**Using Jaxb To Serialize Java-Objects To Xml**

Recently, I had a requirement of serializing some Java objects to xml. Though I had assumed it to be fairly straight forward, during implementation, I found out the hard way that it is not at all intuitive. The following are the ground rules:

1. You need to decorate the class which you want to serialize with the ***@XmlRootElement*** annotation. This rules out having the flexibility of serializing a List or a Map directly. It has to be wrapped within a class having the ***@XmlRootElement*** annotation.
2. Jaxb needs a default constructor. So if you have an immutable class with all fields set through the constructor, you need to provide a default constructor and set all the final fields to null.
3. Though Jaxb works well with the List, it needs a custom adapter to serialize a Map with the ***@XmlJavaTypeAdapter*** annotation, as we will see. This is the single most pain point in using Jaxb.

**The serializer code**

I am using the O/X Mappers from Spring (<http://docs.spring.io/spring-ws/site/reference/html/oxm.html>) for marshalling. I found it very convenient, specially if I want to switch from Jaxb to say, Xstream. The code is pretty simple:

@Override

**public** **void** serialize(Object object, OutputStream outputStream) {

Jaxb2Marshaller marshaller = **new** Jaxb2Marshaller();

marshaller.setClassesToBeBound(EmployeeGroups.**class**);

Map<String, Object> marshallerProperties = **new** HashMap<>();

marshallerProperties.put(javax.xml.bind.Marshaller.***JAXB\_FORMATTED\_OUTPUT***, Boolean.***TRUE***);

marshaller.setMarshallerProperties(marshallerProperties);

Result xmlResultCapturer = **new** StreamResult(outputStream);

**try** {

marshaller.marshal(object, xmlResultCapturer);

} **catch** (Exception e) {

***LOG***.error("could not convert rmi output to xml", e);

}

}

This is the version using plain Jaxb:

@Override

**public** **void** serialize(Object object, OutputStream outputStream) {

JAXBContext jaxbContext;

**try** {

jaxbContext = JAXBContext.*newInstance*(EmployeeGroups.**class**);

} **catch** (JAXBException e) {

**throw** **new** RuntimeException(e);

}

Marshaller marshaller;

**try** {

marshaller = jaxbContext.createMarshaller();

} **catch** (JAXBException e) {

**throw** **new** RuntimeException(e);

}

Result xmlResultCapturer = **new** StreamResult(outputStream);

**try** {

marshaller.marshal(object, xmlResultCapturer);

} **catch** (Exception e) {

***LOG***.error("could not convert rmi output to xml", e);

}

}

**The POJO to be serialized**

The POJO looks like:

@XmlRootElement

**public** **class** EmployeeGroups {

**private** **final** **int** groupSize;

**private** **final** Map<EmployeeRole, List<Employee>> employeeGroups;

**public** EmployeeGroups(**int** groupSize, Map<EmployeeRole, List<Employee>> employeeGroups) {

**this**.groupSize = groupSize;

**this**.employeeGroups = employeeGroups;

}

/\*\*

\* Default constructor, as Jaxb will not work without it

\*/

**public** EmployeeGroups() {

**this**(0, **null**);

}

**public** **int** getGroupSize() {

**return** groupSize;

}

**public** Map<EmployeeRole, List<Employee>> getEmployeeGroups() {

**return** employeeGroups;

}

}

**Attempt 1**

When I try to generate xml, I get a blank document:

<?xml version="1.0" encoding="UTF-8" standalone="yes"?>

<employeeGroups/>

The possible cause might be that since the object is immutable without any setters for the fields, Jaxb cannot figure out the fields to serialize. So, this time we will try to explicitly ask Jaxb to serialize the fields by adding the ***@XmlAccessorType(XmlAccessType.FIELD)*** annotation to the EmployeeGroups.

**Attempt 2**

The 2nd attempt gives rise to the following exception:

Caused by: com.sun.xml.internal.bind.v2.runtime.IllegalAnnotationsException: 1 counts of IllegalAnnotationExceptions

java.util.List is an interface, and JAXB can't handle interfaces.

this problem is related to the following location:

at java.util.List

at private final java.util.Map com.swayam.demo.xml.EmployeeGroups.employeeGroups

at com.swayam.demo.xml.EmployeeGroups

This message is actually misleading. What it means is that its high time for us to write our custom adapter for handling the Map.

**Sources**

The sources can be found here: <https://github.com/paawak/blog/tree/master/code/jaxb-demo>

**Reference**

<https://jaxb.java.net/tutorial/>

<http://docs.spring.io/spring-ws/site/reference/html/oxm.html>