Introduction to Hibernate

SELECT FROM Student – използваме си името на обекта който дефинира таблицата това е **Hibernate Querying Language – HQL**

**Entity Class: Student**

@Entity @Table(name = "students") – Set table name даваме име на таблицата

public class Student { - дефиниране на класа

@Id - главен ключ (Primary key)

@GeneratedValue(strategy = GenerationType.IDENTITY)

@Column(name = "id") - име на колоната

private long id; - типа на данните в колоната

@Column(name = "name", length = 50) – име на колоната и дължина на думата която може да се въведе

private String name; - типа на данните в колоната

Annotations

* @Entity - казваме че даден клас е ентити че трябва да бъде менажирано от Hibernate (че е част базата)
* @Table - декларираме че е таблица и името и
* @Basic – имаме поле (Field) без специфични изисквания към него
* @Transient– локална информация която да не се добавя в базата
* @Id– Конкретното пропърти е главен ключ
  + @GeneratedValue– можем да го използваме за ауто инкремент , но ние най – често ще го използваме за да зададем IDENTITY – или как да се идентифицира атрибутите които сме инициализирали. За другите стратегии за тази анотация да прочета в нета.
    - Automatic, manual, or value taken from a sequence   
      table
* @Column – може да се добавят допълнителни атрибути към нашето поле (Field)

(name = "name", length = 50) във реда – (@Column(name = "name", length = 50)

**JPA(Java Persistence API) Configuration –**

pom.xml

<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/maven-v4\_0\_0.xsd">

<modelVersion>4.0.0</modelVersion>

<groupId>com.javawebtutor</groupId>

<artifactId>JPAMavenExample</artifactId>

<packaging>jar</packaging>

<version>1.0-SNAPSHOT</version>

<name>JPAMavenExample</name>

<url>http://maven.apache.org</url>

…

…

<dependencies>

<dependency>

<groupId>javax.persistence</groupId>

<artifactId>javax.persistence-api</artifactId>

<version>2.2</version>

</dependency>

<dependency>

<groupId>org.hibernate</groupId>

<artifactId>hibernate-core</artifactId>

<version>5.4.30.Final</version>

</dependency>

…

…

<dependency>

<groupId>mysql</groupId>

<artifactId>mysql-connector-java</artifactId>

<version>8.0.25</version>

</dependency>

</dependencies>

</project>

* Create new directory **META-INF** in **resources** folder. After that place persistence.xml in it

persistence.xml

<?xml version="1.0" encoding="UTF-8"?>

<persistence xmlns="http://java.sun.com/xml/ns/persistence" version="2.0">

<persistence-unit name="school">

<properties> <property name = "hibernate.connection.url" value="jdbc:mysql://localhost:3306/school?createDatabaseIfNotExist=true"/>

<property name = "hibernate.connection.driver\_class" value="com.mysql.jdbc.Driver"/>

…

…

<property name = "hibernate.connection.username" value="root"/>

<property name = "hibernate.connection.password" value="1234"/>

<property name = "hibernate.dialect" value="org.hibernate.dialect.MySQL8Dialect"/>

<property name = "hibernate.hbm2ddl.auto" value="update"/>

<property name = "hibernate.show\_sql" value = "true" />

</properties>

</persistence-unit>

</persistence>

<https://plugins.jetbrains.com/plugin/15075-jpa-buddy>

<https://mvnrepository.com/> - от тук си търся репозиторитата и си ги копирам и добавям във файла(горните неща в таблицата)

**persist()** – запазва даденото ентити обект във базата данни(така си го превеждам)/persists given entity object into the DB   
(SQL INSERT)

**remove()** – изтрива дадения обект / deletes given entity into the DB   
(SQL DELETE by primary key)

**refresh()** – явно прави селект заявка за дадения обект / reloads given entity from the DB   
(SQL SELECT by primary key)

entity –единица, обект, субект (единица обект от даден клас)

persist – запазвам

**detach()** – премахва обекта от контекста само в РС /removes the object from the persistence context(PC)

**merge()** – синхронизира състоянието на махнатите обекти със РС / synchronize the state of detached entity with the PC

**contains()** - determine if given entity is managed by the PC

**flush()** – writes the changes from PC in the database