Cocoon Field Day

Cocoon & XML Data Binding

About me

- Consultancy – '*transaction demarcation in the enterprise*'
- October 2003 - Committer for Castor JDO open source project.
- 2000-2003 software engineer for **Morgan Stanley**, listed derivatives
- 1998 – 2000 software engineer for **ClearStream**, securities clearning and settlement
- 1997 MPhil Computer Science, TU Wien
- 1995 MA Business Administration, University of Vienna

Outline

- XML Data Binding
- Castor XML
- Concepts
- Sitemap
- Samples
- Roundup
- Resources

Definitions

- XML Data Binding =
  XML data binding is the binding of XML documents to (Java) objects designed especially for the data in those documents.
- XML Data Binding frameworks
  - Castor: http://castor.exolab.org (open source)
  - JAXB: http://java.sun.com/xml/jaxb/
  - JaxMe: http://ws.apache.org/jaxme/ (open source)
- Focus on Castor XML & *CastorTransformer*
  - Castor XML (http://castor.exolab.org)
  - http://cocoon.apache.org/2.1 - Javadocs

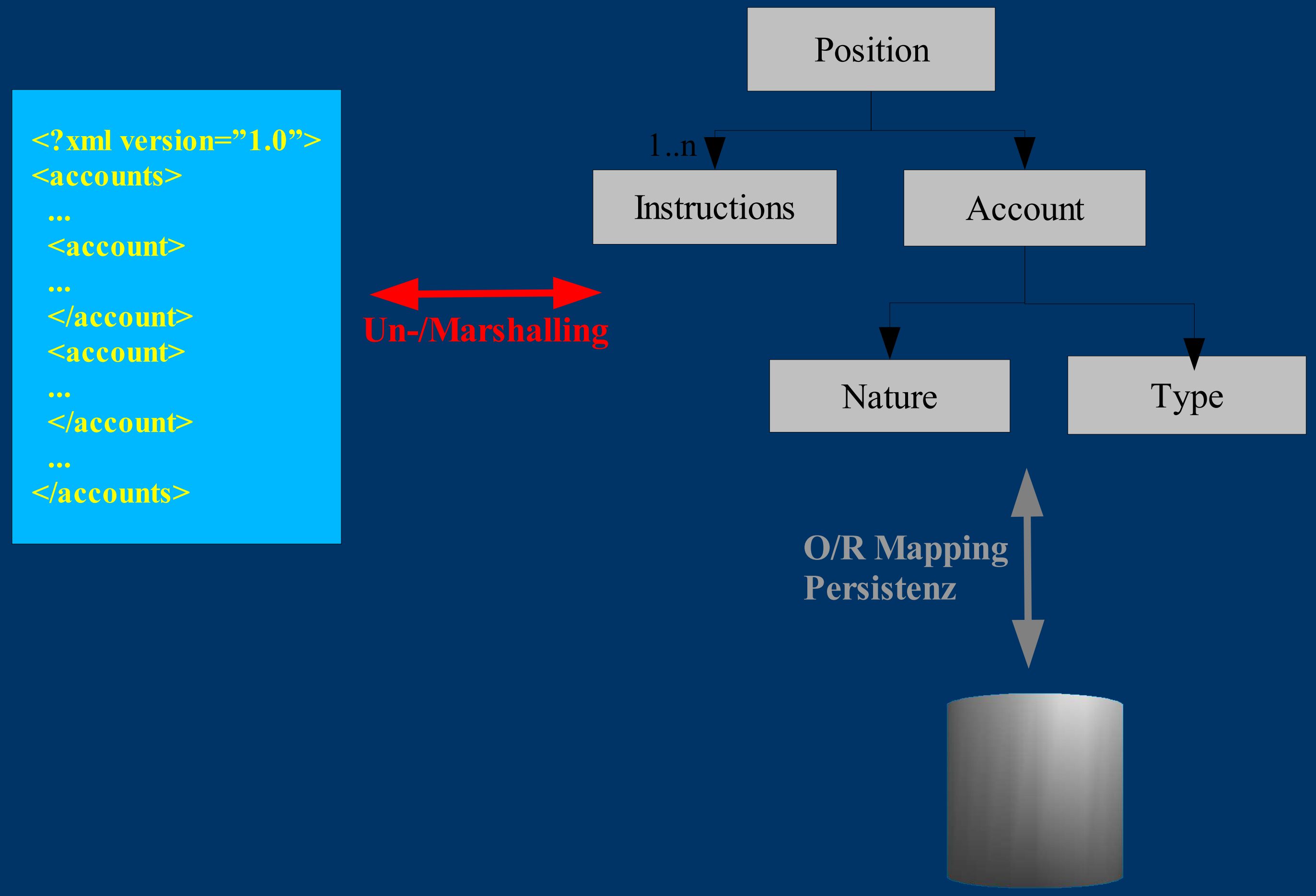
Concepts

- Castor JDO: O/R Mapping Tool
- Castor XML: XML Data Binding Tool
- Two basic approaches:
  - For an existing class hierarchy
    --> Use of a *mapping file* for XML un-/marshalling
  - Use of an XML Schema instance as starting point
    --> Automatic generation of class hierarchy and add-ons to handle XML un-/marshalling

Configuration

- CastorTransformer is a separate component in the *scratchpad* block --> configuration during build time (blocks.properties)
- Requires declaration in the *sitemap*
- Inserts an object graph that is available as Request/Session attribute into pipeline as SAX events --> requires Availability of Castor *mapping file*

Castor – Concepts (2)

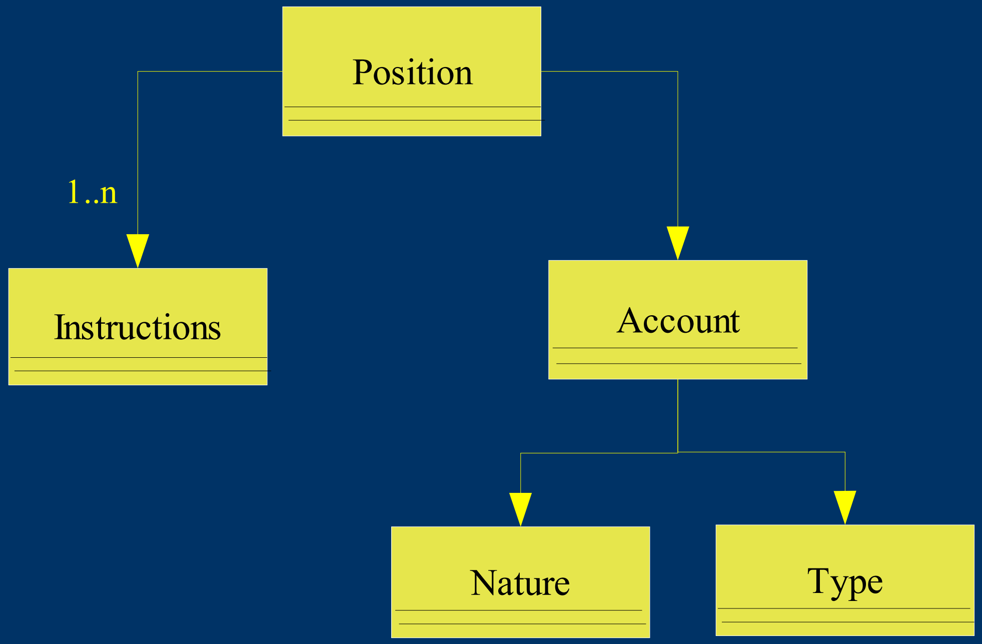


```
<?xml version=”1.0”>
<accounts>
  …
  <account>
  …
  </account>
  <account>
  …
  </account>
  …
</accounts>
```

Castor – Sitemap

```
<map:transformer name=”CastorTransformer”
          src=”org.apache.cocoon.transformation.CastorTransformer”  >
  <mapping>somePath/mapping.xml</mapping>
</map:transformer>
...
<map:pipeline>
  <map:match pattern=”transform/castor/sample”>
    <map:act type=”LoadInstructions” />
    <map:generate type=”file” />
    <map:transform type=”castor” />
      <map:parameter name="mapping-file" value="mapping.xml"/>
    </map:transform>
    <map:serialize type=”xml” />
  </map:match>
</map:pipeline>
```

Sample object hierarchy



Castor (3) – Mapping File

```
<class type=”org.npn.entity.Account”>
  <map-to xml=”account”>
  <field name=”id” type=”short”>
    <bind-to name=”id” node=”attribute” />
  </field>
  <field name=”description” type=”string”>
    <bind-to name=”description” node=”element” />
  </field>
  <field name=”nature” type=”org.npn.entity.AccountNature”/>
</class>
```

Results

```
<account id=”94782312”>
  <description>Erste Bank AG</description>
  <symbol>ESTAG</symbol>
  <account-nature>
    <id>1</id>
    <description>client</description>
  </account-nature>
  <account-type>
    <id>2</id>
    <description>trading</description>
  </account-type>
</account>
```

Amapping file (2)

```
<class type=”org.npn.cocoon.entity.Position”>
   <map-to xml=”position” />
   <field name=”id” type=”short”>
      <bind-to name=”id” node=”element” />
   </field>
   <field name=”value” type=”integer”>
      <bind-to name=”value” node=”element” />
   </field>
   <field name=”instructions” type=”org.npn.entity.Instruction”
            collection=”collection”>
      <bind-to name=”exercized-instructions” node=”element”/>
   </field>
</class>
```

Results (2)

-
  ```
  <position>
    <id>101</id>
    <amount>84</amount>
    <exercised-instructions>
      <instruction id=”457”>
        <value>24</description>
        <price>105.70</price>
      </instruction>
      <instruction id=”458”>
        <value>30</description>
        <price>102.30</price>
      </instruction>
    </exercised-instructions>
  </position>
  ```

Advanced features

- Support for 1:1 and 1..* relations
- Auto-Generation of mapping file through use of xDoclet.
- Use of *SourceGenerator* (advanced feature of Castor XML) to generate class hierarchy from *XML Schema* file -->
  replaces use of mapping file as well, as Castor generates a set of *FieldHandlers* to handle un-/ marshalling.

Roundup

- Seamless integration of XML Data Binding Framework Castor through various transformers, e.g. CastorTransformer
- Easy component setup (block)
- Easy configuration (addition to sitemap)
- Contract with Java 'object world':
  existence of object hierarchy as either a Session or Request attribute
- Marshalling controlled by *mapping file (alternatively through generated classes from XML Schema instance)*.

Resources

- http://wiki.cocoondev.org
  - Useful information on the use of XML data binding tools with Cocoon
- http://cocoon.apache.org
  - Docs, Javadocs
  - Tutorials, Samples
- http://castor.exolab.org
  - General information on Castor inclusing full samples.
- XML Data Binding
  - http://www.rpbourret.com/xml/XMLDataBinding.htm
  - JSR 031: http://java.sun.com/xml/jaxb

Questions ?

?