基于 XFire 和 Axis 构建 Web 服务的研究与实践

拜文娟,马光思
(西安建筑科技大学 信息与控制工程学院,陕西 西安 710055)

摘 要:针对目前在 Web 系统开发中普遍存在的跨平台性和可扩展性需求,基于对 Web 系统开发的业务背景、架构特点和技术选型所进行的深入研究基础之上,结合工程实例,给出了通用 Web 系统的功能需求分析和系统架构设计,同时介绍了构建 Web 服务的设计和实现步骤。其中着重讨论了 XML 格式输入、输出数据的结构设计和利用开源框架 XFire 及 Axis 快速构建 Web 服务的具体过程和技术要点。通过研究和分析,文中给出了借助开源工具快速构建 Web 服务的方法和设计思路。

关键词:Web 服务;开源框架;XML
中图分类号:TP399 **文献标识码**:A **文章编号**:1673-629X(2010)08-0120-04

Research and Practice of Web Services Based on XFire and Axis

BAI Wen-juan, MA Guang-si
(School of Infor. and Control Eng., Xi'an Univ. of Architecture and Tech., Xi'an 710055, China)

Abstract: Aiming at the widespread cross-platform and extensibility demand of development of Web system, with practical applications, give the functional requirements analysis and system architecture design based on the study of business background, architecture features and technology selection of Web system development, and introduce the design and implementation process of Web services. The design of input and output data structure is analyzed based on XML form. It is focused on the specific process and technical points to quickly build Web services using the open-source framework XFire and Axis. With the help of research and analysis, provided reference and design guidance to rapidly build Web service via open-source tools.

Key words: Web services; open-source framework; XML

0 引 言

传统应用系统之间集成复杂,调用受地域、技术平台、应用范围等方面的限制。基于 Web 服务的技术把主要目标集中在构建一个技术层上,不同平台上的应用依靠这个技术层来实施彼此的连接和集成。该技术层采用 XML 表示基本数据格式,采用 WSDL 描述 Web 服务及相关函数、参数和返回值;使用 SOAP 封装服务信息[1]。

基于 Web 服务的应用系统,功能需求包括:

(1)服务发布:服务提供者提供开放的服务接口,注册服务;

(2)服务定位:服务消费者通过特定的服务定位技术(如 UDDI)从服务目录中查找 Web 服务的本地服务代理;

(3)服务调用:服务消费者通过向本地服务代理发送请求消息,再由服务代理对象通过远程调用响应相关请求。

沟通服务提供者和服务请求者的桥梁是用于统一描述、发现和集成的 UDDI 协议[2]。对于调用者来说,只要接口不变,Web 服务实现的任何变更对用户都是透明的。基于以上技术和工程实例,文中分析并讨论了 Web 服务的部分开发工具和应用实现。

1 系统架构分析

1.1 应用系统架构

遵照简单、分层次、面向接口的设计原则,采用数据访问层封装结构化数据,封装遗留系统数据和非结构化数据,目的在于隔离不规则的数据格式造成的访问不一致性。许多成熟的持久层框架和技术[3]如 Hibernate, Ibatis, JPA 等都对数据访问提供了面向对象的封装和支持。内部子系统接口依赖于数据访问层,访问权限也仅限于系统内部,提高其内聚性。业务接口

收稿日期:2009-12-15;**修回日期**:2010-02-27
基金项目:陕西省教育专项科研项目(07JK306)
作者简介:拜文娟(1985-),女,硕士研究生,研究方向为软件架构技术与设计模式;马光思,教授,研究方向为软件理论、软件工程、网络安全。

被发布为 Web Services 供外部应用程序使用。

图 1 给出的 Web 系统架构,描述了外部应用系统如何通过 Web 服务来检索和调用暴露的业务接口。其中,系统边界封装了系统本身的复杂业务,并对外提供透明访问。在系统内部,PublishImpl 实现了 Publish 接口,并依赖数据访问层的 DAO 接口,DAO 可以有多种实现,比如 HibernateDao, IbatisDao 和 JPADao 等,隔离了上层对结构化、非结构化数据和遗产系统数据的访问。

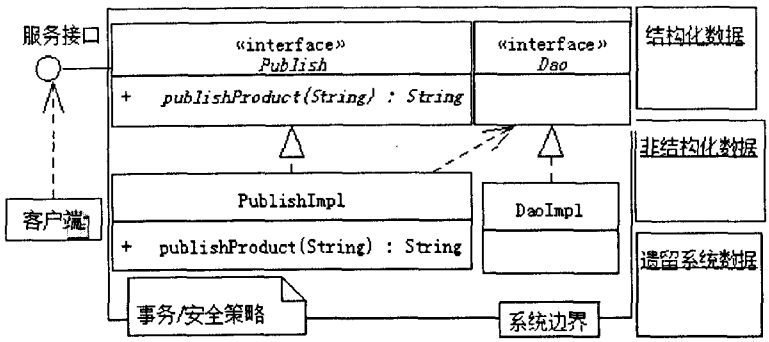

图 1　Web 系统架构图

在安全性方面,可以使用 WS－Security 的信息摘要和数字签名来实现[4]。也可使用结构化信息标准促进组织(OA－SIS)通过的可扩展访问控制标记语言(XACML)来表示控制信息访问的规则和策略。XACML 除给出策略描述语法外,也给出了一个标准化的访问控制决策模型,该模型根据主体、客体与环境的属性进行访问控制[5]。

1.2　服务调用流程

在系统架构实现过程中,外部应用需要遵照固定的顺序来实现服务定位和服务调用。

图 2 的顺序图描述了外部应用检索和调用 Web 服务的流程。外部应用首先通过创建服务定位器 ServiceLocator 来查找本地服务代理,而本地服务代理则会创建远程服务,以后外部应用程序对服务代理的调用,都会通过底层的交换协议(如 SOAP)来转化为对远程业务接口的调用[6]。

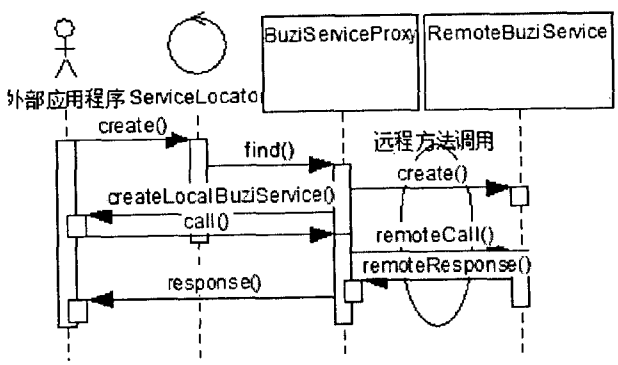

图 2　服务调用顺序图

2　应用实现

Web Services 技术从其产生就受到开源世界的广泛支持。目前流行的几种 Web Services 框架中,由 WSO2 公司提供商业支持的 Apache 开源社区项目 Axis:支持多种数据格式和多语言版本,应用前景广阔,可运行于 Java、C、C++等平台;XFire 使用简单,可以方便快速地从 pojo 发布服务;解析器性能高,支持协议广泛,并且易于和现有成熟框架整合[7]。

XFire 整合 Spring 示例如下:

XFire 发布包中默认提供了一个控制类:org. codehaus. xfire. spring. XFireSpringServlet,首先通过在 web. xml 文件中配置这个 Servlet 发布 Web 服务:

```
<web-app>
  <servlet>
  <servlet-name>XFireServlet</servlet-name>
<display-name>XFireServlet</display-name>
<servlet-class>
    org. codehaus. xfire. spring. XFireSpringServlet
</servlet-class>
  </servlet>
<servlet-mapping> <servlet-name>XFireServlet</servlet-name> <url-pattern>/services/*</url-pattern>
  </servlet-mapping>
</web-app>
```

然后通过下面的代码从 Servlet 获得 Spring 的 ApplicationContext:

```
ApplicationContext appContext = WebApplicationContextUtils. getRequiredWebApplicationContext ( servletConfig. getServletContext ())
```

2.1　使用 XFire 发布服务

服务器端采用 XFire 提供外界使用的服务接口,用于发布信息。接口代码如下:

```
public interface Publish {
public String publishProduct(String input);
}
```

方法 public String publishProduct(String input)的输入参数和返回值都以标准的 XML[8]格式定义,并由 XMLSpy 设计的 XSD 校验[9]。

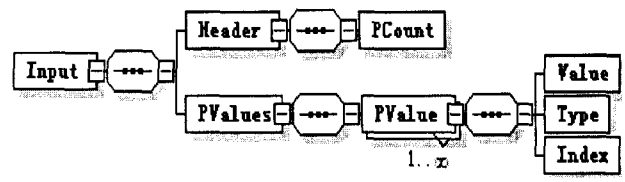

图 3　输入参数数据结构

图 3 描述了所定义输入参数的结构,其中 Header 节点定义了参数数目;PValues 下有多个 PValue,各

PValue 定义具体输入参数，包括参数值、类型和顺序下标。

下面的代码片段展示了调用服务时输入参数的一个应用案例，它以 XML 文本形式表示：

```
<Input>
<Header>
<PCount>2</PCount>
</Header>
<PValues>
<PValue>
<Value>productName1</Value>
<Type>java.lang.String</Type>
<Index>0</Index>
</PValue>
<PValue>
<Value>23.59</Value>
<Type>java.lang.Float</Type>
<Index>1</Index>
</PValue>
</PValues>
</Input>
```

图 4 描述了所定义返回值的结构，其中 Header 节点定义返回值成功与否的编码(需要一个成功或者错误号码对照表)，返回提示信息以及返回值数目；RValues 下有多个 RValue，各 RValue 定义具体的返回值，其中包含返回值、类型和顺序下标。

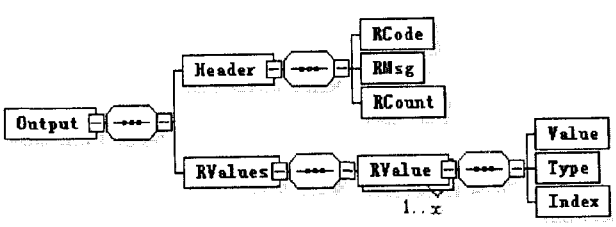

图 4 返回值数据结构

下面的代码片段展示了调用服务成功后返回值的一个应用案例：

```
<Output>
<Header>
<RCode>1</RCode>
<RMsg>success add</RMsg>
<RCount>1</RCount>
</Header>
<RValues>
<RValue>
<Value>1001</Value>
<Type>java.lang.Long</Type>
<Index>0</Index>
</RValue>
</RValues>
</Output>
```

业务接口的实现按照下面的步骤进行：

STEP1：定义 XML 格式的输入参数，发送给 Web 服务接口程序；

STEP2：XSD 对输入参数进行有效性验证[10]；

STEP3：获取输入参数中的调用信息，执行业务操作；

STEP4：对正常操作执行：

(1)构建成功信息和 XML 格式的返回值；

对异常操作执行：

(2)构建错误码和 XML 格式的错误信息；

STEP5：返回客户端。

定义好的接口以服务的形式发布。如下是使用 XFire 发布该接口成为 WSDL 文件，以供外部使用的过程：

STEP1：在 src 下创建文件夹 META - INF，在 META - INF 下创建目录 xfire，在 xfire 目录下创建名为 services.xml 的文件，具体代码如下：

```
<beans>
<service XMLns="http://xfire.codehaus.org/config/1.0">
<name>Publish</name>
<namespace>
http://xfire.xauat.edu.cn
</namespace>
<serviceClass>
cn.edu.xauat.xfire.Publish
</serviceClass>
<implementationClass>
cn.edu.xauat.xfire. PublishImpl
</implementationClass>
</service>
</beans>
```

该文件用于说明接口、实现类的位置，以及用于发布成为 WSDL 文件所需的信息。

STEP2：web.xml 的配置，具体代码如下：

```
<? XML version="1.0" encoding="UTF-8"? >
<web-app xmlns=http://java.sun.com/XML/ns/j2ee xmlns:
xsi="http://www.w3.org/2001/XMLSchema-instance" version
="2.4"
xsi: schemaLocation = " http://java. sun. com/XML/ns/j2ee
http://java.sun.com/XML/ns/j2ee/web-app-2-4.xsd">
<display-name>WebService</display-name>
<servlet>
<servlet-name>XFireServlet</servlet-name><servlet-class
>
org.codehaus.xfire.transport.http.XFireConfigurableServlet
</servlet-class>
</servlet>
```

```
<servlet-mapping>
<servlet-name>XFireServlet</servlet-name>
<url-pattern>/services/*</url-pattern>
</servlet-mapping>
</web-app>
```

该配置文件,用于服务启动时候拦截所有以 services 结尾的请求。

STEP3:启动 Tomcat,在地址栏输入 http://localhost:8080/xfire/services/Publish? wsdl,就可以得到发布的 Web 服务,即 WSDL 文档。

2.2 使用 Axis 访问服务

Axis 框架是 Apache 组织基于 Java 语言新实现的 SOAP 规范。作为第三代 SOAP 工具箱,Axis 能够同现有的 Web 服务器很好地绑定和集成。Axis 的开发方式类似一个小型的应用服务器,它的开发包以 WAR 的形式部署到如 Tomcat 等的 Servlet 容器中[11]。

采用 Axis 来访问 Web 服务,通过解析 Web 服务的 WSDL 文档,得到相应的接口方法,从而可以访问服务器端的方法。具体操作如下:

程序需要下载 Axis 需要的 jar 包。首先得到 service. wsdl,然后新建一个文件名为 generateJavaByWSDL.bat 的文件,该文件实现了通过 Axis 中包含的工具将 service.wsdl 文档快速地生成客户端 java 类,generateJavaByWSDL.bat 具体代码如下:

```
Java -cp lib\axis.jar;lib\commons-logging-1.0.4.jar;lib\commons-discovery-0.2.jar;lib\jaxrpc.jar;lib\saaj.jar;lib\wsdl4j-1.5.1.jar;lib org.apache.axis.wsdl.WSDL2Java -o src src\service.wsdl
```

这样,就可以在 src 目录下得到一个 java 类包,该包将服务器端的 Web 服务方法显示出来,以供客户端用户编写代码调用。客户端可以编写自己的类来实现发布信息的目的。如下例所示:

```
String wsdl="http://localhost:8080/xfire/services/Publish";
Publish service = new PublishLocator();
((PublishLocator)service).setPublishHttpPortEndpointAddress(wsdl);
try {
PublishPortType port = service.getPublishHttpPort();
String in = buildInputXml();//构造参数
String out = port.publishProduct (in);
//...获取返回值
} catch (RemoteException e) {e.printStackTrace();
} catch (ServiceException e)
{ e.printStackTrace();
}
```

通过上面的服务定位,客户端程序就可以透明地访问远程服务接口,获得服务。

3 结束语

文中通过对 Web Services 相关的技术和理论的探讨和对现有 Web 系统应用现状的分析,并通过对比各种 Web Services 开源框架的优劣,结合实例介绍了利用 XFire 和 Axis 构建 Web 服务的具体实现过程。关于构建 Web 服务要考虑的安全性、可靠性、灵活性、封装性和松耦合等非功能性需求,需要在保证完成基本功能性需求的基础上再开展进一步的研究与实践。

参考文献:

[1] 杨　磊,王建斌,马光思,等.基于 SOAP 协议和 Ajax 技术构建 Web 应用[J].计算机技术与发展,2008,18(1):115-117.

[2] 苗　青,陈　钢.基于 WebService 的高校应用集成[J].计算机技术与发展,2008,18(3):17-20.

[3] 李成严,冯慧灵.基于开源技术的 Web 应用架构研究[J].计算机技术与发展,2009,19(8):27-29.

[4] 耿玉波,夏鲁宁,杜　皎,等.一种 Web 服务器安全机制的研究与实现[J].计算机工程,2006(11):189-191.

[5] 戴常英,张会娟.基于信任度的 Web 服务跨域访问控制[J].计算机工程与科学,2009,31(8):42-45.

[6] 宋丽华,刘方爱.基于 Web Service 的网格服务功能的研究[J].计算机技术与发展,2009,19(7):59-61.

[7] Ahmed S. Xfire:The easy and simple way to develop web services. JavaWord.com[EB/OL].2005-01-06.[2007-09-10].http://www.javaworld.com/javaworld/jw-05-2006/jw-0501-xfire.html.

[8] W3C. Extensible Markup Language(XML)[EB/OL].2009-04-16.http://www.w3.org/XML/.

[9] 巢弘坤,陈闳中.XML 数据类型验证算法的改进[J].计算机工程,2009(19):53-55.

[10] W3C. XML Schema Part1[EB/OL].2004-11-28.http://www.w3.org/TR/xmlschema-1/.

[11] Apache.org. Web Services-Axis[EB/OL].2006-04-22.http://ws.apache.org/axis/.