http://speedo.objectweb.org

speedo@objectweb.org

Yoann Bersihand, France Telecom
Research & Development

ObjectWeb
Open Source Middleware

Plan

- Speedo JDO
- Tutorial Environment
- Basics (steps 1-2)
- Mapping (step 3)
- Additional (steps 4-7)
- Properties (step 8)
- Speedo & J2EE (step 9)

Speedo JDO

http://java.sun.com/products/jdo/
http://speedo.objectweb.org

JDO, what is it?

- Java Data Objects
- Many SQL dialects VS JDOQL
- Features:
  - Automatic cache management
  - Query abilities
  - Transactional support
  - Distributed and heterogeneous data stores
  - Integration with EJB

Speedo

- Implementation of Sun's JDO specification
- Part of the ObjectWeb open source community
- Built on top of
  - JORM: mapping objects onto a persistent support
  - MEDOR: query framework
  - Perseus: persistence framework
- Main sponsor: France Telecom

Tutorial Environment

ObjectWeb
Open Source Middleware

Setup

- Ant 1.6.2
- Eclipse 3.0.1
- Java 1.4.2.06
- Speedo
  - ant dist
  - ant to create the speedo.jar
- Database and db manager with HSQLDB
  - *java –cp lib/speedo.jar org.hsqldb.util.DatabaseManager*
  - *java –cp lib/speedo.jar org.hsqldb.Server –database.0 mydb –dbname.0 xdb*

TutorialHelper

- Class to provide a PersistenceManagerFactory
- Main element of the JDO architecture
- PMF creates PersistenceManager
- Configured from the speedo.properties file

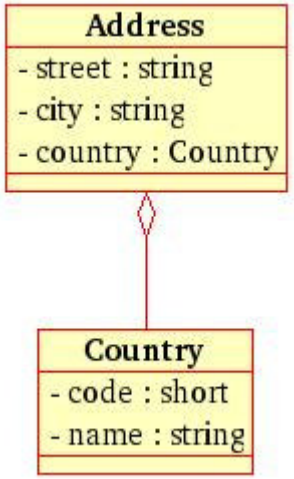
- Organisation
  - pobjects directory
    - Java classes + Jdo files
  - appli directory
    - Examples to run appli directory

Basics
Steps 1-2

ObjectWeb
Open Source Middleware

Step 1-2: object model

- At least one jdo file to describe the mapping
- Object model:



Step 1 : make objects persistent

- Create java objects
- Get a PersistenceManager(PM) from the PMF
- Begin a transaction
- makePersistent(Object o)
- Commit the transaction
- Close the PM

```
PersistenceManager pm = pmf.getPersistenceManager();
pm.currentTransaction().begin();
pm.makePersistent(address1);
pm.currentTransaction().commit();
pm.close();
```

Step 1: run it

- Run the example
  - ant step1
- Results
  - T_BASICS_COUNTRY
  - T_BASICS_ADDRESS

COUNTRY	CITY	STREET
fr	Clermont	impasse St Jacques
fr	Paris	rue Laffiteau
fr	Avignon	rue de Mons

CODE	NAME
fr	France

Step 2 : manage persistent objects (1/2)

- makePersistent and keep a reference via getObjectId(o)

```
pm.currentTransaction().begin();
pm.makePersistent(address);
Object id = pm.getObjectId(address);
pm.currentTransaction().commit();
return id;
```

- Update the persistent object

```
pm.currentTransaction().begin();
Address address = (Address) pm.getObjectById(id, true);
address.setCity("New York");
pm.currentTransaction().commit();
```

Step 2 : manage persistent objects (2/2)

- Delete it

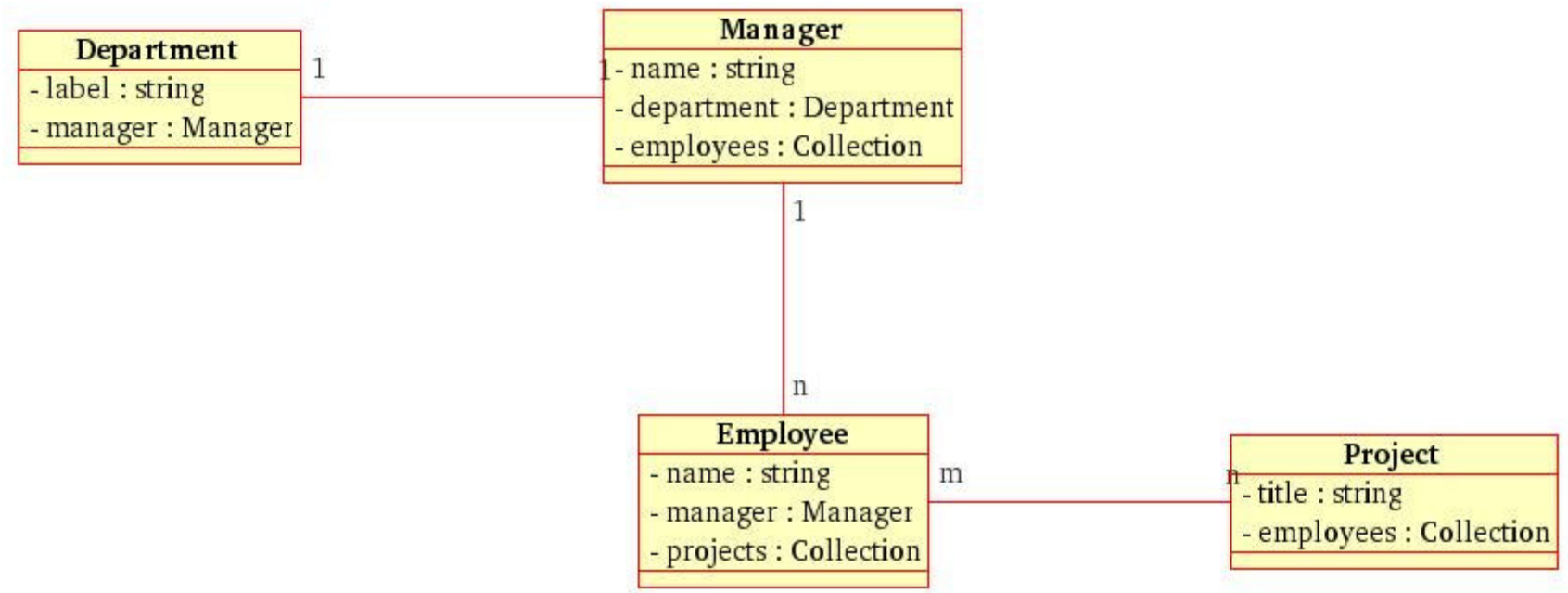
```
pm.currentTransaction().begin();
Address address = (Address) pm.getObjectById(id, true);
pm.deletePersistent(address);
pm.currentTransaction().commit();
```

- Run the example
  - ant step2
- Results
  - T_BASICS_COUNTRY
  - T_BASICS_ADDRESS

Mapping
Step 3

ObjectWeb
Open Source Middleware

Step 3: object model



The jdo file (1/4)

- Manager Class

```
<class name="Manager" identity-type="application">
        <extension vendor-name="speedo" key="sql-name"
        value="T_MAPPING_MANAGER"/>
        <field name="name" primary-key="true">
                <extension vendor-name="speedo" key="sql-name" value="NAME"/>
        </field>
        <field name="department" persistence-modifier="persistent">
                <!-- 1-1 relationship -->
                <extension vendor-name="speedo" key="target-foreign-keys"
                value="LABEL=DPT_LABEL"/>
                <extension vendor-name="speedo" key="reverse-field"
                value="manager"/>
        </field>
        <field name="employees" persistence-modifier="persistent">
                <!-- 1-N relationship -->
                <collection element-type="Employee"/>
                <extension vendor-name="speedo" key="source-foreign-keys"
                value="NAME=MANAGER_NAME"/>
                <extension vendor-name="speedo" key="reverse-field"
                value="manager"/>
        </field>
</class>
```

The jdo file (2/4)

- Department Class

```
<class name="Department" identity-type="application">
        <extension vendor-name="speedo" key="sql-name"
        value="T_MAPPING_DEPARTMENT"/>
        <field name="label" primary-key="true">
                <extension vendor-name="speedo" key="sql-name" value="LABEL"/>
        </field>
        <field name="manager" persistence-modifier="persistent">
                <!-- 1-1 relationship already defined above -->
        </field>
</class>
```

The jdo file (3/4)

- Employee Class

```
<class name="Employee" identity-type="application">
        <extension vendor-name="speedo" key="sql-name"
        value="T_MAPPING_EMPLOYEE"/>
        <field name="name" primary-key="true">
                <extension vendor-name="speedo" key="sql-name" value="NAME"/>
        </field>
        <field name="manager" persistence-modifier="persistent">
                <!-- 1-N relationship already defined above -->
        </field>
        <field name="projects" persistence-modifier="persistent">
                <!-- M-N relationship -->
                <collection element-type="Project"/>
                <extension vendor-name="speedo" key="join-table"
                value="EMPLOYEE_TO_PROJECT"/>
                <extension vendor-name="speedo" key="source-foreign-keys"
                value="NAME=EMPLOYEE_NAME"/>
                <extension vendor-name="speedo" key="target-foreign-keys"
                value="TITLE=PROJECT_TITLE"/>
                <extension vendor-name="speedo" key="reverse-field"
                value="employees"/>
        </field>
</class>
```

The jdo file (4/4)

- Project Class

```
<class name="Project" identity-type="application">
        <extension vendor-name="speedo" key="sql-name"
        value="T_MAPPING_PROJECT"/>
        <field name="title" primary-key="true">
                <extension vendor-name="speedo" key="sql-name" value="TITLE"/>
        </field>
        <field name="employees" persistence-modifier="persistent">
                <!-- M-N relationship already defined above -->
                <collection element-type="Employee"/>
        </field>
</class>
```

Step 3: mapping

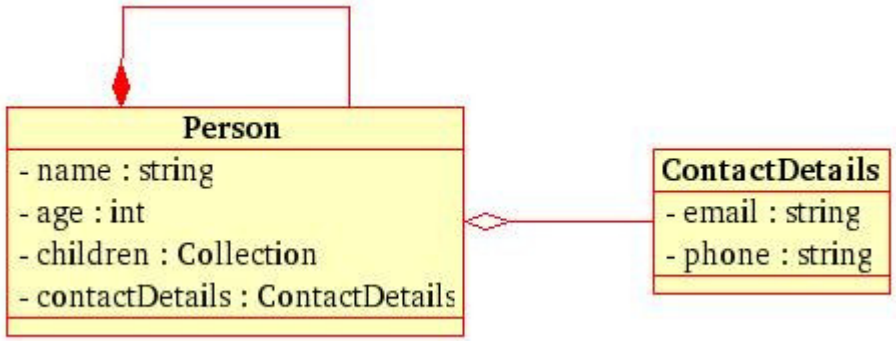
- Run the example
  - ant step3
- Results
  - T_MAPPING_MANAGER
  - T_MAPPING_DEPARTMENT
  - T_MAPPING_EMPLOYEE
  - T_MAPPING_PROJECT
  - EMPLOYEE_TO_PROJECT

Additional Steps 4-7

ObjectWeb
Open Source Middleware

Step 4: object model

- Object model



- makePersistentAll(Collection c)

```
pm.currentTransaction().begin();
pm.makePersistentAll(persons); //where persons is a Collection
pm.currentTransaction().commit();
```

Step 4: JDO queries

- To execute a JDO query:

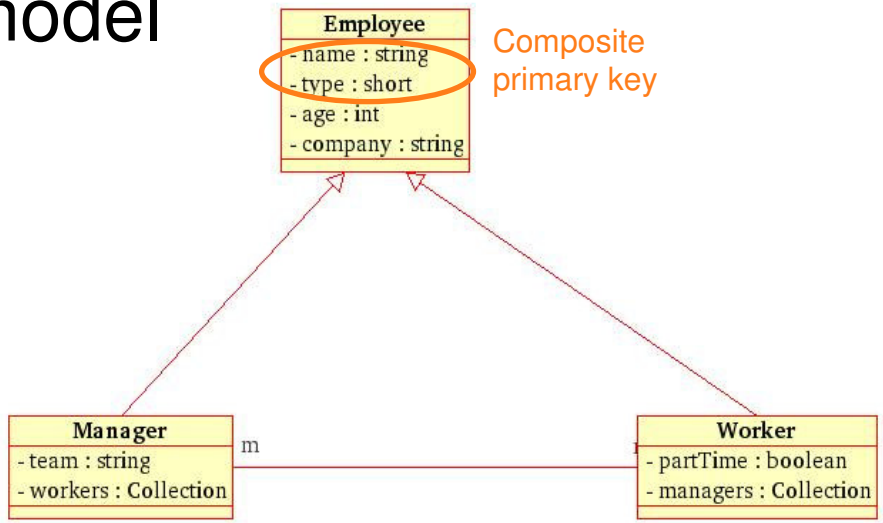
```
//create a new instance of the Query class
Query query = pm.newQuery(Class c);
//declare the parameters you will use
query.declareParameters("String myName");
//declare the filter of the query (your selection criteria)
query.setFilter("(name == myName) || (age > 45)");
//execute the query and get the result set as a java Collection
Collection results = (Collection)query.execute("Lab Loic");
//use of the result set
…
//close the query
query.closeAll();
```

Step 4: queries

- Features
  - Extent (iterate over all the instances of a class)
  - Basic, ordering, parameter passing, composite filter
  - Navigation through fields to define query filters:
    - Single field (contactDetails.phone == xxxxx)
    - Multivalued field (children.contains(child) & child.age<5)
- Run the example
  - ant step4
- Results
  - T_ADDITIONAL_PERSON

Step 5: object model

- Object model



- Composite PK
  - Filtered mapping: filter can be part of the PK or not
  - EmployeeId

Step 5: inheritance mapping

- Super class

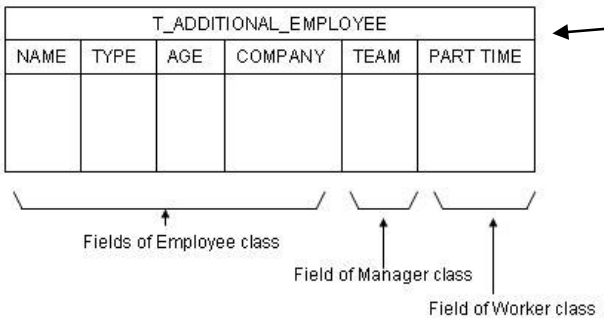
```
<class name="Employee" identity-type="application" objectid-
class="org.objectweb.speedo.tutorial.pobjects.additional.inheritance.EmployeeId">
        <extension vendor-name="speedo" key="sql-name"
        value="T_EMPLOYEE"/>
        <!-- the inheritance filter : the "type" field-->
        <extension vendor-name="speedo" key="inheritance-filter" value="type"/>
        <!-- the value of the filter for the Employee class: "employee" -->
        <extension vendor-name="speedo" key="inheritance-key"     value="employee"/>
        …
</class>
```

- Inherited class

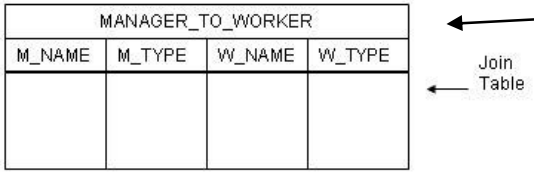
```
<!-- Manager extends Employee-->
<class name="Manager" persistence-capable-superclass="Employee">
        <!-- the type of inheritance: "filtered"-->
        <extension vendor-name="speedo" key="inheritance-mapping"
        value="filtered"/>
        <!-- the value of the filter for the Manager class: "manager" -->
        <extension vendor-name="speedo" key="inheritance-key"     value="manager"/>
        …
</class>
```

Step 5: db representation

- Tables in the database



In this table, all the Employee, Manager & Worker instances will be stored.



Join table with 2 composite FK

Step 5: run it

- Run the example
  - ant step5
- Results
  - T_ADDITIONAL_EMPLOYEE

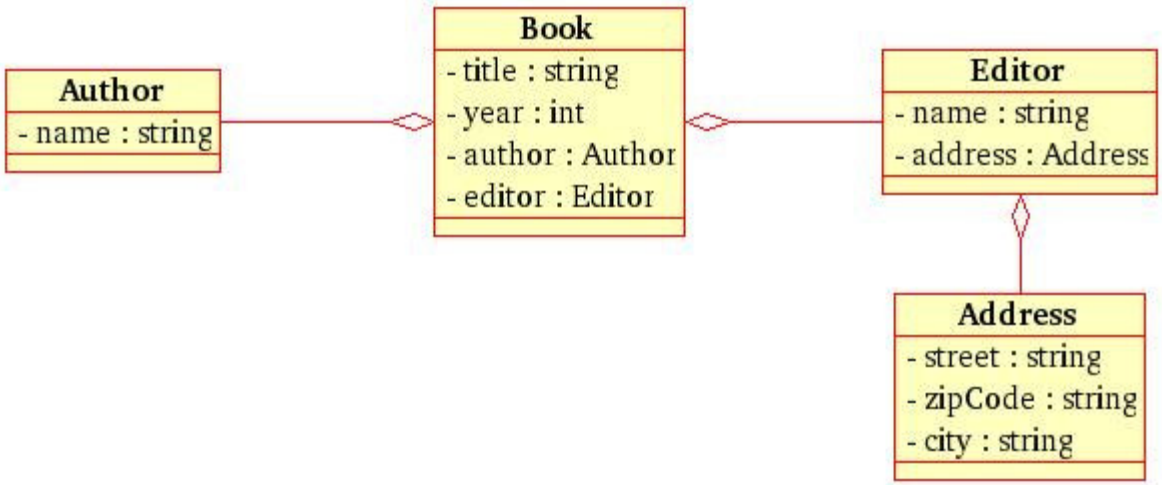
TYPE	AGE	NAME	COMPANY	PART_TIME	TEAM
employee	42	Herve Landry	Tetra Pack	null)	(null)
employee	30	Vincent Racado	Tetra Pack	(null)	(null)
manager	49	Eric Mento	Tetra Pack	(null)	Marketing
manager	52	Jean Duverge	Tetra Pack	(null)	Sales
worker	33	Caroline Bret	Tetra Pack	true	(null)
worker	54	Evelyne Jain	Tetra Pack	false	(null)
worker	28	Tim Jorge	Tetra Pack	false	(null)

Step 6-7: object model (1/2)

- Common object model for
  - Step 6: attach/detach
    - Obtain detached copies of persistent instances
    - Modify them
    - Apply changes to the PM
    - Commit the changes
  - Step 7: fetch plan
    - Define particular loaded states for an object graph
    - Used when detaching, retrieving, refreshing objects
    - 4 predefined fetch groups: default, values, none, all

Step 6-7: object model (2/2)

- Object model



Step 6-7: jdo file

- Classes to be detached must be "detachable"

```
<class name="Author" identity-type="application" detachable="true">
```

- A fetch group is defined at the class level
  - It has a name
  - It defines fields to be loaded
  - It can embed already defined fetch groups

```
<class name="Book" identity-type="application" detachable="true">
        …
        <fetch-group name="editorName">
                <fetch-group name="default"/>
                <field name="editor.name"/>
        </fetch-group>
</class>
```

Step 6: attach/detach

- Make a book persistent
- Detach the book

```
Book copyOfBook = (Book) pm.detachCopy(book);
```

- Modify the book

```
copyOfBook.setYear(2012);
copyOfBook.getEditor().getAddress().setCity("New York");
```

- Attach the book

```
pm.currentTransaction().begin();
Book attachedBook = (Book) pm.attachCopy(copyOfBook,false);
pm.currentTransaction().commit();
```

Step 6: run it

- Run the example
  - ant step6
- Results
  - T_ADDITIONAL_BOOK

YEAR	TITLE	EDITOR_NAME	AUTHOR_NAME
2012	The willow tree	Folio	Hubert Selby Jr
1939	Tropic of Capricorn	Grasset	Henry Miller

Step 7: fetch plan

- Define fetch groups in the jdo file (if needed)
- Get the PM fetch plan (instanciated on the fly)

```
FetchPlan fp = pm.getFetchPlan(); //the default fetch group is set
```

- Add the group(s) you need at the runtime

```
fp.addGroup("myFetchGroup").removeGroup("default");
```

- Detach (retrieve, refresh) persistent object

```
Book myBook = (Book) pm.detachCopy(book);
```

  - Only the fields defined in the active fetch group(s) are loaded

Step 7: run it

- Run the example
  - ant step7
- Results
  - [java] With the default fetchgroup:
    [java] Title can be accessed: The Yage Letters
    [java] Year can be accessed: 1955
    [java] Correct exception caught: Field author cannot be accessed: not loaded when the object has been detached
  - [java] With the all fetchgroup:
    [java] Title can be accessed: The Yage Letters
    [java] Year can be accessed: 1955
    [java] Author can be accessed: William S. Burroughs
    [java] Editor can be accessed: Mille et Une Nuits, address=[Fenton Street, 931ZR2, Leeds]

JDO Driver Properties Editor
Step 8

ObjectWeb
Open Source Middleware

Eclipse plugin (1/2)

- Plugin to edit the speedo.properties file
- Speedo tuning:
  - Pool of connection
  - Cache management (L2 cache)
  - Prefetch of queries
  - Cache of compiled queries
  - Pool of PersistenceManager

Eclipse plugin (2/2)

- Install the plugin
  - Unzip org.objectweb.eclipsejdo.dpe_1.0.0.zip in your $ECLIPSE_HOME/plugin directory
  - Relaunch your Eclipse
  - Click right + 'Open With' -> JDO Driver Properties Editor
- Edit the properties description in your own language

Speedo & J2EE
Step 9

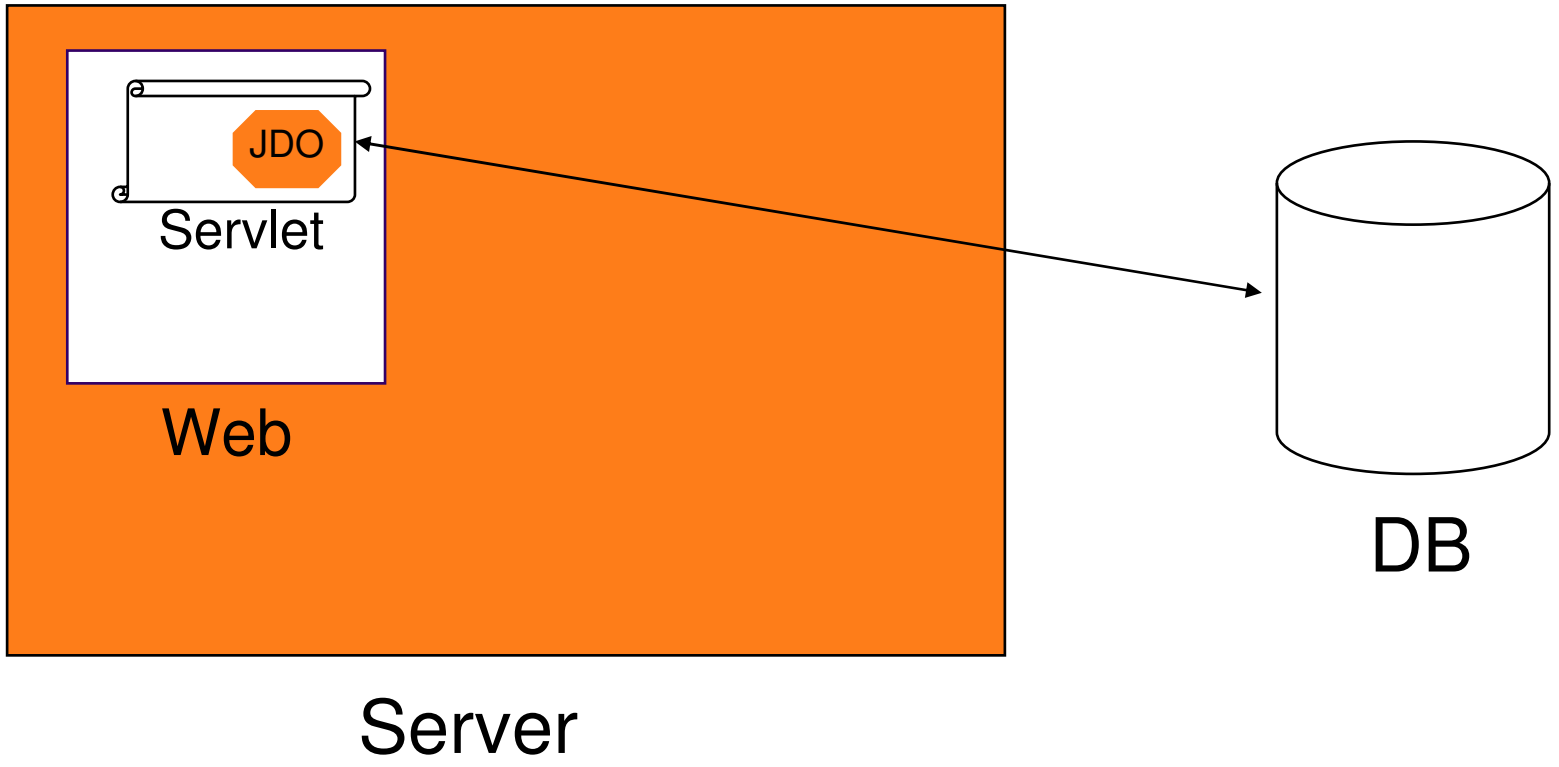
ObjectWeb
Open Source Middleware

Speedo & J2EE servers

- Speedo is fully integrated in JOnAs and WebLogic
- Integration is being achieved in WebSphere
- JDO can be integrated into:
  - A servlet
  - A session bean or a message driven bean (MDB)

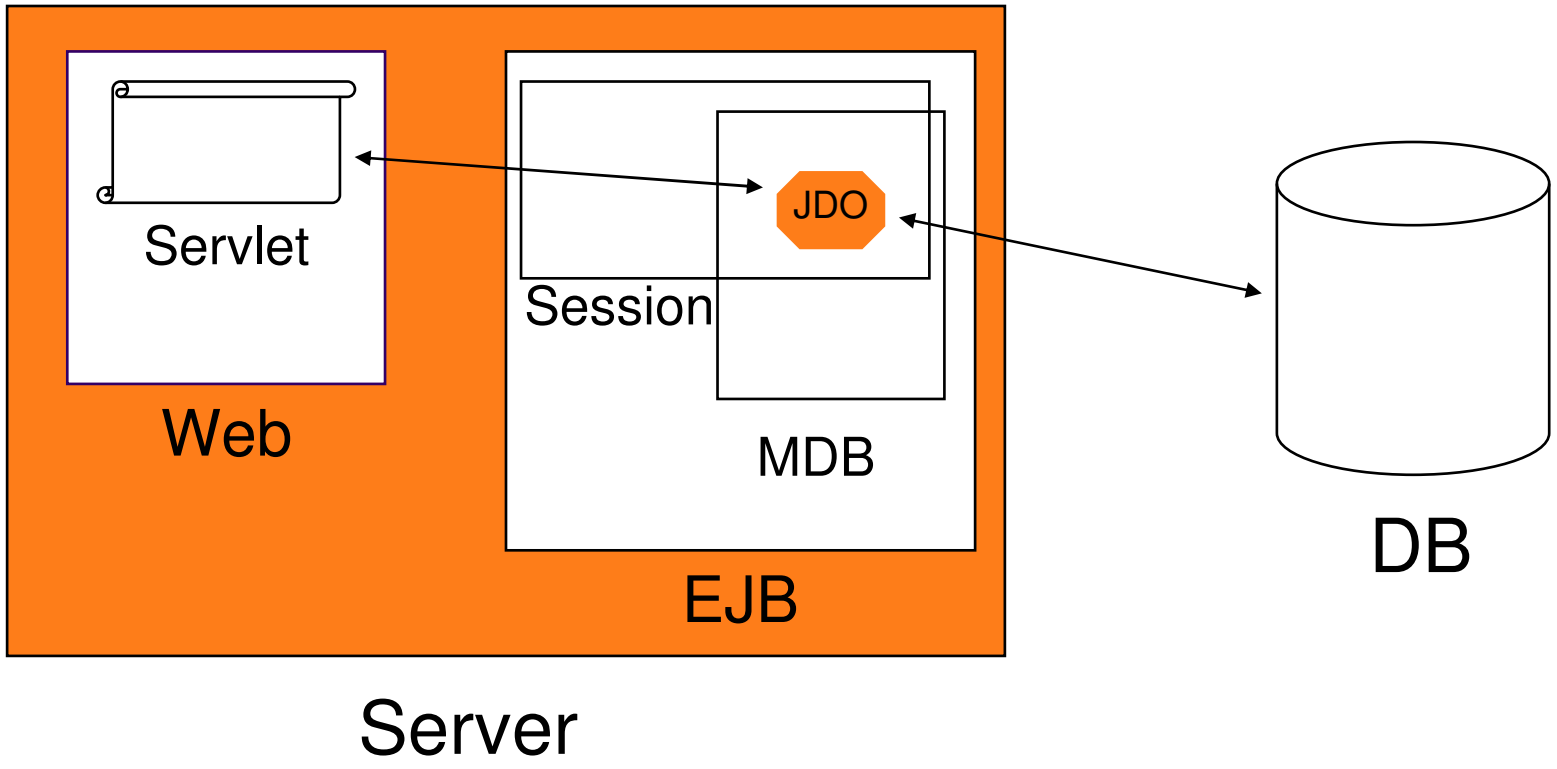
JDO integration (1/2)

- JDO in a servlet



JDO integration (2/2)

- JDO in a session bean/Message Driven Bean



How to get the PMF?

- In a standalone application, via the speedo.properties file

```
Properties p = new Properties();
p.load(new FileInputStream(propertiesFileName));
PersistenceManagerFactory pmf =
      JDOHelper.getPersistenceManagerFactory(p);
```

- In a J2EE application,via the JNDI name of the PMF
  - In the ejb-jar xml descriptor
  - In the server dedicated ejb-jar xml descriptor

```
PersistenceManagerFactory pmf = (PersistenceManagerFactory)
new InitialContext().lookup(PMF_CTX_NAME);
```

Transaction Demarcation (1/2)

- In a standalone application, the user has to write the transaction demarcation

```
public void makeBookPersistent(Book book){
    PersistenceManager pm = pmf.getPersistenceManager();
    pm.currentTransaction().begin();
    pm.makePersistent(book);
    pm.currentTransaction().commit();
}
```

Transaction Demarcation (2/2)

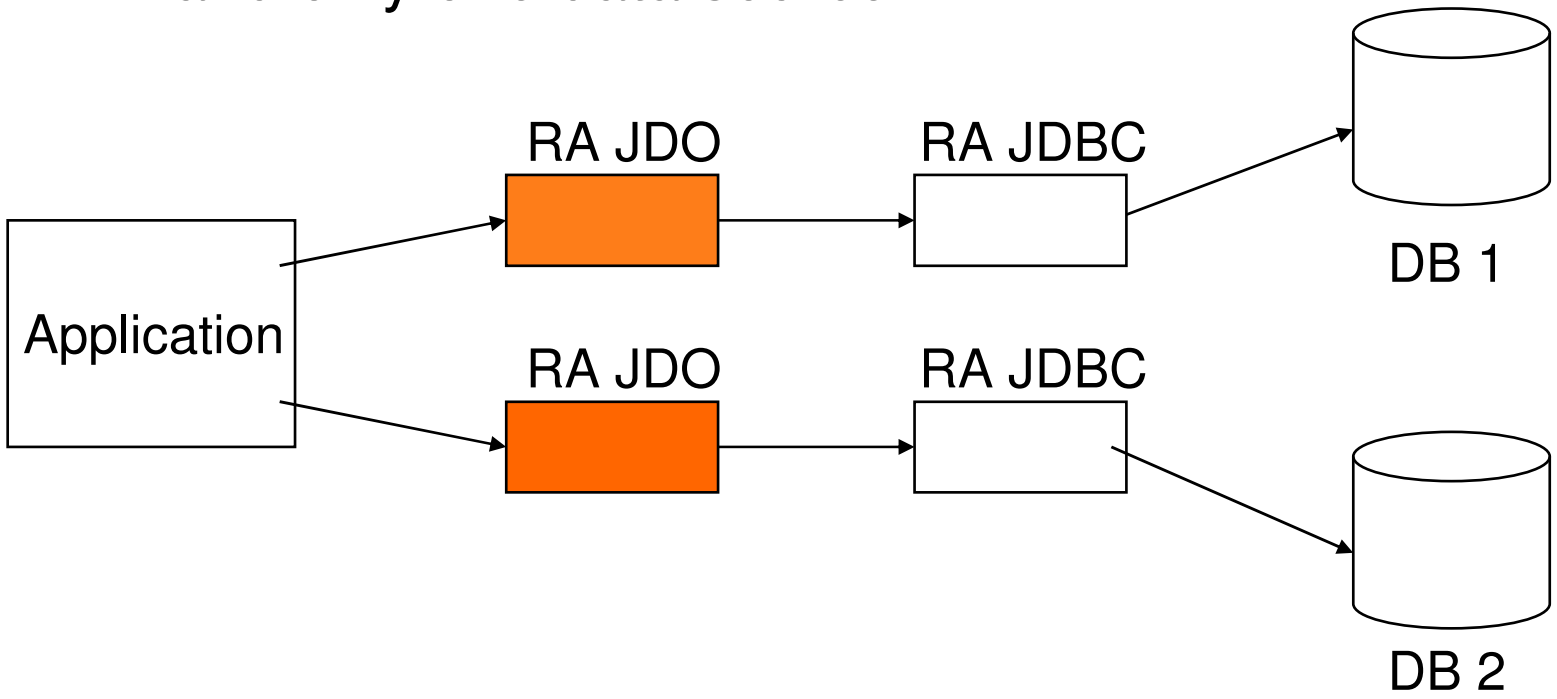
- In a J2EE application, transactions are demarcated by the EJB container
  - Transaction Manager registered into JNDI
  - Required attribute for methods in the xml descriptor

```
public void makeBookPersistent(Book book){
    PersistenceManager pm = pmf.getPersistenceManager();
    pm.makePersistent(book);
}
```

- Always possible for the user to perform transaction demarcation by himself

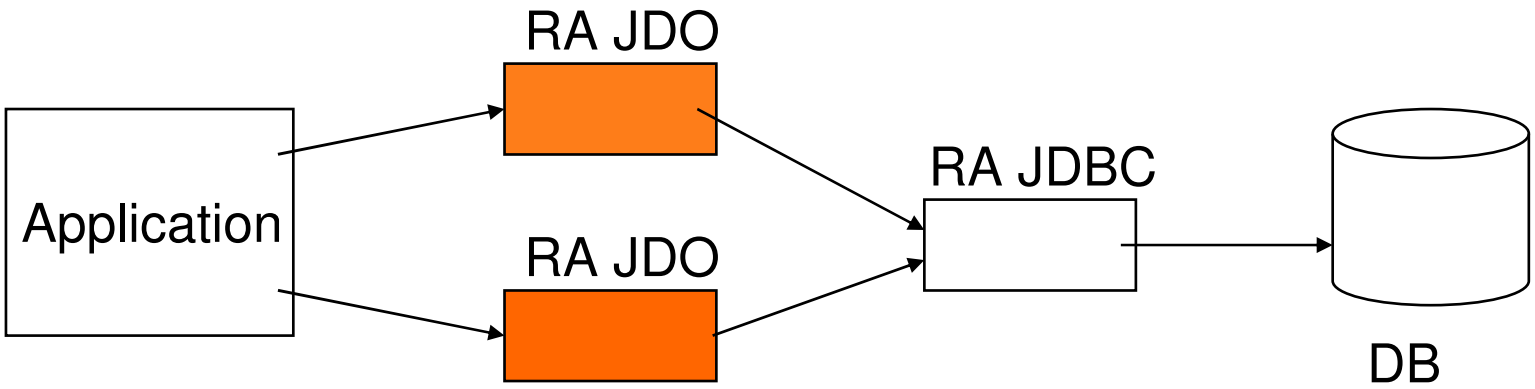
Multiple data sources (1/2)

- Files speedo_ra.rar and jdbc.rar are not grouped
- One speedo_ra.rar per data source
  - An instance of a PMF is registered into JNDI for one and only one data source



Multiple data sources (2/2)

- Another possibility:
  - Many speedo instances provisioning and querying the same data source



Any question?

Email: yoann.bersihand@francetelecom.com

ObjectWeb
Open Source Middleware