JPA Queries

Part 1

Overview

This slide covers the following topics:

- Queries in hibernate
    - HQL
- Query With Condition
- Query with Dynamic Parameters

Queries in Hibernate

What is HQL?

- HQL is a query string used to retrieve data.
- It is similar to SQL but works with Java objects instead of tables.
- Example format: *from Student*
    - Here, Student refers to a Java class mapped to the database table student.

find method

Using find() works only with IDs:

```
Student student = em.find(Student.class, 2);
Tutor tutor = student.getTutor();
Set<Subject> subjects = tutor.getSubjects();
for (Subject subject: subjects) {
   System.out.println(subject);
}
```

Creating a Query in HQL

In order to write queries in hql there is a method ***createQuery*** in EntityManager and Session which creates an object from the query and then we can run ***getResultList()*** on this object which will return a regular Java list.

Retrieving All Rows

To retrieve all records from a table, we use createQuery():

```
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.

Note: When using Query if you get a warning about the type safety. You can ignore it or use this method instead:

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

Query With Condition

To filter specific records, we use the *where* clause (same as SQL):

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

Using LIKE instead of =

```
Query q= em.createQuery("from Student as student where student.name like
'Jim%'");
```

The above queries return one or more results.

Retrieving a Single Record

If we need a single result, we use getSingleResult():

```
Query q= em.createQuery("FROM Student as student WHERE
student.enrollmentID = '1-HEN-2019' ");
Student student = (Student) q.getSingleResult();
```

If no record is found, an exception is thrown: *No entry found for the query.*

Handling Case Sensitivity

Some databases are case-sensitive. To avoid mismatches, use the lower() function:

```
Query q=em.createQuery("FROM Student as student WHERE lower(student.name)
='jimi hendriks'");
Student st = (Student) q.getSingleResult();
```

Query with Dynamic Parameters

To safely pass user input, use placeholders (=:paramName) instead of hardcoding values.

This prevents SQL Injection and improves query safety.

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

Summary

Feature	**Example**
Retrieve All Rows	`from Student`
Retrieve with Condition	`where student.name = 'Jimi Hendriks'`
Retrieve using LIKE	`where student.name like 'Jim%'`
Retrieve Single Result	`getSingleResult()`
Handle Case Sensitivity	`lower(student.name) = 'jimi hendriks'`
Dynamic Parameters	`=:paramName`