

# 一种面向对象的 Web Service API 框架设计

王 雄,褚 伟

(合肥工业大学 计算机网络系统研究所,安徽 合肥 230009)

**摘 要:**通过设计一种面向对象的 Web Service API 框架,使得 Service 调用者通过特定的命令和参数可实现对 Web Service 内部对象的操作,并且这些对象的数据可以通过 XML 格式返回给调用者。传统 Web Service API 设计中存在着接口数与对象数正相关的问题,随着系统中对象的增加,面向功能的接口设计方法会不断地增加实现了相应对象操作方法的接口,而面向对象的 Web Service API 框架可以使用最少的接口实现系统功能,有效地控制了接口的数量,缩短了 Web Service 调用者的学习曲线。这一框架同时实现了对 Session、Cache 和事务管理,并且提供安全验证机制,以及数据合法性的验证机制,有效地封装了业务逻辑和数据存取信息。利用这一 API 框架可以在不增加接口数量的前提下,通过配置文件来添加对象,为系统提供了良好的可拓展性。

**关键词:**Web 服务;接口设计;面向对象;应用程序接口;文档对象模型

**中图分类号:**TP31

**文献标识码:**A

**文章编号:**1673-629X(2009)08-0054-04

## Design of Object Oriented Web Service API Framework

WANG Xiong, CHU Wei

(Institute of Computer Network Systems, Hefei University of Technology, Hefei 230009, China)

**Abstract:** Web service caller can manipulate the object in the web service by many commands defined by the object oriented web service API framework, and the data of the object will return to the caller by the XML format. In traditional web service API design, there is a problem that the amount of the interfaces are increasing when adding more objects in the system. This function oriented design must add some new interfaces of the object to implement the manipulation of the object's attributes, but the object oriented method can use the limited interfaces to implement the functions of the system, and shorten the learning curve of the web service caller. This framework implements the management of session, cache, and transaction, and applies the mechanism of security validation, data validation, encapsulation of the business logic and the persistence of the data to the database, the system can be extendible by adding configuration files of other objects without increasing any other new interfaces.

**Key words:** web service; interface design; object oriented; API; DOM

## 0 引 言

Web 服务是自包含的、自描述的、模块化的应用程序,可以通过 Web 发布、定位和调用。从本质上看,Web Service 是一种完全基于 XML 的技术,应用系统之间可以通过在 Internet 上传送基于 XML 的消息进行互操作<sup>[1,2]</sup>。从客户的角度而言,Web Service 可以被看作一个部署在 Internet 上的对象或者组件。其他系统或组件可以通过 Web Service 暴露在外的接口对该 Web Service 进行访问,用户可以通过 WSDL 文件获取 Web Service 的接口描述<sup>[3]</sup>。

接口是一个系统或对象与外界的接触面,表达了系统或对象的外部观点。由于不同用户对同一个系统有不同的观点,而且同一用户以其不同的目的来使用系统时,也会有不一样的观点。因此,一个系统或对象通常都会有许多个接口,来满足不同目的的用户需要。Web Service 是一种软件系统对系统间的沟通接口,它不是用户接口,而是为前端程序提供服务的。图 1 为这个简单的学生管理方案例子。

可以看到,例子中定义了两个服务程序(没有包含任何实现细节)。这些服务程序的接口都是为了完成任务而定制的。如果客户端程序试图使用这些服务,那它必须针对这些特定接口进行编码——不可能在这些接口定义之前,使用客户程序去有目的地和接口协作。如果能将这些对具体对象的操作进一步地提升到对象无关的层次,则操作不需要与各个对象进行绑定,

收稿日期:2008-11-28;修回日期:2009-03-11

基金项目:国家自然科学基金(90718037)

作者简介:王 雄(1981-),男,湖北黄梅人,硕士研究生,主要研究方向为 J2EE、SOA;褚 伟,副研究员,博士,研究方向为嵌入式系统,计算机网络。

而可以使用统一的接口来处理所有的对象。

从 Web Service 来看,软件系统的各项功能都成为公共的服务,通常并不知道该服务的用户是谁。Web Service 的用户在需求上有明确的目的性,但是在如何使用这一服务时却又面临的多样性,尤其在 SaaS 模式中,服务提供者提供的服务是一个庞大的复杂的功能系统,这就很难确定服务的需求者将会怎样组合系统中的功能,正是由于用户需求的这种不确定性使得面向功能的 Web Service 的接口设计难以符合需求者的个性化要求。同时,在传统的面向功能的或组件的接口设计中,接口的数量会随着对象的增加而不断地增多,这对 Web Service 的调用者来说无疑增加了学习成本,而面向对象的设计思想为 Web Service 的 API 设计提供了另一种思路。

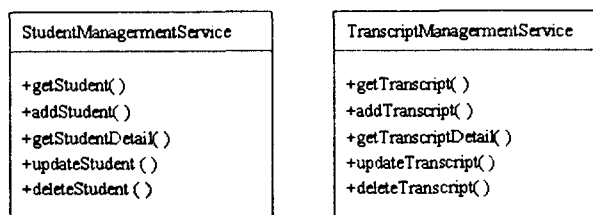


图 1 两个服务程序

## 1 面向对象的 Web Service API 设计思路

Web Service 的接口设计遵循一般接口设计的原则,即:实现必需功能;隐藏实现细节;统一接口风格;在较低层次发现错误和在较高层次处理错误等等。同时,接口是自说明的,用户在使用接口的时候通过接口名称和参数以及返回值,能了解接口的实现功能。作为 Web Service 的接口还要具备以下功能:

- (1)接口的返回值应该是简单的、语言无关的;
- (2)将 UI 和 Service 以及数据库隔离;
- (3)良好的可扩展性能,对系统的扩展不需要修改代码或只需要修改少量的代码;
- (4)安全策略, Session 和事务处理功能应封装在 Web Service 内部<sup>[4]</sup>;
- (5)提供对象 Cache 的管理。

其次,Web Service 的接口粒度要合适,接口的参数要尽量简单,如果只是把类的方法暴露出来作为服务接口,这样其实是把原来在本地的调用放到远程调用中去执行,影响了接口的性能。粒度太大,则需要大量的参数来映射该服务各种情况,给服务调用者带来不便。再者,要提供对接口参数和返回值的校验,调用者操作的响应应该明确给出成功与否,如果失败则应提供足够的错误定位信息,使调用者很快找到错误的原因,以便修改输入的参数。针对以上原则和功能,可

以将用 XML 语言描述的对象作为系统的返回值,而使用一些对象操作的方法作为参数来设计接口的功能,这样可以用最少的接口数量实现系统的功能,并且接口的数量不会随着对象的增加而增多,因为针对对象的操作方法是一致的。Web Service API 框架暴露给用户的是对象的 Schema 和 WSDL 文件<sup>[5]</sup>,将对象的处理逻辑和数据存取封装在 API 框架的内部,很好地满足了 Web Service API 的功能需要。

### 1.1 接口类型

系统提供两种外部接口类型:REST 和 SOAP<sup>[6,7]</sup>。REST 更适合描述为一个带有 XML 数据的 URL API 类型<sup>[8]</sup>。REST 接口是使用 Struts 实现的。使用 Struts 中的 Form 和 Action 类来接收提交来的数据,然后根据命令的不同进行分配处理。REST 接口主要由 UI 类型的应用调用<sup>[9]</sup>,这种应用一般希望得到数据请求的快速响应。SOAP 接口是使用 AXIS 实现的,系统中的类 WSAPIBindingImpl 包含了连接 SOAP 接口与系统其他部分的逻辑。实际上,它同样构造了一个类似的 Form 来利用 SOAP 调用中的数据。SOAP 接口的 WSDL 文件结构如图 2 所示。

### 1.2 对象模型

对象模型是对 DOM 所代表的对象一个包装类, entity 接口定义了所有 object 级别的操作命令和 commit 以及 rollback 方法。GeneralEntity 是一个简单的对象,它实现了基于配置文件定义的对象级别的操作。在 GeneralEntity 对象中使用 DOM4j 来对 DOM 树进行操作。DOM 树是利用在对象的配置文件中定义的 mapping 来构建的,这些 mapping 指定了节点路径与数据库字段名之间的对应关系。具体的对象类对 GeneralEntity 的部分方法实现了重载,为对象提供特有的操作。对象模型的目的在于可以通过在对象配置文件中增加节点及 mapping 信息来达到扩展性,并能提供 XML 格式的对象输出。对象的实例在系统内部是以具体实现类的形式存在,但是对象的操作都是基于 DOM 的,所有的改变也是存储在 DOM 中,调用者使用命令调用 Web Service 过程如图 3 所示。

### 1.3 对象的配置

在 Web Service 的框架内部对象是通过 XML 描述的对象结构文件来定义的,在这个描述文件中包含以下内容:对象的 XML 结构,对象的属性;对象的 key (即对象的唯一标示符 id);存放对象信息的表,以及关于这些表的 select, update, delete, insert 操作的语句的名称;XML 中循环节点;与该对象关联的外部对象;XML 节点与数据库表字段的映射;一个节点值的列举;终端节点的初始化值;更新的前提条件;安全策略;

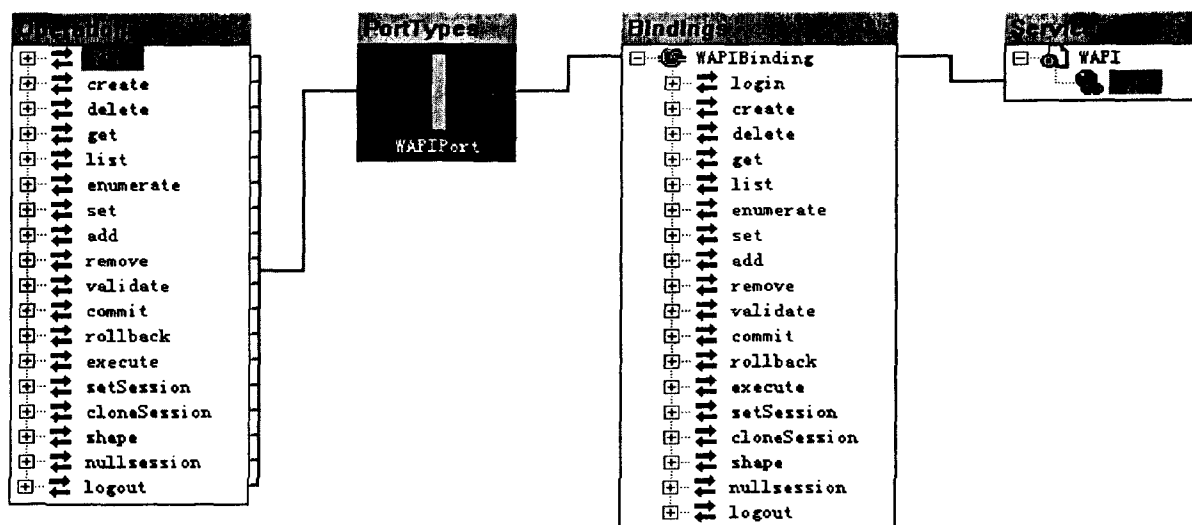


图 2 WSDL 结构图

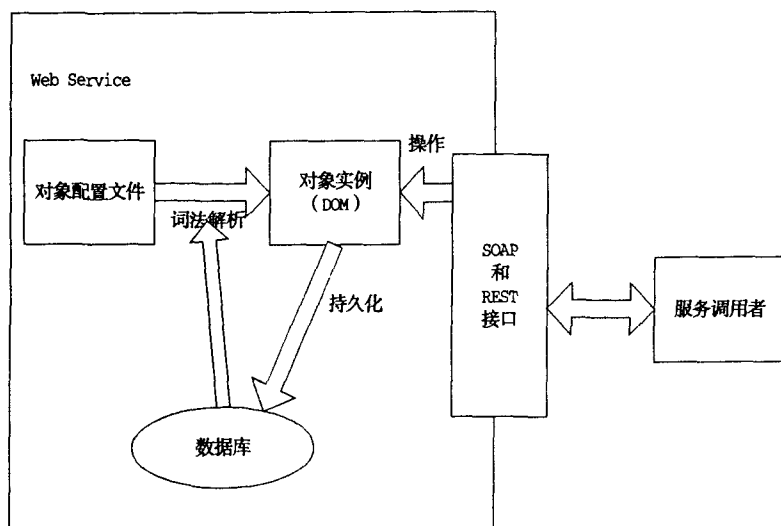


图 3 接口命令的调用过程

验证;触发器(对某一节点的改变可能影响其他节点的值)。在框架的内部,对象是以 DOM 的形式存在的,对象的属性被映射为节点的值。

#### 1.4 框架中的 Cache 和 DOM

框架中存在三种不同的 Cache: Session Cache, Original Cache, Global Cache。Session Cache 是用来保存还没有提交的对象,这种 Cache 是每个 Session 中都存在的,也就是说,Session A 不能看见 Session B 中改变的對象。Original Cache 是保存提交后的对象的,它和数据库中的对象数据保持一致。Original Cache 是跨 Session 的,也就是说,所有的 Session 都可以知道 Original Cache 中对象。Global Cache 是跨服务器保存提交的对象,可以通过 JGroup 机制来同步不同服务器上的对象数据。

系统中的 DOM 存在形式有: Original DOM, Changed DOM, Merger DOM。Original DOM 是改变之

前的对象 DOM 树,系统只保存一个原始对象的拷贝,所有 session 中的对象都是对这个对象的引用,Changed DOM 保存的是改变了的 DOM,可能是部分的 DOM 树,Changed DOM 和 Original DOM 合并为 Merger DOM,成为包含最后需要持久化数据的 DOM。在执行 commit 命令以后,Original Cache 会被清空,当再次引用这个对象的时候,系统会重新载入这个 Original DOM。

## 2 Web Service API 参数设计

### 2.1 参数的设计

下面用 REST 接口的方式来介绍参数的设计。框架对外提供的命令参数包括以下几个不同层次的命令(见表 1)。

命令参数指定的形式为:cmd=命令名称,如:cmd=login。

命令参数只是参数的一部分,在使用具体命令的时候还需要更详细的参数:

Login 需要提供 username 和 password 参数。

Create 命令需要指定创建对象的类型:因此要指定 type 参数,例如 type=student。Credential 是除 login 命令外其他命令必须指定的参数。

Get 需要指定对象的类型,同时要指定对象的唯一标识符 id 或者 where 从句。Where 从句定义对象需要满足的条件。还可以使用 select 参数,select 参数以 XPath 形式指定需要获取信息的节点,这样就可以使得获取的信息更有针对性。

Set 命令要确定是对某个或某些特定的对象,因此

需要对象的唯一标识符 id 或者 where 从句,如 studentID=12345。同时 set 还必须要指定修改的 DOM 节点的内容,因此也是一个 DOM 结构形式,这个 DOM 结构用 XML 参数来指定,例如:xml = < student > < name > Johe < /name > < /student >。Delete 命令和 get 与 set 命令一样需要指定 id 或者 where 从句。

表 1 命令参数及其功能

级别	命令	功能
Session	login	用户首次访问 Web Service 时使用,在身份识别成功后产生一个唯一的认证字符串(credential)作为 Session 的标示,在同一个 session 中使用其他命令参数时必须使用这一认证字符串来唯一标示这个 session 和用户的访问权限,session 结束以后这一认证字符串将失效。如果失败则返回登录失败的错误信息
	logout	用来登出
	commit	将改变了的对象属性持久化到数据库中
	rollback	回滚到对象改变以前的状态
Object	shape	返回对象的结构信息,用户可以使用这一命令来获取对象的结构信息。结构信息中包含对象描述的节点的层次结构,但并不暴露数据存取的相关信息及与其他对象的关系等等
	create	创建一个新的对象
	get	获取一个或一系列满足条件的对象
	set	修改对象的属性
	delete	删除一个对象
	add	Add 和 remove 命令是对对象内部的循环节点进行操作。Add 增加一个循环节点,Remove 删除一个循环节点
	remove	
	validate	对对象的内容进行校验,在进行调用 commit 之前会被自动调用
	enumerate	列出节点的可能取值

Shape 命令只需要 type 参数和 select 参数。

Add 和 remove 命令需要指定对象 id 或者 where 从句以及循环节点的内容。

Validate 命令需要参数 type 和 id。

Enumerate 要求指定 type 和 id 外还有 select 参数。

Execute 命令需要指定 taskname 和 taskdate 参数。Taskdate 指定需要的数据。

Rollback 和 commit 命令仅仅需要 Credential 参数。

## 2.2 安全上下文(Security Context)与状态

安全上下文包含了当前的安全认证书,这一证书是用户登录时产生的,并作为 session 的唯一标示,系统定义了安全认证书的过期时间。在继续操作 Web Service 的过程中,必须使用同样的认证书,直到 logout 系统。安全认证书是每次响应必须包含的元素,这样就避免了在本机保存这一证书的安全隐患。

状态反映的是调用者操作的执行成功与否,在执行过程如果出现错误,则返回对应的错误信息。

## 2.3 plugin 组件程序

框架提供了一组用来扩充功能的插件程序,它们可以用来进行复杂的逻辑处理,这些插件程序包括以

下几个类型:

(1)一般的 Call,在条件节点或者终端节点配置,用来根据同一对象内部其他节点或者其他对象的节点值来计算当前节点值,或者判断节点是否需要满足一定的前提条件。

(2)DAO Call,框架中对表的操作是由 SQL statement 定义的,SQL statement 有两种形式,一种是 SQL 形式,另外一种 Call 形式,DAO Call 定义了 select,insert,update, generateID, delete 等方法,依据 Entity 的信息来对数据库进行操作。

(3)Translate,当直接从数据库中获取的值不能被直接利用,需要经过一定的转化如从 bit 位到布尔型的转换,将取得的值进行分段等等,可以利用 Translate 完成这些工作。

(4)Where Evaluator,处理与 where 从句相关的逻辑,包括缓存和分页,将条件节点定义的语句转化为 SQL 的 where 从句,再在 iBatis 中生成动态的 SQL 调用。

(5)Key Helper,为 Global Cache 和 Session Cache 产生 key。

(6)Execute Call,实现 Execute 的逻辑,如发出 e-mail 等。

## 2.4 对象的验证和持久化

对象的验证机制在 Web Service API 框架中有几个不同级别的实现方式。首先是安全级别(Security),安全级别的验证主要针对 get, set, create, delete 命令,这些验证在调用对应命令的时候被执行。Policy 级别的验证,当这个 Policy 在上下文中时,检验就会在对象中激活。Prerequisite 级别,当改变某个节点的值的时候,该级别的验证才被执行。Validation 级别,这个级别的验证是作为提交的一部分自动执行的。如果验证失败,则 commit 操作被转化为 rollback 操作。

对象的持久化是通过 iBatis 来实现的,依据生成的动态的 SQL 来实现数据的增删改查。

## 3 结束语

面向对象的 Web Service API 的设计方法,在保障了系统接口功能的同时,大大降低 Web Service 调用者的学习成本,在接口不变的前提下,以可配置的对象文件的方式实现了系统功能的扩展,并以灵活的可配置性和 plugin 组件程序实现了对象之间低耦合的目标。随着 Web Service 应用的推广,SaaS 模式得到了越来越多的认可,Web Service 的 API 要实现轻量级的调用方式,面向对象的 API 设计框架为解决这一问题提供了

(下转第 61 页)

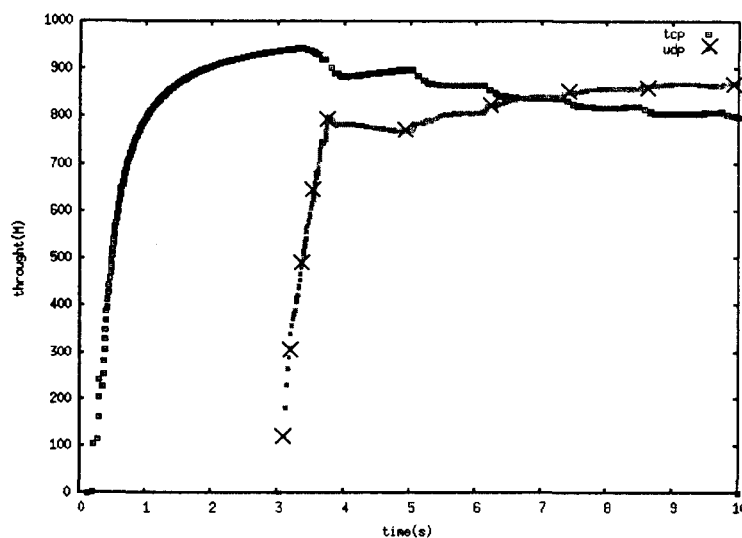


图2 TCP流、TFRC流共存网络下的带宽公平性

#### 4 结束语

卫星网络具有诸多地面网络无法比拟的优点,这使得它成为当前研究的热点,但是它的大时延、高误码等特点又使传统协议性能无法很好地发挥。文中针对卫星网络的这些特点,对传统的TCP、UDP进行了深入分析并提出改进意见,对于UDP流的资源争用,引入了TFRC协议机制,并通过仿真实验给出了引入前后的对比效果,结果表明该协议很好地实现了网络带宽的公平性。

#### 参考文献:

- [1] 依那,金野,梁庆林.卫星链路TCP传输性能分析[J].真空电子技术,2003,32(2):25-29.
- [2] 龙飞,王勇前,曹志刚,等.卫星因特网接入TCP/IP协议的改进与发展[J].清华大学学报,2001,41(7):17-20.

(上接第57页)

一个可选方案<sup>[10]</sup>。

#### 参考文献:

- [1] 周潇.基于Web Service的PKI研究与实现[D].上海:华东师范大学,2007.
- [2] 饶元,冯博琴.新网络体系结构——Web Service研究综述[J].计算机科学,2004,31(5):1-4.
- [3] 李艳霞,巩九洲,黎玉琴.一种基于Web Services的信息集成方案[J].计算机技术与发展,2008,18(9):105-107.
- [4] 陈振邦,王戟,董威,等.面向服务软件体系结构的接口模型[J].软件学报,2006,17(6):1459-1469.
- [5] 吴广印.基于Web Service构架的资源共享技术研究与应用[J].情报学报,2007,26(6):851-857.
- [6] 顾宁,刘家茂,柴晓路.Web Service原理与研发实践

- [3] 贾刚,钟亦平,张世永.UDP数据流对TCP数据流影响的分析和模拟[J].计算机工程,2003,29(7):143-146.
- [4] 王彬,吴铁军.基于显式速率的TCP友好的UDP拥塞控制策略[J].电路与系统学报,2003,8(5):43-48.
- [5] 郑四海,李腊元.一种实时多媒体数据流传输算法及其在NS2上的实现[J].武汉理工大学学报,2008,32(1):47-50.
- [6] 张怡.无线网络中TCP友好拥塞控制[D].南京:南京理工大学,2006.
- [7] 王彬.TCP/IP网络拥塞控制策略研究[D].杭州:浙江大学,2004.
- [8] 王满喜,胡向晖,马刘非.混合式网络拥塞控制算法[J].电子科技大学学报,2006,36(3):20-25.
- [9] 赵飞,叶震.UDP协议与TCP协议的对比分析与可靠性改进[J].计算机技术与发展,2006,16(9):120-122.
- [10] 胡吉明,刘少君.状态协议分析技术在TCP中的应用[J].计算机技术与发展,2006,16(3):212-215.
- [11] 曾晶萍,杨文俊,彭力.TCP友好速率控制协议的分析及应用[J].计算机技术与发展,2007,17(1):111-113.
- [12] Biaz S, Vaidya N. Discrimination congestion losses from wireless losses using interarrival times at the receiver[C]//In Proc. IEEE Symp. Application-Specific Systems and Software Engineering and Technology. Richardson, TX: [s. n.], 1999: 10-17.
- [13] Handley M, Floyd S. TCP Friendly Rate Control (TFRC): Protocol Specification[S]. IETF RFC 3448, 2003.
- [14] Wen P, Cao J, Li Y. Design of High-performance Networked Real-time Control Systems[J]. Control Theory & Applications, 2007, 1(5):1329-1335.

[M].北京:机械工业出版社,2007.

- [7] Curbera F. Unraveling the Web Services Web: An Introduction to SOAP, WSDL, and UDDI[J]. IEEE Internet Computing, 2002, 6(2):86-93.
- [8] Tilkov S. A Brief Introduction to REST[EB/OL]. 2007-12-10. <http://www.infoq.com/articles/rest-introduction;jsessionid=5464B81EB79C0B256B3DA9F7CD942DF6>.
- [9] Fielding R T. Architectural Styles and the Design of Network-based Software Architectures[D]. California: University of California, 2000.
- [10] Beyer D, Chakrabarti A, Henzinger T A. Web service interfaces[C]//Proceedings of the Fifteenth International World Wide Web Conference (WWW 2005). Chiba: ACM Press, 2005.