# 在中间件中支持用户自定义连接子*

曹东刚[1+], 梅 宏[1], 曹建农[2]

[1](北京大学 信息科学技术学院 软件研究所,北京 100871)

[2](香港理工大学 电子计算学系,香港)

## Supporting User-Defined Connector in Middleware

CAO Dong-Gang[1+], MEI Hong[1], CAO Jian-Nong[2]

[1](Institute of Software, School of Electronics Engineering and Computer Science, Peking University, Beijing 100871, China)

[2](Department of Computing, The Hong Kong Polytechnic University, Hong Kong, China)

+ Corresponding author: Phn: +86-10-62757670, E-mail: caodg@sei.pku.edu.cn, http://www.pku.edu.cn

Received 2004-09-30; Accepted 2005-02-04

**Cao DG, Mei H, Cao JN. Supporting user-defined connector in middleware. *Journal of Software*, 2005,16(8): 1378−1385.** DOI: 10.1360/jos161378

**Abstract**: Connector is an important concept in software architecture design, but it doesn't get explicit support from runtime. This paper describes the work on supporting user-defined connector in J2EE application server PKUAS. The PKUAS connector model takes advantage of the Aspect technology. A construct named Advice is used to model crosscutting concerns related to interaction behaviours. These advices are organized into connector and are invoked dynamically at runtime. At deployment time, PKUAS will generate a client-side connector object and publish it to a naming service. Application clients download the connector object to a local machine to manage the component interactions. Runtime connectors correspond well to those in the design time, thus narrowing the concept gap greatly. This flexible and modular approach effectively facilitates the separation of concerns.

**Key words**: connector; middleware; application server; non-functional constrain

摘 要: 连接子是软件体系结构设计阶段的重要概念,但在运行阶段却缺少显式的支持.描述了在 J2EE 应用服务器 PKUAS 中显式支持用户自定义连接子的机制.PKUAS 连接子模型借鉴 Aspect 的思想,连接子由若干 Advice 组成,Advice 对各种交互约束进行建模,Advice 在运行时刻被动态调用.连接子在部署时刻由 PKUAS 动态生成并发布到名字服务中,由客户端在运行时刻下载到本地加载,负责构件交互.运行时刻的连接子可以和设计阶段的连接子相对应,从而保持概念的连贯性和一致性,体现了良好的关注点分离原则.

* Supported by the National Natural Science Foundation of China under Grant Nos.60125206, 60233010, 60303004 (国家自然科学基金); the National High-Tech Research and Development Plan of China under Grant No.2004AA113060 (国家高技术研究发展计划(863)); the National Grand Fundamental Research 973 Program of China under Grant No.2002CB312003 (国家重点基础研究发展规划(973)); the Key Grant Project of Sci. & Tech. Research of MOE of China under Grant No.0214 (国家教育部科学技术研究重大项目)

作者简介: 曹东刚(1975−),男,山东文登人,博士,讲师,主要研究领域为软件工程,Internet 技术;梅宏(1963−),男,博士,教授,博士生导师,CCF 高级会员,主要研究领域为软件工程,软件构件技术;曹建农(1964−),男,副教授,CCF 高级会员,主要研究领域为分布计算技术.

© 中国科学院软件研究所 http://www.jos.org.cn

关键词: 连接子;中间件;应用服务器;非功能约束
中图法分类号: TP311 文献标识码: A

在软件体系结构研究中,作为和构件同等重要的一阶实体,连接子得到了高度关注.但是,连接子的研究进展却并不尽如人意.目前连接子的研究大多局限于体系结构设计阶段的建模与验证,大多数体系结构描述语言只提供了很简单的连接子模型,连接子的行为也大多仅局限于通信.尤为重要的是,在软件生命周期的另一个重要阶段——运行时刻,一直缺少对连接子的显式支持机制.

中间件为分布应用程序提供了多种交互机制,如远程过程调用、消息传递等,并提供各种基础设施服务,例如安全服务、事务服务等,在某种意义上可以认为,中间件充当了连接子的角色.然而,中间件并没有显式的连接子的概念,因此从体系结构设计阶段到实现、部署和运行阶段存在着较大的鸿沟.同时,中间件对构件交互的支持多是通过预定义的通信基础设施实现,功能相对比较固定,用户难以扩展.

例如,在客户机/服务器交互模式下,中间件会分别为客户端程序和服务器端程序编译接口规约,产生 stub 和 skeleton,这些 stub/skeleton 在运行时刻代理实际的网络调用.由于 stub/skeleton 是由编译器产生的,用户通常不能直接修改,为了增加新的功能而又尽量不影响已有的业务代码,又出现了 Interceptor[1]和 Smart Proxy[1]等元编程技术.然而,Interceptor 和 Smart Proxy 技术需要客户掌握相当的底层知识,对于已有代码而言并不是完全透明的,仍然需要客户程序做预处理,例如注册 Interceptor、生成 Smart Proxy 等.同时,客户端必须要有 Interceptor 和 Smart Proxy 的代码,当一个分布应用程序有很多客户端的时候,对各个分布节点上的 Interceptor 和 Smart Proxy 代码的管理、维护和演化会变得很困难.

JINI[2]提供了一种服务代理动态发布与下载的机制.JINI 服务发布的时候,会把服务代理(service proxy)发布到 Lookup 服务中,客户程序通过 Lookup 服务将该服务代理透明地下载到本地,由该服务代理负责对目标服务的访问.JINI 的好处是客户端非常简洁,不需要了解任何交互细节,不需要编译生成 stub/skeleton 代码,所有的访问逻辑都被封装在动态下载的服务代理中,访问过程透明.然而,JINI 也没有 Connector 的概念,没有提供如何系统化地生成 JINI 服务代理以及扩展服务代理功能的方法.

本文描述了一种在 J2EE 应用服务器 PKUAS[3]中显式支持用户自定义连接子的技术.PKUAS 的连接子模型借鉴了面向 Aspect 程序设计[4]的思想,用户通过 Advice 定义各种功能或者非功能约束,这些 Advice 和构件的业务方法调用通过元编程技术在连接子中组织在一起.引入自定义连接子后,应用开发者和部署者就可以在不必修改已有业务逻辑的前提下,方便、灵活地在应用的部署或运行时刻在连接子中通过 Advice 动态添加新的处理逻辑.PKUAS 连接子作为运行时刻的一阶实体,从软件工程的角度为开发人员提供了看待底层中间件基础设施的视点,从而在一个较高的抽象层次提供了模块化的封装贯穿特性的机制.PKUAS 连接子的发布则借鉴了 JINI 的思想,连接子在应用部署时刻由应用服务器动态生成并发布到名字服务中,由客户端在运行时刻透明地下载到本地加载.运行时刻的连接子可以和设计阶段的连接子相对应,从而保持概念的连贯性和一致性.同时,这种方法体现了良好的关注点分离原则,交互机制灵活,模块化程度高,有利于系统的维护和演化.

本文第 1 节介绍相关研究工作.第 2 节介绍 PKUAS 的连接子模型.第 3 节介绍 PKUAS 连接子支持机制的设计和实现.第 4 节给出一个示例.最后一节总结全文.

## 1 相关研究工作

在软件体系结构的研究中,计算(构件)和交互(连接子)被分开,连接子成为和构件同等重要的一阶实体.然而长期以来,人们对于连接子(交互)并无深刻理解.Mehta 和 Medvidovic 等人深入分析了构件交互的本质及构件基本构成单位[5],并对现有连接子进行了分类:按照在交互中扮演的角色,连接子可以分为通信(communication)、协调(coordination)、转换(conversion)、协助(facilitation)等类型;按照实现交互服务方式的不同,连接子可分为过程调用、事件、数据访问、连接、流、仲裁器、分布器等类别.在连接子提供的众多服务中,数据传递和控制传递经常是共生的,是连接子的主要角色.其他角色可以认为具有贯穿(crosscutting)特性,主

© 中国科学院软件研究所 http://www.jos.org.cn

要对数据/控制传递进行约束.

在中间件支持连接子的研究方面,文献[6]分析了连接子和中间件提供的通信服务的关系,然而该文对连接子的讨论仅限于“通信服务”.DADO[7]描述了一种在中间件中支持可靠性、安全、日志等贯穿特性的方法,这些贯穿特性实际上可以和文献[5]中连接子的协调、转换、协助等角色对应.DADO 用一对 Adaplet 来对一个贯穿特性进行描述,在运行时刻,Adaplet 被分别附加到应用的 stub 和 skeleton 对象上.DADO 没有显式的连接子的概念,用 Adaplet 对贯穿特性进行描述也不如 Aspect 自然.并且,Adaplet 要附加到生成的 stub/skeleton 上,应用程序也必须事先知道 Adaplet 的代码,这就降低了 Adaplet 的灵活性.

针对当前中间件产品提供的交互机制难以扩展、灵活性不高的情况,文献[8]提出了一种在 CORBA 中支持移动代理(mobile proxy),即动态下载 stub 和 skeleton 的方法.文献[9]描述了一种轻量的在 RMI 中支持 Smart Proxy 和 Interceptor 的方法,J2EE 应用服务器 Jboss[10]也采用了类似的技术.然而,上述各种方法都有一个根本的问题:运行时刻没有显式的连接子的概念,也没有系统的对构件交互进行描述与规约的方法,从而造成了体系结构设计阶段到实现、部署和运行阶段的概念鸿沟.

南京大学提出用移动 Agent 来负责构件交互的 Cogent 后组装机制[11],具有灵活性强、动态调配性能好和对网络环境具有良好适应性的特点.不过该方法易受制于移动 Agent 技术本身的限制,同时该方法不支持符合工业标准的商业 COTS 可复用构件.

## 2 PKUAS 连接子模型

J2EE 应用服务器 PKUAS[3]显式支持连接子,连接子作为一个实体存在于运行时刻,并具有丰富的功能.PKUAS 连接子完成应用构件进行业务计算所需要的交互行为.连接子的交互行为主要是数据/控制的传递以及与之相关的各种约束.

一个应用系统会有若干关注点,其中最主要的是业务功能关注点(business functional concern),由构件完成.除此之外还有若干非功能关注点(non-functional concern),例如日志、安全、事务、可靠性等,通常贯穿于整个系统,被称为贯穿特性(crosscutting feature)[12].实际上,大多数贯穿特性是对交互的约束,这种约束可以是直接的,也可以是间接的.例如,日志记录的是交互情况,或者由于交互引起的状态改变;安全和事务也是针对交互的;负载均衡则要解决由于交互引起的负载分布问题.我们也将上述贯穿特性的行为称为交互行为.

PKUAS 借鉴 AOP(aspect oriented programming)[4]的思想,用 Advice 描述交互行为.由于交互通常是双边行为(多方参与的交互可以分解为若干双边交互),因此,PKUAS 的 Advice 分为源端(AdviceSource)和宿端(AdviceSink)两部分,二者协同完成一个完整的交互.例如,对于安全认证而言,需要源端准备认证信息,宿端接收到认证信息后,执行实际的认证.有的 Advice 只有源端行为,如传统 Smart Proxy 技术解决的客户端缓存等问题.有的 Advice 则只有宿端行为,如服务器端日志等.源端 Advice 和宿端 Advice 之间有时候需要交流信息,这些信息由业务调用的请求消息和返回结果捎带(piggy-back).

在 AOP 技术中,Aspect 代码(advice)需要和业务代码在特定的交互点(joint point)编排(weaving)到一起.PKUAS 用连接子在运行时刻将各 Advice 组织在一起.与 Advice 分为两部分对应,PKUAS 连接子也分为两部分:源端(ConnectorSource)和宿端(ConnectorSink).当构件进行业务调用时,源端和宿端连接子利用 Interceptor 技术,截取构件请求,调用各个 Advice,动态地将 Advice 与构件业务方法编排到一起.由于 PKUAS 在连接子中编织贯穿特性,因此不必像传统 AOP 技术那样修改构件源代码,支持构件黑盒复用.

图 1 描述了 PKUAS 的连接子模型,该模型包括 3 个平面:业务平面、贯穿平面和支撑平面.业务平面主要从系统功能的角度关注应用的业务逻辑.非功能属性等贯穿特性在贯穿平面和业务逻辑“贯穿”——连接子在执行构件业务交互时动态调用 Advice,对业务交互进行约束.通常,Advice 需要调用系统提供的公共服务,例如,中间件的安全服务、事务服务等,这些公共服务位于支撑平面.

PKUAS 连接子模型中的 3 个平面各有关注,体现了关注点分离(separation of concern)的软件工程思想:应用的业务开发者只需在业务平面关注业务逻辑,将安全、事务等贯彻特性交给贯穿平面.系统开发人员通过 Advice 描述贯穿特性,Advice 封装了对底层支撑平面公共服务的调用.PKUAS 连接子和 Advice 为系统开发人

© 中国科学院软件研究所 http://www.jos.org.cn

员提供了一种灵活的、模块化的贯穿特性处理机制.用户可以在不改动构件业务代码的情况下,在应用部署时刻定义新的连接子,引入新的 Advice,对构件交互施加各种约束,从而在保持构件源代码不变的情况下,改变应用的交互行为,支持黑盒构件复用.

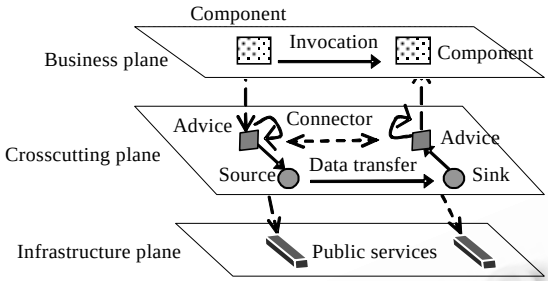


Fig.1 PKUAS connector model

图 1 PKUAS 连接子模型

## 3 系统设计与实现

J2EE 应用服务器 PKUAS 的连接子机制的一个重要原则是对应用程序代码透明.构件的提供者和使用者完全遵循 J2EE 规范进行开发,意识不到构件交互是由连接子机制负责的.

### 3.1 工作机制

PKUAS 的连接子工作机制借鉴了 JINI 的服务发布思想,即服务提供者(EJB 构件)负责提供对自己的访问逻辑,并通过查找机制(名字服务)对外发布.服务的使用者(客户程序)通过查找机制透明下载访问逻辑,客户程序不必在运行前编译生成 stub/skeleton 等分布计算设施,从而减轻了开发、管理、维护的负担.图 2 描述了 PKUAS 连接子从部署、生成、发布到客户端下载并实例化的完整工作流程.

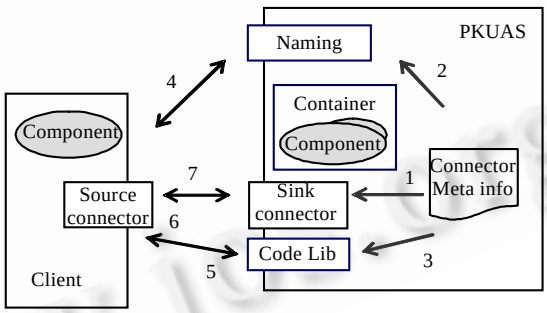


Fig.2 PKUAS connector working mechanism

图 2 PKUAS 连接子工作机制

在应用部署时刻,PKUAS 会解析部署描述文件 pkuas.xml,生成每个连接子的元信息,然后:

(1) 生成宿端连接子并和容器绑定.PKUAS 用一个容器管理一个构件类型的所有实例,这些实例共享一个连接子.容器和宿端连接子绑定之后,构件就可以接受外界的服务请求了.

(2) 生成并发布源端连接子对象.源端连接子封装了客户端的构件交互逻辑.PKUAS 首先实例化源端连接子对象并序列化,以二进制的形式发布到名字服务中,供客户下载.

(3) 发布源端连接子代码.Java 对象的实例化需要首先装载对象的类代码.PKUAS 连接子的构成是动态的,可能包含若干客户自定义的 Advice.而客户程序可能并没有这些 Advice 代码,因此 PKUAS 需要将相关的代码(class 文件)发布到代码库中,供客户程序下载.PKUAS 内置了一个 http 服务器,提供代码下载服务.

(4) 客户程序在名字服务中查找并下载源端连接子对象.此时,源端连接子对象是以序列化(serialized)的形

© 中国科学院软件研究所 http://www.jos.org.cn

式存在,需要实例化为内存中的对象后才可以使用.

(5) 客户程序从代码库中下载实例化源端连接子所需要的 Advice 代码.

(6) 将序列化的源端连接子对象实例化为内存中的对象.

(7) 连接子准备就绪,此时构件就可以进行交互了.

### 3.2 连接子与Advice

PKUAS 的连接子实现技术综合运用了 Java 反射技术、动态代理技术与 RMI 技术,以及 Smart Proxy 与 Interceptor 等元编程技术.

源端连接子的关键技术是要让客户程序能按照已有的Java语义进行调用,将从名字服务处下载的对象cast到构件的接口类型.PKUAS 利用 Java 的动态代理技术(dynamic proxy)实现该目标,在应用部署时刻根据构件的接口为构件生成动态代理,该动态代理可以cast到构件的接口类型.动态代理在创建时注册了InvocationHandler回调句柄,当接收到用户调用请求时,动态代理会将该方法调用转化为用元数据表示的消息请求 Invocation,然后将该消息派发到InvocationHandler的invoke方法.ConnectorSource继承了InvocationHandler接口,在其invoke方法中将调用请求信息封装到Invocation对象中,以此作为参数调用AdviceSource的advice方法进行截取处理.在图 2 中,发布到名字服务中的对象其实是动态代理,但是由于动态代理是匿名的,其中心在于InvocationHandler,因此我们就用 ConnectorSource 指代动态代理.

AdviceSource 组织成一个链表,被 ConnectorSource 依次调用,分别完成对各关注点的处理逻辑.AdviceSource 可以在 Invocation 中捎带信息给对端的 AdviceSink,AdviceSink 也可以在返回的元消息 Result 中捎带消息给 AdviceSource.

宿端连接子 ConnectorSink 的机制相对简单,主要是要和负责底层通信的 daemon 绑定,还有就是在将请求派发到目标构件之前,委托容器调用 AdviceSink 链表进行截取处理.细节不再详述.

### 3.3 部署描述信息

应用组装者和部署者在部署描述文件中描述连接子的构成.图 3 给出了部署描述文件 pkuas.xml 中连接子的 schema 定义,其中 Advices 中的 advice 描述了需要加载的 Advice 类名.组装者或部署者在不必改动构件源代码的情况下,只需在连接子中定义不同的 Advices,即可为构件交互添加各种约束,甚至改变构件交互行为.第 4 节给出了一个连接子的定义示例.

```
〈xs:complexType name=“Connector”〉
  〈xs:sequence〉
    〈xs:element name=“source” type=“Advices” minOccurs=“0”/〉
    〈xs:element name=“sink” type=“Advices”minOccurs=“0”/〉
  〈/xs:sequence〉
〈/xs:complexType〉
〈xs:complexType name=“Advices”〉
  〈xs:sequence〉
    〈xs:element name=“advice” type=“xs:string” minOccurs=“0”/〉
  〈/xs:sequence〉
〈/xs:complexType〉
```

Fig.3 PKUAS connector schema

图 3 PKUAS 连接子 schema

### 3.4 性能分析

本节将衡量并分析 PKUAS 连接子机制的性能.不考虑线程调度策略、缓存等因素的影响,设计如下的测试用例:一个独立(standalone)客户端构件向服务器端构件发送固定长度的消息(参数长度为 1,100,200,…,1000 字节);服务器端构件接收到消息后返回确认消息;客户端构件记录请求的响应时间.为了减小误差,客户端连续调用 10 000 次,记录这 10 000 次总共的请求响应时间.

服务器配置为:800MHz Intel Pentium 处理器,256M 内存,Redhat Linux 8.0.客户机配置为:1.4GHz Intel Pentium M 处理器,512M 内存,Microsoft Windows XP Professional SP1a.服务器和客户机运行 Java 2 SDK

© 中国科学院软件研究所 http://www.jos.org.cn

1.4.2,100M 以太网.

首先,我们以开放源码 J2EE 应用服务器 JBoss 为基准进行比较.在该次测试中,PKUAS 连接子除了通信 Advice 外没有安装额外的 Advice.图 4 给出了 PKUAS 连接子和 JBoss 3.2.5 的性能对比结果.从图中可以看出,采用连接子机制的 PKUAS 的响应时间比 JBoss 高大约 5%~8%.由于 PKUAS 连接子机制没有进行任何优化,作为原型系统,这个结果是可以接受的.

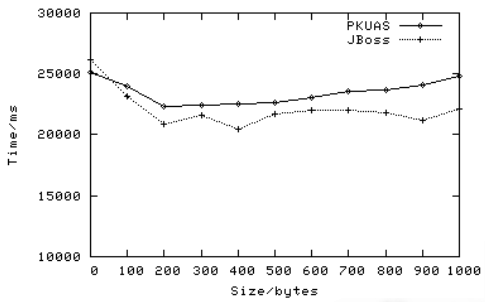


Fig.4 PKUAS vs JBoss

图 4 PKUAS 和 JBoss 性能对比

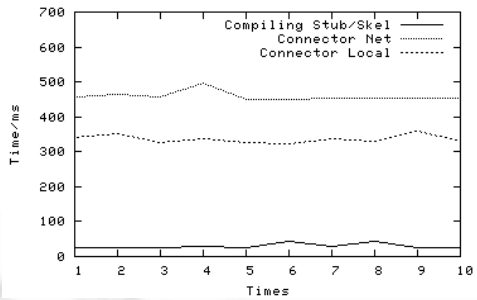


Fig.5 Connector loading performance

图 5 Connector 下载启动时间

然后,我们衡量连接子的下载启动开销,结果如图 5 所示.“Compiling stub/skel”是 PKUAS 在采用通常的编译生成 stub/skeleton 的方式下,加载 stub/skeleton 的时间.其他两个曲线是 PKUAS 连接子的加载情况,其中,“Connector Net”表示 PKUAS 客户端和服务器分布在两台机器的情况下,PKUAS 客户端下载连接子代码并加载的时间,“Connector Local”表示 PKUAS 客户端和服务器在一台机器的情况下,从本地加载连接子的时间.从图中可以看出,PKUAS 加载连接子的时间大约是加载预编译生成的 stub/skeleton 时间的 8~10 倍.不过,应用程序在初次将连接子代码下载到本地后,在随后的调用中就可以从本地直接加载,无须再次下载.对于长生命周期的应用程序,连接子的下载开销占整个开销的比例很小.

从上面的测试结果中可以看出,PKUAS 连接子确实在性能方面有损失.不过在实际系统中,应用开发人员可以很方便地设计各种提升性能的 Advice,例如客户端缓存、客户端计算等.在带来灵活性和简便性的同时,必然会带来一定的性能上的损失,用户需要进行折衷.

## 4 应用示例

本节通过一个简单的 Time 应用演示如何在 PKUAS 中自定义连接子.

Time 应用业务逻辑很简单,服务器端构件 SesTimeBean 对外提供 time 服务,返回系统当前时间.Time 应用的重点在于其如下非业务功能需求:1) 安全认证:只有合法的客户才能获得服务;2) 时间度量:记录 time 请求的传输时间以及在服务器端的执行时间.

时间度量通常的做法,是客户程序计算本次调用总共的时间(传输时间+服务执行时间),服务器记录请求在服务器端的服务执行时间,然后用总时间减去服务执行时间得到传输时间.然而这种方法存在两个问题:首先,需要修改应用程序源代码;其次,两部分的时间是由物理分布的节点分开计算的,服务器的计算结果不能马上传递给客户程序,通常是写到 Log 中,这就失去了时效性.

PKUAS 的连接子机制提供了灵活、方便的处理方法.用户无须修改应用程序源代码,只需定义两个 Advice:宿端的 MetricAdviceSink 在 Result 中把服务执行时间捎带给源端的 MetricAdviceSource,即可计算出网络传输时间;然后,在部署描述文件 pkuas.xml 中修改连接子配置即可,从而以简洁的模块化方式解决了前面提到的两个问题.限于篇幅,图 6 只给出了时间度量 Advice 的部分代码,图 7 给出了连接子配置.

这种方式同样可以应用于实时性能监控、负载均衡等场合.

© 中国科学院软件研究所 http://www.jos.org.cn

## 5 结束语

连接子作为软件体系结构设计阶段的重要概念,应该在运行阶段得到中间件的显式支撑.本文描述了在J2EE应用服务器PKUAS中显式支持用户自定义连接子的机制.PKUAS借鉴Aspect的思想,用户通过Advice定义各种非功能约束,这些Advice和构件的业务方法调用在连接子中组织在一起.应用系统开发者和部署者就可以在不必修改应用系统已有的业务逻辑的前提下,方便、灵活地在应用系统的部署或运行时刻通过Advice动态添加新的处理逻辑.连接子作为运行时刻的一阶实体,从软件工程的角度为开发人员提供了看待底层中间件基础设施的视点,从而在一个较高的抽象层次提供了模块化的封装贯穿特性的机制.PKUAS连接子的发布借鉴了JINI的思想,连接子在部署时刻由应用服务器动态生成并发布到名字服务中,由客户端在运行时刻透明地下载到本地加载.运行时刻的连接子可以和设计阶段的连接子相对应,从而保持概念的连贯性和一致性.同时,这种方法体现了良好的关注点分离原则,交互机制灵活,模块化程度高,有利于系统的维护和演化.

```
public class MetricAdviceSource extends EJBAdviceSource {
  public Result advice(Invocation mi) throws Exception {
    long bef = System.currentTimeMillis();
    Result result = getNext().advice(mi);
    long aft = System.currentTimeMillis();
    long total = aft−bef;
    ContextKey key = new ContextKey(“Metric”, 101);
    Object ctx = result.getPiggybackContext(key);
    Long ttran = total−((Long)ctx).longValue();
    return result;
  }
}
```

```
public class MetricAdviceSink extends EJBAdviceSink {
 public Result advice(Invocation mi) throws Exception {
  long bef = System.currentTimeMillis();
  Result result = getNext().advice(mi);
    long aft = System.currentTimeMillis();
    ContextKey key = new ContextKey(“Metric”, 101);
    result.setPiggybackContext(key, new Long(aft-bef));
    return result ;
  }
}
```

Fig.6 Part of MetricAdvice source code in application Time

图6 Time应用的度量Advice部分代码

```
〈connector〉
 〈source〉
  〈advice〉pku.as.application.advice.source.StatelessEJBAdviceSource〈/advice〉
  〈advice〉pku.as.connector.advice.SecurityAdviceSource〈/advice〉
  〈advice〉pkuas.test.time.MetricAdviceSource〈/advice〉
 〈/source〉
 〈sink〉
  〈advice〉pku.as.application.advice.sink.SecurityEJBAdviceSink〈/advice〉
  〈advice〉pkuas.test.time.MetricAdviceSink〈/advice〉
 〈/sink〉
〈/connector〉
```

Fig.7 Connector configuration for Time

图7 Time应用中连接子配置

**References**:


[1] Wang N, Parameswaran K, Schmidt D. The design and performance of meta programming mechanisms for object request broker middleware. In: Proc. of the 6th USENIX Conf. on Object-Oriented Technologies and Systems (COOTS 2001). 2001.

[2] SUN Microsystem. Jini Architecture Specification, Version 1.2, 2001.

[3] Huang G, Wang QX, Cao DG, Mei H. PKUAS: A domain-oriented component operating platform. Acta Electronica Sinica, 2002,30(12A):1938−1942 (in Chinese with English abstract).

[4] Kiczales G, Lamping J, Mendhekar A, Maeda C, Lopes CV, Loingtier J-M, Irwin J. Aspect oriented programming. In: Proc. of the 11th European Conf. on Object-Oriented Programming (ECOOP’97). 1997. 220−243.

[5] Mehta NR, Medvidovic N, Phadke S. Towards a taxonomy of software connectors. In: Proc. of the 22nd Int’l Conf. on Software Engineering (ICSE 2000). 2000. 178−187.

[6] Dashofy EM, Medvidovic N, Taylor RN. Using off-the-shelf middleware to implement connectors in distributed software architectures. In: Proc. of the 21st Int’l Conf. on Software Engineering (ICSE’99). 1999. 3−12.


© 中国科学院软件研究所 http://www.jos.org.cn

[7] Wohlstadter E, Jackson S, Devanbu P. Dado: Enhancing middleware to support crosscutting features in distributed, heterogeneous systems. In: Proc. of the 25th Int'l Conf. on Software Engineering (ICSE 2003). 2003. 174−186.

[8] Aziz B, Jensen C. Adaptability in CORBA: The mobile proxy approach. In: Proc. of the Int'l Symp. on Distributed Objects and Applications. 2000. 295−304.

[9] Santos N, Marques P, Silva L. A framework for smart proxies and interceptors in rmi. In: Proc. of the 15th ISCA Int'l Conf. on Parallel and Distributed Computing Systems (ISCA PDCS-02). 2002.

[10] JBoss Group, 2004. http://www.jboss.org

[11] Hu HY, Yang M, Tao XP, LÜ J. Research and implementation of late assembly technology in cogent. Acta Electronica Sinica, 2002, 30(12),1823−1827. (in Chinese with English abstract).

[12] Cao DG, Mei H. Aspect-Oriented Programming—A New Programming Paradigm. Computer Science, 2003,30(9):1565−1572 (in Chinese with English abstract).

附中文参考文献:

[3] 黄罡,王千祥,曹东刚,梅宏.PKUAS:一种面向领域的构件运行支撑平台.电子学报,2002,30(12A):1938−1942.

[11] 胡海洋,杨玫,陶先平,吕建.Cogent 后组装技术研究与实现.电子学报,2002,30(12):1823−1827.

[12] 曹东刚,梅宏.面向 Aspect 的程序设计——一种新的编程范型.计算机科学,2003,30(9):1565−1572.

© 中国科学院软件研究所 http://www.jos.org.cn