HIBERNATE

Paweł Gągała
pawelgagala@gmail.com

infoShare
<academy/>

About me

Paweł Gągała:

- Developer & team lead in Roche
- In past worked also in Lufthansa, Jeppesen

Agenda

- Why database ?
- Why JPA (Java Persistence API) ?
- Why Hibernate ?
- Hibernate architecutre

Agenda

- How use hibernate ?
- States of entity
- Detais about session interaction
- Integration with JEE

Agenda

- Collection mappings
- Inhertiance
- Validation
- SQL

Agenda

- Named query
- Criteria
- HQL
- Cache

Feel free to interrupt anytime You have a questions or doubts :)

Why database ?

- We want to store somewhere persistently data
- db is just stored collection of information
- db can be relational (e.g. MySQL, Oracle) - used by most companies and non relational (e.g. MongoDB)

Why database ?

- Relational db storing data in tables made of rows and columns

Rows

```
mysql> select * from Car;
+----+---------------+
| id | model         |
+----+---------------+
|  1 | Bmw X1        |
|  2 | Volkswagen CC |
+----+---------------+
```

Why database ?

- Relational db storing data in tables made of rows and columns

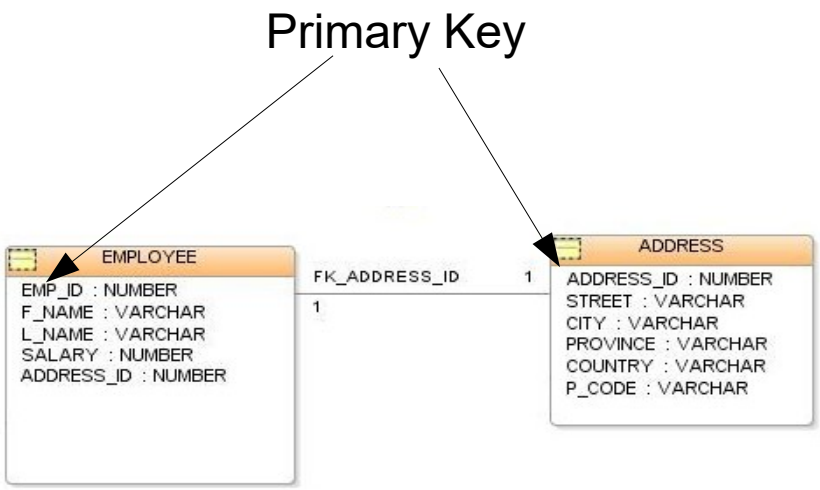
Rows

```
mysql> select * from Car;
+----+---------------+
| id | model         |
+----+---------------+
|  1 | Bmw X1        |
|  2 | Volkswagen CC |
+----+---------------+
```

Columns

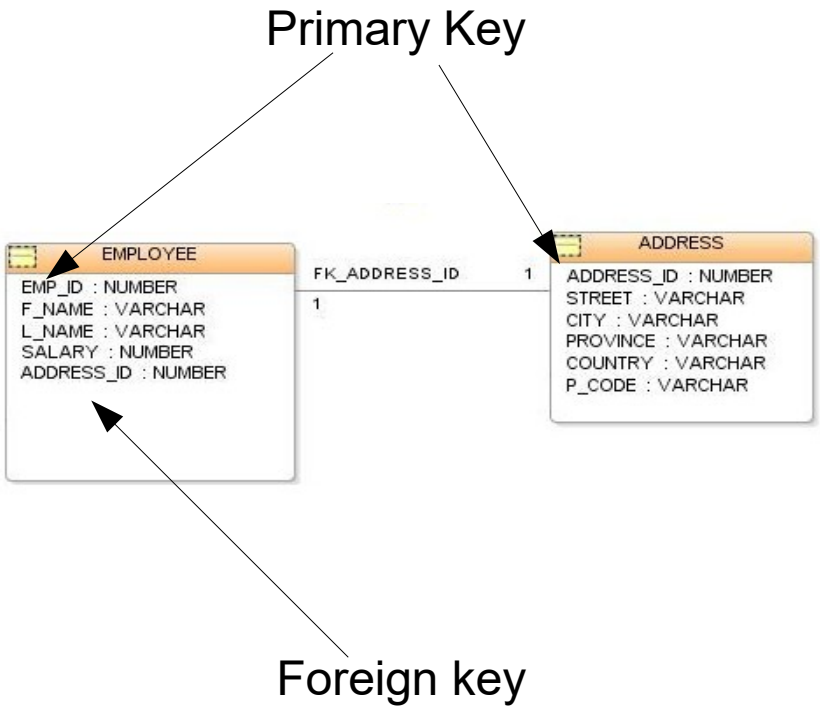
Why database ?

- Data in tables are identified by primary keys
- Relationships between tables use foreign keys



Why database ?

- Data in tables are identified by primary keys
- Relationships between tables use foreign keys

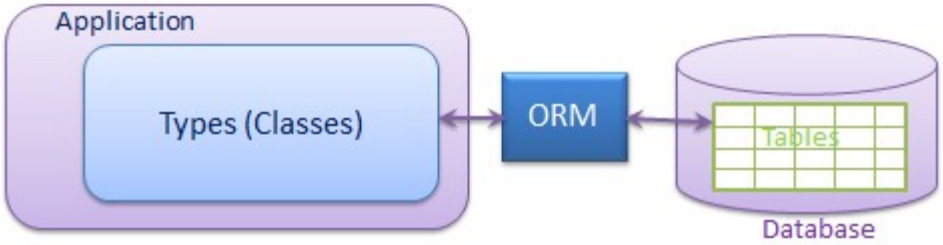


Why JPA ?

- Database vocabulary is unknown for Java (in java we have inheritence, abstract classes, collections etc. in db relational world we don't have such a structures)
- When JVM stops or garbage collector cleans memory content - object and his state disappear

Why JPA ?

- JPA brings standard - ORM (object relational mapping). ORM connects world of database and java objects

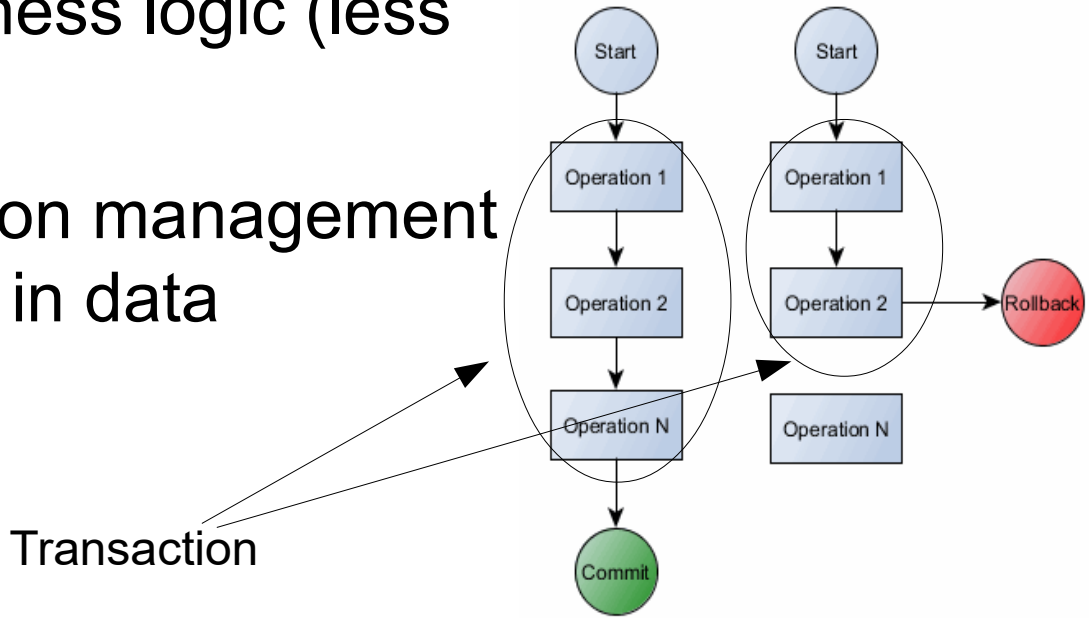


Why JPA ?

- The example of implementation ORM standard is hibernate (there are others like TopLink, JDO – Java data objects)

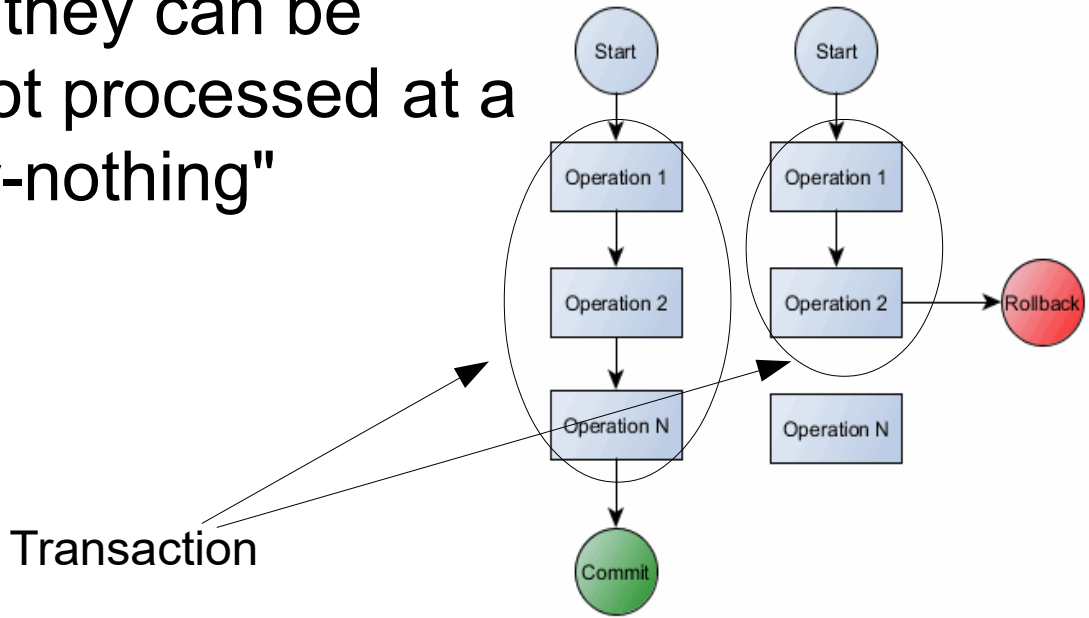
Why Hibernate ?

- Remove JDBC boiler-plate code, we can focus on business logic (less code == better)
- Supports transaction management - no inconsistency in data



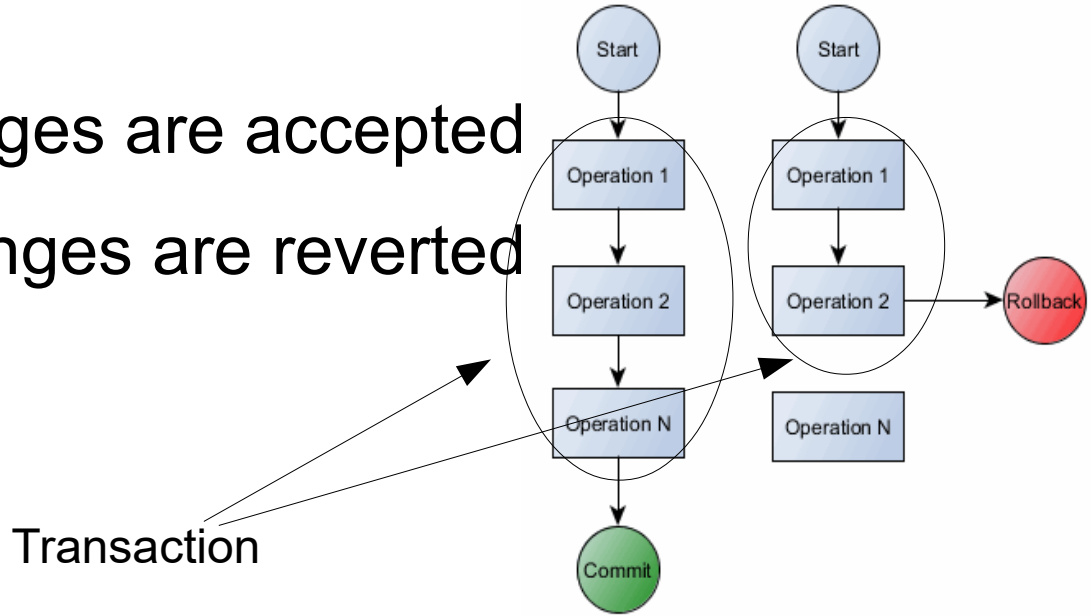
Why Hibernate ?

- Transaction – couple of operations gathered together they can be processed all or not processed at a (provide an "all-or-nothing" proposition)



Why Hibernate ?

- Transaction can be commited or rollbacked
- Commit – all changes are accepted
- Rollback – all changes are reverted

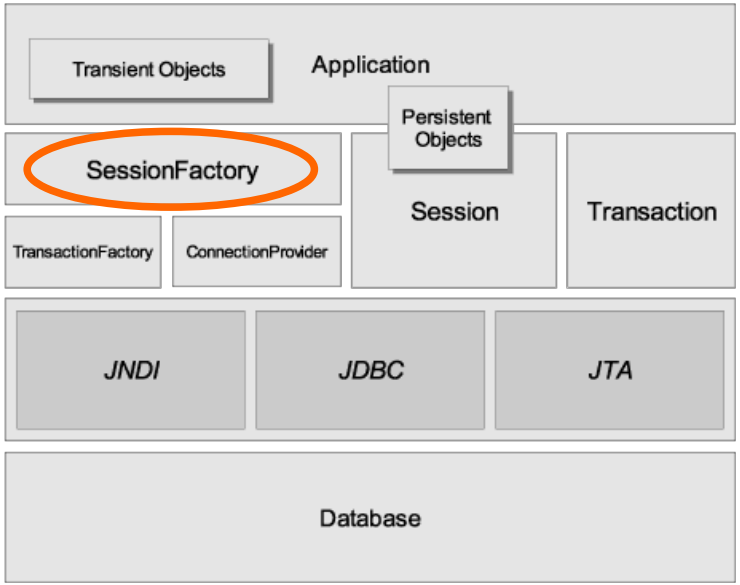


Why Hibernate ?

- Easy to integrate with JEE, other frameworks (e.g. Spring)
- Widely used in companies
- HQL - powerful query language similar to SQL (but fully object oriented), cache, criteria etc. etc.

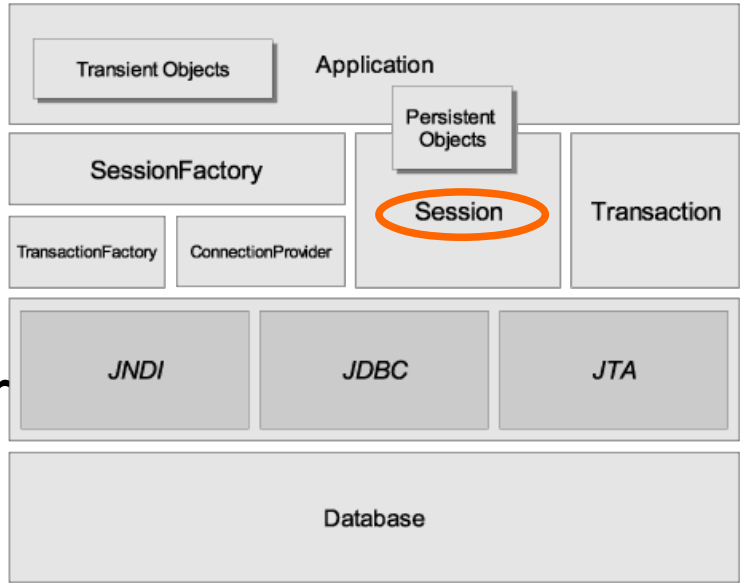
Hibernate Architecutre

- org.hibernate.SessionFactory - thread-safe cache with mappings for database, it provides Session instance



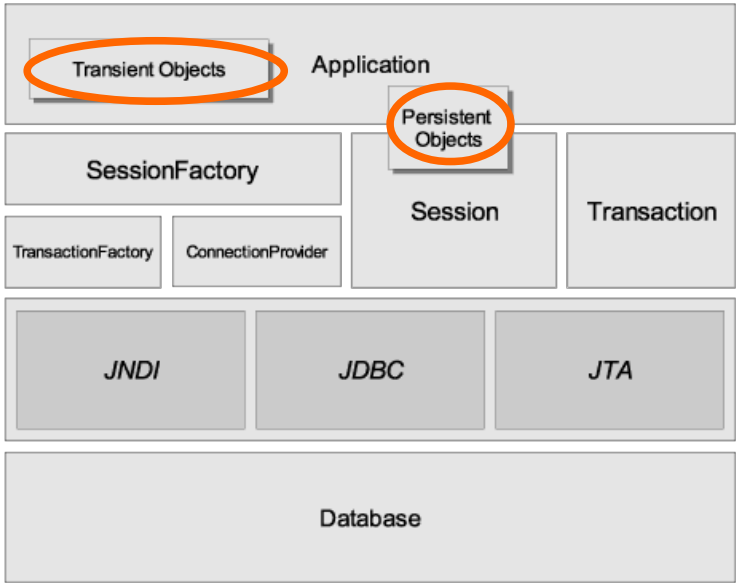
Hibernate Architecutre

- org.hibernate.Session - represents conversion between application and persistent store. Wraps JDBC Connection object. Factory for Transaction object



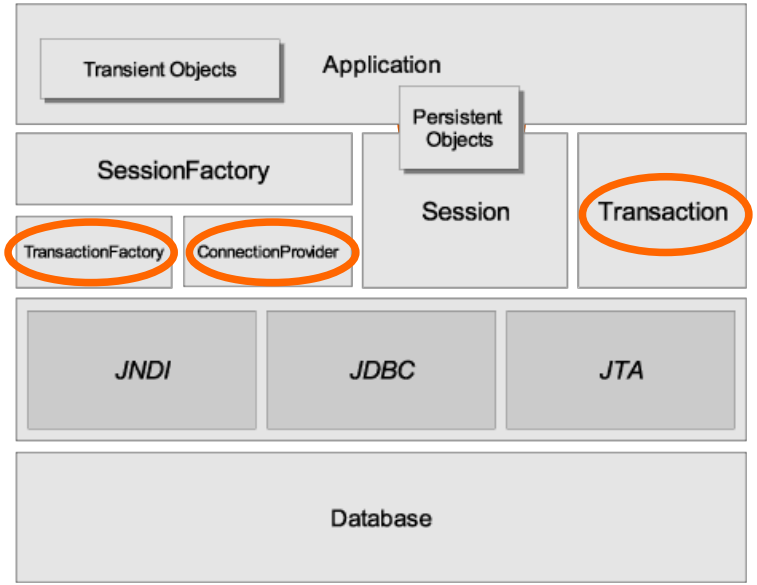
Hibernate Architecutre

- Persistent objects – Java objects connected with session
- Transient objects - Java objects not connected with session (e.g. if session was closed)



Hibernate Architecutre

- org.hibernate.Transaction-Factory - provides transactin instance
- org.hibernate.Connection-Provider - factory for jdbc connection
- org.hibernate.Transaction - atomic units of work



How use hibernate ?

- Database installation (we will use MySQL)
- Add hibernate-core dependency to Maven
- Add database connector dependency to Maven (we will use mysql-connector-java dependency)

How use hibernate ?

- Add hibernate config file
- Add Entity (Java class) with special annotations which will reflects Your table in database

How use hibernate ?

- Add hibernate config file
- Add Entity (Java class) with special annotations which will reflects Your table in database

How use hibernate ?
- MySQL installation

- Open terminal and type "sudo apt-get update"
- Type "sudo apt-get install mysql-server"

How use hibernate ?
- MySQL installation

- Type "sudo mysql_secure_installation" :
  - install validation password plugin
  - set password validation policy for 0
  - don't change password for root
  - don't remove annymous user
  - don't disallow root login remotely
  - remove test database and access to it
  - reload privilege tables now

How use hibernate ?
- MySQL installation

- Check if mysql is running by typing : "service mysql status"

```
pawel@pawel-VirtualBox:~/test/jjdz4-materialy-hibernate$ service mysql status
● mysql.service - MySQL Community Server
   Loaded: loaded (/lib/systemd/system/mysql.service; enabled; vendor preset: enabled)
   Active: active (running) since sob 2017-12-02 12:41:19 CET; 5h 30min ago
  Process: 893 ExecStartPost=/usr/share/mysql/mysql-systemd-start post (code=exited, status=0/SUCCESS)
  Process: 882 ExecStartPre=/usr/share/mysql/mysql-systemd-start pre (code=exited, status=0/SUCCESS)
 Main PID: 892 (mysqld)
   CGroup: /system.slice/mysql.service
           └─892 /usr/sbin/mysqld
```

How use hibernate ?
- MySQL installation

- Login to db using "mysql -u root -p"
- Create own user typing: "create user my_username identified by 'my_password';"
- Create database by typing: "CREATE DATABASE my_user_databse";

How use hibernate ?
- MySQL installation

- Grant access to database for own user typing: "grant all on my_user_databse.* to 'my_username' with grant option;"
- Type "exit" and login as own user (mysql -u root -p my_username)
- Switch to created table by typing: "use my_user_databse"

How use hibernate ?
- MySQL installation

- Create table by typing:

```
CREATE TABLE Car (
    id bigint(5) NOT NULL AUTO_INCREMENT,
    model varchar(50) DEFAULT NULL,
    PRIMARY KEY(id)
    );
```

How use hibernate ?
- MySQL installation

Primary key with integer type

- Create table by typing:

```
CREATE TABLE Car (
  id bigint(5) NOT NULL AUTO_INCREMENT,
  model varchar(50) DEFAULT NULL,
  PRIMARY KEY(id)
  );
```

How use hibernate ?
- MySQL installation

Cannot be null

- Create table by typing:

```
CREATE TABLE Car (
    id bigint(5) NOT NULL AUTO_INCREMENT,
    model varchar(50) DEFAULT NULL,
    PRIMARY KEY(id)
    );
```

How use hibernate ?
- MySQL installation

- Create table by typing:

Value of id will be automatically incremented with new record

```
CREATE TABLE Car (
    id bigint(5) NOT NULL AUTO_INCREMENT,
    model varchar(50) DEFAULT NULL,
    PRIMARY KEY(id)
    );
```

How use hibernate ?
- MySQL installation

- Create table by typing:

model column with varchar type (String) and default value null if we won't provide any

```
CREATE TABLE Car (
    id bigint(5) NOT NULL AUTO_INCREMENT,
    model varchar(50) DEFAULT NULL,
    PRIMARY KEY(id)
    );
```

How use hibernate ?
- MySQL installation

- Verify created table by typing: "show tables;"

```
mysql> show tables;
+-------------------+
| Tables_in_pawelDb |
+-------------------+
| Car               |
+-------------------+
1 row in set (0,00 sec)
```

Simple Example

- We will connect our Car table with java code through Hibernate

Simple Example

- We will connect our Car table with java code through Hibernate
- Add java class reflecting Car table - entity

Simple Example

- We will connect our Car table with java code through Hibernate
- Add java class reflecting Car table - entity
- Add util service for managing session

Simple Example

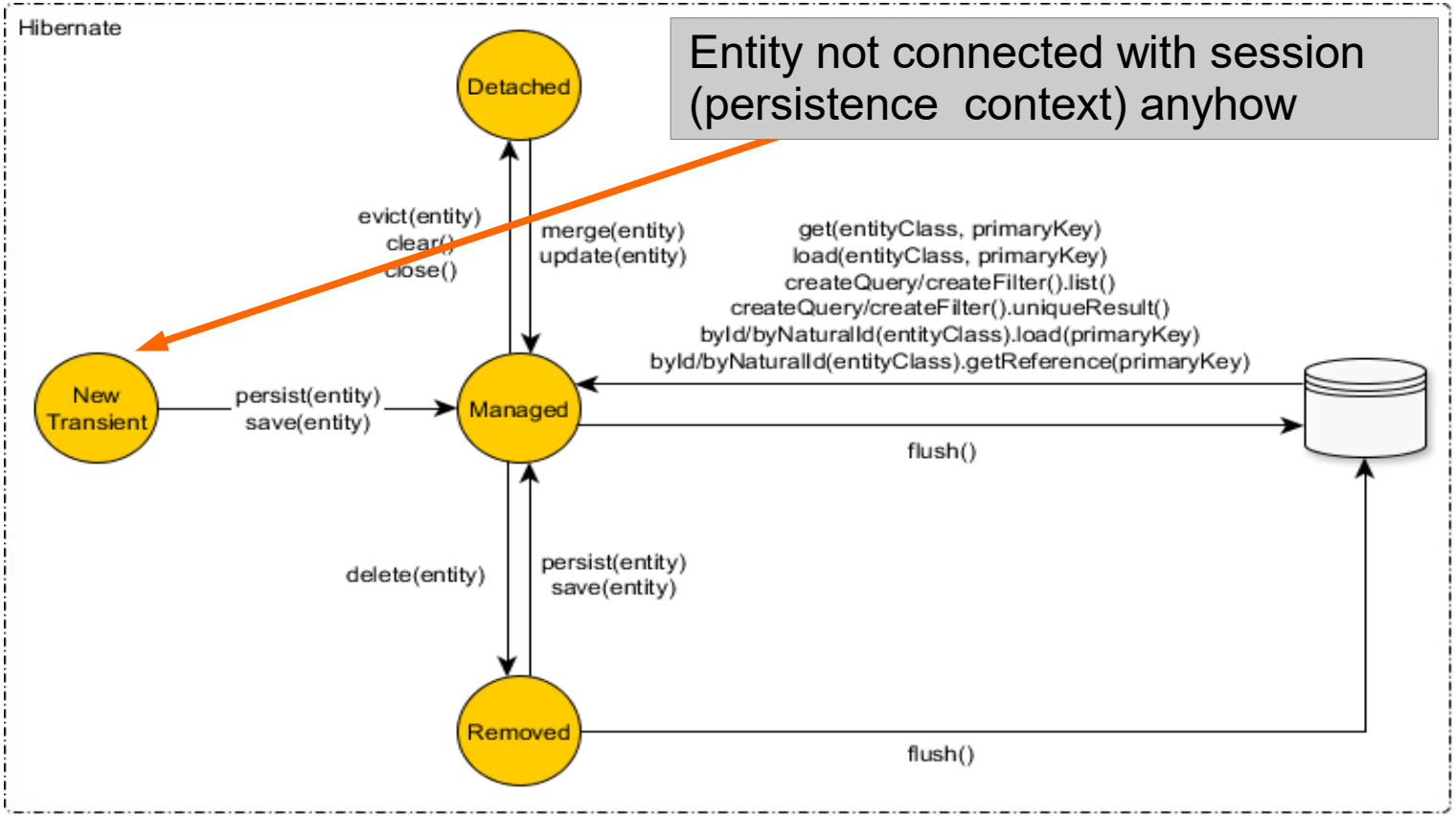
- We will connect our Car table with java code through Hibernate
- Add java class reflecting Car table - entity
- Add util service for managing session
- Add service with logic responsible for adding records to db / printing records

Excercise 1

- Clone repository
- Create table "Webpage" which will contain id, address
- Add entity reflecting Webpage table
- Add service with logic responsible for adding records to db / printing records

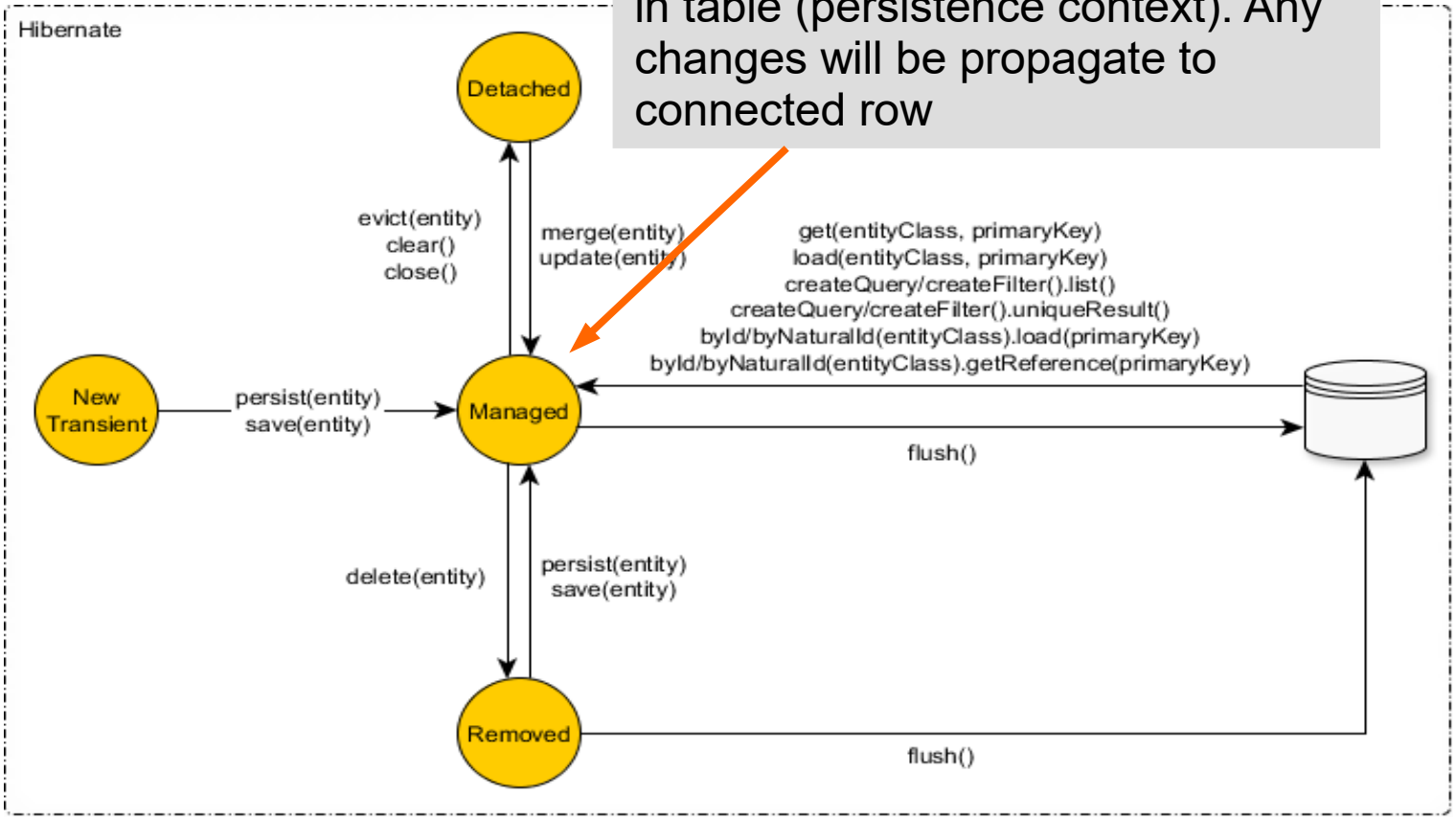
More complex example

States of entity



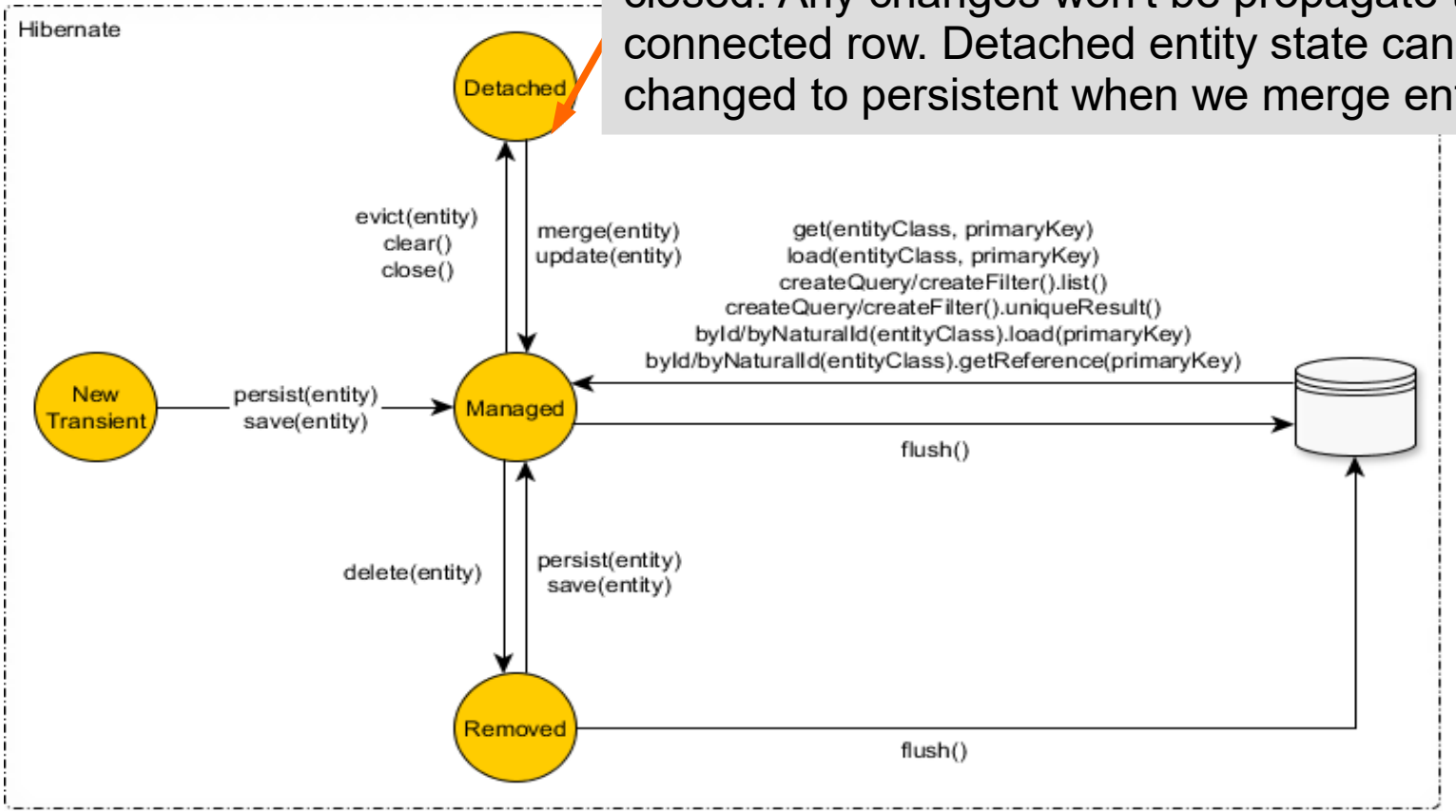
States of entity

Entity connected with particular row in table (persistence context). Any changes will be propagate to connected row

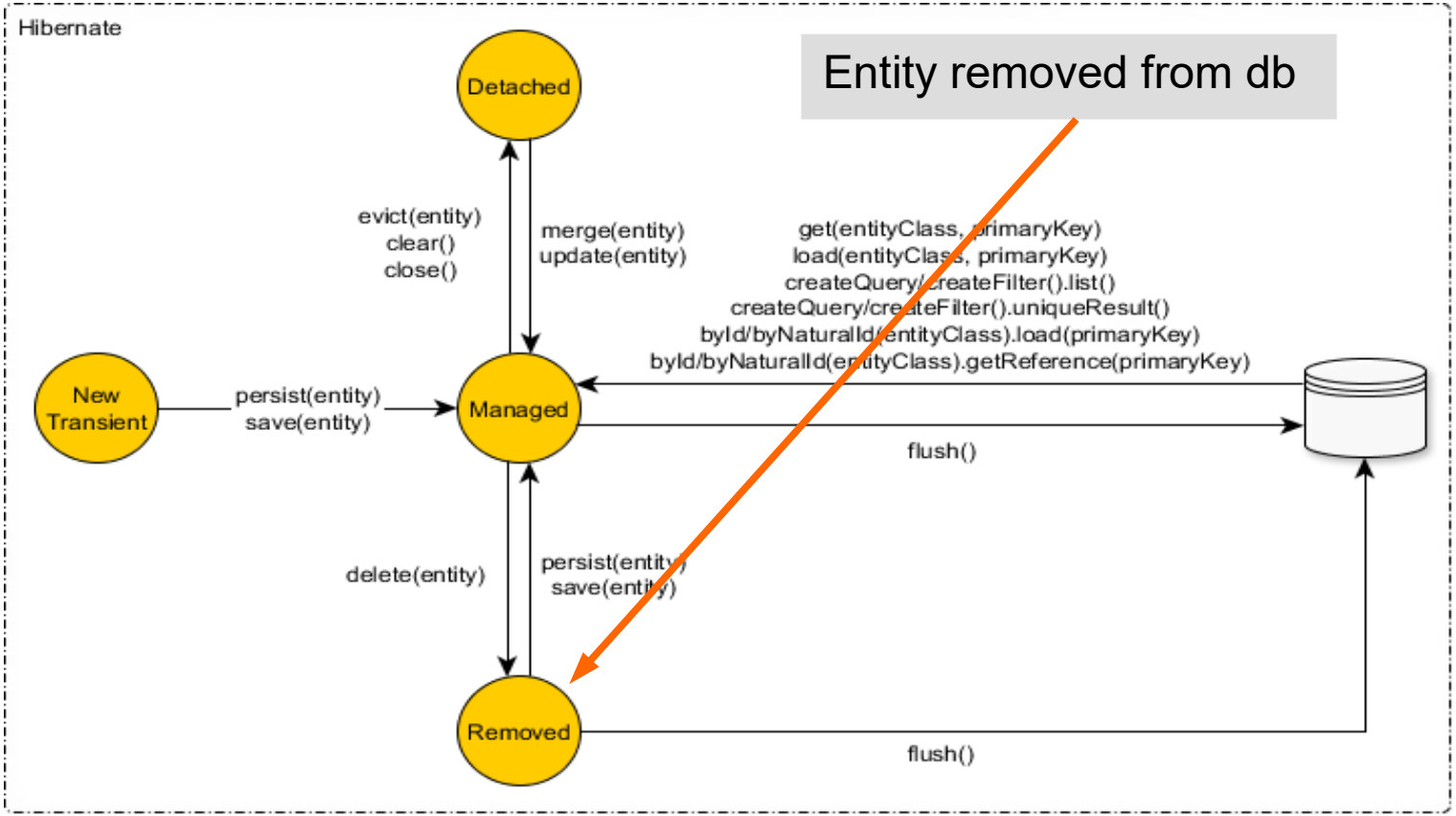


States of entity

When session connected with entity will be closed. Any changes won't be propagate to connected row. Detached entity state can be changed to persistent when we merge entity



States of entity



Detais about session interaction – obtaining session

- getCurrentSession - fetches session bound to the context. We don't have to explicitly close it (it will be closed when sessionFactory will be closed). As session is not thread safe getCurrentSession shouldn't be use in multi threaded environment

Detais about session interaction – obtaining session

- openSession - creates new session and open it. It should be always close after database operations will be done. It should be used in multi threaded application (e.g. in web appliaction for every request)

Detais about session interaction – obtaining session

- openStatelessSession - don't use first lvl cache, second lvl cache, doesn't support dirty checking, cascading operations, transactional write-behind. Useful when someone have lots of records and very less memory (otherways out of memory error in cache can occurs)

Detais about session interaction – obtaining session

- Dirty checking - auto-detect incoming changes of entity
- Transactional write-behind - optimize performance (e.g. every property change in the entity does not cause a separate sql update to be executed)
- Caches - in hibernate exists first lvl cache and second - will be explained later in separate chapter

Detais about session interaction – saving/updating table

- save - return generated id of entity after call, can be call without transaction
- update - transforms passed object from detached state to persistent and update its state in db. Throws error if passed entity is in transient state

Detais about session interaction – saving/updating table

- merge - method conforming jpa specification (EJB persistence), same as update but return managed object
- saveOrUpdate – as name suggest combination of save or update

Detais about session interaction – fetching data

- get - return object from hibernate cache or from db - using when we want make sure that data exists
- load - return reference to object which can not exists at all. It loading object from cache or db when its properties will be accessed. Return immediately object with id. It should be used if it is known that data exists

Integration with JEE

- We should use standards and standard is entity manager
- Hibernate is easy to integrate with JEE, Spring
- Interaction with entity manager is really similar to Session object, but easier and less code need to achieve that

Integration with JEE

- Entity manager under the hood use Hibernate Session

Integration with JEE - steps

- Open JEE project
- Add following dependencies to maven file:

```
<dependency>
    <groupId>org.hibernate</groupId>
    <artifactId>hibernate-core</artifactId>
    <version>5.2.12.Final</version>
    <scope>provided</scope>
 </dependency>
```

Integration with JEE - steps

- Add following dependencies to maven file:

```
<dependency>
    <groupId>mysql</groupId>
    <artifactId>mysql-connector-java</artifactId>
    <version>8.0.8-dmr</version>
</dependency>
```

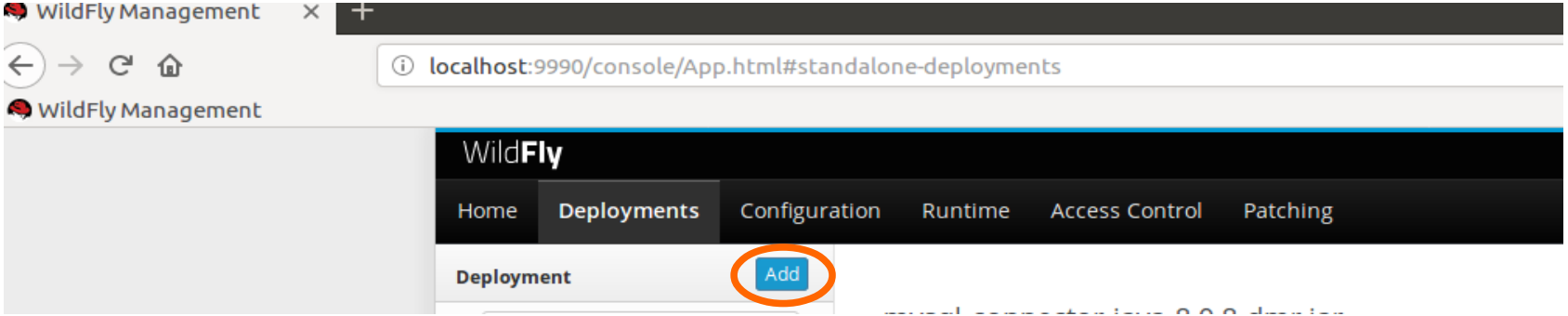
- Run maven „mvn package” command form jee project

Integration with JEE - steps

- Create account for wildfly („sh add-user.sh” in wildfly/bin folder)
- Start Wildfly („sh standalone.sh” in wildfly/bin folder)
- Login to admin panel (http://localhost:9990/) using created account

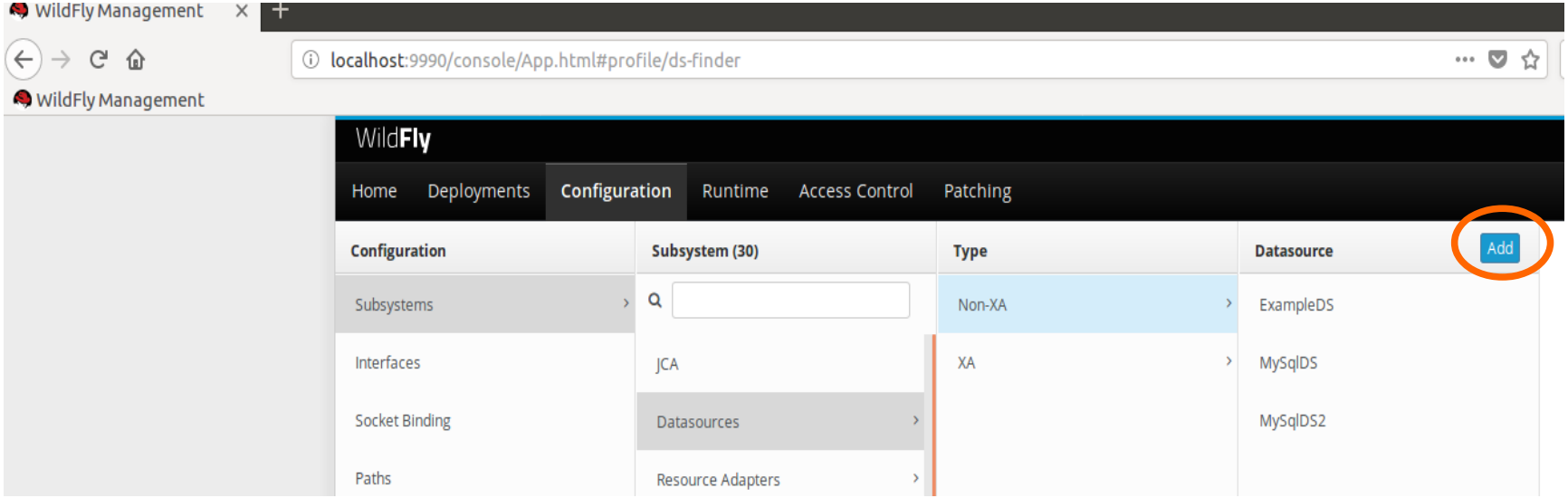
Integration with JEE - steps

- Deploy MySQL connector (mysql-connector-java-8.0.8-dmr.jar) – can be taken from target folder created earlier via maven package command



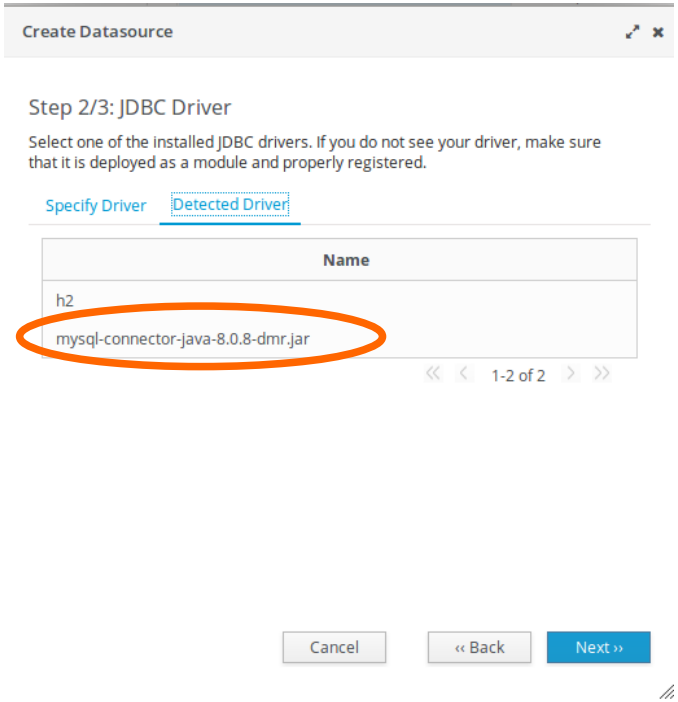
Integration with JEE - steps

- Add datasource connected with Your database



Integration with JEE - steps

- Use detected mysql driver in second step



Integration with JEE - steps

- Copy hibernate.cfg.xml from hibernate project src/main/resources to JEE project src/main/resources/META-INF

Integration with JEE - steps

- Rename hibernate configuration to persistence.xml and adjust it:

```
<?xml version="1.0" encoding="UTF-8"?>
<persistence xmlns="http://xmlns.jcp.org/xml/ns/pers
             xmlns:xsi="http://www.w3.org/2001/XMLSc
             xsi:schemaLocation="http://xmlns.jcp.or
             http://xmlns.jcp.org/xml/ns/persistence/persistence_2_1.xsd"
             version="2.1">

    <persistence-unit name="pUnit">
        <provider>org.hibernate.jpa.HibernatePersistenceProvider</provider>
        <jta-data-source>java:/mysql</jta-data-source>
        <properties>
            <property name="hibernate.archive.autodetection" value="class"/>
            <property name="hibernate.show_sql" value="true"/>
            <property name="hibernate.format_sql" value="true"/>
            <property name="hbm2ddl.auto" value="validate"/>
        </properties>
    </persistence-unit>
</persistence>
```

Unit which will be used to connect EntityManager to out datasource

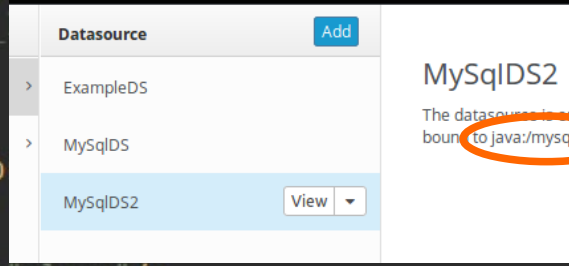
Integration with JEE - steps

- Rename hibernate configuration to persistence.xml and adjust it:

```xml
<?xml version="1.0" encoding="UTF-8"?>
<persistence xmlns="http://xmlns.jcp.org/xml/ns/pers
             xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance
             xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/persis
             http://xmlns.jcp.org/xml/ns/persistence/persistence_2_
             version="2.1">

    <persistence-unit name="pUnit">
        <provider>org.hibernate.jpa.HibernatePersistenceProvider</p
        <jta-data-source>java:/mysql</jta-data-source>
        <properties>
            <property name="hibernate.archive.autodetection" value="class"/>
            <property name="hibernate.show_sql" value="true"/>
            <property name="hibernate.format_sql" value="true"/>
            <property name="hbm2ddl.auto" value="validate"/>
        </properties>
    </persistence-unit>
</persistence>
```

Data source added in Wildfly



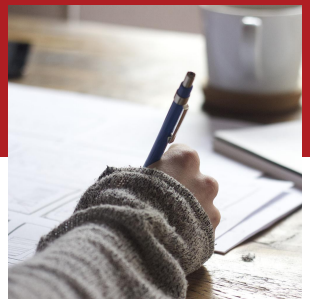
Integration with JEE - steps

- Rename hibernate configuration to persistence.xml and adjust it:

```
<?xml version="1.0" encoding="UTF-8"?>
<persistence xmlns="http://xmlns.jcp.org/xml/ns/pers
             xmlns:xsi="http://www.w3.org/2001/XMLSc
             xsi:schemaLocation="http://xmlns.jcp.or
             http://xmlns.jcp.org/xml/ns/persistence
             version="2.1">

    <persistence-unit name="pUnit">
        <provider>org.hibernate.jpa.HibernatePersistenceProvider</provider>
        <jta-data-source>java:/mysql</jta-data-source>
        <properties>
            <property name="hibernate.archive.autodetection" value="class"/>
            <property name="hibernate.show_sql" value="true"/>
            <property name="hibernate.format_sql" value="true"/>
            <property name="hbm2ddl.auto" value="validate"/>
        </properties>
    </persistence-unit>
</persistence>
```

Automatically detectes all entities in project (less code comparing with hibernate config)



Thanks!!

Questions?