Hibernate

Using databases, the popular way

JDBC vs ORM

- При JDBC
  - Работим директно с базата, използваме SQL
  - Най-бързият вариант, но при големи системи става сложно
  - Ние сме отговорни да съпоставим таблици към java класове
  - Трудно е да се имплементира MVC модела
- При ORM
  - Бизнес кода работи с обекти, вместо с таблици
  - Под капака пак си е JDBC

Hibernate

Хиайбернейт мапва java класове и database таблици чрез XML файлове

Ако има промяна в Java обекта, само XML-а трябва да се ъпдейтне

Работи се само с Java типове от данни

Няма нужда от application server

Лесен достъп до данни и търсене

Важни обекти

- Configuration Object - задължителен, настройва връзката с базата
- SessionFactory Object - през него се създават сесиите. Много тежък, но thread-safe. Една база -> един такъв обект
- Session Object - съзщинската връзка с базата. Много лек. За всяка поредица от интеракции с базата се създава нова сесия - този обект не е thread safe и не трябва да се държи отворен дълго.
- Transaction & Query Objects - в тях се обозначава типа операция,

hibernate.cfg.xml

```
<?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>
   <property name="hibernate.dialect">
      org.hibernate.dialect.MySQLDialect
   </property>
   <property name="hibernate.connection.driver_class">
      com.mysql.jdbc.Driver
   </property>

   <!-- Assume test is the database name -->
   <property name="hibernate.connection.url">
      jdbc:mysql://localhost/test
   </property>
   <property name="hibernate.connection.username">
      root
   </property>
   <property name="hibernate.connection.password">
      root123
   </property>

   <!-- List of XML mapping files -->
   <mapping resource="Employee.hbm.xml"/>

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

Използване

```
Session session = factory.openSession();
Transaction tx = null;
try {
   tx = session.beginTransaction();
   // do some work
   ...
   tx.commit();
}
catch (Exception e) {
   if (tx!=null) tx.rollback();
   e.printStackTrace();
}finally {
   session.close();
}
```

Mapping

- Чрез XML файл
- Чрез анотации
  - @Entity - обозначава Hibernate entity
  - @Table - съдържа детайли за таблицата
  - @Id - за първичния ключ
  - @GeneratedValue - чрез него се задава custom стратегия за създаване на нови ключове
  - @Column - определя полето като колона, може да има следните свойства

Пример

```java
import javax.persistence.*;

@Entity
@Table(name = "EMPLOYEE")
public class Employee {
   @Id @GeneratedValue
   @Column(name = "id")
   private int id;

   @Column(name = "first_name")
   private String firstName;

   @Column(name = "salary")
   private int salary;

   public Employee() {}
```

Пример

```
public static void main(String[] args) {
      try{
         factory = new AnnotationConfiguration().
                   configure().
                   addAnnotatedClass(Employee.class).
                   buildSessionFactory();
      }catch (Throwable ex) {
         System.err.println("Failed to create sessionFactory object." + ex);
      }
      ManageEmployee ME = new ManageEmployee();

      /* Add few employee records in database */
      Integer empID2 = ME.addEmployee("Daisy", "Das", 5000);
      Integer empID3 = ME.addEmployee("John", "Paul", 10000);

      /* Update employee's records */
      ME.updateEmployee(empID1, 5000);

      /* Delete an employee from the database */
      ME.deleteEmployee(empID2);

      /* List down new list of the employees */
      ME.listEmployees();
   }
```

Пример

```
/* 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();
         employee.setFirstName(fname);
         employee.setLastName(lname);
         employee.setSalary(salary);
         employeeID = (Integer) session.save(employee);
         tx.commit();
      }catch (HibernateException e) {
         if (tx!=null) tx.rollback();
         e.printStackTrace();
      }finally {
         session.close();
      }
      return employeeID;
   }
```

Пример

```
/* Method to  READ all the employees */
   public void listEmployees( ){
      Session session = factory.openSession();
      Transaction tx = null;
      try{
         tx = session.beginTransaction();
         List employees = session.createQuery("FROM Employee").list();
         for (Iterator iterator =
                              employees.iterator(); iterator.hasNext();){
            Employee employee = (Employee) iterator.next();
            System.out.print("First Name: " + employee.getFirstName());
            System.out.print("  Last Name: " + employee.getLastName());
            System.out.println("  Salary: " + employee.getSalary());
         }
         tx.commit();
      }catch (HibernateException e) {
         if (tx!=null) tx.rollback();
         e.printStackTrace();
      }finally {
         session.close();
      }
   }
```

MVC