3) Hibernate

- Un *framework per la gestione della persistenza* consente di mappare direttamente una tabella in un oggetto, istanza di una classe (**O-R mapping service**).
- Non è necessario scandire il database. Il risultato popolerà direttamente una collezione di oggetti.
- Suggerimento: utilizzare "**Hibernate**" per velocizzare il processo.di programmazione (che non è l'obiettivo principale del progetto)
- Java: l'IDE consigliato è **IntelliJ IDEA 15** [**https://www.jetbrains.com/student/**]
- Il progetto d'esempio è su GitHub (https://github.com/jackbergus/javahibernateexample)

Java: Maven (pom.xml)

- **Apache Maven** è un software per la gestione di progetti Java e *build automation*.
- **Apache Maven** usa un costrutto conosciuto come *Project Object Model* (POM); un file XML (pom.ml) che descrive le dipendenze fra il progetto e le varie versioni di librerie necessarie nonché le dipendenze fra di esse.
  - In questo caso si inseriscono le dipendenze per Hibernate, per il driver di accesso al server MySQL, e la libreria esterna per la gestione delle *transactions*.

```
<dependencies>

    <!-- Adding the hibernate Dependencies -->
    <dependency>
        <groupId>org.hibernate</groupId>
        <artifactId>hibernate</artifactId>
        <version>3.2.6.ga</version>
    </dependency>
    <dependency>
        <groupId>org.hibernate</groupId>
        <artifactId>hibernate-annotations</artifactId>
        <version>3.3.1.GA</version>
    </dependency>

    <!-- Adding the mysql driver for hibernate -->
    <dependency>
        <groupId>mysql</groupId>
        <artifactId>mysql-connector-java</artifactId>
        <version>5.1.6</version>
    </dependency>

    <!-- java.lang.NoClassDefFoundError: javax/transaction/Synchronization -->
    <dependency>
        <groupId>javax.transaction</groupId>
        <artifactId>jta</artifactId>
        <version>1.1</version>
    </dependency>

</dependencies>
```

Java: Hibernate

(/src/resources/hibernate.cfg.xml)

Il file di configurazione principale va inserito tra le risorse.

```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE hibernate-configuration SYSTEM
        "http://www.hibernate.org/dtd/hibernate-configuration-3.0.dtd">

<hibernate-configuration>
    <session-factory>
        <!-- Using the MySQL database -->
        <property name="hibernate.dialect">org.hibernate.dialect.MySQLDialect</property>
        <!-- Specifiying the MySQL Driver, imported from pom.xml -->
        <property name="hibernate.connection.driver_class">com.mysql.jdbc.Driver</property>

        <!-- Updates and creates the tables -->
        <property name="hibernate.hbm2ddl.auto">update</property>

        <!-- Assume test is the database name, connect to it -->
        <property name="hibernate.connection.url">jdbc:mysql://localhost/test</property>
        <property name="hibernate.connection.username">root</property>
        <property name="hibernate.connection.password">pwd</property>

        <!-- List of XML mapping files (Employee class)
             The xml configuration for the class should be stored in the resources path in the same configuration path
             of the class -->
        <mapping resource="it/db/bergamigiacomo/mysql/Employee.hbm.xml"/>

    </session-factory>
</hibernate-configuration>
```

Da rimuovere se non si vogliono creare le tabelle automaticamente.

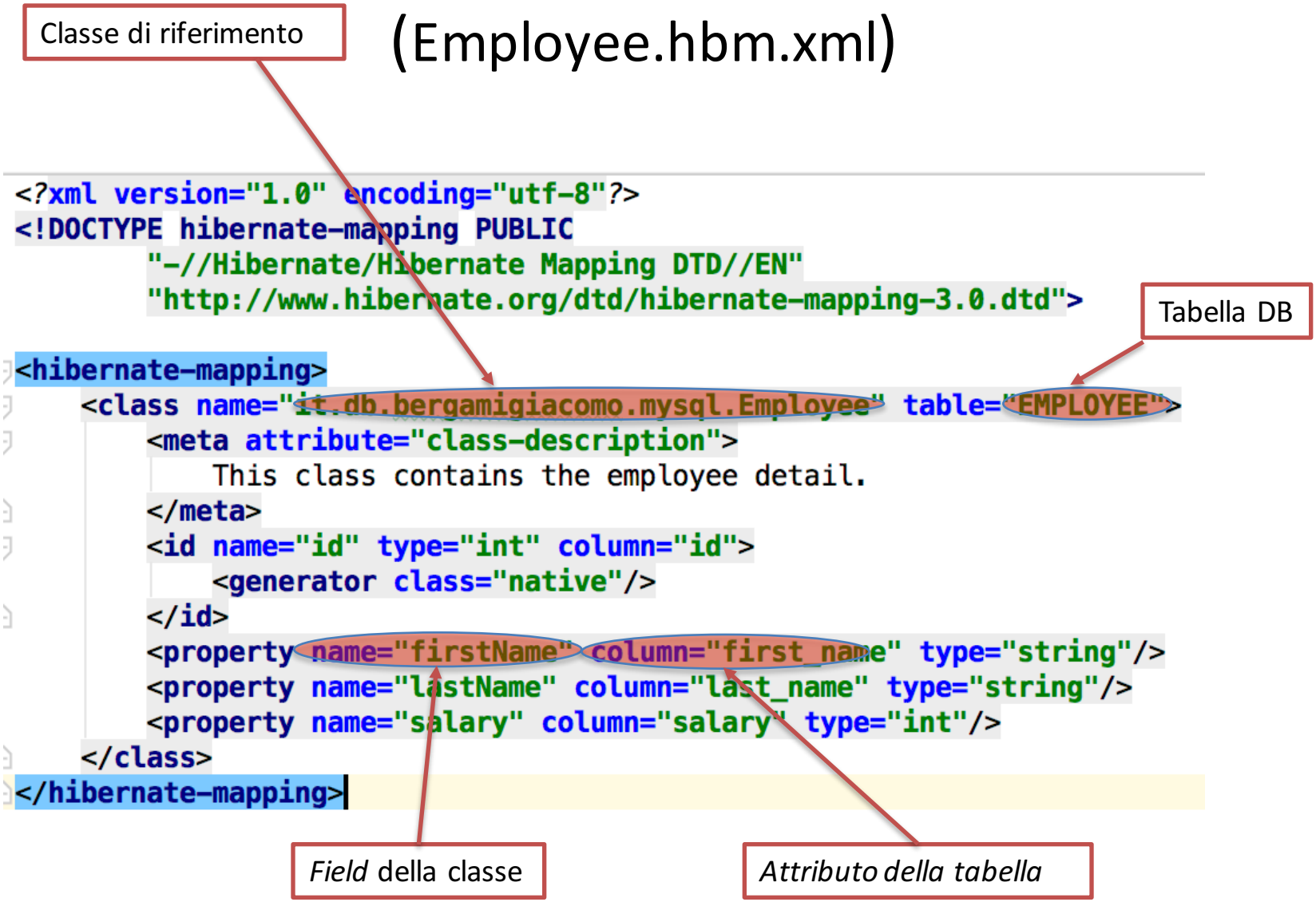
Java: Hibernate
(Employee.java)

- Ad una tabella nel database può corrispondere una classe java **POJO**: i field privati verranno inizializzati da Hibernate e, per ciascuno di questi, dev'essere implementato un getter ed un setter.
- Ad ogni classe java corrisponde un file di configurazione xml (segue)

```java
package it.db.bergamigiacomo.mysql;

public class Employee {
    private int id;
    private String firstName;
    private String lastName;
    private int salary;

    public Employee() {}

    public Employee(String fname, String lname, int salary) {
        this.firstName = fname;
        this.lastName = lname;
        this.salary = salary;
    }
    public int getId() { return id; }
    public void setId( int id ) { this.id = id; }
    public String getFirstName() { return firstName; }
    public void setFirstName( String first_name ) { this.firstName = first_name; }
    public String getLastName() { return lastName; }
    public void setLastName( String last_name ) { this.lastName = last_name; }
    public int getSalary() { return salary; }
    public void setSalary( int salary ) { this.salary = salary; }
}
```

Java: Hibernate

(Employee.hbm.xml)



Java: Hibernate
Popolare una tabella (I)

- Lo scopo principale di queste librerie è quello di evitare SQL injection: conseguentemente la maggior parte delle operazioni utilizza o processori di query o di metodi appositi all'interno di una determinata sessione per eseguire le operazioni sensibili (creazione, eliminazione, aggiornamento)
- Il **singleton** per l'accesso al database è:

```
SessionFactory factory = new Configuration().configure().buildSessionFactory();
```

Java: Hibernate
Popolare una tabella (II)

- Ogni *factory* consente di aprire una *sessione*, all'interno della quale possono essere eseguite delle *transazioni* **tx**, che consentono di eseguire delle operazioni sul database.
- La creazione di una nuova riga corrisponde alla creazione di un nuovo oggetto e del salvataggio dello stesso (**save**)

```
/* Method to CREATE an employee in the database */
public Integer addEmployee(String fname, String lname, int salary){
    Session session = factory.openSession();
    Transaction tx = null;
    Integer employeeID = null;
    try{
        tx = session.beginTransaction();
        Employee employee = new Employee(fname, lname, salary);
        employeeID = (Integer) session.save(employee);
        tx.commit();
    }catch (HibernateException e) {
        if (tx!=null) tx.rollback();
        e.printStackTrace();
    }finally {
        session.close();
    }
    return employeeID;
}
```

Java: Hibernate
Altre operazioni utili

- **session.get(clazz,id);**

  Ottiene l'entry con identificativo **id** dalla classe **clazz** (es. Employee.class).

- **session.update(obj);**

  Aggiorna un oggetto **obj** ottenuto tramite la libreria.

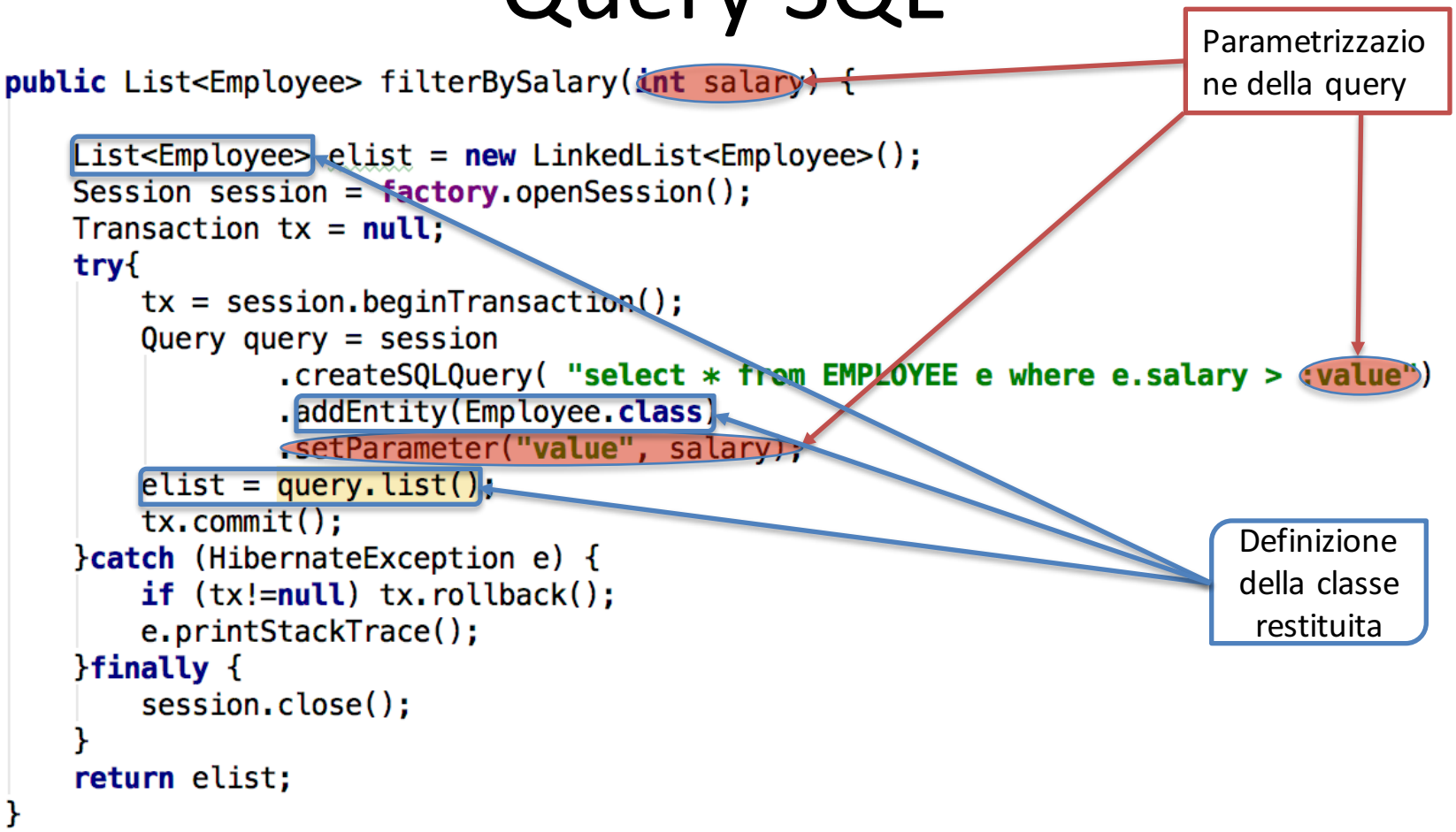
- **session.delete(obj);**

  Rimuove un oggetto **obj** ottenuto tramite la libreria.

- **session.createSQLQuery(string);**

  Crea una query che può essere eseguita. Restituisce una lista di oggetti.

Java: Hibernate Query SQL



Link

- La distribuzione corrente è scaricabile dal sito:

http://dev.mysql.com/downloads/mysql/

- Una documentazione completa è reperibile all'URL:

http://dev.mysql.com/doc/refman/5.7/en/index.html

- Libro on-line sull'utilizzo di Hibernate:

https://www.safaribooksonline.com/library/view/just-hibernate/9781449334369/