Cocoon & Persistence

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

- Introduction
- ESQL Logicsheet
- SQL Transformer
- O/R Mapping Tools (Object/Relational Bridges)
  - Castor

Definitions

- Persistence (for the purpose of this presentation)= Query/Insertion/Update of data stored in a RDBMS.
- RDBMS = relational database system, e.g. DB2, Oracle, mySQL, etc.
- Does NOT cover XML Datenbanken (--> presentation on Tamino) and non-relational DBMS
  NB: basically all database systems for which there is a compliant JDBC driver.

Configuration

- RDBMS access internally realised through Avalon components (Excalibur DataSource)
- Builds upon standard J2EE technology JDBC
- Avalon components require configuration
  - Register JDBC driver
  - Configure DataSources (Avalon Excalibur)
    - Configure internal JDBC DataSource (e.g. Cocoon CLI)
    - Configure external JDBC DataSource (e.g. *web container)*

Datasource configuration

- <datasources>

```
<jdbc name="excaliburConnection">
  <pool-controller min="5" max="10"/>
  <dburl>dbc:mysql://localhost:3306/castor</dburl>
  <user>me</user>
  <password>*********</password>
</jdbc>

<j2ee name=”externalConnection">
  <dbname>externalDataSource</dbname>
</j2ee>

</datasources>
```

Cocoon & Persistent

ESQL

ESQL Logicsheet

- One of many ***logicsheets*** for XSP Pages
- Specification of a database query through use of predefined XML tags, e.g. <esql:connection>, <esql:query>, <esql:results>, etc.
- Inserts result of database query into pipeline as a stream of SAX events.

ESQL – Sitemap

```
<xmap:pipelines>
  ...
  <xmap:pipeline>
    <xmap:match pattern=”esql/sample”>
      <xmap:generate type=”serverpages” />
      <xmap:serialize type=”xml” />
    </xmap:match>
  </xmap:pipeline>
  ...
</xmap:pipelines>
```

ESQL - Structure

```
<xsp:page xmlns:xsp="http://apache.org/xsp"
          xmlns:esql="http://apache.org/cocoon/SQL/v2">
  <page>
    <esql:connection>
      <esql:pool />
      <esql:execute-query />
        <esql:query />
        <esql:results />
        <!-- <esql:no-results /> -->
        <!-- </esql:error /> -->
      </esql:execute-query />
    </esql:connection>
  </page>
</xsp:page>
```

ESQL Logicsheet

- ESQL Logicsheet is a *thin wrapper* for JDBC
- Supports in principle the complete functionality of what's possible when coding JDBC in Java.
- Has abstractions for extended functions:
  - Grouping
  - Nested queries
  - Prepared Statemenst/Stored Procedures

ESQL - Sample

```
...
<esql:connection>
  <esql:pool=”mySQL-dataSource”>
  <esql:execute-query>
    <esql:query>select * from ACCT</esql:query>
    <esql:results>
      <esql:row-results>
        <esql:get-columns />
      </esql:row-results>
    </esql:results>
  </esql:execute-query>
</esql:connection>
...
```

ESQL – Results 1

<id>94782312</id>
<desc>Erste Bank AG</desc>
<symb>ESTAG</symb>

<id>94782314</id>
<desc>Credit Suisse First Boston</desc>
<symb>CSFB</symb>

ESQL – ResultSet

- <esql:row-results> as abstraction for JDBC ResultSet
- Facilitates the insertion into an existing document structure.

```
<accounts>
  <esql:row-results>
    <account>
      <esql:get-columns />
    </account>
  </esql:row-results>
</accounts>
```

ESQL – Results 2

```
<accounts>
  <account>
    <id>94782312</id>
    <desc>Erste Bank AG</desc>
    <symb>ESTAG</symb>
  </account>
  <account>
    <id>94782314</id>
    <desc>Credit Suisse First Boston</desc>
    <symb>CSFB</symb>
  </account>
</accounts>
```

ESQL – Output co ntrol

- <esql:get-columns/> uses table column names for to name the XML elemenst
- If not desired, or if only a subset of the complete set of table columns is required, use <esql:get-XXX>.

```
<account>
  <id><esql:get-int column=”id” /></id>
  <description></esql:get-string column=”desc” /></description>
</account>
```

ESQL – Results 3

```
<accounts>
  <account>
    <id>94782312</id>
    <description>Erste Bank AG</description>
  </account>
  <account>
    <id>94782314</id>
    <description>Credit Suisse First Boston</description>
  </account>
</accounts>
```

SQL Transformer

SQL Transformer

- **Transformer** (NOT a generator; no compilation required; pipeline caching in principle possible)
- Specification of a database query through use of predefined XML tags, e.g. <esql:execute-query>, <esql:query>, etc.
- Inserts result of database query into pipeline as a stream of SAX events.
- Syntax **different to** ESQL Logicsheet

SQLT (2) – Sitemap

```
<map:pipeline>
  <map:match pattern=”transform/sql/sample”>
    <xmap:generate type=”file” />
    <xmap:transform type=”sql” />
      <map:parameter name=”use-connection”
                     value=”mySQL-dataSource” />
      <map:parameter name="show-nr-of-rows" value="true"/>
    </xmap:transform>
    <xmap:serialize type=”xml” />
  </xmap:match>
</xmap:pipeline>
```

SQLT - Sample

```
<page xmlns:sql=”http://apache.org/cocoon/SQL/2.0”>
  <sql:execute-query>
    <sql:query name=”accounts”>
      select * from ACCT
    </sql:query>
  </sql:execute-query>
</page>
```

SQLT – Results

```
<rowset name=”accounts”>
  <row>
    <id>94782312</id>
    <desc>Erste Bank AG</desc>
    <symb>ESTAG</symb>
  </row>
  <row>
    <id>94782314</id>
    <desc>Credit Suisse First Boston</desc>
    <symb>CSFB</symb>
  </row>
</rowset>
```

ESQL/SQLT - Shortcomings

ESQL – Complex sample

```
...
<esql:execute-query>
  <esql:query>
    SELECT p.id, p.desc, p.acct, a.desc, t.desc, n.desc, ...
    FROM posn p, instr i, acct a, acct o
                acct_natr n, acct_type t
    WHERE p.id = i.id AND
                p.acct = a.id AND
                p.dest = o.id AND
                a.natr_id = n.id AND
                a.type_id = t.id
  </esql:query>
  <esql:results> ...</esql:results>
</esql:execute-query>
...
```

ESQL & SQLT - Disadvantages

- Difficult/impossible maintainnance:
  SQL statements embedded in XML code
- Complex *development cycle*, as failure analysis happens during pipiline execution

--> Use of O/R mapping tools

Cocoon & Persistenz

O/R Mapping

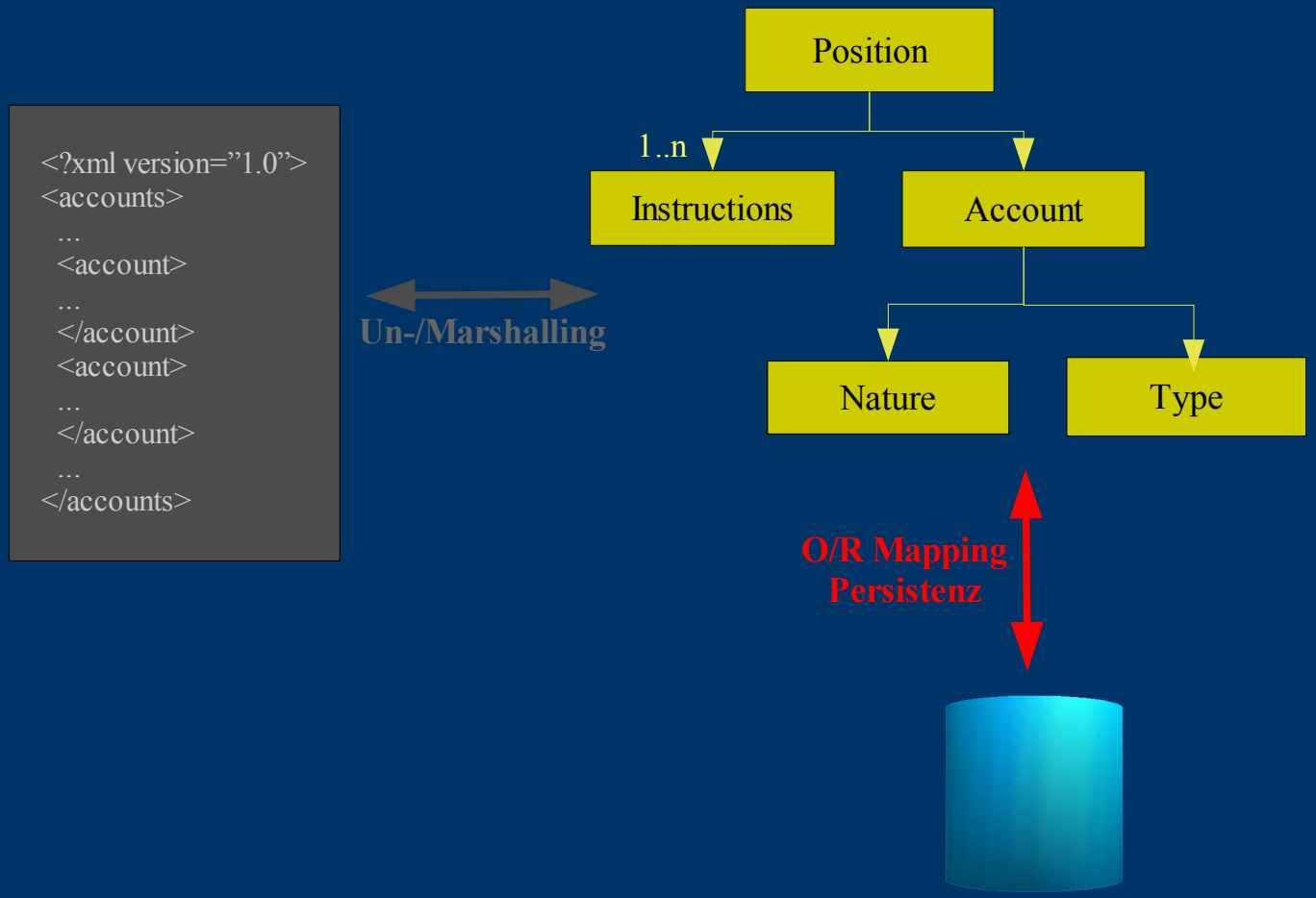
O/R Mapping - Overview

- Tool for transparent, non-intrusive persistence of Java objects in RDBMS
- Seamless integration with *business logic* in form of (*Enterprise) JavaBeans possible --> ideal replacement to* container managed persistence (CMP) in form of BMP + JDO
- Automatic SQL generation
- Existing open source solutions
  - OJB (http://db.apache.org/ojb) **--> presentation**
  - Castor (http://castor.exolab.org)
  - Hibernate (http://hibernate.sf.net)

Castor (1) - JDO & XML

- Castor - O/R mapping & XML data binding tool
- Various usage scenarios:
  - Existing data stored in RDBMS
    --> use *mapping file* both for persistence and XML un-/marshalling; create class hierarchy manually or tool-based
  - Existing class *hierarchy*
    --> use *mapping file* both for persistence and XML un-/marshalling; create Db schemas manually or tool-based
  - Use XML Schema instance
    --> automatic generation of class hierarchy and accompanying functionality for XML un-/marshalling

Castor - Concepts



Castor – Mapping file

```
<class type="org.npn.entity.Account"
        id="id">
  <map-to table="ACCT" />
  <field name="id" type="short">
    <sql name="idtr" />
  </field>
  <field name="description" type="string">
    <sql name="desc" type="varchar" />
  </field>
  <field name="nature" type="org.npn.entity.AccountNature">
    <sql key="natr_id" />
  </field>
</class>
```

Castor - Integration

- Use/creation of a *mapping file*
- Implement a code fragment that uses Castor JDO to deal with persistence, e.g. load all Instruction instances of a specific user.
  NB: Castor always loads a complete object graph, with the exception of objects where *lazy loading* is defined.
- Integrate with Cocoon by implementing this functionality as part of a Cocoon action.

Castor - code fragment

- List instructions = new ArrayList();
  ...
  Database db = JDO.getDatabase();
  db.begin(); // transaction demarcation
  OQLQuery query =
  db.createQuery ("select * from Instruction i where i.owner = $1");
  query.bind (1928456);
  OQLResult result = query.execute (Database.READ_ONLY);
  while (result.hasMore()) {
  instructions.add ((Instruction) result.next());
  }
  db.commit(); // transaction demarcation

  request.setAttribute ("instruction", instructions);

Castor – Sitemap

```
<map:pipeline>
  <map:match pattern=”transform/sql/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>
```

Castor – Advances features

- ***Supports use of performance caches***
- **Lazy loading** – When dealing with complex object graphs, specified parts of this graph will only be loaded ***on-demand*, i.e. when accessed.**
- Support for 1:1, 1:n und **m:n** relationships

Castor – Advances features (2)

```
<class type=”org.npn.cocoon.entity.Position” identity=”id”>
  <map-to table=”POSN” />
  <cache-type type="count-limited" capacity="50"/>
  <field name=”id” type=”short”>
    <sql name=”idtr” />
  </field>
  <field name=”instructions” type=”org.npn.entity.Instruction”
         collection=”arrayList”>
    <sql many-key=”pos_id” />
  </field>
  <field name=”beneficiaries” type=”org.npn.entity.Accounts”
         collection=”vector” lazy=”true”>
    <sql many-key=”pos_id” />
  </field>
</class>
```

Castor - Gotchas

- Limitations when o/r mapping tools used in parallel with other approaches to access the same data. In general, one is facing the same problems as with other types of applications, e.g. concurrent data updates.
- Caching of limited use.
  - In the case of external **updates**, don't use it.
  - In the case of multiple Castor instances, no snynchronisation available right now
    NB: Implementation of distributed caches in scope for future releases of Castor, OJB. --> with this it will be possible to use e.g. Castor in application server clusters.

Roundup

- Different solutions for different problems (*web publishing* vs. *web applications*)
- “Keep it simple” --> O/R mapping tools are o*verkill* for simple web publishing solutions.
- Use O/R mapping tools for complex *web applications,* especially if a data model exists already and performance tuning is vital to your application.

Cocoon & Persistenz

- http://wiki.cocoondev.org
  - plenty of informationen on *database connectivity* incl. *the integration* of various o/r mapping tools, e.g. Hibernate and OJB with Cocoon
- http://cocoon.apache.org
  - docs, Javadocs
  - tutorial, samples
- http://castor.exolab.org
  - General information on Castor incl. samples.

Questions ?