Castor

Overview

Castor is a tool to take an existing schema, generate Java objects, and serialize those objects at runtime to XML and/or a DB.

Advantages:

- Works well with Apache Axis when the schema is pre-defined.
- Works great when serializing to/from raw XML files.
- Can serialize to DB.

Disadvantages:

- Not good with very complex schemas.
- Generated Java objects are quite complex.
- Numerous bugs at this time – especially for XML used as rendering markup rather than data wrapping.
- When using with Axis, use of helper tools (such as XDoclet) is limited.
- Castor handles document-centric XML very poorly. This is not well-documented by Castor, but it is well understood by several people in the Digital Library. It can be summarized by demonstrating that Castor will round-trip this: <a>b<c>d</c>e as: <a>be<c>d</c>. (Furthermore, more complex mixed content will inappropriately fail Castor's parsing validation.) This problem is a fatal issue with EAD, and likely would be equally as fatal with TEI.

Alternatives

If the schema is not pre-defined, consider other alternatives:

- If Axis is used, consider using XDoclet to generate Java objects and WSDL, etc.
- If ORM is used, consider a more mature tool, such as Hibernate.

Case Study

I have several metadata schemas that work well with Castor, and I'm just going to ignore them for now. This case study is an example of challenges with Castor: the EAD2002 Schema.

Step 1: Get Schema

The first problem is that there is no official schema for EAD2002. I have 2 options here, the first is to create my own schema, and the second is to use a non-official schema generated by another group. I will look at both options.

Option 1: I used <oXygen/> to generate a schema based on 3 EAD2002 XML files which I merged together, then saved it as generated_ead.xsd

Option 2: I grabbed the princeton schema from http://diglib.peinceton.edu/ead/dtd/2002/ead.xsd, and saved it as princeton_ead.xsd

Step 2: Generate Java Code

To have Castor generate the Java code, I ran the following conversions:
java org.exolab.castor.builder.SourceGenerator -package generated_ead -types j2 -i generated_ead.xsd
java org.exolab.castor.builder.SourceGenerator -package princeton_ead -types j2 -i princeton_ead.xsd

This reveals the first set of errors:

- Princeton
    - Warning: The W3C datatype 'ENTITY' is not currently supported by Castor Source Generator.

To work around this issue, I have created a variant princeton schema where the xs:ENTITY references have been replace by xs:string. This is named princeton2_ead.xsd, and is handled appropriately by the Castor schema generation.

Step 3: Generate Test Class, and test

At this point, I want to de-serialize (read) and serialize (write) the XML to test if and how well the Castor generated code will handle the actual XML. I will generate a test class for each schema, 1 at a time, and see how it handles each of my 4 input files (the original 3, plus the merged file).

First problem is that the princeton_ead code is broken. All sorts of undefined references to an ENTITY type. I'll just skip that option since it had errors before as well.

Next, I'll test the princeton2_ead code. The code is attached, and you might notice the import at the top which will determine which ead codebase to use.

The princeton2_ead compiles well, but fails to read any of the 4 tests:

- hohen_ead.xml:
    - Illegal Text data found as child of: _m_render
- ussteel_ead.xml:

        - Illegal Text data found as child of: _m_render
- hoagy_ead.xml:
    - The container object (princeton2_ead.M_render) cannot accept the child object associated with the element 'lb' because the container is already full!
- big_ead.xml:
    - The container object (princeton2_ead.M_render) cannot accept the child object associated with the element 'lb' because the container is already full!

However, after looking at the source of the error, I know from experience that this is a Castor bug. For some reason it tends to not like certain child-element mixed content. In particular it hates the empty <lb /> tags. So, as a hack, I'll replace all the <lb /> to a entity (which refers to a carriage return character).

As a side note, there are several bugs related to this issue. For the infrastructure project, I don't think it will be a major issue, but regeneration of XML in the form of <a>text<b/>text<b/>text is problematic. I won't go into it here, but beware of such issues. (try looking at mapping files and custom parsers for workaround options)

So, with the <lb /> hack, they princeton2_ead results in:

- hohen_ead.xml:
    - The container object (princeton2_ead.M_blocks) cannot accept the child object associated with the element 'p' because the container is already full!
- ussteel_ead.xml:
    - The container object (princeton2_ead.M_blocks) cannot accept the child object associated with the element 'p' because the container is already full!
- hoagy_ead.xml:
    - The container object (princeton2_ead.M_blocks) cannot accept the child object associated with the element 'p' because the container is already full!
- big_ead.xml:
    - Illegal Text data found as child of: _items

Next, I'll recompile the test code to use the generated_ead.xml

This also compiled well, and also failed to read any of the (<lb />) data

- hohen_ead.xml:
    - Illegal Text data found as child of: _items
- ussteel_ead.xml:
    - Illegal Text data found as child of: _items
- hoagy_ead.xml:
    - The field '_c04Choice' is a required field of class 'generated_ead.C04
- big_ead.xml:
    - Illegal Text data found as child of: _items

This also compiled well, and also failed to read most of the (non-<lb />) data

- hohen_ead.xml:
    - The field '_controlaccessChoice' is a required field of class 'generated_ead.Controlaccess
- hoagy_ead.xml:
    - The field '_c04Choice' is a required field of class 'generated_ead.C04
- big_ead.xml:
    - Illegal Text data found as child of: _items

But succeeded on:

- ussteel_ead.xml

So, things aren't looking very good.

Step 4: Abandon hope of purist approach, make Castor work.

Now, I can't see how to consistantly use the princeton schema, so I'll abaodon it, and go with a generated schema. Since this is only for internal use, and external validation is done against the princeton schema, all I have to do is provide a broader accaptability and it should be fine.

I'll pull the XML files and the schema into Oxygen for analysis.

The first thing I found is that the schema is not namespaced well. So I've added the princeton namespace to the generated schema.

Next, I found that there are way too many required attributes. Since the validation is external, I'm not going to worry about rules here, just structure. Therefore the "required" can be globally changed to "optional".

Next, I found that there are too many required elements. This is not as easy to combat since there is no simple search/replace option. However, I've changed the following:

- /schema/filedesc/notestmt --> minOccurs="0"
- /schema/descgrp/bioghist --> minOccurs="0"
- /schema/archdesc/descgrp --> minOccurs="0"
- /schema/archdesc/odd --> minOccurs="0"
- /schema/archdesc/controlaccess --> minOccurs="0"
- /schema/archdesc/userestrict --> minOccurs="0"
- /schema/archdesc/prefercite --> minOccurs="0"

- /schema/archdesc/custodhist --> minOccurs="0"
- /schema/archdesc/processinfo --> minOccurs="0"
- /schema/archdesc/bioghist --> minOccurs="0"
- /schema/archdesc/arrangement --> minOccurs="0"
- /schema/profiledesc/creation --> minOccurs="0"

Next, I found I had omitted some optional attributes in the big_ead.xml file, so I added:

- /schema/archdesc@type --> <xs:attribute name="type" use="optional" type="xs:NCName"/>
- /schema/archdesc@audience --> <xs:attribute name="audience" use="optional" type="xs:NCName"/>
- /schema/archdesc@relatedencoding --> <xs:attribute name="relatedencoding" use="optional" type="xs:NCName"/>
- /schema/frontmatter@audience --> <xs:attribute name="audience" use="optional" type="xs:NCName"/>

Next I found I had changed the order of certain elements in the big_ead.xml file, so I switched:

- /schema/archdesc/did and /schema/archdesc/descgrp

Also, some issues that were revealed by the last Castor read attempt:

- /schema/c04/choice --> minOccurs="0"
- /schema/controlaccess/choice --> minOccurs="0"

I'm quite sure there is more loosening that will be needed with future EAD2002 XML files, but that's my schema for now... Now I'll regenerate the code and test again.

Step 5: (re)Generate Test Class, and (re)test

Going throught the generation test yields a good run for the 3 main files:

- hohen_ead.xml:
- ussteel_ead.xml
- hoagy_ead.xml:

But still fails on

- big_ead.xml:
    - Illegal Text data found as child of: _items

This is regarding the multiple <date> tags embedded within the <creation> tag. (As a side note, this also **should** have existed in the Hoagy file, but I just now realized that the hoagy file in use here is not the best one. It doesn't really matter, as you will see shortly.)

After commenting out one of the <date> tags, the big_ead.xml file is processed normally.

Case study conclusion.

Castor stinks for rendering-centric XML markup. But it should be adaquate for our infrastructure needs, so long as we can tweak some of the fields, particularly repetative fields, like <date>. Also, some pre- and post- processing will be needed for certain tags like <lb/>.

I have a test case that demonstrates some of the problems with Castor, which I have attached as test.xml and test.xsd. It could probably be flushed out a little more, but it's a start.