JPA- Queries 1- HQL Workshop

Overview

In this workshop, we want to do some queries. These are the subjects that we go through in this workshop:

- Query to get all the data (all the rows)
- Query With Condition
- Query with Dynamic Parameters

Preparation

We will use a new HibernateTest.java. In this class we have created several students, tutors and subjects to work with.

- Download this *HibernateTest.java* here.
- Download the domain classes here. (We will use a table for two classes where Address is an embedded class in Student.)

Drop tables student and tutor manually.

Change update to create in persistence.xml
Run the code. (If you get an error indicating that there is no table to drop, it is ok. Run it one more time). Check the database and see if you have created the tables and you have data in them.

Create a query

Now we want to do some queries in our main method and retrieve all the rows from student table:
(this code should be between tx.begin and tx.commit.

```
Query q = em.createQuery("from Student");
List<Student> students = q.getResultList();
for(Student student:students) {
   System.out.println(student);
}
```

Query belongs to *jakarta.persistence* library.
Here you may have a warning about the type safety. You can ignore it or use this method instead:

```
TypedQuery<Student> q = em.createQuery("from Student", Student.class);
```

Run the code, the result will be three students.

Query With Condition

The above statement will return all the data in the table. But if we are interested in getting a specific person and not all the rows in the table, we add *'where'* exactly the same as we do in sql.

```
TypedQuery<Student> q1 = em.createQuery("from Student as student where
student.name = 'Jimi Hendriks' ", Student.class);
List<Student> myStudents = q1.getResultList();
for(Student student:myStudents) {
   System.out.println(student);
}
```

In the above query we could use *'like'* instead of *'='* too.

```
TypedQuery<Student> q1 = em.createQuery("from Student as student where
student.name like 'Jim%'", Student.class);
List<Student> myStudents = q1.getResultList();
for(Student student:myStudents) {
   System.out.println(student);
}
```

In the queries above, we could get one or more rows of data back.

But if we want to get a unique result (just one row), we use the method *getSingleResult* instead. Now we search on the enrollmentID property to find a specific student:

```
q= em.createQuery("FROM Student as student WHERE student.enrollmentID =
'1-HEN-2019' ", Student.class);
Student theStudent = (Student) q.getSingleResult();
System.out.println(theStudent);
```

If there is no record, we get *'No entry found for the query'*.

Using lower()

You should be careful about case sensitivity. Some databases care about it. To solve the problem, we can use:

```
q=em.createQuery("FROM Student as student WHERE lower(student.name) ='jimi
hendriks'", Student.class);
Student st = (Student) q.getSingleResult();
System.out.println(st);
```

Query with Dynamic Parameters

The safer way to create a statement query is using a placeholder.
So first we will specify what we want to search in the database by creating a variable.
Then we write the query, in the sql statement we use *=:* and *the field* that we are searching in the table.
There is a method setParameter which we can use and set the parameter to the field in it.

```
String requiredName = "jimi hendriks";
q=em.createQuery("FROM Student as student WHERE lower(student.name)
=:name", Student.class);
q.setParameter("name", requiredName);
List<Student>QueryResult =q.getResultList();
for(Student st1:QueryResult) {
   System.out.println(st1);
}
```