How to use spring to marshal and unmarshal xml?

Asked 7 years, 4 months ago Modified 1 year, 11 months ago Viewed 119k times

28

I have a spring boot project. I have a few xsds in my project. I have generated the classes using maven-jaxb2-plugin. I have used this tutorial to get a sample spring boot application running.

```
import org.kaushik.xsds.XOBJECT;

@SpringBootApplication
public class JaxbExample2Application {

public static void main(String[] args) {
    //SpringApplication.run(JaxbExample2Application.class, args);
    XOBJECT xObject = new XOBJECT('a',1,2);

    try {
        JAXBContext jc = JAXBContext.newInstance(User.class);

        Marshaller marshaller = jc.createMarshaller();
        marshaller.setProperty(Marshaller.JAXB_FORMATTED_OUTPUT, true);
        marshaller.marshal(xObject, System.out);

    } catch (PropertyException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    } catch (JAXBException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    }
  }
}
```

But my concern is that I need to have all the jaxb classes of the schema mapped. Also is there something in Spring that I can use to make my task easier. I have looked at the Spring OXM project but it had application context configured in xml. Does spring boot have anything that I can use out of the box. Any examples will be helpful.

Edit

I tried xerx593's answer and I ran a simple test using main method

```
    JaxbHelper jaxbHelper = new JaxbHelper();
    Jaxb2Marshaller marshaller = new Jaxb2Marshaller();
    marshaller.setClassesToBeBound(XOBJECT.class);
    jaxbHelper.setMarshaller(marshaller);
    XOBJECT xOBJECT= (PurchaseOrder)jaxbHelper.load(new StreamSource(new
FileInputStream("src/main/resources/PurchaseOrder.xml")));
    System.out.println(xOBJECT.getShipTo().getName());
```

It ran perfectly fine. Now I just need to plug it in using spring boot.

spring spring-boot jaxb spring-oxm

Share Edit Follow

edited Jun 23, 2017 at 9:14

asked Jun 21, 2017 at 12:44
Kaushik Chakraborty
699 ● 1 ● 9 ● 16

How you configure the mappers doesn't matter. XML is just a means to an end. Just create a `Jaxb2Marshaller` and use it. – M. Deinum Jun 21, 2017 at 13:26

@M.Deinum All the examples show xml configuration of jaxb2Marshaller, I am looking for a Java configuration example. – Kaushik Chakraborty Jun 21, 2017 at 13:39

What is so hard about `new Jaxb2Marshaller()` ? – M. Deinum Jun 21, 2017 at 13:47

4 Answers

Sorted by: Highest score (default)

36

OXM is definitely the right for you!

A simple java configuration of a Jaxb2Marshaller would look like:

```
//...
import java.util.HashMap;
import org.springframework.oxm.jaxb.Jaxb2Marshaller;
//...

@Configuration
public class MyConfigClass {
    @Bean
    public Jaxb2Marshaller jaxb2Marshaller() {
        Jaxb2Marshaller marshaller = new Jaxb2Marshaller();
        marshaller.setClassesToBeBound(new Class[]{
           //all the classes the context needs to know about
           org.kaushik.xsds.All.class,
           org.kaushik.xsds.Of.class,
           org.kaushik.xsds.Your.class,
           org.kaushik.xsds.Classes.class
        });
        // "alternative/additiona - ly":
          // marshaller.setContextPath(<jaxb.context-file>)
          // marshaller.setPackagesToScan({"com.foo", "com.baz", "com.bar"});

        marshaller.setMarshallerProperties(new HashMap<String, Object>() {{
          put(javax.xml.bind.Marshaller.JAXB_FORMATTED_OUTPUT, true);
          // set more properties here...
        }});

        return marshaller;
    }
}
```

In your Application/Service class you could approach like this:

```java
import java.io.InputStream;
import java.io.StringWriter;
import javax.xml.bind.JAXBException;
import javax.xml.transform.Result;
import javax.xml.transform.stream.StreamResult;
import javax.xml.transform.stream.StreamSource;
import org.springframework.oxm.jaxb.Jaxb2Marshaller;

@Component
public class MyMarshallerWrapper {
   // you would rather:
   @Autowired
   private Jaxb2Marshaller  marshaller;
   // than:
   // JAXBContext jc = JAXBContext.newInstance(User.class);
   // Marshaller marshaller = jc.createMarshaller();

   // marshalls one object (of your bound classes) into a String.
   public <T> String marshallXml(final T obj) throws JAXBException {
      StringWriter sw = new StringWriter();
      Result result = new StreamResult(sw);
      marshaller.marshal(obj, result);
      return sw.toString();
   }

   // (tries to) unmarshall(s) an InputStream to the desired object.
   @SuppressWarnings("unchecked")
   public <T> T unmarshallXml(final InputStream xml) throws JAXBException {
      return (T) marshaller.unmarshal(new StreamSource(xml));
   }
}
```

See Jaxb2Marshaller-javadoc, and a related Answer

Share Edit Follow

edited Oct 15, 2019 at 10:05

answered Jun 21, 2017 at 14:21

xerx593
13.1k ● 5 ● 35 ● 72

1 I'll try this when I get back to work, looks just like what I need. But looking at the javadoc, you might as well do `marshaller.setPackagesToScan("org.kaushik.xsds")` (which would contain my xjc-generated JAXB-classes), right? – daniu Dec 2, 2017 at 7:02

10

If you just want `serializing/deserializing` bean with XML. I think `jackson fasterxml` is one good choice:

```java
ObjectMapper xmlMapper = new XmlMapper();
String xml = xmlMapper.writeValueAsString(new Simple());  // serializing

Simple value = xmlMapper.readValue("<Simple><x>1</x><y>2</y></Simple>",
     Simple.class); // deserializing
```

maven:

```
<dependency>
    <groupId>com.fasterxml.jackson.dataformat</groupId>
    <artifactId>jackson-dataformat-xml</artifactId>
</dependency>
```

Refer: https://github.com/FasterXML/jackson-dataformat-xml

Share Edit Follow

edited Aug 27, 2018 at 18:31
Community Bot
1 ● 1

answered Nov 22, 2017 at 5:14
bluearrow
884 ● 2 ● 12 ● 29

2 `fasterxml` have a lot of limitations - look at them carefull before using. – Cherry Feb 1, 2019 at 12:35

7

Spring BOOT is very smart and it can understand what you need with a little help.

To make XML marshalling/unmarshalling work you simply need to add annotations @XmlRootElement to class and @XmlElement to fields without getter and target class will be serialized/deserialized automatically.

Here is the DTO example

```
package com.exmaple;

import lombok.AllArgsConstructor;
import lombok.Getter;
import lombok.NoArgsConstructor;
import lombok.Setter;
import lombok.ToString;

import javax.xml.bind.annotation.XmlElement;
import javax.xml.bind.annotation.XmlRootElement;
import java.io.Serializable;
import java.util.Date;
import java.util.Random;

@AllArgsConstructor
@NoArgsConstructor
@ToString
@Setter
@XmlRootElement
public class Contact implements Serializable {
    @XmlElement
    private Long id;

    @XmlElement
    private int version;

    @Getter private String firstName;

    @XmlElement
    private String lastName;

    @XmlElement
    private Date birthDate;

    public static Contact randomContact() {
        Random random = new Random();
        return new Contact(random.nextLong(), random.nextInt(), "name-" +
random.nextLong(), "surname-" + random.nextLong(), new Date());
    }
}
```

And the Controller:

```
package com.exmaple;

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.bind.annotation.ResponseBody;

@Controller
@RequestMapping(value="/contact")
public class ContactController {
    final Logger logger = LoggerFactory.getLogger(ContactController.class);

    @RequestMapping("/random")
    @ResponseBody
    public Contact randomContact() {
        return Contact.randomContact();
    }

    @RequestMapping(value = "/edit", method = RequestMethod.POST)
    @ResponseBody
    public Contact editContact(@RequestBody Contact contact) {
        logger.info("Received contact: {}", contact);
        contact.setFirstName(contact.getFirstName() + "-EDITED");
        return contact;
    }
}
```

You can check-out full code example here: https://github.com/sergpank/spring-boot-xml

Any questions are welcome.

Share Edit Follow edited Jan 9, 2019 at 11:46 answered Sep 7, 2017 at 13:33

sergpank
988 ● 10 ● 18

You can use `StringSource` / `StringResult` to read / read xml source with spring

4

```
@Autowired
   Jaxb2Marshaller jaxb2Marshaller;

   @Override
   public Service parseXmlRequest(@NonNull String xmlRequest) {
       return (Service) jaxb2Marshaller.unmarshal(new StringSource(xmlRequest));
   }

   @Override
   public String prepareXmlResponse(@NonNull Service xmlResponse) {
       StringResult stringResult = new StringResult();
       jaxb2Marshaller.marshal(xmlResponse, stringResult);
       return stringResult.toString();
   }
```

Share Edit Follow

edited Jan 9, 2019 at 8:09
piet.t
11.9k ● 21 ● 44 ● 55

answered Nov 28, 2018 at 15:41
Mohamed.Abdo
2,180 ● 1 ● 19 ● 12

This reply needs more description to get votes. – CodeSlave Dec 14, 2022 at 11:55