通过变量引用值)。

0.3.2　控制对实例变量的存取

实例变量通过访问设置来控制其他对象的存取，实例变量可以指定为：

- Public(公共的)：可以由其他任何对象存取。
- Protected(保护的)：可以由对象自身的程序和它的子孙对象存取。
- Private(私有的)：只可以由对象自身的程序存取。

例如：

```
//Public 变量和常量,任何对象可以引用
public integer ii_currentvalue
CONSTANT public integer WARPFACTOR = 1.2
//Protected 变量,子孙对象可以使用
protected string is_starship
//Private 变量,只能用于内部计算
```

```
private integer ii_maxrpm
private integer ii_minrpm
```

在以上的基础上，对 Public 和 Protected 变量可以进一步限定：

- PRIVATEREAD——私有读
- PRIVATEWRITE——私有写
- PROTECTEDREAD——保护读
- PROTECTEDWRITE——保护写

例如：

```
public privatewrite integer ii_averagerpm
```

说明了一个可供公共读、但只能内部写的变量。

0.3.3 解决命名冲突

在用 PowerScript 编程时可能会出现两类命名的冲突：

- 在不同范围内定义了同名的变量。例如，一个全局变量和一个实例变量或局部变量有同样的名称，对这样的冲突编译会给出警告，但是不一定要重新命名变量。
- 子孙对象和祖先对象有同名的函数或事件。

为了引用被"隐藏"的变量和祖先对象的事件或函数，可以使用圆点表示法限定或使用范围操作符。

引用实例变量

如果一个实例变量和一个局部变量、或全局变量、或共享变量有同样的名称，则使用对象名来限定实例变量：

```
<对象名>.<实例变量>
```

例如，如果窗口的一个局部变量和一个实例变量都命名为 birthdate，那么为了存取实例变量则要写为：

```
w_main.birthdate
```

再如，如果一个窗口定义了局部变量 x，这个名称与窗口的 X 属性命名冲突，因此要限定 X 属性，例如比较这两个值的语句：

```
IF x > w_main.X THEN
```

引用全局变量

如果一个全局变量和一个局部变量或共享变量有同样的名称，那么可以用范围操作符(::)来存取全局变量：

```
::<全局变量>
```

如下的表达式比较同名的局部变量和全局变量：

```
IF total < ::total THEN ...
```

同名函数和事件的处理

当子孙对象和祖先对象有同名的函数和事件时，在子孙对象中使用范围操作符引用祖先对象的函数或事件。

例如，下面的语句调用祖先对象 w_ancestor 中的 of_func 函数(范围操作符前是祖先类的名称，不是变量名)：

```
result = w_ancestor:: FUNCTION of_func(arg1, arg2)
```

也可以使用 Super 代词代替祖先类的名称，Super 引用对象的直接祖先。如下的语句执行祖先对象的 ue_process 用户事件：

```
result = Super:: EVENT ue_process()
```

0.3.4　祖先对象程序的返回值

如果在子孙对象的事件中执行某个处理，而这个处理依赖于祖先对象程序的返回值，这时可以直接使用 AncestorReturnValue 变量的值，该变量是系统自动说明并被赋值为祖先对象事件的返回值。

当编译程序第一次遇到调用祖先对象事件程序的 CALL 语句时，编译程序立刻生成说明 AncestorReturnValue 变量并为它赋值(祖先对象事件的返回值)的语句。变量 AncestorReturnValue 的数据类型与事件返回值的数据类型是一致的。

例如，在子孙对象事件的程序中可以有如下语句：

```
IF AncestorReturnValue = 1 THEN
    //执行某些程序代码
ELSE
    //执行另外一些程序代码
END IF
```

0.3.5　函数和事件的参数类型

在定义函数和用户事件时，可以为它们指定参数、参数的数据类型及其传递方式。

参数的传递方式有三种：

- 值传递(Value)：默认的参数传递方式。PowerBuilder 传递一个值的拷贝给函数或事件，在函数或事件内对参数的任意修改，对原来的值都没有任何影响。
- 变量传递(Reference)：PowerBuilder 传递一个变量的地址指针给函数或事件，这时在函数或事件内、外操作的都是同一个变量，即在函数或事件内对参数值的任