The [OXM](http://docs.spring.io/spring-framework/docs/3.0.x/reference/oxm.html) module provides an abstraction layer that supports Object/XML mapping implementations for JAXB, Castor, XMLBeans, JiBX and XStream.

<http://www.mkyong.com/java/jaxb-hello-world-example/>

No extra jaxb libraries are required if you are using JDK1.6 or above, because [JAXB is bundled in JDK 1.6](http://jaxb.java.net/guide/Which_JAXB_RI_is_included_in_which_JDK_.html).

1. [**Diff between STAX and SAX**](#stax_sax)

JAXB:

**JAXB**, stands for **Java Architecture for XML Binding**, using JAXB annotation to convert Java object to / from XML file. In this tutorial, we show you how to use JAXB to do following stuffs.

1. **Marshalling** – Convert a Java object into a XML file.
2. **Unmarshalling** – Convert XML content into a Java Object.

technologies used in this article

1. JDK 1.6
2. JAXB 2.0

Working with JAXB is easy, just annotate object with JAXB annotation, later use jaxbMarshaller.marshal() orjaxbMarshaller.unmarshal() to do the object / XML conversion.

Difference between JAXB and Castor?  
Please note that JAXB is an API, and there are multiple implementations available.

Sun provides a reference implementation and package it with J2EE (its available in J2SE 1.6 also). Castor was born before JAXB came out from Sun, and offers some extra features. But if all you want is plain XML binding, then the reference Sun implementation should work great.

storm

FOR FASTER reading XML to Object., use XMLInputFactory and XMLStreamReader

**public** **class** JAXBExample {

**public** **static** **void** main(String[] args) {

**try** {

File file = **new** File("C:\\Users\\cherukun\\Desktop\\SURV3\\Data migration\\DEV\\mastersheetdata11k.xml");

/\*JAXBContext jaxbContext = JAXBContext.newInstance(DataSet.class);

Unmarshaller jaxbUnmarshaller = jaxbContext.createUnmarshaller();

DataSet dataset = (DataSet) jaxbUnmarshaller.unmarshal(file);

System.out.println(dataset.getRecord().size());\*/

FileReader fr = **new** FileReader(file);

JAXBContext jc = JAXBContext.*newInstance*(DataSet.**class**);

Unmarshaller unmarshaller = jc.createUnmarshaller();

XMLInputFactory xmlif = XMLInputFactory.*newInstance*();

//XMLEventReader xmler = xmlif.createXMLEventReader(fr);

XMLStreamReader xmler = xmlif.createXMLStreamReader(fr);

**long** beginTime = System.*currentTimeMillis*();

DataSet dataset = (DataSet)unmarshaller.unmarshal(xmler);

**long** endTime = System.*currentTimeMillis*();

System.*out*.println(dataset.getRecord().size());

System.*out*.println("time taken to read xml to object in Ms "+(endTime-beginTime));

/\*int i=0;

for (Record r: dataset.getRecord()){

System.out.println(i+++" ->"+r.getSUID());

}\*/

} **catch** (Exception e) {

e.printStackTrace();

}

}

}

**Diff between STAX and SAX**

* StAX-enabled clients are generally easier to code than SAX clients. While it can be argued that SAX parsers are marginally easier to write, StAX parser code can be smaller and the code necessary for the client to interact with the parser simpler.
* StAX is a bidirectional API, meaning that it can both read and write XML documents. SAX is read only, so another API is needed if you want to write XML documents.
* SAX is a push API, whereas StAX is pull. The trade-offs between push and pull APIs outlined above apply here.