



VisiBroker for Java Reference
VisiBroker for Java Programmer's Guide

Borland / Inprise 公司 著 李文军 周晓聪 李师贤 等译

VisiBroker for Java

开发人员指南



机械工业出版社
China Machine Press

本书提供了使用VisiBroker for Java 4.0 开发基于对象的分布式应用程序的详细资料。CORBA是公共对象请求代理体系结构的缩写,是由对象管理组(OMG)制定的一种面向对象应用软件体系结构的规范标准。由Inprise公司开发的VisiBroker产品使程序员能够遵循CORBA规范开发与调配基于对象的分布式应用程序。该产品最新推出的4.0版本完全遵从了最新的CORBA规范2.3版。

本书内容分为上下两篇:上篇是“VisiBroker for Java程序员指南”,包括了CORBA与VisiBroker入门、服务程序与客户程序的基本概念、VisiBroker ORB及其CORBA服务的配置与管理、各种工具与服务等知识。下篇是“VisiBroker for Java参考手册”,包括了VisiBroker所提供的程序员工具、IDL到Java的映射的介绍以及所提供的方法及参数的详细说明。

本书适用于熟悉面向对象开发方法以及Java程序设计语言的编程人员。

Inprise Corporation: VisiBroker for Java Programmer's Guide、VisiBroker for Java Reference.

Original edition copyright © Inprise Corporation. All rights reserved.

Chinese edition copyright © 2000 by China Machine Press. All rights reserved.

本书中文简体字版由美国Inprise公司授权机械工业出版社独家出版。未经出版者书面许可,不得以任何方式复制或抄袭本书内容。

版权所有,侵权必究。

本书版权登记号:图字:

图书在版编目(CIP)数据

VisiBroker for Java开发人员指南 / 宝兰-英博思公司著;李文军等译. - 北京:机械工业出版社, 2000.11

(Borland/Inprise核心技术丛书)

书名原文: VisiBroker for Java Programmer's Guide、VisiBroker for Java Reference

ISBN 7-111-08236-2

I.V… II.①宝… ②李… III.JAVA语言-程序设计 IV.TP312

中国版本图书馆CIP数据核字(2000)第45475号

机械工业出版社(北京市西城区百万庄大街22号 邮政编码 100037)

责任编辑:瞿静华

北京忠信诚胶印厂印刷·新华书店北京发行所发行

2000年11月第1版第1次印刷

787mm×1092mm 1/16·33.5印张

印数:0 001-4 000册

定价:79.00元(附光盘)

凡购本书,如有倒页、脱页、缺页,由本社发行部调换

序

分布式运算技术一直是业界的热门话题，CORBA(Common Object Request Broker Architecture)作为OMG组织制定的分布式体系架构，目前在全球掀起了一个又一个浪潮。随着CORBA技术应用在越来越多的关键领域(电信、金融、银行、企业)，并取得了极大成功，相信业界人士也会越来越意识到CORBA技术为IT产业带来的极大影响。尤其当今Internet的应用正如火如荼，企业迫切需求完整的分布式解决方案，应用于复杂的异构环境，实现不同硬件平台、软件系统、网络环境及数据库系统间的有效集成，CORBA更显示了其卓越的生命力。

Borland Inprise公司的VisiBroker作为CORBA标准的ORB(Object Request Broker)产品，以其卓越的性能受到业界的广泛关注，正被应用于越来越多的关键领域。本书作为Inprise VisiBroker for Java 4.0的标准开发手册，以CORBA的体系架构为基础，详细介绍了CORBA的基本理论和应用，诸如IDL、POA、DII、DSI等关键技术。全面细致地讲解了如何使用VisiBroker开发强大的分布式应用系统，内容不仅包括了IDL Compiler、Smart Agent、OAD、IREP等基本功能，而且还对Naming Service、Event Service等标准VisiBroker服务作了细致讲解。相信无论是VisiBroker的初学者还是CORBA的专家都会从此书受益。

北京宝兰-英博思信息技术有限公司是Borland/Inprise公司在中国大陆的授权代理机构，负责向中国大陆地区用户提供Borland/Inprise公司全系列产品及其技术服务。这里向VisiBroker的使用者和爱好者特别推荐本系列丛书，相信它一定会成为您的良师益友。

北京宝兰-英博思信息技术有限公司

www.inprise.com.cn

译者序

CORBA是公共对象请求代理体系结构的缩写，是由对象管理组(OMG)制定的一种面向对象应用软件体系结构的规范标准。由Inprise公司开发的VisiBroker产品使程序员能够遵循CORBA规范开发与调配基于对象的分布式应用程序。该产品最新推出的4.0版本完全遵从了最新的CORBA 2.3版本规范。

本书提供了使用VisiBroker for Java 4.0 开发基于对象的分布式应用程序的详细资料。本书内容分为上下两篇：上篇包括了CORBA与VisiBroker入门、服务程序与客户程序的基本概念、VisiBroker ORB及其CORBA服务的配置与管理、各种工具与服务等，并讨论了动态激发接口、动态框架接口、拦截器和对象包装器等高级概念。下篇包括了VisiBroker所提供的程序员工具、IDL到Java的映射的介绍，核心接口与类、动态接口与类的声明、所提供的方法及参数的详细说明，以及关于接口库、激活、命名服务、事件服务、异常、服务质量、位置服务等接口与类的声明及方法的详细介绍。本书要求读者熟悉面向对象开发方法以及Java程序设计语言。

在翻译过程中，译者对原文的错漏进行了修正。本书的翻译主要由中山大学计算机科学系的李文军、周晓聪、李师贤完成，李娜、谢红、张珞玲及郑红等也参与了其中的部分工作。由于水平有限，加之时间匆忙，译文谬误之处在所难免，恳请广大读者不吝批评指正。

李文军 周晓聪 李师贤
2000年6月于广州康乐园

目 录

序

译者序

第1章 概述	1
1.1 最新特性	1
1.2 手册约定	2
1.3 查找补充资料	3
1.4 联系Inprise公司的开发者支持	3

上篇 VisiBroker for Java程序员指南

第一部分 基本概念

第2章 理解CORBA模型	7
2.1 什么是CORBA	7
2.2 什么是VisiBroker	8
2.3 VisiBroker for Java的特性	8
2.3.1 VisiBroker智能代理体系结构	8
2.3.2 利用位置服务增强对象查找	8
2.3.3 对对象实现和对象激活的支持	9
2.3.4 健壮的线程与连接管理	9
2.3.5 IDL编译器	9
2.3.6 使用DII和DSI动态调用	9
2.3.7 接口库与实现库	9
2.3.8 服务端可移植性	10
2.3.9 用拦截器与对象包装器定制ORB	10
2.3.10 命名服务中的后备存储	10
2.3.11 Web命名	10
2.3.12 不用IDL定义接口	10
2.3.13 Gatekeeper	10
2.4 VisiBroker遵循CORBA规范	11
2.5 VisiBroker开发环境	11
2.6 Java开发环境	11
2.6.1 Java运行环境	12
2.6.2 VisiBroker的要求	12

2.6.3 支持Java的Web浏览器	12
2.7 与VisiBroker for C++的互操作性	12
2.8 与其他ORB产品的互操作性	13
2.9 从IDL到Java的映射	13
第3章 环境设置	14
3.1 设置PATH环境变量	14
3.1.1 在Windows平台修改PATH	14
3.1.2 在Windows NT平台修改PATH	14
3.1.3 在UNIX平台修改PATH	14
3.2 CLASSPATH	14
3.3 设置VBROKER_ADM环境变量	15
3.3.1 在Windows平台设置 VBROKER_ADM	15
3.3.2 在UNIX平台设置 VBROKER_ADM	15
3.4 设置OSAGENT_PORT环境变量	15
3.5 日志输出	16
第4章 用VisiBroker开发一个应用 程序例子	17
4.1 开发过程	17
4.2 步骤1: 定义对象接口	17
4.3 步骤2: 生成客户程序的桩 与服务程序的servant	18
4.4 步骤3: 实现客户程序	20
4.4.1 Client.java	20
4.4.2 AccountManagerHelper.java	21
4.5 步骤4: 实现服务程序	21
4.6 步骤5: 建立例子	22
4.7 步骤6: 启动服务程序并运行例子	22
4.7.1 启动智能代理	23
4.7.2 启动服务程序	23
4.7.3 运行客户程序	23
4.8 用VisiBroker调配应用程序	23

第5章 处理异常	27
5.1 CORBA模型中的异常	27
5.2 系统异常	27
5.2.1 获取结束状态	28
5.2.2 捕获系统异常	28
5.2.3 将异常向下传给系统异常	29
5.2.4 捕获特定类型的系统异常	30
5.3 用户异常	30
5.3.1 定义用户异常	30
5.3.2 修改对象以引发异常	31
5.3.3 捕获用户异常	31
5.3.4 添加用户异常的域	32

第二部分 服务程序的概念

第6章 服务程序基础	33
6.1 概述	33
6.2 初始化ORB	33
6.3 创建POA	33
6.3.1 获取根POA的一个引用	34
6.3.2 创建子POA	34
6.3.3 实现servant的方法	35
6.3.4 激活POA	36
6.4 激活对象	36
6.5 等待客户程序请求	36
6.6 完整的例子	37
第7章 使用POA	38
7.1 什么是可移植对象适配器	38
7.1.1 POA的术语	38
7.1.2 创建与使用POA的步骤	39
7.2 POA策略	39
7.2.1 线程策略	39
7.2.2 生命期策略	40
7.2.3 对象ID唯一性策略	40
7.2.4 ID指派策略	40
7.2.5 servant保留策略	40
7.2.6 请求处理策略	40
7.2.7 隐式激活策略	41
7.2.8 绑定支持策略	41

7.3 创建POA	41
7.3.1 POA命名约定	41
7.3.2 获取根POA	42
7.3.3 设置POA属性	42
7.3.4 创建并激活POA	42
7.4 激活对象	43
7.4.1 显式激活对象	43
7.4.2 按要求激活对象	44
7.4.3 隐式激活对象	44
7.4.4 用缺省servant激活	44
7.4.5 冻结对象	45
7.5 使用servant与servant管理器	46
7.5.1 servant激活器ServantActivators	47
7.5.2 servant定位器ServantLocators	49
7.6 使用POA管理器管理POA	51
7.6.1 取当前状态	52
7.6.2 持有状态	52
7.6.3 活动状态	52
7.6.4 丢弃状态	53
7.6.5 非活动状态	53
7.7 设置监听和分派的属性	54
7.7.1 设置服务程序引擎的属性	54
7.7.2 设置服务程序连接管理器的属性	55
7.7.3 何时使用这些属性	56
7.8 适配器激活器	57
7.9 处理请求	58
第8章 管理线程与连接	59
8.1 使用VisiBroker的线程	59
8.2 VisiBroker提供的线程策略	59
8.3 线程缓冲池策略	59
8.4 每会话线程策略	62
8.5 VisiBroker提供的连接管理	64
8.6 设置分派策略和属性	64
8.6.1 线程缓冲池策略	65
8.6.2 每会话线程策略	65
8.6.3 编码注意事项	65
第9章 使用纽带机制	66
9.1 纽带机制如何工作	66

9.2 示例程序.....66	11.5.1 位置服务.....81
9.2.1 使用纽带机制示例程序的位置66	11.5.2 命名服务.....81
9.2.2 对Server类的改动66	11.5.3 接口库.....81
9.2.3 对AccountManager类的改动67	11.5.4 实现库.....81
9.2.4 对Account类的改动.....68	11.5.5 服务程序管理器.....81
9.2.5 编译纽带例子68	11.5.6 Gatekeeper82
第三部分 客户程序的概念	第12章 使用ORB服务浏览器83
第10章 客户程序基础69	12.1 简介83
10.1 初始化ORB69	12.2 位置服务浏览器83
10.2 绑定到对象69	12.2.1 访问位置服务浏览器83
10.3 调用一个对象的操作70	12.2.2 刷新活动对象列表84
10.4 操纵对象引用70	12.3 命名服务浏览器84
10.4.1 将引用转换为字符串70	12.3.1 访问命名服务85
10.4.2 获取对象与接口名字71	12.3.2 浏览命名服务85
10.4.3 确定对象引用的类型71	12.3.3 浏览VisiBroker命名服务的簇86
10.4.4 确定被绑定对象的位置与状态72	12.4 实现库浏览器86
10.4.5 窄化对象引用72	12.5 接口库浏览器87
10.4.6 宽化对象引用72	12.5.1 查看接口库87
10.5 使用服务质量73	12.5.2 访问接口库88
10.5.1 理解服务质量73	12.5.3 浏览接口库89
10.5.2 QoS接口73	第13章 使用服务程序管理器90
10.5.3 QoS实例76	13.1 什么是服务程序管理器90
第四部分 配置与管理	13.1.1 查看顶层容器90
第11章 使用VisiBroker控制台77	13.1.2 服务程序管理器的浏览器90
11.1 什么是VisiBroker控制台.....77	13.1.3 使用VisiBroker 4.0的服务 程序例子90
11.2 启动VisiBroker控制台.....78	13.1.4 设置服务程序管理器的安全性92
11.3 配置控制台78	13.2 使用服务程序管理器的浏览器92
11.4 VisiBroker控制台导航.....80	13.2.1 查看服务程序的内容92
11.4.1 菜单栏.....80	13.2.2 启用服务程序93
11.4.2 工具栏.....80	13.2.3 调用方法93
11.4.3 状态栏.....80	13.2.4 设置属性94
11.4.4 下拉式菜单或上下文菜单.....80	第14章 设置属性98
11.4.5 导航面板.....80	14.1 概述98
11.4.6 内容面板.....81	14.2 属性文件98
11.5 支持的ORB服务81	14.3 环境变量99
	14.4 通过命令行和HTML设置属性100

14.5 VisiBroker for Java属性	100	16.7.2 迁移实例化的对象	116
第五部分 工具与服务		16.7.3 迁移注册到OAD的对象	117
第15章 使用IDL	101	16.8 报告所有对象与服务	117
15.1 IDL简介	101	16.9 绑定到对象	117
15.2 IDL编译器如何生成代码	101	第17章 使用位置服务	118
15.3 生成的代码	102	17.1 什么是位置服务	118
15.3.1 _<interface name>Stub.java	102	17.2 位置服务组件	119
15.3.2 <interface name>.java	102	17.2.1 位置服务代理	119
15.3.3 <interface name>Helper.java	102	17.2.2 触发器	121
15.3.4 <interface name>Holder.java	104	17.3 向代理查询	122
15.3.5 <interface name>Operations .java	104	17.3.1 查找接口的所有实例	122
15.3.6 <interface name>POA.java	104	17.3.2 查找智能代理知道的所有信息	123
15.3.7 <interface name>POATie.java	105	17.4 编写并注册触发器处理程序	125
15.4 在IDL中定义接口的属性	106	第18章 使用命名服务	128
15.5 指定无返回值的单向方法	106	18.1 概述	128
15.6 在IDL中指定继承其他接口的接口	106	18.2 理解名字空间	129
第16章 使用智能代理	108	18.2.1 命名上下文对象	129
16.1 什么是智能代理	108	18.2.2 命名上下文工厂对象	130
16.1.1 查找智能代理	108	18.2.3 Name和NameComponent	130
16.1.2 通过智能代理协作查找对象	108	18.2.4 名字解析	131
16.1.3 与OAD协作以连接对象	108	18.3 运行命名服务	131
16.1.4 启动智能代理	109	18.3.1 安装命名服务	132
16.1.5 保证智能代理的可用性	109	18.3.2 配置命名服务	132
16.2 使用ORB域	110	18.3.3 启动命名服务	132
16.3 连接不同本地网络中的智能代理	111	18.4 引导命名服务	133
16.4 使用多重初始地址的主机	112	18.4.1 调用resolve_initial_references 方法	133
16.5 使用点对点通信	114	18.4.2 使用-DSVCnameroot选项	133
16.5.1 以运行时参数指定主机	114	18.4.3 使用-DORBInitRef选项	133
16.5.2 用环境变量指定IP地址	115	18.4.4 使用-DORBDefaultRef选项	134
16.5.3 用agentaddr文件指定主机	115	18.5 NamingContext	134
16.6 保证对象的可用性	115	18.6 NamingContextExt	135
16.6.1 调用无状态对象的方法	115	18.7 缺省命名上下文	135
16.6.2 提高保持状态对象的容错性	116	18.8 可接插的后备存储	136
16.6.3 复制注册到OAD的对象	116	18.8.1 后备存储的类型	136
16.7 在主机间迁移对象	116	18.8.2 配置与使用	137
16.7.1 迁移保持状态的对象	116	18.9 簇	139

18.9.1 簇的标准	139	19.8.12 PushSupplier	163
18.9.2 Cluster和ClusterManager接口	140	第20章 使用对象激活监控进程	165
18.9.3 创建一个簇	141	20.1 对象和服务程序的自动激活	165
18.9.4 负载均衡	142	20.1.1 查找实现库数据	165
18.10 失败转移	142	20.1.2 激活服务程序	165
18.11 Java的import语句	143	20.2 启动对象激活监控进程	166
18.12 程序样例	143	20.2.1 在Windows平台启动对 象激活监控进程	166
第19章 使用事件服务	146	20.2.2 在UNIX平台启动对象激活 监控进程	166
19.1 概述	146	20.3 使用对象激活监控进程工具	167
19.1.1 消费者和供应商的代理	147	20.3.1 将接口名字转换为库ID	167
19.1.2 OMG公共对象服务规范	147	20.3.2 使用oadutil list列出对象	167
19.2 通信模型	148	20.3.3 使用oadutil reg注册对象	168
19.2.1 外推模型	148	20.3.4 区分一个对象的多个实例	171
19.2.2 回拉模型	149	20.3.5 使用CreationImplDef类设置 激活属性	171
19.3 使用事件信道	150	20.3.6 动态改变ORB实现	171
19.4 外推型供应商和消费者的例子	151	20.3.7 使用OAD::reg_implementation 进行OAD注册	172
19.4.1 运行外推模型的例子	151	20.3.8 创建和注册对象的例子	173
19.4.2 运行回拉模型的例子	152	20.3.9 由OAD传递的参数	173
19.4.3 PullSupply	152	20.4 注销对象	174
19.4.4 运行PullSupply	152	20.4.1 使用oadutil工具注销对象	174
19.4.5 PullConsume	154	20.4.2 使用OAD操作注销对象	175
19.4.6 运行PullConsume	154	20.4.3 显示实现库的内容	175
19.5 启动事件服务	156	20.5 OAD的IDL接口	175
19.6 进程内事件信道	157	第21章 使用接口库	177
19.7 Java的import语句	158	21.1 什么是接口库	177
19.8 接口参考资料	158	21.1.1 接口库包含什么	177
19.8.1 EventChannel	159	21.1.2 可有多少个接口库	178
19.8.2 EventLibrary	159	21.2 使用irep创建和浏览接口库	178
19.8.3 ConsumerAdmin	160	21.2.1 使用irep创建接口库	178
19.8.4 SupplierAdmin	160	21.2.2 浏览接口库中的内容	179
19.8.5 ProxyPullConsumer	160	21.3 使用idl2ir更新接口库	179
19.8.6 ProxyPushConsumer	161	21.4 理解接口库的结构	180
19.8.7 ProxyPullSupplier	161	21.4.1 标识接口库中的对象	181
19.8.8 ProxyPushSupplier	161		
19.8.9 PullConsumer	162		
19.8.10 PushConsumer	162		
19.8.11 PullSupplier	163		

21.4.2 存储在接口库中的对象类型	181
21.4.3 继承的接口	182
21.5 访问接口库	182
21.6 例子程序	183

第六部分 高级概念

第22章 使用动态调用接口	185
22.1 什么是动态调用接口	185
22.1.1 DII主要概念介绍	185
22.1.2 动态调用对象操作的步骤	188
22.1.3 查找使用DII的例子程序	188
22.1.4 使用idl2java编译器	188
22.2 获取通用对象引用	188
22.3 创建和初始化Request	189
22.3.1 Request接口	189
22.3.2 创建和初始化DII请求的方法	189
22.3.3 使用_create_request方法	190
22.3.4 使用_request方法	190
22.3.5 创建Request对象的例子	191
22.3.6 为Request设置参数	191
22.3.7 使用Any类安全地传递类型	192
22.3.8 使用TypeCode类表示参数 或属性的类型	193
22.4 发送DII请求和接收结果	195
22.4.1 调用请求	195
22.4.2 使用send_deferred方法发送 延迟DII请求	195
22.4.3 使用send_oneway方法发送 异步DII请求	196
22.4.4 发送多个请求	196
22.4.5 接收多个请求	196
22.5 在DII中使用接口库	197
第23章 使用动态框架接口	198
23.1 什么是动态框架接口	198
23.2 动态创建对象实现的步骤	198
23.3 继承DynamicImplementation类	199
23.3.1 设计动态请求对象的例子	199

23.3.2 指定库ID	201
23.4 查看ServerRequest类	202
23.5 实现Account对象	202
23.6 实现AccountManager对象	202
23.7 服务程序的实现	203
第24章 使用可移植拦截器	205
24.1 概述	205
24.2 拦截器的接口和管理器	205
24.2.1 客户程序拦截器	206
24.2.2 服务程序拦截器	207
24.2.3 缺省拦截器类	209
24.2.4 在VisiBroker ORB注册拦截器	209
24.2.5 创建拦截器对象	209
24.2.6 装入拦截器	210
24.3 拦截器例子	210
24.3.1 代码例子	210
24.3.2 代码列表	211
24.4 在拦截器之间传递信息	216
第25章 使用对象包装器	217
25.1 概述	217
25.1.1 类型化和非类型化的对象 包装器	217
25.1.2 idl2java的特殊要求	217
25.1.3 应用程序示例	218
25.2 非类型化对象包装器	218
25.2.1 使用多个非类型化对象包装器	218
25.2.2 pre_method的调用次序	219
25.2.3 post_method的调用次序	219
25.3 使用非类型化对象包装器	219
25.3.1 实现非类型化对象包装器工厂	220
25.3.2 实现非类型化对象包装器	220
25.3.3 创建和注册非类型化对象 包装器工厂	221
25.3.4 删除非类型化对象包装器	223
25.4 类型化对象包装器	223
25.4.1 使用多个类型化对象包装器	224
25.4.2 调用次序	224

25.4.3	共位客户和服务程序的类型 化对象包装器	225
25.5	使用类型化对象包装器	225
25.5.1	实现类型化对象包装器	225
25.5.2	为客户程序注册类型化对象 包装器	226
25.5.3	为服务程序注册类型化对象 包装器	227
25.5.4	删除类型化对象包装器	228
25.6	联合使用非类型化和类型化 对象包装器	228
25.6.1	类型化对象包装器的命令行 参数	228
25.6.2	类型化包装器的初始化程序	229
25.6.3	非类型化对象包装器的命令行 参数	230
25.6.4	非类型化包装器的初始化程序	230
25.6.5	运行应用示例	231
第26章	在IIOP上使用RMI	234
26.1	概述	234
26.2	使用java2iiop	234
26.2.1	支持的接口	234
26.2.2	运行java2iiop	235
26.2.3	完成开发步骤	235
26.3	RMI-IIOP的Bank例子	235
26.4	支持的数据类型	237
26.4.1	映射基本数据类型	237
26.4.2	映射复杂数据类型	237
26.4.3	接口	238
26.4.4	数组	238
第27章	使用动态管理类型	239
27.1	概述	239
27.2	DynAny类型	239
27.2.1	用法限制	239
27.2.2	创建DynAny	239
27.2.3	初始化和访问DynAny的值	240
27.3	复合数据类型	240
27.3.1	DynEnum	240
27.3.2	DynStruct	241
27.3.3	DynUnion	241
27.3.4	DynSequence和DynArray	241
27.4	IDL示例	241
27.5	客户应用程序示例	242
27.6	服务应用程序示例	243
第28章	使用值类型	248
28.1	理解值类型	248
28.1.1	具体值类型	248
28.1.2	抽象值类型	249
28.2	实现值类型	249
28.2.1	定义值类型	249
28.2.2	编译IDL文件	250
28.2.3	继承值类型的基类	250
28.2.4	实现Factory类	250
28.2.5	在ORB上注册Factory对象	251
28.3	实现工厂对象	251
28.3.1	工厂对象和值类型	251
28.3.2	注册值类型	251
28.4	封装值类型	252
28.5	抽象接口	252
28.6	定制值类型	253
28.7	可截断值类型	253
第29章	使用URL命名	254
29.1	URL命名服务	254
29.2	注册对象	254
29.3	使用URL查找对象	256
第七部分 向后兼容性		
第30章	在VisiBroker 4.0中使用BOA	259
30.1	使用VisiBroker 4.0编译BOA代码	259
30.2	支持的BOA选项	259
30.3	使用BOA的限制	259
30.4	使用对象激活器	259
30.5	BOA的对象命名	259
第31章	迁移VisiBroker代码	261
31.1	迁移工具	261
31.1.1	对程序包名前缀的改变	261

第七部分 向后兼容性

第30章	在VisiBroker 4.0中使用BOA	259
30.1	使用VisiBroker 4.0编译BOA代码	259
30.2	支持的BOA选项	259
30.3	使用BOA的限制	259
30.4	使用对象激活器	259
30.5	BOA的对象命名	259
第31章	迁移VisiBroker代码	261
31.1	迁移工具	261
31.1.1	对程序包名前缀的改变	261

31.1.2 对类名的改变	261	34.5.3 Java null	292
31.1.3 对API调用的改变	262	34.5.4 Boolean	292
31.1.4 将BOA改为POA	263	34.5.5 Char	292
31.1.5 改变拦截器的使用	263	34.5.6 Octet	292
31.2 调用迁移工具	263	34.5.7 String	292
31.3 使用迁移后的代码	264	34.5.8 WString	292
31.4 将BOA手工迁移到POA	264	34.5.9 整数类型	292
31.4.1 一个例子	264	34.5.10 浮点数类型	293
31.4.2 映射BOA类型到POA策略	266	34.6 Helper类	293
31.5 迁移到新的程序包名字	267	34.7 常量	294
31.6 迁移到新的类名字	267	34.7.1 接口中的常量	294
31.7 迁移到新的API调用	268	34.7.2 不在接口中的常量	295
31.8 迁移拦截器	268	34.8 复合类型	295
第32章 使用对象激活工具	271	34.8.1 Enum	295
32.1 延迟对象激活	271	34.8.2 Struct	297
32.2 Activator接口	271	34.8.3 Union	298
32.3 使用服务激活的途径	272	34.8.4 Sequence	299
32.3.1 使用服务激活器延迟对象激活	272	34.8.5 Array	300
32.3.2 对服务延迟对象激活的例子	273	34.9 接口	301
下篇 VisiBroker for Java 参考手册		34.9.1 参数传递	303
第33章 程序员工具	278	34.9.2 使用继承实现服务程序	304
33.1 选项	278	34.9.3 使用委派实现服务程序	305
33.2 idl2ir	278	34.9.4 接口作用域	306
33.3 ir2idl	279	34.10 异常的映射	306
33.4 idl2java	279	34.11 用户自定义异常	306
33.5 java2idl	281	34.12 系统异常	307
33.6 java2iio	282	34.13 Any类型的映射	307
33.7 vbj	283	34.14 嵌套类型的映射	308
33.8 其他VisiBroker工具	285	34.15 Typedef的映射	308
第34章 IDL到Java的映射	286	34.15.1 简单IDL类型	308
34.1 名字	286	34.15.2 复杂IDL类型	308
34.2 保留名字	286	第35章 生成的接口和类	310
34.3 保留字	287	35.1 概览	310
34.4 模块	287	35.1.1 基调和操作类	310
34.5 基本类型	288	35.1.2 辅助类	310
34.5.1 IDL扩充类型	288	35.1.3 可移植桩和框架接口	311
34.5.2 Holder类	289	35.2 <interface_name>Operations	311
		35.3 <type_name>Helper	311

35.3.1 所有Helper类都有的方法	311	36.12 PortableServer.ServantActivator.....	347
35.3.2 为接口生成的方法	312	36.13 PortableServer.ServantLocator	347
35.3.3 为对象包装器生成的方法	313	36.14 PortableServer.ServantManager	349
35.4 <type_name>Holder	314	36.15 Principal	349
35.4.1 成员数据	315	36.15.1 IDL定义	349
35.4.2 方法	315	36.15.2 Principal的方法	349
35.5 <interface_name>Stub.....	315	第37章 动态接口与类	350
35.6 <interface_name>POA	315	37.1 Any	350
35.7 <interface_name>POATie	315	37.1.1 Any的方法	350
第36章 核心接口与类	317	37.1.2 Any的提取方法	351
36.1 BindOptions	317	37.1.3 Any的插入方法	351
36.1.1 IDL定义	317	37.2 ARG_IN	352
36.1.2 BindOptions的构造方法	317	37.3 ARG_INOUT	352
36.2 BOA	317	37.4 ARG_OUT	353
36.2.1 IDL定义	318	37.5 ContextList	353
36.2.2 BOA的方法	319	37.5.1 IDL定义	353
36.3 CompletionStatus	320	37.5.2 ContextList的方法	353
36.3.1 IDL定义	320	37.6 DynAny	354
36.3.2 CompletionStatus的方法.....	320	37.6.1 重要的用法限制说明	354
36.4 Context	321	37.6.2 DynAny的方法	354
36.4.1 IDL定义	321	37.6.3 DynAny的提取方法	355
36.4.2 Context的方法	321	37.6.4 DynAny的插入方法	356
36.5 InvalidName	322	37.7 DynArray	356
36.6 Object	323	37.7.1 重要的用法限制说明	357
36.6.1 org.omg.CORBA Object定义	323	37.7.2 DynArray的方法	357
36.6.2 org.omg.Object的方法.....	324	37.8 DynAnyFactory	357
36.6.3 VisiBroker对Object的扩充	325	37.8.1 重要的用法限制说明	357
36.6.4 VisiBroker对Object扩充的方法.....	326	37.8.2 DynAnyFactory的方法	357
36.7 ORB	327	37.9 DynEnum	358
36.7.1 JDK的ORB定义	327	37.9.1 重要的用法限制说明	358
36.7.2 JDK ORB的方法	329	37.9.2 DynEnum的方法	358
36.7.3 OMG ORB定义	334	37.10 DynFixed	359
36.7.4 VisiBroker ORB的扩充	335	37.11 DynSequence	359
36.7.5 VisiBroker ORB的方法	335	37.11.1 重要的用法限制说明	359
36.8 PortableServer.AdapterActivator	337	37.11.2 DynSequence的方法	359
36.9 PortableServer.Current	337	37.12 DynStruct	360
36.10 PortableServer.POA	338	37.12.1 重要的用法限制说明	360
36.11 PortableServer.POAManager	345	37.12.2 DynStruct的方法	360

37.13 DynUnion	361	第38章 关于接口库的接口与类	379
37.13.1 重要的用法限制说明	361	38.1 AliasDef	379
37.13.2 DynUnion的方法	361	38.2 ArrayDef	379
37.14 DynValue	362	38.3 AttributeDef	380
37.15 DynamicImplementation	363	38.4 AttributeDescription	381
37.15.1 构造方法	363	38.4.1 AttributeDescription的变量	381
37.15.2 DynamicImplementation		38.4.2 AttributeDescription的方法	381
的方法	363	38.5 AttributeMode	382
37.16 Environment	364	38.6 ConstantDef	382
37.17 ExceptionList	364	38.7 ConstantDescription	383
37.17.1 IDL定义	364	38.7.1 ConstantDescription的变量	383
37.17.2 ExceptionList的方法	365	38.7.2 ConstantDescription的方法	383
37.18 InputStream	365	38.8 Contained	384
37.19 Invalid	366	38.8.1 IDL定义	384
37.20 InvalidSeq	366	38.8.2 Contained的方法	384
37.21 NamedValue	366	38.9 ContainedPackage.Description	385
37.21.1 IDL定义	366	38.9.1 ContainedPackage.Description	
37.21.2 NamedValue的方法	367	的变量	386
37.22 NameValuePair	367	38.9.2 ContainedPackage.Description	
37.22.1 NameValuePair的变量	367	的方法	386
37.22.2 NameValuePair的构造方法	367	38.10 Container	386
37.23 NVList	367	38.10.1 IDL定义	386
37.23.1 IDL定义	368	38.10.2 Container的方法	388
37.23.2 NVList的方法	368	38.11 ContainerPackage.Description	392
37.24 OutputStream	369	38.11.1 ContainerPackage.Description	
37.25 Request	370	的变量	392
37.25.1 IDL定义	370	38.11.2 ContainedPackage.Description	
37.25.2 Request的方法	371	的方法	392
37.26 ServerRequest	373	38.12 DefinitionKind	392
37.26.1 IDL定义	373	38.12.1 DefinitionKind的方法	393
37.26.2 ServerRequest的方法	374	38.12.2 DefinitionKind的常量	393
37.27 TCKind	374	38.13 EnumDef	393
37.27.1 IDL定义	374	38.14 ExceptionDef	394
37.27.2 TCKind的方法	375	38.15 ExceptionDescription	394
37.28 TypeCode	375	38.15.1 ExceptionDescription的变量	395
37.28.1 IDL定义	375	38.15.2 ExceptionDescription的方法	395
37.28.2 TypeCode的方法	376	38.16 FixedDef	395
37.29 UnknownUserException	378		

38.17 FullValueDescription	396	38.32.2 PrimitiveKind的变量	409
38.17.1 FullValueDescription的变量	396	38.33 Repository	410
38.17.2 FullValueDescription的方法	397	38.34 SequenceDef	411
38.18 IDLType	397	38.35 StringDef	412
38.18.1 IDL定义	398	38.36 StructDef	412
38.18.2 IDLType的方法	398	38.37 StructMember	413
38.19 InterfaceDef	398	38.37.1 StructMemeber的变量	413
38.19.1 IDL定义	398	38.37.2 StructMemeber的方法	413
38.19.2 InterfaceDef的方法	399	38.38 TypedefDef	413
38.20 Interface DefPackage. FullInterface Description	400	38.39 TypeDescription	414
38.20.1 Interface DefPackage. Full- Interface Description 的变量	400	38.39.1 TypeDescription的变量	414
38.20.2 Interface DefPackage. Full- Interface Description的方法	401	38.39.2 TypeDescription的方法	414
38.21 InterfaceDescription	402	38.40 UnionDef	415
38.21.1 InterfaceDescription的变量	402	38.41 UnionMember	415
38.21.2 InterfaceDescription的方法	402	38.41.1 UnionMember的变量	416
38.22 IObject	403	38.41.2 UnionMember的方法	416
38.22.1 IDL定义	403	38.42 ValueBoxDef	416
38.22.2 IObject的方法	403	38.43 ValueDef	417
38.23 ModuleDef	403	38.44 ValueDescription	419
38.24 ModuleDescription	403	38.44.1 ValueDescription的变量	419
38.24.1 ModuleDescription的变量	404	38.44.2 ValueDescription的方法	420
38.24.2 ModuleDescription的方法	404	38.45 ValueMemberDef	420
38.25 NativeDef	404	38.46 WstringDef	420
38.26 OperationDef	404	第39章 关于激活的接口与类	422
38.27 OperationDescription	406	39.1 ActivationImplDef	422
38.27.1 OperationDescription的变量	406	39.1.1 IDL定义	422
38.27.2 OperationDescription的方法	406	39.1.2 ActivationImplDef方法	422
38.28 OperationMode	407	39.2 Activator	422
38.29 ParameterDescription	407	39.2.1 IDL定义	422
38.29.1 ParameterDescription的变量	408	39.2.2 Activator方法	423
38.29.2 ParameterDescription的方法	408	39.3 CreationImplDef	423
38.30 ParameterMode	408	39.3.1 IDL定义	423
38.31 PrimitiveDef	409	39.3.2 激活策略	424
38.32 PrimitiveKind	409	39.3.3 例子	424
38.32.1 PrimitiveKind的方法	409	39.3.4 环境变量	424
		39.3.5 显式传播或传递的环境变量	425
		39.3.6 CreationEmplDef方法	425
		39.4 ImplementationDef	427

39.5 OAD	427	41.5.1 IDL定义	446
39.5.1 IDL定义	427	41.5.2 Java定义	446
39.5.2 ImplementationStatus	428	41.5.3 ProxyPushConsumer方法	447
39.5.3 OAD方法	428	41.6 ProxyPullSupplier	447
第40章 关于命名服务的接口与类	433	41.6.1 IDL定义	447
40.1 NamingContext	433	41.6.2 Java定义	447
40.1.1 IDL定义	433	41.6.3 ProxyPullSupplier方法	447
40.1.2 NamingContext方法	433	41.7 ProxyPushSupplier	447
40.2 NamingContextExt	437	41.7.1 IDL定义	448
40.2.1 IDL定义	437	41.7.2 Java定义	448
40.2.2 NamingContextExt方法	438	41.7.3 ProxyPushSupplier方法	448
40.3 Binding和BindingList	439	41.8 PullConsumer	448
40.4 BindingIterator	440	41.8.1 IDL定义	448
40.4.1 IDL定义	440	41.8.2 Java定义	448
40.4.2 BindingIterator方法	440	41.8.3 PullConsumer方法	449
40.5 NamingContextFactory	441	41.9 PushConsumer	449
40.5.1 IDL定义	441	41.9.1 IDL定义	449
40.5.2 NamingContextFactory方法	441	41.9.2 Java定义	449
40.6 ExtendedNamingContextFactory	441	41.9.3 PushConsumer方法	449
40.6.1 IDL定义	442	41.10 PullSupplier	449
40.6.2 ExtendedNamingContext- Factory方法	442	41.10.1 IDL定义	449
第41章 关于事件服务的接口与类	443	41.10.2 Java定义	450
41.1 ConsumerAdmin	443	41.10.3 PullSupplier方法	450
41.1.1 IDL定义	443	41.11 PushSupplier	450
41.1.2 Java定义	443	41.11.1 IDL定义	450
41.1.3 ConsumerAdmin方法	443	41.11.2 Java定义	450
41.2 EventChannel	443	41.11.3 PushSupplier方法	451
41.2.1 Java定义	444	41.12 SupplierAdmin	451
41.2.2 EventChannel方法	444	41.12.1 IDL定义	451
41.3 EventLibrary	444	41.12.2 Java定义	451
41.3.1 Java定义	444	41.12.3 SupplierAdmin方法	451
41.3.2 EventLibrary方法	445	第42章 关于异常的类	452
41.4 ProxyPullConsumer	445	42.1 简介	452
41.4.1 IDL定义	446	42.2 SystemException	452
41.4.2 Java定义	446	42.3 UserException	453
41.4.3 ProxyPullConsumer方法	446	第43章 关于拦截器和对象包装器的 接口与类	455
41.5 ProxyPushConsumer	446	43.1 简介	455

43.2 拦截器管理器	455	43.15.1 import语句	463
43.3 IOR模板	455	43.15.2 ServerRequestInterceptor方法	463
43.4 InterceptorManager	455	43.16 ServerRequestInterceptorManager	464
43.5 InterceptorManagerControl	456	43.16.1 import语句	464
43.5.1 import语句	456	43.16.2 ServerRequestInterceptor- Manager方法	464
43.5.2 InterceptorManagerControl 方法	456	43.17 IORCreationInterceptor	464
43.6 BindInterceptor	456	43.17.1 import语句	465
43.6.1 import语句	456	43.17.2 IORCreationInterceptor方法	465
43.6.2 BindInterceptor方法	456	43.18 IORInterceptorManager	465
43.7 BindInterceptorManager	458	43.18.1 import语句	465
43.7.1 import语句	458	43.18.2 IORInterceptorManager方法	465
43.7.2 BindInterceptorManager方法	458	43.19 Location	465
43.8 ClientRequestInterceptor	458	43.20 ChainUntypedObjectWrapper Factory	466
43.8.1 import语句	458	43.20.1 IDL定义	466
43.8.2 ClientRequestInterceptor方法	458	43.20.2 import语句	466
43.9 ClientRequestInterceptorManager	459	43.20.3 ChainUntypedObjectWrapper- Factory方法	466
43.9.1 import语句	459	43.21 UntypedObjectWrapper	467
43.9.2 ClientRequestInterceptor Manager方法	459	43.21.1 IDL定义	467
43.10 POALifeCycleInterceptor	460	43.21.2 UntypedObjectWrapper方法	467
43.10.1 import语句	460	43.22 UntypedObjectWrapperFactory	468
43.10.2 POALifeCycleInterceptor方法	460	43.22.1 IDL定义	468
43.11 POALifeCycleInterceptorManager	461	43.22.2 import语句	468
43.11.1 import语句	461	43.22.3 UntypedObjectWrapper- Factory方法	468
43.11.2 POALifeCycleInterceptor- Manager方法	461	第44章 关于服务质量的接口与类	470
43.12 ActiveObjectLifeCycleInterceptor	461	44.1 PolicyManager	470
43.12.1 import语句	461	44.1.1 IDL定义	470
43.12.2 ActiveObjectLifeCycle- Interceptor方法	461	44.1.2 PolicyManager方法	470
43.13 ActiveObjectLifeCycle- InterceptorManager	462	44.2 PolicyCurrent	471
43.13.1 import语句	462	44.3 Object	471
43.13.2 ActiveObjectLifeCycle- InterceptorManager方法	462	44.3.1 org.omg.CORBA.Object方法	471
43.14 ForwardRequestException	462	44.3.2 com.inprise.vbroker.Object方法	472
43.15 ServerRequestInterceptor	462	44.4 RebindPolicy	473
		44.4.1 IDL定义	474
		44.4.2 策略值	474