java关于对象(javabean)与xml之间的相互转换 - beyondLi71的博客 - CSDN博客

今天我们要探讨的一个问题是，如何将xml格式的字符串转换成对象，以及对象转换成xml字符串。

简介

现如今，虽然数据的传输大部分都是用json格式来进行传输，但是xml毕竟也会有一些老的项目在进行使用，身为一个万能的程序员。我们又怎能不会使用它呢？正常的老式方法是通过获取节点来进行一系列操作，个人感觉太过于复杂、繁琐。今天推荐一套简单的api。XStream类。好了废话不多说，直接上代码。(为了讲述的更加清晰。我们全程不使用任何注解，只调用api来达到效果。在理解了原理的情况下看下注解的语法即会使用)

对象转xml

首先我们从简单的对象转xml为例来进行讲解，因为xml转对象会相对复杂，我们由浅到深。

第一步：导入jar包，本人项目以gradle搭建。jar包下载引用如下(普通项目从网上找一下jar包放到lib文件夹下即可)

compile(“com.thoughtworks.xstream:xstream:1.4.10”)

第二步：创建对象(创建User与Customer，不要被名字误导。。。没什么关系)

1.创建User对象(使用lombok创建getset方法，不了解的同学直接手动创建即可)

```
package com.kingboy.springboot.domain.dto;import lombok.Data;import java.util.List;/** * Created by beyondLi on 2017/6/14. */@DatapublicclassUser {private String name;    private Integer age;    private List customer;}
```

- 1
- 2
- 3
- 4
- 5
- 6
- 7
- 8
- 9
- 10
- 11
- 12
- 13
- 14
- 15

- 1
- 2
- 3
- 4
- 5
- 6
- 7
- 8
- 9
- 10
- 11
- 12
- 13
- 14
- 15

2.创建Customer对象(同上)

```
package com.kingboy.springboot.domain.dto;import lombok.Data;/** * Created by beyondLi on 2017/6/14. */@DatapublicclassCustomer {private String commodity;}
```

- 1
- 2
- 3
- 4
- 5
- 6
- 7
- 8
- 9
- 10
- 11

- 1
- 2
- 3
- 4
- 5
- 6
- 7
- 8
- 9
- 10
- 11

第三步：测试类

```
//创建user对象与customer对象并赋值        User user = new User();        Customer customer1 = new Customer();        Customer customer2 = new Customer();        customer1.setCommodity("商品1");        customer2.setCommodity("商品2");        List list = new ArrayList<>();        list.add(customer1);        list.add(customer2);        user.setName("beyondLi");        user.setAge(23);        user.setCustomer(list);        //创建xStream对象        XStream xStream = new XStream();        //调用toXML将对象转成字符串        String s = xStream.toXML(user);        System.out.println(s);
```

- 1
- 2
- 3
- 4
- 5
- 6
- 7
- 8
- 9
- 10
- 11
- 12
- 13
- 14
- 15
- 16
- 17

- 1
- 2
- 3
- 4
- 5
- 6
- 7
- 8

- 9
- 10
- 11
- 12
- 13
- 14
- 15
- 16
- 17

第四步：输出结果

```
<com.kingboy.springboot.domain.dto.User>
  <name>beyondLi</name>
  <age>23</age>
  <customer>
    <com.kingboy.springboot.domain.dto.Customer>
      <commodity>商品1</commodity>
    </com.kingboy.springboot.domain.dto.Customer>
    <com.kingboy.springboot.domain.dto.Customer>
      <commodity>商品2</commodity>
    </com.kingboy.springboot.domain.dto.Customer>
  </customer>
</com.kingboy.springboot.domain.dto.User>
```


第五步：优化

上述我们看到了结果，但是貌似和我们想要的不太一样呀，怎么有些是全路径名称呢？这里解决方法很简单。主要讲思路。因为关系到xml转对象时候封装是否报错的问题。默认情况下当对象没有名字的时候，例如最一开始的对象 以及对象中的集合类型中泛型的类型，这些我们都没办法给他起名字的，默认情况下它是全路径名称。所以如果我们不处理，当xml转对象的时候传来的xml直接都是名字，而我们如果直接调用api封装就会出现因为名字不一致所以封装失败而报错的问题。所以我们要给全路径名称的类起别名（**注！最好不要无脑性的给所有对象都起别名，虽然也可以解决问题。但是我们最好理解透彻，只给有需要的类起别名**），代码如下

```
//创建user对象与customer对象并赋值        User user = new User();        Customer customer1 = new Customer();        Customer customer2 = new Customer();        customer1.setCommodity("商品1");        customer2.setCommodity("商品2");        List list = new ArrayList<>();        list.add(customer1);        list.add(customer2);        user.setName("beyondLi");        user.setAge(23);        user.setCustomer(list);        //创建xStream对象        XStream xStream = new XStream();        //给指定类起别名        xStream.alias("User",User.class);        xStream.alias("Customer",Customer.class);        //调用toXML 将对象转成字符串        String s = xStream.toXML(user);        System.out.println(s);
```

- 1
- 2
- 3
- 4
- 5
- 6
- 7
- 8
- 9
- 10
- 11
- 12
- 13
- 14
- 15
- 16
- 17
- 18
- 19
- 20

- 1
- 2

- 3
- 4
- 5
- 6
- 7
- 8
- 9
- 10
- 11
- 12
- 13
- 14
- 15
- 16
- 17
- 18
- 19
- 20

结果：

```
<User>
  <name>beyondLi</name>
  <age>23</age>
  <customer>
    <Customer>
      <commodity>商品1</commodity>
    </Customer>
    <Customer>
      <commodity>商品2</commodity>
    </Customer>
  </customer>
</User>
```


希望阅读到这里的同学将上面那个起别名理解清楚，什么时候起别名，什么时候不需要起别名，这样才能在xml转对象的时候不出现任何的偏差和错误。

XML转对象

第一步：导入jar包。同上

第二步：创建User与Customer对象。同上

第三部：测试类

首先我们展示一下上面提示的不起别名报错问题，我模拟了一个xml类型的字符串，但是不起别名

```
//模拟一个xml格式字符串        String xml = "\n" +                "  beyondLi\n" +                "  23\n"
+                "  \n" +                "    \n" +                "      商品1\n" +                "
</Customer>\n" +                "    \n" +                "      商品2\n" +                "
</Customer>\n" +                "  </customer>\n" +                "</user>";        //创建xStream对象
XStream xstream = new XStream();        //起别名，先不写，让其报错        User user2 = (User)
xstream.fromXML(xml);        System.out.println(user2);
```

- 1
- 2
- 3
- 4
- 5
- 6
- 7
- 8
- 9
- 10
- 11
- 12
- 13
- 14

- 15
- 16
- 17
- 18

- 1
- 2
- 3
- 4
- 5
- 6
- 7
- 8
- 9
- 10
- 11
- 12
- 13
- 14
- 15
- 16
- 17
- 18

结果

```
Security framework of XStream not initialized, XStream is probably vulnerable.

com.thoughtworks.xstream.mapper.CannotResolveClassException: user

    at com.thoughtworks.xstream.mapper.DefaultMapper.realClass(DefaultMapper.java:81)
    at com.thoughtworks.xstream.mapper.MapperWrapper.realClass(MapperWrapper.java:125)
    at com.thoughtworks.xstream.mapper.DynamicProxyMapper.realClass(DynamicProxyMapper.java:55)
    at com.thoughtworks.xstream.mapper.MapperWrapper.realClass(MapperWrapper.java:125)
    at com.thoughtworks.xstream.mapper.PackageAliasingMapper.realClass(PackageAliasingMapper.java:88)
    at com.thoughtworks.xstream.mapper.MapperWrapper.realClass(MapperWrapper.java:125)
    at com.thoughtworks.xstream.mapper.ClassAliasingMapper.realClass(ClassAliasingMapper.java:79)
    at com.thoughtworks.xstream.mapper.MapperWrapper.realClass(MapperWrapper.java:125)
    at com.thoughtworks.xstream.mapper.ArrayMapper.realClass(ArrayMapper.java:74)
    at com.thoughtworks.xstream.mapper.MapperWrapper.realClass(MapperWrapper.java:125)
```


和明显，虽然我们对象user和customer起名字都没有错误。但是报错说找不到。这个报错结合上面的对象转xml的结果。我们就可以明显的看出问题出在了哪里。

正确代码如下

```
//模拟一个xml格式字符串        String xml = "\n" +                "  beyondLi\n" +                "  23\n"
+                "  \n" +                "    \n" +                "      商品1\n" +                "
</Customer>\n" +                "    \n" +                "      商品2\n" +                "
</Customer>\n" +                "  </customer>\n" +                "</user>";        //创建xStream对象
XStream xstream = new XStream();        //将别名与xml名字相对应        xstream.alias("user", User.class);
xstream.alias("Customer", Customer.class);        User user2 = (User) xstream.fromXML(xml);
System.out.println(user2);
```

- 1
- 2
- 3
- 4
- 5
- 6
- 7
- 8
- 9
- 10
- 11

- 12
- 13
- 14
- 15
- 16
- 17
- 18
- 19
- 20
- 21

- 1
- 2
- 3
- 4
- 5
- 6
- 7
- 8
- 9
- 10
- 11
- 12
- 13
- 14
- 15
- 16
- 17
- 18
- 19
- 20
- 21

结果:

```
Security framework of XStream not initialized, XStream is probably vulnerable.
User(name=beyondLi, age=23, customer=[Customer(commodity=商品1), Customer(commodity=商品2)])
------------------------------------------------------------------------
Picked up JAVA_TOOL_OPTIONS: -Dfile.encoding=UTF-8

Process finished with exit code 0
```


转换成功

其实xml与对象之间的相互转换并没有难度，但是经常出错和不成功其实就是因为在别名上的问题。而导致无法匹配。希望这篇文当可以帮助更多的同学解决问题。

以上观点仅是个人理解。如有错误或不完善，还望指出，共同成长