

JAXB Tips and Tricks Part 2 Generating Java Classes from XML Schema

By

Rob Ratcliff



What is JAXB?

- ▶ Java Architecture for XML Binding
- ▶ Maps an XML Schema into Java Objects
 - ▶ Experimental support for DTD, RelaxNG and WSDL
 - ▶ Generates classes by default, but can specify existing target classes
- ▶ Generates a XML Schema from annotated Java Objects
- ▶ Saves programmers labor and debug time compared to manual parsing, especially as schemas evolve



History

- ▶ Version 1.0 released in 2003 as part of JSR 31
 - ▶ Marshalling and Unmarshalling code generated
- ▶ Version 2.0 released in 2006 as part of JSR 222 for Java 6
 - ▶ Reflection based marshalling and unmarshalling code - much cleaner
- ▶ Version 2.2 released in 2009
- ▶ Version 2.2.3 bundled with Java 7
- ▶ Version 2.2.8 bundled with Java 8
 - ▶ Support for 3rd party implementations of JAXB such as EclipseLink/MOXY
 - ▶ jaxb-impl.jar has to be on classpath not in endorsed directory or endorsed classpath
 - ▶ jaxb-api.jar in endorsed directory or endorsed classpath
- ▶ Latest release is 2.2.11
- ▶ JAXB has a git repository

Top 10 Reasons JAXB is a Fav!

1. Built into the JDK, but overridable with new versions or 3rd party implementations
2. Easy to use
3. Well documented
4. Logical API
5. Robust and Efficient implementation
6. Complete control of the fields that get serialized to XML
7. Handles mapping from XML Schema or Annotated Java classes
8. Community generated plugins to customize code generation
9. Good default mappings
10. Default mappings can be completely overridden

XML Schema to Java Classes



Default XML Schema to Java Bindings

XML Schema Data Type	Java Data Type (lower case indicates a primitive data type)
anySimpleType (for xsd:element of this type)	java.lang.Object
anySimpleType (for xsd:attribute of this type)	java.lang.String
base64Binary	byte[]
boolean	boolean or (Boolean)
byte	byte (or Byte)
date	java.xml.datatype.XMLGregorianCalendar
dateTime	javax.xml.datatype.XMLGregorianCalendar
decimal	java.math.BigDecimal
double	double (or Double)
duration	javax.xml.datatype.Duration
float	float (or Float)
g	java.xml.datatype.XMLGregorianCalendar
hexBinary	byte[]
int	int (or Integer)
integer	java.math.BigInteger
long	long or (Long)
NOTATION	javax.xml.namespace.Qname
Qname	javax.xml.namespace.Qname
short	short (or Short)
string	java.lang.String
time	java.xml.datatype.XMLGregorianCalendar
unsignedByte	short (or Short)
unsignedInt	long (or Long)
unsignedShort	int

Generating Java Classes from Schema

```
xjc -extension geometry.xsd -b geometry.xjb
```



Converting Java Classes from Schema

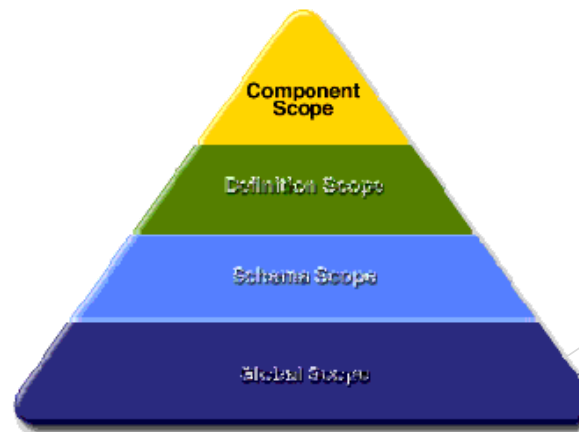
► `xjc geometry.xsd -b geometry.xjb`

► Binding File

```
<globalBindings>
  [ collectionType = "collectionType" ]
  [ fixedAttributeAsConstantProperty = "true" | "false" | "1" | "0" ]
  [ generateIsSetMethod = "true" | "false" | "1" | "0" ]
  [ enableFailFastCheck = "true" | "false" | "1" | "0" ]
  [ choiceContentProperty = "true" | "false" | "1" | "0" ]
  [ underscoreBinding = "asWordSeparator" | "asCharInWord" ]
  [ typesafeEnumBase = "typesafeEnumBase" ]
  [ typesafeEnumMemberName = "generateName" | "generateError" ]
  [ enableJavaNamingConventions = "true" | "false" | "1" | "0" ]
  [ bindingStyle = "elementBinding" | "modelGroupBinding" ]
  [ <javaType> ... </javaType> ]*
</globalBindings>
```


JAXB Binding File

```
<jaxb:bindings version="2.0" xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:jaxb="http://java.sun.com/xml/ns/jaxb" xmlns:xjc="http://java.sun.com/xml/ns/jaxb/xjc"
jaxb:extensionBindingPrefixes="xjc">
  <jaxb:bindings schemaLocation="geometry.xsd">
    <jaxb:globalBindings>
      <jaxb:serializable/>
      <xjc:javaType name="java.sql.Timestamp" xmlType="xs:dateTime" adapter="TimeConverter"/>
    </jaxb:globalBindings>
    <jaxb:schemaBindings>
    </jaxb:schemaBindings>
  </jaxb:bindings>
</jaxb:bindings>
```



Custom Mapping

- ▶ Color
- ▶ Polygons
- ▶ DateTime
- ▶ Wrapper Elements



Some Useful JAXB Plugins for Customizing Generated Java Classes from XML Schemas

- ▶ Fluent API Plugin - Generate fluent design pattern for setters
 - ▶ i.e. returns “this” for each setter so that “sets” can be chained as in:

```
person.setFirstName(firstname).setLastName(lastName).setAddress(address);
```
- ▶ JAXB2 Basics Plugin - generates toString(), equals(), hashCode(), etc.
- ▶ Value Constructor - generates a constructor with arguments
- ▶ HyperJAXB3 - Generates JPA annotated classes
<http://confluence.highsource.org/display/HJ3/Home>
- ▶ See <https://java.net/projects/jaxb2-commons/pages/Home> for more plugins
- ▶ JAK - Java API for KML - <https://github.com/micromata/javaapiforkml>

Converting XML to Java

```
StringReader reader = new StringReader(xmlString);  
JAXBContext context = JAXBContext.newInstance(ObjectFactory.class);  
ShapeList shape = (ShapeList)  
context.createUnmarshaller().unmarshal(reader);
```

nillable vs. minOccurs=0?

▶ Variants

- ▶ `<myElement attr1='true'>some content</myElement>`
- ▶ `<myElement/>`
- ▶ `<myElement xsi:nil='true'/>`
- ▶ Missing `<myElement/>`
- ▶ `minOccurs=1` will map to the Java primitive
- ▶ `minOccurs=0` or `nillable=true` will map to the Object version of primitives
- ▶ `minOccurs=0` and `nillable=true` will map to `JAXBElement<Double>`

Validating Against a Schema

- ▶ No validation by default
- ▶ JAXB tries to be forgiving when there is an extra field or missing field to help deal with version
- ▶ Just `Unmarshaller.setValidating(true)` to begin validating

Converting Java to XML

```
JAXBContext context = JAXBContext.newInstance(ObjectFactory.class);  
  
StringWriter stringWriter = new StringWriter();  
Marshaller marshaller = context.createMarshaller();  
marshaller.setProperty("jaxb.formatted.output", Boolean.TRUE);  
marshaller.marshal(jaxbElement, stringWriter);  
return stringWriter.toString();
```

Tips

1. Performance
 1. Cache the Threadsafe JAXBContext for much better performance - put in a static or singleton
 2. Marshallers and Unmarshallers are not Threadsafe so if performance is critical use pools of these
2. Don't embed JAXB customizations in schema, put in a binding file
3. Good mapping requires iterations on schema, binding customizations and the resulting Java code

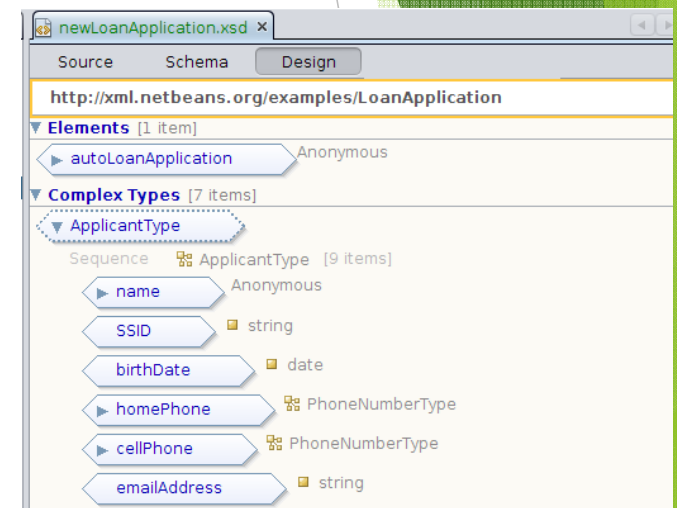
Schema Evolution/Versioning

- ▶ Schema Versioning Approaches
 - ▶ Add version as part of the namespace (don't use a raw number)
 - ▶ Add a version field in the XML
- ▶ Java package name generated from target namespace by default
 - ▶ Advantage: Allows multiple versions of the schema to be handled in the same classloader
 - ▶ Disadvantage: Package names of consuming code has to be changed for new versions of the schema
- ▶ Use a different classloader (e.g. Session Bean) for each version accessed through a common Java interface
 - ▶ Advantage: Do not need to change consuming code
 - ▶ Disadvantage: Added complexity

XML Schema Editors

▶ <http://deadlock.netbeans.org/hudson/job/xml/lastSuccessfulBuild/artifact/build/updates/updates.xml>

▶ http://wiki.eclipse.org/index.php/Introduction_to_the_XSD_Editor



Current Deficits

1. Annotations/documentation doesn't get translated to JavaDoc
2. Restrictions don't get translated into validation code (MOXy may do this)



Questions?

Graphs using ID and IDREF

