主题：**Apache CXF**入门范例以及对传递**List<Map>**类型的疑惑

作者

冰火特蕾莎

等级: 初级会员

文章: 18

积分: 80

来自: 北京

正文

发表时间：2009-11-28 最后修改：2009-11-29

< > 猎头职位：上海：上海：天会皓闻诚聘资深Java架构师

在选择WebService框架的过程中，偶最终选择了Apache CXF，純粹侭諟錮爲听说它与Spring的无缝整合想当初用Axis的时候，因为没有太好的办法让Spring能够集成Axis，只好平白无故地多出一个WebService代理类，让偶的感觉很是不爽

偶要在此记载一下CXF的一些入门知识

首珧，偺網哋址諟http://cxf.apache.org/，里面可以找到User's Guide和download地址，偶的版本是目前最新的

apache-cxf-2.2.5

先来做一个最简单的入门级别例子吧,也就是经典的HelloWord

Server端代码

WebService接口HelloService.java

Java代码

```
1.  package cfx.server;
2.
3.  import javax.jws.WebMethod;
4.  import javax.jws.WebParam;
5.  import javax.jws.WebService;
6.
```

相关文章:

- cxf+spring+struts2的helloWorld
- Apache CXF 学习笔记
- CXF中使用dataBinding时遇到异常的解决办法

推荐圈子: GT-Grid

更多相关推荐

```java
@WebService
public interface HelloService {
        @WebMethod
        String sayHi(@WebParam String name);
}
```

实现类HelloServiceImpl.java

Java代码

```java
public class HelloServiceImpl implements HelloService {
        public String sayHi(String name) {
                System.out.println("HelloServiceImpl.sayHi called");
                return "Hello"+name;
}
```

WebService配置文件：cxf-servlet.xml（可放置于WEB-INF目录下）

Xml代码

```xml
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xmlns:jaxws="http://cxf.apache.org/jaxws"
      xmlns:soap="http://cxf.apache.org/bindings/soap"
      xsi:schemaLocation="
http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans/spring-beans.xsd
http://cxf.apache.org/bindings/soap http://cxf.apache.org/schemas/configuration/soap.xsd
http://cxf.apache.org/jaxws
http://cxf.apache.org/schemas/jaxws.xsd">
  <jaxws:server id="jaxwsService" serviceClass="cfx.server.HelloService" address="/hello">
        <jaxws:serviceBean>
                <bean class="cfx.server.HelloServiceImpl" />
        </jaxws:serviceBean>
  </jaxws:server>
</beans>
```

web.xml代码，用于添加CXFServlet这个处理webservice请求的控制器类

Xml代码

```xml
<?xml version="1.0" encoding="UTF-8"?>
<web-app version="2.4" xmlns="http://java.sun.com/xml/ns/j2ee" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://java.sun.com/xml/ns/j2ee
        http://java.sun.com/xml/ns/j2ee/web-app_2_4.xsd">

  <servlet>
    <description>Apache CXF Endpoint</description>
    <display-name>cxf</display-name>
```

```
    <servlet-name>cxf</servlet-name>
    <servlet-class>org.apache.cxf.transport.servlet.CXFServlet</servlet-class>
    <load-on-startup>1</load-on-startup>
  </servlet>
  <servlet-mapping>
    <servlet-name>cxf</servlet-name>
    <url-pattern>/services/*</url-pattern>
  </servlet-mapping>
  <session-config>
    <session-timeout>60</session-timeout>
  </session-config>
</web-app>
```

Client端测试代码

Java代码

```
public class CXF {
        public static void main(String[] args) {
                JaxWsProxyFactoryBean factory = new JaxWsProxyFactoryBean();
                factory.getInInterceptors().add(new LoggingInInterceptor());
                factory.getOutInterceptors().add(new LoggingOutInterceptor());
                factory.setServiceClass(HelloService.class);
                factory.setAddress("http://localhost:8080/cxf/services/hello");
                HelloService client = (HelloService) factory.create();
                String reply = client.sayHi("特蕾莎");
                System.out.println("Server said: " + reply);
}
```

****

怎么样，是不是很简单啊！现在再来一个和Spring整合的例子

注意，Server端和Client端都要通过Spring-bean的方式整合

Server端现在有四个文件，假设是

HelloService.java

HelloServiceImpl.java

HelloDao.java

HelloDaoImpl.java

在HelloServiceImpl中存在一个HelloDao的属性，代码省略如下

Java代码

```
public class HelloServiceImpl implements HelloService {
        private HelloDao dao;
        public String sayHi(String name) {
                System.out.println("HelloServiceImpl.sayHi called");
                return dao.getString(name);
        }
}
```

HelloDaoImpl用于处理持久化，代码省略咯
需要修改的是配置文件，此时可以这样改
首先在web.xml里加入Spring监听器

Xml代码

```xml
<?xml version="1.0" encoding="UTF-8"?>
<web-app version="2.4" xmlns="http://java.sun.com/xml/ns/j2ee" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://java.sun.com/xml/ns/j2ee
        http://java.sun.com/xml/ns/j2ee/web-app_2_4.xsd">
  <listener>
    <listener-class>org.springframework.web.context.ContextLoaderListener</listener-class>
  </listener>
  <context-param>
    <param-name>contextConfigLocation</param-name>
    <param-value>classpath:applicationContext*.xml</param-value>
  </context-param>
  <servlet>
    <description>Apache CXF Endpoint</description>
    <display-name>cxf</display-name>
    <servlet-name>cxf</servlet-name>
    <servlet-class>org.apache.cxf.transport.servlet.CXFServlet</servlet-class>
    <load-on-startup>1</load-on-startup>
  </servlet>
  <servlet-mapping>
    <servlet-name>cxf</servlet-name>
    <url-pattern>/services/*</url-pattern>
  </servlet-mapping>
  <session-config>
    <session-timeout>60</session-timeout>
  </session-config>
</web-app>
```

樊鎮WEB-INF/cxf-servlet這個忮件可以省略咯
把一个标准的spring-bean文件放在src下（即classes目录下），要让人一看就知道spring大哥进来咯
applicationContext.xml

Xml代码

```xml
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:jaxws="http://cxf.apache.org/jaxws"
  xsi:schemaLocation="
http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans/spring-beans.xsd
http://cxf.apache.org/jaxws http://cxf.apache.org/schemas/jaxws.xsd">
```

```
7.
8.    <import resource="classpath:META-INF/cxf/cxf.xml" />
9.    <import resource="classpath:META-INF/cxf/cxf-extension-soap.xml" />
10.   <import resource="classpath:META-INF/cxf/cxf-servlet.xml" />
11.   <bean id="helloDao" class="cfx.server.HelloDaoImpl" />
12.   <jaxws:server id="jaxwsService" serviceClass="cfx.server.HelloService" address="/hello">
13.     <jaxws:serviceBean>
14.       <bean id="helloService" class="cfx.server.HelloServiceImpl">
15.         <property name="dao" ref="helloDao" />
16.       </bean>
17.     </jaxws:serviceBean>
18.   </jaxws:server>
19. </beans>
```

這樣啟動服務器的时候，spring就自动进行bean的注入以及WebService服务的发布了
接下来是客户端代码
鋂爲諟普通Java，所以就简单配一下塞戶端的spring文件了

Xml代码

```
1.  <?xml version="1.0" encoding="UTF-8"?>
2.
3.  <beans xmlns="http://www.springframework.org/schema/beans" xmlns:xsi="http://www.w3.org/2001/XMLSchema-inst
    ance"
4.    xmlns:jaxws="http://cxf.apache.org/jaxws"
5.    xsi:schemaLocation="
6.  http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans/spring-beans.xsd
7.  http://cxf.apache.org/jaxws http://cxf.apache.org/schema/jaxws.xsd">
8.
9.    <bean id="HelloService" class="cfx.server.HelloService" factory-bean="clientFactory" factory-method="cre
    ate" />
10.   <bean id="clientFactory" class="org.apache.cxf.jaxws.JaxWsProxyFactoryBean">
11.     <property name="serviceClass" value="cfx.server.HelloService" />
12.     <property name="address" value="http://localhost:8080/cxf/services/hello" />
13.   </bean>
14.
15. </beans>
```

CXFClientTest.java

Java代码

```
1.  public static void main(String[] args) {
2.          ClassPathXmlApplicationContext context = new ClassPathXmlApplicationContext(new String[] { "cfx/cli
    ent/client-beans.xml" });
3.          HelloService client = (HelloService) context.getBean("HelloService");
4.          testString(client);
```

```
}
static void testString(HelloService client) {
        String reply = client.sayHi("特蕾莎");
        System.out.println("Server said: " + reply);
}
```

****

然后是复杂数据类型的问题，经过测试，发觉基本数据类型和List都是没有问题的，我的测试方法包括

Java代码

```
@WebMethod
String sayHi(@WebParam String name);

@WebMethod
List<Integer> getList(@WebParam List<String> strs);

@WebMethod
List<User> getJavaBean();
```

但是传递Map时，就出现问题了，所以参照了user's guide,得到如下解决办法
测试某个方法的参数和返回值都是Map类型

Java代码

```
@WebMethod
@XmlJavaTypeAdapter(MapAdapter.class)
Map<String, String> getMap(@WebParam @XmlJavaTypeAdapter(MapAdapter.class) Map<String, String> map);
```

MapAdapter是我自己写的用於數據類型轉換的适配器类，代码如下

Java代码

```
public class MapAdapter extends XmlAdapter<MapConvertor, Map<String, Object>> {

        @Override
        public MapConvertor marshal(Map<String, Object> map) throws Exception {
                MapConvertor convertor = new MapConvertor();
                for(Map.Entry<String, Object> entry:map.entrySet()){
                        MapConvertor.MapEntry e = new MapConvertor.MapEntry(entry);
                        convertor.addEntry(e);
                }
                return convertor;
        }

        @Override
```

```
14.         public Map<String, Object> unmarshal(MapConvertor map) throws Exception {
15.                 Map<String, Object> result = new HashMap<String,Object>();
16.                 for(MapConvertor.MapEntry e :map.getEntries()){
17.                         result.put(e.getKey(), e.getValue());
18.                 }
19.                 return result;
20.         }
21.
22. }
```

MapConvertor.java Map格式转换类

Java代码

```
1.  @XmlType(name = "MapConvertor")
2.  @XmlAccessorType(XmlAccessType.FIELD)
3.  public class MapConvertor {
4.
5.          private List<MapEntry> entries = new ArrayList<MapEntry>();
6.
7.          public void addEntry(MapEntry entry){
8.                  entries.add(entry);
9.          }
10.
11.         public static class MapEntry{
12.                 public MapEntry() {
13.                         super();
14.                 }
15.                 public MapEntry(Map.Entry<String,Object> entry) {
16.                         super();
17.                         this.key = entry.getKey();
18.                         this.value = entry.getValue();
19.                 }
20.                 public MapEntry(String key,Object value) {
21.                         super();
22.                         this.key = key;
23.                         this.value = value;
24.                 }
25.                 private String key;
26.                 private Object value;
27.                 public String getKey() {
28.                         return key;
29.                 }
30.                 public void setKey(String key) {
31.                         this.key = key;
32.                 }
```

```
33.                     public Object getValue() {
34.                             return value;
35.                     }
36.                     public void setValue(Object value) {
37.                             this.value = value;
38.                     }
39.             }
40.
41.             public List<MapEntry> getEntries() {
42.                     return entries;
43.             }
44.     }
```

经过这个MapAdapter，算是完成了Map类型的数据传递
接下来，就是更为复杂的一点的这种情况了：List<Map<String,Object>>
这个情况实在不太好办，目前偶也没有找到更好的办法，偶只能这样做

Java代码

```
1.  @WebMethod
2.  List<MapConvertor> getListMap(List<MapConvertor> listmap);
```

就是把MapConvertor当成Map来使，但偶觉得这不是一个很妥善的办法。

其实偶觉得，WebService里应该尽量减少使用javabean对象进行传输
一个JavaBean可以转换成一个Map
一个包含多个JavaBean的List可以转换成一个包含多个Map的List
所以如果对Map支持得好的话，就应该多用Map和List来实现数据传递
当然在调用的时候，也最好能够像Axis一样使用Call来实现无接口调用
这样在Client端就不需要得到Server提供的任何jar了
这样才是最松散耦合的系统
只可惜现在对Map支持得不是很好，最起码在List<Map>时用的方式感觉不太好

不知道各位GGJJ们是怎么做的呢？

推荐链接

返回顶楼

kiki

等级:

文章: 28

积分: 262

我现在离线

发表时间：2009-12-05

这样更省事

Java代码

```java
package cn.com.pansky.dss.common.ws;

import java.util.HashMap;
import javax.xml.bind.annotation.adapters.XmlAdapter;
import com.thoughtworks.xstream.XStream;
import com.thoughtworks.xstream.io.xml.DomDriver;
/**
 * XML to Map及Map to XML序列化
 * @author shenwujie
 *
 */
public class MapAdapter extends XmlAdapter<String,HashMap> {

        @Override
        public String marshal(HashMap map) throws Exception {
                XStream xs = new XStream(new DomDriver());
                return xs.toXML(map);
        }

        @Override
        public HashMap unmarshal(String xmlData) throws Exception {
                XStream xs = new XStream(new DomDriver());
                return (HashMap) xs.fromXML(xmlData);
        }

}
```

返回顶楼　　回帖地址　　0　　0 请登录后投票

taiwei.peng

等级: 初级会员

发表时间：2010-02-02

博主，你的

@WebMethod

List<User> getJavaBean();

这个方法是怎么弄的啊？？？？？？？？？？？？？？？？？？

文章: 2

积分: 30

来自: 深圳

我现在离线

返回顶楼 回帖地址 0 0 请登录后投票

yangkai1217

等级: 初级会员

发表时间：2010-03-02

如果方法中用了这种map 会提示一下内容,请问该如何解决呢

警告: {http://service.ws.upc.red.com/}findUser requires a wrapper bean but problems with ASM has prevented creating one. Operation may not work correctly.

文章: 4

积分: 30

来自: 武汉

我现在离线

返回顶楼 回帖地址 0 0 请登录后投票

论坛首页 → Java编程和Java企业应用版

跳转论坛: Java编程和Java企业应用

上海：□□□□□□J2EE高级工程师（WEB）

□□: □□□□□□java□□□□□□□□□□

□□: □□□□□□□□□□□□ java□□□□□□

□□: □□□□□□JAVA□□□□□□