Evan
Only let oneself become strong enough, good enough, can afford the life that you want to.

CXF+JAXB处理复杂数据

标签： CXF JAXB 复杂数据

2017-06-12 16:41 62人阅读 评论(0)

分类： CXF JAXB

目录(?) [-]

1. Table of Contents
2. 数据复杂性的分类
    1. 1 简单数据类型
    2. 2 自定义类型
    3. 3 集合类型
    4. 4 复杂类型
3. JAXB对数据复杂性的支持
4. 常用技巧
    1. 1 使用自定义的XmlAdapter支持Map
    2. 2 断开循环引用的回路
    3. 3 使用XmlSeeAlso标注处理继承关系
5. 代码
    1. 1 maven工程文件
    2. 2 Map适配器
    3. 3 Map适配器使用的key-value结构
    4. 4 JavaBean父类
    5. 5 JavaBean子类
    6. 6 webservice接口定义
    7. 7 webservice实现类
    8. 8 测试用例

CXF默认使用JAXB 来实现对象和XML之间的映射。在前面的例子 中，使用CXF发布的Webservice，其方法的参数和返回值都是简单类型。 本文讨论对象复杂性的分级，验证对于各种复杂度JAXB的支持情况，以及使用JAXB时对于Map，循环引用，继承等情况的处理办法。 文中的例子没有直接调用JAXB的API，而是用CXF发布webservice的形式验证对象到xml的marshal和unmarshal， 所以本文也可以作为使用CXF的参考资料。

Table of Contents

- 1 数据复杂性的分类
    - 1.1 简单数据类型
    - 1.2 自定义类型
    - 1.3 集合类型
    - 1.4 复杂类型
- 2 JAXB对数据复杂性的支持
- 3 常用技巧
    - 3.1 使用自定义的XmlAdapter支持Map
    - 3.2 断开循环引用的回路
    - 3.3 使用@XmlSeeAlso标注处理继承关系
- 4 代码

- 4.1 maven工程文件：pom.xml
- 4.2 Map适配器：MapAdapter.java
- 4.3 Map适配器使用的key-value结构：MapEntity.java
- 4.4 JavaBean:User.java
- 4.5 JavaBean:MyUser.java
- 4.6 服务接口定义：CXFDemo.java
- 4.7 服务实现类：CXFDemoImpl.java
- 4.8 测试代码：TestEndpoint.java

1 数据复杂性的分类

大体来说，**Java**中的数据/数据对象按照其复杂度可以分为以下几类：

1.1 简单数据类型

包括基本类型和Java对基本类型的封装，主要有：

基本类型	封装类
float	Float
double	Double
byte	Byte
short	Short
int	Integer
long	Long
char	Character
boolean	Boolean
char[]	String

1.2 自定义类型

在C里面叫做struct，在Java里面叫做JavaBean，包含自定义属性和getter/setter方法。

1.3 集合类型

Java的集合类（Collection）主要分为List，Set，Map三个系列。List实现了元素的序列（顺序），Set实现不重复的集合，Map实现了key-value的映射。

1.4 复杂类型

更复杂的情况是对于上述三种类型的组合运用，比如在自定义类型中使用集合，或者集合的嵌套等。 复杂类型还会涉及到循环引用和继承关系等问题。

2 JAXB对数据复杂性的支持

- 简单类型

对于简单的数据类型，JAXB不需要任何处理就完全能够支持

- 自定义类型

JAXB对于一般的JavaBean也能够支持，比如下面的例子：

User.java

```
public class User {
	private Integer id;
	private String name;

	public Integer getId() {
		return id;
	}

	public void setId(Integer id) {
		this.id = id;
	}

	public String getName() {
		return name;
	}

	public void setName(String name) {
		this.name = name;
	}
}
```

不需要JavaBean实现Serializable接口，也不需要增加@XmlRootElement声明。

- 集合类型

  JAXB能够内置支持List和Set集合，但是对于Map的支持需要自己处理。

- 复杂类型

  JAXB支持简单类型、自定义类型、集合类型等的嵌套，但是对于循环引用、继承等情况需要增加额外的处理。

3 常用技巧

3.1 使用自定义的XmlAdapter支持Map

JAXB可以在变量上添加@XmlJavaTypeAdapter标注，指定对该变量专门的适配器进行处理。 适配器继承XmlAdapter类，并覆盖了marshal和unmarshal方法，分别用于对象到XML的映射和XML到对象的映射。

使用XmlAdapter可以实现对Map类型的映射。

比如对于要通过CXF发布的WebService接口方法上，可以增加标注：

```
@XmlJavaTypeAdapter(MapAdapter.class)
Map<String,User> getUserMap();
```

```
Integer setUserMap(@XmlJavaTypeAdapter(MapAdapter.class)Map<String,User> users);
```

其中的MapAdapter就是自己实现的Map适配器，代码如下：

⊞MapAdapter.java

MapEntity是自己定义的一个简单结构，用于保持Map中的key-value关系：

⊟

```
public class MapEntity{
    public Object key;
    public Object value;
}
```

经过这样的处理，就能够实现Map与XML之间的映射。

3.2 断开循环引用的回路

对象之间的引用很有可能出现回路。最简单的情况是两个对象之间互相引用。这在ORM中很常见。如果我们在前面的User类中增加父子关系，如下：

⊞User.java

当同时在两个方向设置引用关系时，就发生了循环引用：

```
child.parent = parent;
parent.children.put(child.getName(), child);
```

发生循环引用时，JAXB就会抛出异常。而处理的办法就是断开其中一个方向的引用。具体做法就是使用@XmlTransient标注，表明该属性在marshal是不作处理。 如上面的User中，我们可以只处理parent到child的引用，而不处理child到parent的引用：

```
@XmlTransient
public User parent;
```

这样虽然解决了循环引用的问题，但是会导致得到User对象的parent属性为null。为使用带来不变。 解决的办法是在JavaBean中增加afterUnmarshal()方法，当JAXB从xml恢复出对象后，会自动调用这个方法。我们可以在方法中将丢失的信息补全：

⊟

```
public void afterUnmarshal(Unmarshaller u,Object parent) {
    for(Iterator itor = this.children.values().iterator();itor.hasNext();){
        User user = (User)itor.next();
        user.parent = this;
    }
}
```

3.3 使用@XmlSeeAlso标注处理继承关系

继承关系在ORM中已经处理得非常完善了，JAXB处理继承关系更加简单，只需要在继承树的根类上增加@XmlSeeAlso标注，声明所有的子类即可。 比如我们定义了一个User的子类：

```
public class MyUser extends User {...}
```

则只需要在User类上面增加标注：

```
@XmlSeeAlso({
    MyUser.class
})
public class User {...}
```

4 代码

本文相关的所有代码如下：

4.1 maven工程文件

```
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/maven-v4_0_0.xsd">
    <modelVersion>4.0.0</modelVersion>
    <groupId>com.hysec</groupId>
    <artifactId>cxfdemo</artifactId>
    <packaging>jar</packaging>
    <version>1.0-SNAPSHOT</version>
    <name>cxfdemo</name>



    <dependencies>
        <dependency>
            <groupId>junit</groupId>
            <artifactId>junit</artifactId>
            <version>4.11</version>
            <scope>test</scope>
        </dependency>
        <dependency>
            <groupId>org.apache.cxf</groupId>
            <artifactId>apache-cxf</artifactId>
            <version>2.4.1</version>
            <type>pom</type>
        </dependency>
    </dependencies>
</project>
```

4.2 Map适配器

```
package com.hysec.utils.jaxb;


import java.util.HashMap;
import java.util.Iterator;
import java.util.Map;
import java.util.Set;


import javax.xml.bind.annotation.adapters.XmlAdapter;



public class MapAdapter extends XmlAdapter<MapEntity[], Map> {


    @Override
    public MapEntity[] marshal(Map map) throws Exception {
        // TODO Auto-generated method stub
        MapEntity[] list = new MapEntity[map.size()];
        Set keyset = map.keySet();
        int index =0;
        for(Iterator itor=keyset.iterator();itor.hasNext();){
            MapEntity item = new MapEntity();
            item.key = itor.next();
            item.value = map.get(item.key);
            list[index++] = item;
        }
        return list;
    }


    @Override
    public Map unmarshal(MapEntity[] list) throws Exception {
        // TODO Auto-generated method stub
        Map map = new HashMap();
        for(int i=0;i<list.length;i++){
            MapEntity item = list[i];
            map.put(item.key, item.value);
        }


        return map;


    }
```

```
}
```

4.3 Map适配器使用的key-value结构

```
package com.hysec.utils.jaxb;

public class MapEntity{
    public Object key;
    public Object value;
}
```

4.4 JavaBean父类

```
package cxfdemo;

import java.util.HashMap;
import java.util.Iterator;
import java.util.Map;

import javax.xml.bind.Unmarshaller;
import javax.xml.bind.annotation.XmlSeeAlso;
import javax.xml.bind.annotation.XmlTransient;

@XmlSeeAlso({
    MyUser.class
})
public class User {

    private Integer id;

    private String name;

    @XmlTransient
    public User parent;

    public Map<String,User> children = new HashMap<String,User>();
```

```
    public Integer getId() {
        return id;
    }

    public void setId(Integer id) {
        this.id = id;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public void afterUnmarshal(Unmarshaller u,Object parent) {
        for(Iterator itor = this.children.values().iterator();itor.hasNext();){
            User user = (User)itor.next();
            user.parent = this;
        }

    }


}
```

4.5 JavaBean子类

```
package cxfdemo;

public class MyUser extends User {
    public String myProp;
}
```

4.6 webservice接口定义

```
package cxfdemo;

import java.util.List;
import java.util.Map;
import java.util.Set;

import javax.jws.WebService;
import javax.xml.bind.annotation.adapters.XmlJavaTypeAdapter;

import com.hysec.utils.jaxb.MapAdapter;

@WebService
public interface CXFDemo {
    String sayHello(String foo);
    String sayHelloToUser(User user);
    User getUser(String name);

    List<User> getUsers();
    Integer setUsers(List<User> users);

    Set<User> getUserSet();
    Integer setUserSet(Set<User> users);

    @XmlJavaTypeAdapter(MapAdapter.class)
    Map<String,User> getUserMap();
    Integer setUserMap(@XmlJavaTypeAdapter(MapAdapter.class)Map<String,User> users);


    User addChild(User parent,User child);
}
```

4.7 webservice实现类

```
package cxfdemo;

import java.util.ArrayList;
import java.util.HashMap;
import java.util.HashSet;
```

```
import java.util.List;

import java.util.Map;

import java.util.Set;


import javax.jws.WebService;


@WebService()

public class CXFDemoImpl implements CXFDemo {


    public String sayHello(String foo) {

        return "hello "+foo;

    }


    public String sayHelloToUser(User user){

        return "hello "+user.getName();

    }


    public User getUser(String name){

        User user = new User();

        user.setName(name);

        return user;

    }


    public List<User> getUsers(){

        List<User> users = new ArrayList<User>();

        users.add(new User());

        return users;

    }


    public Integer setUsers(List<User> users){

        return users.size();

    }


    public Set<User> getUserSet(){

         Set<User> set = new HashSet<User>();

         set.add(new User());

         set.add(new User());

         return set;

    }

    public Integer setUserSet(Set<User> users){
```

```
        return users.size();
    }

    public Map<String,User> getUserMap(){
        HashMap<String,User> map = new HashMap<String,User>();
        User user1 = new User();
        user1.setName("Holbrook");
        map.put("Holbrook", user1);

        User user2 = new User();
        user2.setName("wanghaikuo");
        map.put("wanghaikuo", user2);

        return map;
    }

    public Integer setUserMap(Map<String,User> users){
        return users.size();
    }

    public User addChild(User parent,User child){
        child.parent = parent;
        parent.children.put(child.getName(), child);
        return parent;
    }
}
```

4.8 测试用例

```
package cxfdemo.test;

import java.util.ArrayList;
import java.util.HashMap;
import java.util.HashSet;
import java.util.List;
import java.util.Map;
import java.util.Set;
```

```
import javax.xml.ws.Endpoint;


import junit.framework.Assert;
import junit.framework.TestCase;


import org.apache.cxf.jaxws.JaxWsProxyFactoryBean;


import cxfdemo.CXFDemo;
import cxfdemo.CXFDemoImpl;
import cxfdemo.MyUser;
import cxfdemo.User;


public class TestEndpoint extends TestCase {


    private static final String ADDRESS = "http://localhost:9000/cxfdemo";
    private static CXFDemo service;


    @Override
    protected void setUp() throws Exception {
        // TODO Auto-generated method stub
        super.setUp();
        if(null==service){
            System.out.println("Starting Server");
            CXFDemoImpl demo = new CXFDemoImpl();


            Endpoint.publish(ADDRESS, demo);
            System.out.println("Start success");



        JaxWsProxyFactoryBean factory = new JaxWsProxyFactoryBean();
        factory.setServiceClass(CXFDemo.class);
        factory.setAddress(ADDRESS);
        service = (CXFDemo)factory.create();
        }


    }


    public void testSayHello(){
        Assert.assertEquals(service.sayHello("foo"), "hello foo");
    }
```

```
public void testSayHelloToUser(){

    User user = new User();
    user.setName("Holbrook");
    String result = service.sayHelloToUser(user);
    Assert.assertEquals(result,"hello Holbrook");
}

public void testGetUser(){
    User user = service.getUser("Holbrook");
    Assert.assertEquals("Holbrook",user.getName());
}

public void testGetUsers(){
    List<User> users = service.getUsers();
    Assert.assertEquals(1,users.size());
}

public void testSetUsers(){
    List<User> users = new ArrayList<User>();
    users.add(new User());
    users.add(new User());
    users.add(new User());

    Assert.assertEquals(3,service.setUsers(users).intValue());
}

public void testGetUserSet(){
    Set<User> userSet = service.getUserSet();
    Assert.assertEquals(2,userSet.size());
}

public void testSetUserSet(){
     Set<User> set = new HashSet<User>();
     set.add(new User());
     set.add(new User());

     Assert.assertEquals(2, service.setUserSet(set).intValue());
}
```

```
public void testGetUserMap(){
    Map<String,User> map = service.getUserMap();
    Assert.assertTrue(map.containsKey("Holbrook"));
    Assert.assertTrue(map.containsKey("wanghaikuo"));
}

public void testSetUserMap(){
    HashMap<String,User> map = new HashMap<String,User>();
    User user1 = new User();
    user1.setName("Holbrook");
    map.put("Holbrook", user1);

    User user2 = new User();
    user2.setName("wanghaikuo");
    map.put("wanghaikuo", user2);

    Assert.assertEquals(2, service.setUserMap(map).intValue());
}

public void testAddChild(){
    User root = new User();
    root.setName("root");
    //root.parent = root;

    User child = new User();
    child.setName("child");
    User parent = service.addChild(root, child);

    Assert.assertTrue(parent.children.containsKey("child"));
    Assert.assertEquals(parent.children.get("child").parent, parent);
}

public void testInheritance(){
    User parent = new User();
    MyUser child = new MyUser();
    child.setName("child");
    child.myProp = "subclass Prop";
    User root = service.addChild(parent, child);
```

```
        User newChild = root.children.get("child");
        System.out.println(newChild instanceof MyUser);
        System.out.println(((MyUser)newChild).myProp);
    }

}
```

Reference：

https://my.oschina.net/u/246522/blog/151160

http://www.cnblogs.com/hoojo/archive/2011/03/30/1999563.html

- 上一篇 Restful 接口传递参数
- 下一篇 Maven 本地仓库，远程仓库，中央仓库，Nexus私服，镜像 详解

相关文章推荐

- CXF+JAXB处理复杂数据
- WebService编程（1:Axis；2:Axis2/XFire；3:CX...
- CXF webservice JAXB 处理复杂数据类型方法
- WebService介绍(WebService基础知识、XFire、...
- CXF：几点认识
- 【WebService】CXF处理javaBean等复合类型以...
- WebService-CXF
- 设计美好的服务器(7)--Apache CXF笔记
- Apache CXF学习笔记二-复杂数据类型
- JAX-WS CXF Web services

猜你在找

【直播】机器学习&深度学习系统实战（唐宇迪）
【直播回放】深度学习基础与TensorFlow实践（王琛）
【直播】机器学习之凸优化（马博士）
【直播】机器学习之概率与统计推断（冒教授）
【直播】TensorFlow实战进阶（智亮）
【直播】Kaggle 神器：XGBoost 从基础到实战（冒教授）
【直播】计算机视觉原理及实战（屈教授）
【直播】机器学习之矩阵（黄博士）
【直播】机器学习之数学基础
【直播】深度学习30天系统实训（唐宇迪）

查看评论

暂无评论