Intro, Persistence, Object Relational Mapping and JPA

JPA Slides #1

Persistence

Persistent object: Object that outlives the execution of the program

- Have to store for later retrieval (next execution)

Many persistence mechanisms

- Flat files
- Serialization
- XML
- Different types of databases …
- …we will use a <u>relational database</u> (de facto standard)

The OO-Relational Mismatch

Relational databases and object orientation doesn't fit!

Object orientation: Objects

Relational databases: **Sets of tuples**
- No objects, classes
- No inheritance, polymorphism, generics...

Major clash, the OO-relational mismatch
- Relational databases won't change, mathematical foundations...
- Unsolved problem, ...

Handling the Mismatch, Option 1

Surrender : I.e. don't use OO

Possible solution (good for massive reads)

- Example: Product Catalog to web
- Just use primitive types, String, int, ...
- Fastest possible solution
- Not a solution for complex cases

Handling the Mismatch, Option 2

Try to fix the mismatch

- Map between objects and tuples, **object relational mapping**, **ORM**

No general best strategy

- Must know how database in going to be used
- Mostly reads? Mostly writes?
- Different strategies
- Very complex task to implement (we don't)
- We use some **middleware** (glue layer)

ORM Cases to Handle

Associations? Multiplicity! Inheritance? Generics?

Object graphs! Lazy fetching? Lazy object creation? Caching? Concurrency? Transactions?...

Ad hoc searching
- Possible don't need objects (ex. statistics)

Should database or application do the work?
- Databases very efficient at searching/sorting … we prefer!

Java API's for Persistence

Java database connectivity, JDBC

- Low level API, no ORM (not used by us)
- Using embedded SQL strings as parameters
- JEE spec. makes JDBC mandatory

Java Data Objects, JDO

- Very (too?) general, relational database, object database , ...
- Not used in course, possible fading away...?

Java Persistence API, JPA 2.x

- Supports <u>only</u> relational databases
- Built on top of JDBC
- This <u>will be our middleware</u> (glue application and database)

Java Persistence API, JPA

“The Java Persistence API provides Java developers with an object/relational mapping facility for managing relational data in Java applications.”

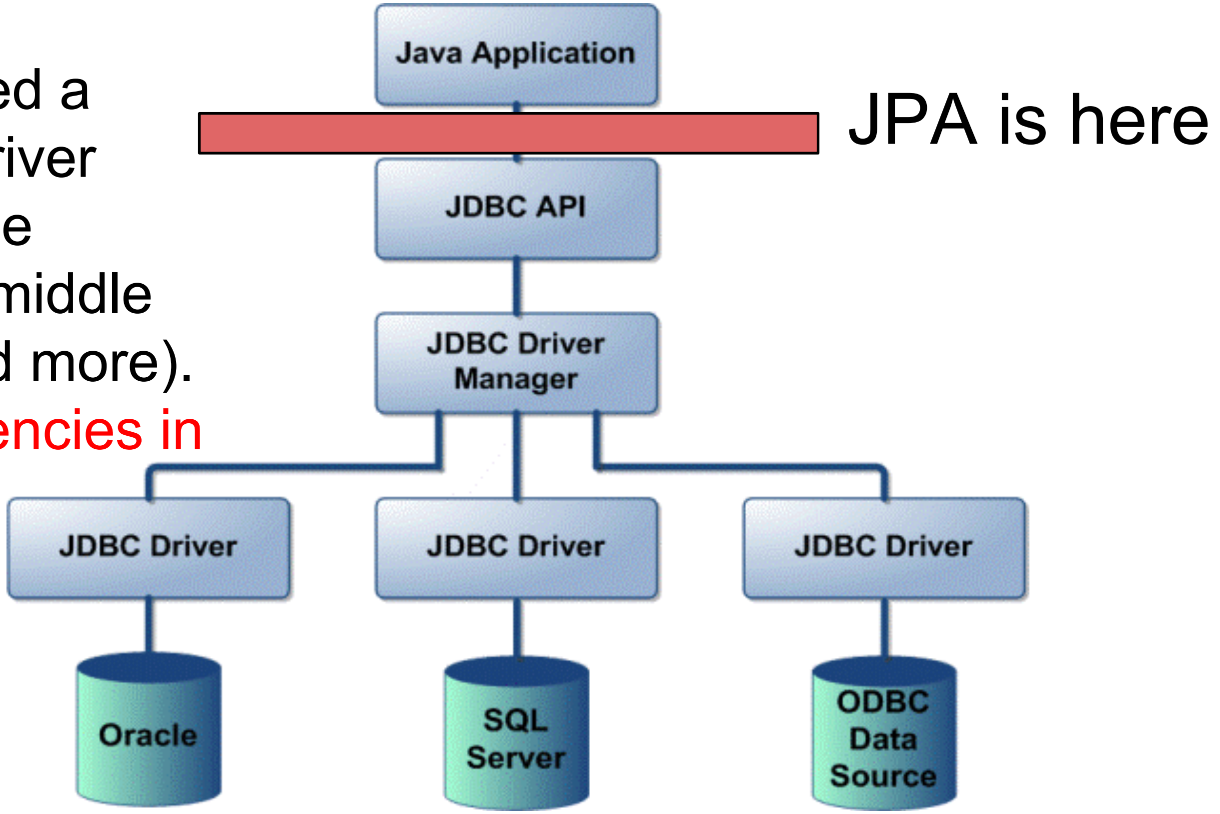
Java Persistence consists of four areas:

- The Java Persistence API, to handle persistent objects
- The Query language, to query database in an OO fashion
- The Java Persistence Criteria API, same as above by typesafe
- Object/relational mapping metadata, annotations

JPA and JDBC

JPA built on top of JDBC JDBC

We'll need a JDBC Driver (database specific middle ware and more). Dependencies in pom



Executing JPA Applications

Possible to use JPA in JEE and JSE environments

JSE, Tomcat or JUnit

- Have to supply <u>many</u> dependencies
- Have to handle a lot in application (more to code)

JEE, GlassFish, …

- Fewer dependencies
- Container will handle a lot. <u>We use!</u>

JPA Config Files

There will be (at least) 2 config files involved
- src/main/setup/**glassfish-resources.xml**, technical data for the database, location, JDBC driver and much more (server specific). Information is noted as a **data source**. Data sources have names (like "jdbc/mydatasource" always leading jdbc, also there's an interface java.sql.DataSource)
- src/main/resources/META-INF/**persistence.xml**, containing **persistence units (PU)**. A PU defines how persistent objects should be handled. A PU has a reference to a data source

Generated by NetBeans, possible some tweaking

JavaDB (Derby)

As noted in the crash course

- We'll use JavaDB (aka Derby) a relational database bundled with NetBeans!
- Create/drop databases from inside Netbeans
- Create/drop tables (all tables should belong to a "schema" APP)
- CRUD operations on table data from inside NetBeans (NOTE: Must commit to make persistent, click small button in table heading)
- Run queries from Netbeans
- Sample database supplied (good for testing queries)

Databases stored as files in ~/.netbeans-derby directory
Possible to delete database by erasing files