面向方面的SOA构件装配方法研究与实现①

邓子云　陈玉林　杨晓峰 (湖南现代物流职业技术学院 物流信息系 湖南 长沙 410131)

摘　要： 异种构件间协议解耦和构件装配的问题一直困扰着软件开发人员，Web服务组合和SCA装配规范提供了可选的解决方案，但对问题的解决都还存在一些局限性。为此，提出了面向方面的SOA构件装配方法的思想，借助实现了SCA规范的Tuscany和AOP软件Spring，给出了解决问题的思路，利用银行的内部转账的业务实例验证了思想与解决问题思路。

关键词： 面向方面编程；构件装配；协议解耦；实例验证

Research and Implementation of Aspect Oriented Assembly Methods for SOA Components

DENG Zi-Yun, CHEN Yu-Lin, YANG Xiao-Feng

(Logistics Information Department, Hunan Modern Logistics Occupation Technical College, Changsha 410131, China)

Abstract: The problem of protocol decoupling and assembling for different components had plagued the software developers. Web service composition and SCA assembly specification provided optional solutions. However, these solutions have limitations. In this paper, the idea of the assembly about SOA component is put forward. Through Tuscany with SCA specification and Spring with AOP, this paper proposes a solution to the problem. It also verifies the solution through the bank's internal transfer of business.

Keywords: AOP; component assemble; protocol decoupling; verify with an example

SOA(Service Oriented Architecture，面向服务的架构)是一种面向服务的架构方法，SOA的核心概念是重用和互操作，它将企业的IT资源整合成可操作的，基于标准的服务，使其能被重新组合和应用[1]。

服务并非仅指Web Service，Web服务只是服务中的最为典型和常用的一种，其它很多的构件封装形式也可以称为服务。一个构件向外界暴露接口以供访问，这个构件就称为一个服务。

1　构件互操作需解决的关键性问题

构件的种类有很多，如EJB(Enterprise Java-Bean，企业级JavaBean构件)、JMS(Java Messaging Service，Java消息服务)、COM/DCOM、CORBA (Common Object Request Broker Architecture，公共对象请求代理体系结构)等，异种构件的互操作与访问问题一直困扰着软件开发领域的工程师们，这要解决以下的主要2个关键性的问题：

(1) 异种构件间协议解耦的问题。比如EJB构件与Web Service构件，访问EJB常使用RMI(Remote Method Invocation，远程方法调用)方式，而访问Web Service则是采用的SOAP(Simple Object Access Protocol，简单对象访问协议)方式，如果两者要互操作就要解决一个协议转换的问题。

(2) 构件装配的问题。面对不同技术实现的异种构件，如何在分布式环境下将这些构件装配起来，形成更大的构件或组合成应用，这需要找到解决的办法。

① 基金项目:湖南省"十一五"重点科技计划(2008GK2019)

收稿时间:2009-09-16;收到修改稿时间:2009-11-02

2 解决问题的思路

针对上文提出的 2 个关键性的问题，有人提出了各种不同的解决办法，也建立起了一起标准的规范。

2.1 Web 服务组合

WS-BPEL(Web Services Business Process Execution Language，Web 服务业务流程执行语言)用于将 Web Service 进行服务编制(Orchestration)和服务编排(Choreography)[2]。服务编制用总控过程来控制涉及到的 Web 服务，并协调 Web 服务不同操作的执行；服务编排方式集中在消息的交换，不依赖中央的总控协调过程。也有学者在研究 Web Service 组合的算法、自动组合、发现算法，比如基于语义的 Web 服务[3]。

2.2 SCA 与 Tuscany

SCA(Service Component Architecture，服务构件架构)规范是一个基于 SOA 思想来描述了建立应用和系统的规范集合，这个标准是开放式的，主要围绕如何实现和扩展服务来展开说明[4]。

在 SCA 中，构件通过引用、提升、连线等处理，形成更大的组合构件，比如如图 1 所示。在这个图中，构件 A 引用了构件 B，构件 A 和构件 B 这两个构件形成了组合构件 A。构件 A 的服务接口被提升为组合构件 A 的服务接口，可以绑定 Web Service、JMS 等协议供访问。构件 B 的引用被提升为组合构件 A 的引用。构件 A 的一个属性和构件 B 的一个属性被设置为组合构件 A 的属性。

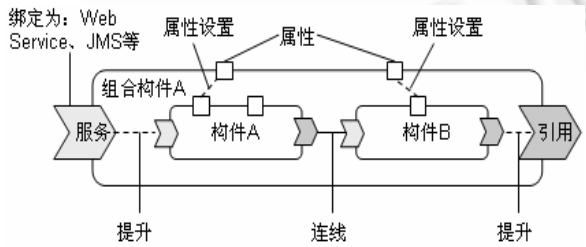

图 1 组合构件装配图示例

SCA 中的构件可以绑定为不同的协议来作为服务对外发布，比如绑定为 Web Service、RMI、Ajax、CORBA、EJB 等；构件的技术实现也可以采用各种不同的编程语言，如 Java、BPEL、JavaScript、Ruby、Python、Groovy、OSGI 等。

2.3 AOP 与 Spring

通过 Spring 提供的 AOP(Aspect Oriented Programming，面向方面编程)功能，可以方便地进行面向方面的编程，许多不容易用传统 OOP(Object Oriented Programming，面向对象编程)实现的功能可以通过 AOP 轻松应付[5]。Spring AOP 构建于 IoC (Inversion of Control，控制反转)之上，并和 IoC 浑然一体。

AOP 通过切点(Pointcut)来定位连接点，一个切点可以匹配多个连接点。在 Spring 中切点通过 org.springframework.aop.Pointcut 接口来描述。Spring 通过切点来向多个连接点织入增强。连接点(Join- point)是指的程序执行的某个特定位置，比如类的某个方法调用之前、调用之后、方法抛出异常之后等。

增强(Advice)是织入到连接点上的一段代码。Spring 中有多种增强，如：前置增强、后置增强、异常增强、引介增强等。切面(Aspect)由切点和增强两个部分组成，如图 2 所示。

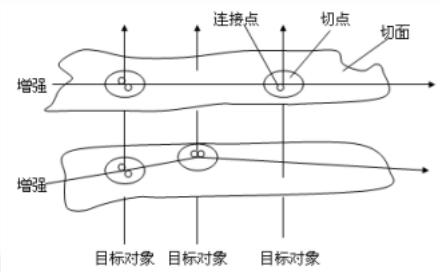

图 2 AOP 思想描述图

那么切点又如何进行匹配呢？比较常用的方法是使用切点表达式。切点表达式由关键字和操作参数组成，如切点表达式 execution(* sayHello(..))中，execution 是关键字，“* sayHello(..)”为操作参数，而 execution()实际上是一个切点函数。

2.4 面向方面的 SOA 构件装配

Web 服务的组合方式目前存在的问题主要有：(1)许多内容尚不成熟，离工程应用还有一定的距离；(2)专门面向 Web 服务而不能直接处理多种协议的异种构件。

SCA 从 2007 年 7 月发布 SCA 规范 1.0 正式版至今也还只有 2 年多的时间，实现 SCA 的容器还很少，但已有开源的实现，如 Tuscany Java SCA[6]。

构件与构件之间通过引用建立起了一种依赖关系，不过这种依赖关系是通过配置文件来声明的，而不是写在软件程序中，这些就具有了相当好的柔性。不过在 SCA 构件装配规范下构件的装配仍然不够理想，主要存在有以下的问题：

(1) 构件分布在各种容器中，通过引用建立起依赖关系，对于比较常用的构件，被引用时需要重复声明。

(2) SCA 的构件工厂在生成构件实例时需要采取单例、复例、Web 应用、Web 会话等多种生命周期模型，以面对复杂的应用需要，而现有的 Tuscany Java SCA 容器尚不支持。

面向方面的 SOA 构件装配思想是要将 SCA 与 AOP 两种理念综合运用起来，比如将 Tuscany Java SCA 与 Spring 集成起来将形成很好地解决上述问题的办法，解决问题的思路如下：

(1) 用 Spring 作为面向方面的中间件软件，将构件工厂的角色交给这种中间件软件来完成，从而实现多种生命周期模型；特别是细粒度的构件应尽量交给面向方面容器来管理，在其它构件需要进行增强时，即行织入。

(2) 协议解耦的工作交给 SCA 中间件软件(如 Tuscany)来实现，以方便将构件对外以各种服务方式发布供外界访问。

(3) 构件装配工作尽量利用 AOP 方式在 Spring 容器中完成，粗粒度且为复例方式的构件可考虑放在 SCA 容器中。

3 面向方面的SOA构件装配实例

为证明前文所述构件装配思路，以银行的内部转账业务为例进行了测试。银行的内部转账的业务可简单地理解为从一个账户上转钱到另一个账户上，也就是说，一个账户要减钱，另一个账户要加钱，两个账户金额的变动数额相等，这样就能平账(实际上银行的转账操作一般采用复式记账法，会有中转账户，以便于监控资金结算情况，因此会有多笔会计分录产生)。

3.1 设计思路

现在想这样来设计，将转账操作作为一个 Web Service 构件用 Tuscany Java SCA 来对外发布，转账操作的构件和账户构件均由 Spring Bean 来实现，它们之间的调用关系如图 3 所示。

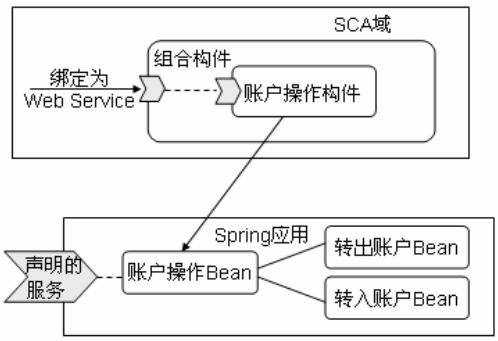

图 3　转账操作相关的构件设计图

3.2 配置情况

根据这个设计思路，composite 文件中的关键配置内容如下。

```
<!--服务名称与提升设置-->
<service name="accountOperationService"
  promote="accountOperation">
  <!--Java 类型接口设置-->
  <interface.java interface="spring. AccountOperationInterface"/>
  <!--绑定为 Web Service，并设置 URI-->
  <binding.ws uri="http://localhost:8089/_/ExchangeMoneyWebService"/>
</service>
<!--组件名称设置-->
<component name="accountOperation">
<!--组件用 Spring 实现，并定位 Spring 中的 XML-->
<implementation.spring location="spring/springContext.xml"/>
</component>
```

以上配置设置 accountOperation 构件由 Spring 来实现，并将 accountOperation 构件绑定为 Web Service 对外发布。Spring 的配置文件中的关键代码片断如下。

```
<!--SCA 服务名称、类型与目标 Bean 设置-->
<sca:service name="accountOperation- Service"
  type="spring.AccountOperationInterface"
  target="accountOperation"/>
<!--Bean ID 与实现类设置-->
<bean id="accountOperation"
  class="spring.AccountOperation">
```

```
    <!--Bean 组件的属性设置-->
    <property name="add_a" ref="add_a_ bean"/>
    <property name="sub_b" ref="sub_b_ bean"/>
  </bean>
  <!--Bean ID 与实现类设置-->
  <bean id="add_a_bean" class="spring.Account">
    <!--此处省去账户属性设置值的配置代码-->
  </bean>
  <!--Bean ID 与实现类设置-->
  <bean id="sub_b_bean" class="spring.Account">
    <!--此处省去账户属性设置值的配置代码-->
  </bean>
```

3.3 切入的日志记录

如果要在转账前和转账后进行日志记录，记录当前账户的资金情况，该如何做成切面呢？这需要开发一个增强类，代码如下所示。

```
AdviceExchangeMoney.javapackage spring;
/**
* 转账交易的增强类
**/
public class AdviceExchangeMoney {
  public Account add_a;//转入账户
  public Account sub_b;//转出账户
  //前置增强方法
  public void before_exchange_money(float money){
  /*转账前的日志代码*/
  }
  //后置增强方法
  public void after_exchange_money(float money){
  /*转账后的日志代码*/
  }
  /*此处省去本类所有属性 get 与 set 方法源代码*/
}
```

增强类有 2 个属性，对应着转入账户对象和转出账户对象，从而可以在增强方法中得到转入账户和转出账户的信息。前置增强方法和后置增强方法都有一个参数 money，即为转多少钱。

在 Spring 应用的配置文件中有关增强的配置内容如下：

```
<aop:config>
  <!--引用的切面-->
  <aop:aspect ref="adviceExchangeMoney">
    <!--切点定义-->
    <aop:pointcut id="exchangeMoneyPointcut" expression="target(spring.AccountOperation) and execution(* exchange_money(float)) and args (money,..)"/>
    <!--前置增强设置-->
    <aop:before method="before_exchange_money" pointcut-ref="exchangeMoneyPointcut"/>
    <!--后置增强设置-->
    <aop:after method="after_exchange_money" pointcut-ref="exchangeMoneyPointcut"/>
  </aop:aspect>
</aop:config>
<!--被增强的 Bean-->
<bean id="adviceExchangeMoney" class="spring.AdviceExchangeMoney">
  <!--引用转入账户 Bean-->
  <property name="add_a" ref="add_a_bean"/>
  <!--引用转出账户 Bean-->
  <property name="sub_b" ref="sub_b_bean"/>
</bean>
```

adviceExchangeMoney 是声明的增强类的 Bean 对象，引用了配置文件中此前声明的 add_a_bean 和 sub_b_bean 这 2 个 Bean 对象，因为 Spring 配置是使默认用的 singleton(单例)作用域，所以实际上 adviceExchangeMoney 构件和 accountOperation 构件引用的是相同的两个账户，从而方便得到同样的账户信息，如图 4 所示。

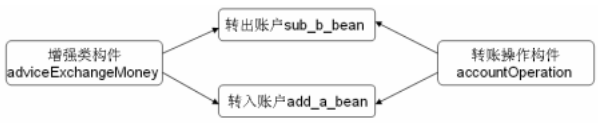

图 4 账户构件被引用的情况

声明的切面引用了 adviceExchangeMoney 增强类构件。切点 exchangeMoneyPointcut 表达式表示

目标类为 spring.AccountOperation(账户操作类)，当执行 exchange_money()方法，该方法的有且仅有一个参数，参数类型为 float，参数名称为 money 时进行增强。

前置增强执行的增强方法为 before_exchange_money，切点为 exchangeMoneyPointcut；后置增强执行的增强方法为 after_exchange_money，切点为 exchangeMoneyPointcut。

3.4 实现结果

根据前文所述的设置思想、配置情况及切入的构件情况，通过测试过实现了面向方面的 SOA 构件装配，其中 SOA 服务器端程序运行的结果如图 5 所示。

图 5　装配实例实现结果

4　结语

面向方面的 SOA 构件装配方法将构件在分布式应用的情况下，将构件的功能进行增强和装配，SOA 中间件软件进行协议解耦，而支持面向方面功能的中间件软件以 SOA 构件为对象进行织入、装配和增强。

使用 Tuscany 和 Spring 配合将能较好地实现面向方面的 SOA 构件装配方法，但毕竟 Spring 是紧密耦合的 AOP 工具，也还需要再进一步改进，比如改造为分布式的 SOA 切面工具。

参考文献

1 李长青,张为群.基于SOA 的异构软件自动化测试方法研究.计算机科学, 2007,(12):278－282.

2 Moitra D, Ganesh J. Web services and flexible business process: Towards the adaptive enterprise. Information and Management, 2005,42:921－933.

3 刘剑.面向服务体系结构的服务重组关键技术研究[博士学位论文].武汉:华中科技大学,2006.

4 Chappell D. Introducing SCA.July 2007. http://www._davidchappell.com/articles/IntroducingSCA.pdf

5 邓子云.贯通Java Web轻量级应用开发:JSP+ Struts+ Hibernate+Spring 实例精解.北京:电子工业出版社,2008.530.

6 Tuscany SCA Java. [2009-10-22]. http://tuscany.apache.org/sca-java._html