

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。

版权所有, 翻印必究。

本书封面贴有清华大学出版社激光防伪标签, 无标签者不得销售。

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	1
0.1 面向对象的术语	1
0.2 面向对象技术在 PowerBuilder 中的实现	2
0.2.1 继承	2
0.2.2 封装	2
0.2.3 多态性	2
0.3 PowerScript 语言使用的几个专题	3
0.3.1 常量说明	3
0.3.2 控制对实例变量的存取	3
0.3.3 解决命名冲突	4
0.3.4 祖先对象程序的返回值	5
0.3.5 函数和事件的参数类型	5
0.3.6 祖先和子孙变量	6
第1章 快速捷径	8
1.1 关于样本程序	8
1.2 运行样本程序	9
1.2.1 浏览和运行样本程序	9
1.2.2 查找自己想要的内容	11
第2章 MDI 应用程序	13
2.1 什么是 MDI	13
2.1.1 MDI 窗口的构成	13
2.1.2 标准 MDI 窗口和定制 MDI 窗口	13
2.2 建立 MDI 窗口	15
2.3 在 MDI 窗口中打开工作窗口	16
2.3.1 OpenSheet 函数	16
2.3.2 OpenSheetWithParm 函数	17
2.4 MDI 窗口的 MicroHelp	19
2.4.1 菜单项的 MicroHelp	19
2.4.2 控件的 MicroHelp	20
2.5 MDI 窗口的工具栏	21
2.5.1 MDI 窗口工具栏的一些概念	21

2.5.2	在 Menu 画板中设置工具栏的属性	22
2.5.3	建立工具栏中的下拉图标列表框	23
2.5.4	在 Window 画板中设置工具栏的属性	24
2.5.5	在 Application 画板中设置工具栏的属性	24
2.5.6	在 MDI 应用程序中灵活使用工具栏	26
2.6	调整客户区域的尺寸	29
第 3 章	多窗口实例	32
3.1	概述	32
3.2	多窗口实例的使用	33
3.3	窗口数组	34
3.3.1	窗口数组的说明和使用	34
3.3.2	操作窗口数组	35
3.3.3	混合窗口数组	35
3.4	实例窗口中控件的引用	36
第 4 章	事务对象与数据库操作	38
4.1	事务对象的一些概念	38
4.2	事务对象的使用	41
4.2.1	事务的基本概念	41
4.2.2	事务对象的赋值	42
4.2.3	同时操作多个数据库	43
4.2.4	数据库事务池	45
4.3	使用事务对象调用存储过程	46
4.4	嵌入 SQL 与数据库操作	51
4.4.1	SELECT 查询语句	51
4.4.2	操作语句	52
4.4.3	事务提交与撤消	53
4.4.4	使用游标的语句	54
4.4.5	利用存储过程进行查询的语句	57
4.4.6	查询和更新 Blob 类型字段的语句	59
4.4.7	动态 SQL 语句	61
第 5 章	数据管道程序设计	71
5.1	概述	71
5.2	准备工作	71
5.2.1	建立 Data Pipeline 对象	72
5.2.2	建立相关的用户对象	73

5.2.3 建立 Window 对象	74
5.3 数据管道操作及处理程序	75
5.3.1 准备工作	75
5.3.2 数据管道操作	76
5.3.3 出错处理	78
第 6 章 灵活运用图形表现数据	79
6.1 Graph 控件简介	79
6.2 生成单一产品的销售走势图	82
6.2.1 界面设计	82
6.2.2 建立 DataWindow 对象	84
6.2.3 编写程序代码	84
6.3 生成多个产品销售走势对比图	86
6.4 动态改变图形的类型	89
第 7 章 窗口中几个控件的使用	92
7.1 Tab 标签控件	92
7.1.1 术语及实例说明	92
7.1.2 准备工作	93
7.1.3 建立 Tab 控件和选项卡	94
7.1.4 管理 Tab 控件及其选项卡	95
7.1.5 写 Tab 控件的程序	96
7.2 ListView 控件	100
7.2.1 利用 ListView 控件进行查询	100
7.2.2 ListView 控件的详细资料方式	102
7.3 TreeView 控件	104
7.3.1 用 TreeView 控件实现带层次的查询	104
7.3.2 TreeView 控件应用技术	107
7.4 轨迹条控件	114
7.5 进度条控件	117
7.6 超级链接控件	118
第 8 章 Rich Text 应用	120
8.1 RichTextEdit 控件的使用	120
8.1.1 RichTextEdit 控件的属性设置	120
8.1.2 RichTextEdit 控件的文本	121
8.1.3 打开和保存文件的实例	123
8.2 RichTextEdit 控件中的输入域	126

8.2.1	在文档中插入输入域	126
8.2.2	为输入域指定值	127
8.2.3	为日期和页码指定输入域	128
8.3	预览和打印	128
8.3.1	预览	128
8.3.2	打印	129
8.4	使用数据库中的数据	129
8.5	RichText 数据窗口	130
第9章	OLE 技术及应用	133
9.1	OLE 的基本概念	133
9.2	OLE 控件和可插入的对象	134
9.2.1	建立和设置 OLE 控件	134
9.2.2	链接和嵌入	136
9.2.3	Offsite 或 In-place 激活	136
9.3	OLE 定制控件	137
9.4	可编程的 OLE 对象	138
9.4.1	OLEObject 对象数据类型	139
9.4.2	OLE 控件、OLE 定制控件和 OLEObject 对象之间的赋值	140
9.4.3	OLEObject 的事件	141
9.5	OLE 程序设计	141
9.5.1	自动操作接口	142
9.5.2	自动操作与 Any 数据类型	144
9.6	Browser 画板中的 OLE 信息	144
9.7	操作 OLE 对象的高级方法	146
9.7.1	OLE 存储的结构	146
9.7.2	存储和流的对象类型	147
9.7.3	打开和保存存储	147
9.7.4	建立和使用存储的例子	149
9.7.5	OLE 流的概念和应用	151
9.7.6	使用存储的策略	154
第10章	PowerBuilder 自动服务器	155
10.1	Automation Server 的基本概念	155
10.2	用户对象作为自动服务器	156
10.2.1	建立作为服务器的类用户对象	156
10.2.2	建立对象的运行时库	156
10.2.3	注册用户对象	157

10.2.4	编写访问用户对象的客户端程序	159
10.3	使用 PowerBuilder 作为自动服务器	160
10.3.1	建立要访问的用户对象	160
10.3.2	生成运行时库	161
10.3.3	编写访问 PowerBuilder 和用户对象的客户端程序	161
10.4	命名服务器的建立和使用	163
第 11 章	动态数据交换	165
11.1	动态数据交换的概念	165
11.2	动态数据交换的编程	166
11.2.1	DDE 客户端程序设计	166
11.2.2	DDE 服务器程序编程	169
11.2.3	DDE 的事件和函数	171
11.3	动态数据交换应用实例	186
11.3.1	DDE 服务器程序设计实例	186
11.3.2	DDE 客户程序设计实例	189
11.4	使用 Excel 打印 PowerBuilder 的报表	191
第 12 章	几种常用编程技术	193
12.1	使用初始化文件和 Windows 注册表	193
12.1.1	使用初始化文件	193
12.1.2	使用 Windows 注册表	194
12.2	处理 Blob 数据	197
12.2.1	处理 Blob 数据的 SQL 语句	197
12.2.2	利用 Blob 数据完成对图片或大文本的处理	198
12.2.3	数据窗口中的 Blob 列	199
12.3	动态链接库与外部函数的调用	202
12.3.1	说明外部函数	202
12.3.2	外部函数调用举例	204
12.4	嵌入邮件应用	206
12.4.1	与邮件应用相关的技术	207
12.4.2	邮件会话编程	209
12.5	与目录和文件操作有关的一组函数	211
12.5.1	目录管理函数	211
12.5.2	文件操作函数	213
12.5.3	文件管理函数	220

第 13 章 分布式应用概述	223
13.1 为什么要使用分布式应用技术	223
13.1.1 客户/服务器模式存在的问题	223
13.1.2 分布式计算提供的解决方案	224
13.2 EAServer	225
13.2.1 什么是 EAServer	225
13.2.2 EAServer 服务器结构	226
13.2.3 EAServer 客户结构	227
13.2.4 关于 Jaguar CTS	229
13.3 微软事务服务器(MTS)	230
 第 14 章 建立 EAServer 组件	231
14.1 建立 EAServer 组件的方法	231
14.1.1 使用向导建立 EAServer 组件	231
14.1.2 建立 EAServer 配置文件(profile)	231
14.1.3 建立 EAServer 组件的步骤	232
14.1.4 建立 EAServer 组件的实例	233
14.2 共享组件和服务组件	239
14.2.1 共享组件	240
14.2.2 服务组件	241
14.3 实例池(Instance Pooling)	242
14.3.1 为什么使用实例池	242
14.3.2 在向导的选项中定义实例池	242
14.3.3 控制放入实例池中的实例状态	243
14.3.4 组件的生命周期	243
14.4 对事务提供支持	244
14.4.1 为什么使用 EAServer 事务支持	245
14.4.2 标明组件如何支持事务	245
14.4.3 使用事务服务内容对象(transaction service context object)	246
14.4.4 Automatic Demarcation/Deactivation(自动划分/解除)	247
14.4.5 提交和撤消	247
14.4.6 事务和组件的生命周期	247
14.5 从 EAServer 组件访问数据库	248
14.5.1 概述	248
14.5.2 使用连接缓存(connection caching)	249
14.5.3 实现检索操作	250
14.5.4 实现修改	251
14.5.5 传送结果集	256

14.6	定义组件接口	257
14.6.1	说明接口	258
14.6.2	接口中包含的内容	258
14.6.3	方法的命名与方法的重载	258
14.6.4	数据类型	258
14.6.5	通过引用传送参数	259
14.6.6	传递只读数据	259
14.6.7	传递对象	259
14.6.8	对空值提供支持	260
14.6.9	EAServer 有效性	260
14.7	实施现存的接口	260
14.7.1	选择接口	261
14.7.2	在向导中设置选项	261
14.7.3	在画板中编辑用户对象	261
14.7.4	插接组件到 EAServer	262
14.7.5	使用不同的工程	262
14.8	调用其他服务器的组件方法	262
14.9	存取组件属性	263
14.10	测试和调试组件	265
14.10.1	活动编辑	265
14.10.2	远程调试	266
14.10.3	写一些信息到 EAServer 日志中	268
第 15 章	建立 EAServer 客户端应用	269
15.1	建立 EAServer 客户端应用的方法	269
15.2	建立与 EAServer 服务器的连接	269
15.2.1	手工编写代码	270
15.2.2	使用向导创建 Connection 对象	271
15.3	生成 EAServer 代理(Proxy)对象	275
15.4	调用组件方法	278
15.4.1	调用 EAServer 服务器组件的方法	278
15.4.2	实现对 n_stock_app 组件上的方法 businessdays 进行调用	279
15.4.3	清除代理对象实例	281
15.5	使用 JaguarORB 对象	281
15.5.1	概述	281
15.5.2	使用 String_To_Object 函数实例化代理对象	282
15.5.3	使用命名服务 API 实例化	284
15.6	划分到客户端和划分到组件的事务	285

15.6.1	两阶段提交	285
15.6.2	将组件标记为 OTS 类型	285
15.6.3	初始化 CORBACurrent 对象	285
15.6.4	开始和结束事务	286
15.6.5	获取有关事务的信息	287
15.6.6	暂停和恢复事务	287
15.6.7	为事务设置超时 (timeout) 属性	287
15.7	请求从服务器返回信息	288
15.8	处理通信错误	290
15.8.1	概述	290
15.8.2	CORBA 异常	291
15.8.3	编写 Connection 对象 Error 事件的程序	293
15.8.4	编写 SystemError 事件的程序	294
15.9	插接客户应用程序	294
第 16 章	建立 COM 或 MTS 组件	295
16.1	建立 COM 和 MTS 组件概述	295
16.1.1	使用向导	295
16.1.2	开发步骤	296
16.2	组件对象模型	296
16.2.1	PowerBuilder 的 COM 服务器	296
16.2.2	自动服务器和 PB COM 服务器	297
16.3	定义组件接口	297
16.3.1	方法和数据类型	297
16.3.2	编码时的限制	299
16.4	从 COM 和 MTS 组件访问数据库	300
16.4.1	传送结果集	300
16.4.2	从客户端存取 MTS 组件的结果集	301
16.4.3	在 PowerBuilder 中使用 ADO ResultSets	301
16.4.4	从 MTS 组件中返回结果集	302
16.5	提供对事务的支持	303
16.6	为控制对象生存期提供支持	304
16.7	调用另一个服务器组件的方法	304
16.7.1	使用 OLEObject 对象	305
16.7.2	使用 TransactionServer 对象	305
16.8	在日志文件中记录错误	305
16.9	安全问题	305
16.9.1	在 Project 画板或向导中设定权限	305

16.9.2	标明安全	305
16.10	在 Project 画板中建立 COM 和 MTS 组件	306
16.10.1	使用 Project 画板建立 COM 服务器	306
16.10.2	自动注册组件	307
16.10.3	插接组件到 MTS	307
16.10.4	选择定制或双接口	307
16.11	运行 PowerBuilder 的 COM 对象	308
16.12	插接 PB COM 服务器	309
16.12.1	使用带有可用 COM 应用程序的 PB COM 服务器	309
16.12.2	插接一个 PB COM 对象到 MTS	309
16.12.3	插接文件到 MTS 服务器计算机	310
16.12.4	导入 PB COM 对象到 MTS	310
16.12.5	创建客户插接文件	311
16.12.6	安装客户插接文件	311
16.13	从客户端访问 PB COM 服务器	312
16.13.1	VB 作为客户	312
16.13.2	C++ 作为客户	313
16.13.3	使用 PB COM 服务器和 DCOM 对象	314
第 17 章	建立 COM/MTS 客户应用	318
17.1	建立 COM/MTS 客户应用程序	318
17.2	连接到 COM 服务器	318
17.3	与 COM 组件交互	319
17.3.1	调用组件方法	319
17.3.2	传送结果集	319
17.3.3	处理执行错误	320
17.4	从客户端控制事务	320

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。

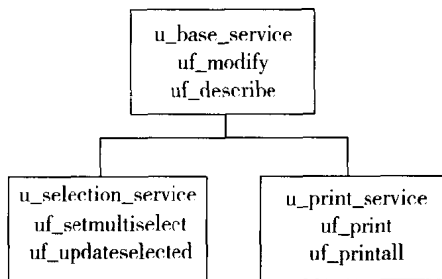


图 0-1 继承建立非可视对象

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：

- ◆ PRIVATE_READ——私有读
- ◆ PRIVATE_WRITE——私有写
- ◆ PROTECTED_READ——保护读
- ◆ PROTECTED_WRITE——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