

JAXB annotations

Motivation:	Map metadata from ISO-19139 to existing classes
Contact:	Martin Desruisseaux , Cédric Briçon
Tracker:	TBD
Attachment:	jaxb-metadata.xml (example of marshalled XML)

Description

Java objects can be binded to XML using annotations. Many frameworks exist and JAXB is one of them. The traditional approach was to keep `org.geotools.metadata` as plain JavaBeans implementations without annotations, and let peoples generate their own annotated classes in separated modules. Those annotated classes are typically generated automatically from XSD using some processor - JAXB among others can do that. However like every automatically generated code, the result are plain containers with no intelligence in the sense of no method doing non-trivial tasks related to GIS. It is possible to modify the generated classes afterward, but adding intelligence in the generated classes duplicate the core library.

Our proposal is to annotate the existing metadata classes with JAXB, providing that:

- Those annotations do not add any dependency to the distributed JARs, neither for Java 5 or Java 6 users.
- Those annotations do not imply any API change. More specifically `Collection<SomeInterface>` stay unchanged - the metadata module must continue to work with arbitrary interfaces. This is possible through the definition of [XmlAdapter](#).
- Those annotations must map to ISO 19139 compliant XML - no custom XML.

The first point is possible since JAXB 2.0 is bundled with Java 6 (we will suggest later an approach for Java 5 users). JAXB is the result of [JSR-222](#) with the participation of *Sun Microsystems, IBM, Oracle, Novell, BEA* and others, so it is as close to a standard as possible in Java world. We would not support dependencies toward external frameworks like Hibernate or XStream in core GeoTools library, but JAXB is special because of its inclusion in the JDK.

Why JAXB annotations

- JAXB is a JCP standard, bundled in JDK 6 and supported (according JSR-222 member list) by Sun, IBM, *etc*. Because of its status, JAXB is expected to be more "well known" than other frameworks.
- JAXB doesn't introduce any dependencies in distributed JARs. Every Java 6 users have JAXB 2.0 out of the box. JAXB is "special" in this regard compared to Hibernate, XStream, *etc*.
- Using JAXB doesn't exclude usage of other frameworks. JAXB annotations are invisible in the Javadoc since they are not declared as `@Documented`, so users who don't care will even not notice that they are there.
- If we don't add JAXB annotations ourself, users wanting them in their own `org.geotools.metadata` subcl asses will have to edit and rebuild GeoTools themself, since they would need to add `@XmlTransient` on all super-classes.
- Users wanting JAXB in a separated set of classes (probably generating them from XSD using JAXB processor) while get a lot of duplication. If they want to implement GeoAPI interfaces, significant hand-work would be required anyway.
- The hand-written `org.geotools.metadata` classes offer more flexibility than generated classes. They also contains additional functionalities, like `contains(point)` methods. Those methods could be added to generated classes, but doing so duplicate the GeoTools core library. It still doable for metadata classes, but

would be more difficult for referencing classes.

- Annotations keep XML binding close to the code. The advantage is similar to javadoc comments which keep documentation close to the code, and thus increase the chance that the binding/documentation is updated in case of code change.
- Annotations allow usage of *Annotation Processing Tools* (APT), which may open interesting perspectives in the future. A short term APT could be to compare XML binding against the UML names declared in GeoAPI interfaces.

Inconvenient

- Increase the metadata JAR size for Java 6 users, since the class files are expanded with annotations. Adapters also introduce new non-public classes (invisible to users).
- Introduce yet another XML technology in GeoTools.

Java 5 users

We will configure the build in such a way that Java 5 users will not be affected. First, we will create a new module in the `org.geotools.maven` group, which will be used **at compile time only**. This module will contain a copy of JAXB annotations we use, except that their retention policy will be changed as below:

```
@Retention(RetentionPolicy.SOURCE)
```

According to Javadoc: "*Annotations are to be discarded by the compiler*". The results for Java 5 build will be as if no annotation had been declared.

The metadata module will depend on this temporary `gt2-mock-jaxb` module with `<scope>provided</scope>`, which implies that this dependency is used for compilation only and does not appear in the distributed JAR. Other modules will not inherit this dependency.

Java 6 users

Everything described in the previous section does **not** apply to Java 6 users. The `gt2-mock-jaxb` module is never declared for Java 6 users, and the metadata classes are compiled with the annotations bundled in the JDK.

Status

This proposal was voted on; added to the 2.5.x series - and then withdrawn due to build issues with Java 5 vs Java 6.

- [Andrea Aime](#) +1
- [Ian Turton](#) (waiting for email vote)
- [Justin Deoliveira](#)
- [Jody Garnett](#) +1
- [Martin Desruisseaux](#) +1
- [Simone Giannecchini](#) +0

Community support:

- [Gabriel Roldán](#) +1

Tasks

	no progress	✓	done	✗	impeded	⚠	lack mandate /funds/time	?	volunteer needed
--	-------------	---	------	---	---------	---	--------------------------	---	------------------

[Cédric Briçon](#) will be doing the following work:

1. ✓ Add a build/maven (or build/mock, or other name to be determined) module for allowing Java 5 users to get annotation erased.
2. ✓ Updated `modules/library/metadata/pom.xml` with Java 5 and Java 6 profiles.
3. ✓ Add adapters in `org.geotools.resources.jaxb` or some other package invisible to users.
4. ✓ Add annotations to the Metadata implementations.
5. ✓ User document page for the metadata module showing how to generate an XML document.
6. ✓ User document page for the metadata module showing how to parse an XML document

Progression state

Proposal completed; and withdrawn since the build was not stable between Java 6 and Java 5 JAXB differences.

API Change

The API change consists of the addition of annotations only; as such the functionality of the code has not changed but the information you can determine using reflection has changed. This information is used by jaxb when marshalling and unmarshalling.

Only implementations are annotated.

BEFORE

```
public class CitationImpl extends
MetadataEntity implements Citation {
...
    /**
     * Returns the name by which the cited
resource is known.
     */
    public InternationalString getTitle() {
        return title;
    }

    /**
     * Set the name by which the cited
resource is known.
     */
    public synchronized void setTitle(final
InternationalString newValue) {
        checkWritePermission();
        title = newValue;
    }
...
}
```

AFTER

```
@XmlType(propOrder={"title", ...})
@XmlRootElement(name = "CI_Citation")
public class CitationImpl extends
MetadataEntity implements Citation {
...
    /**
     * Returns the name by which the cited
resource is known.
     */
    @XmlElement(name = "title", required =
true, namespace =
"http://www.isotc211.org/2005/gmd")
    public InternationalString getTitle() {
        return title;
    }

    /**
     * Set the name by which the cited
resource is known.
     */
    public synchronized void setTitle(final
InternationalString newValue) {
        checkWritePermission();
        title = newValue;
    }
...
}
```