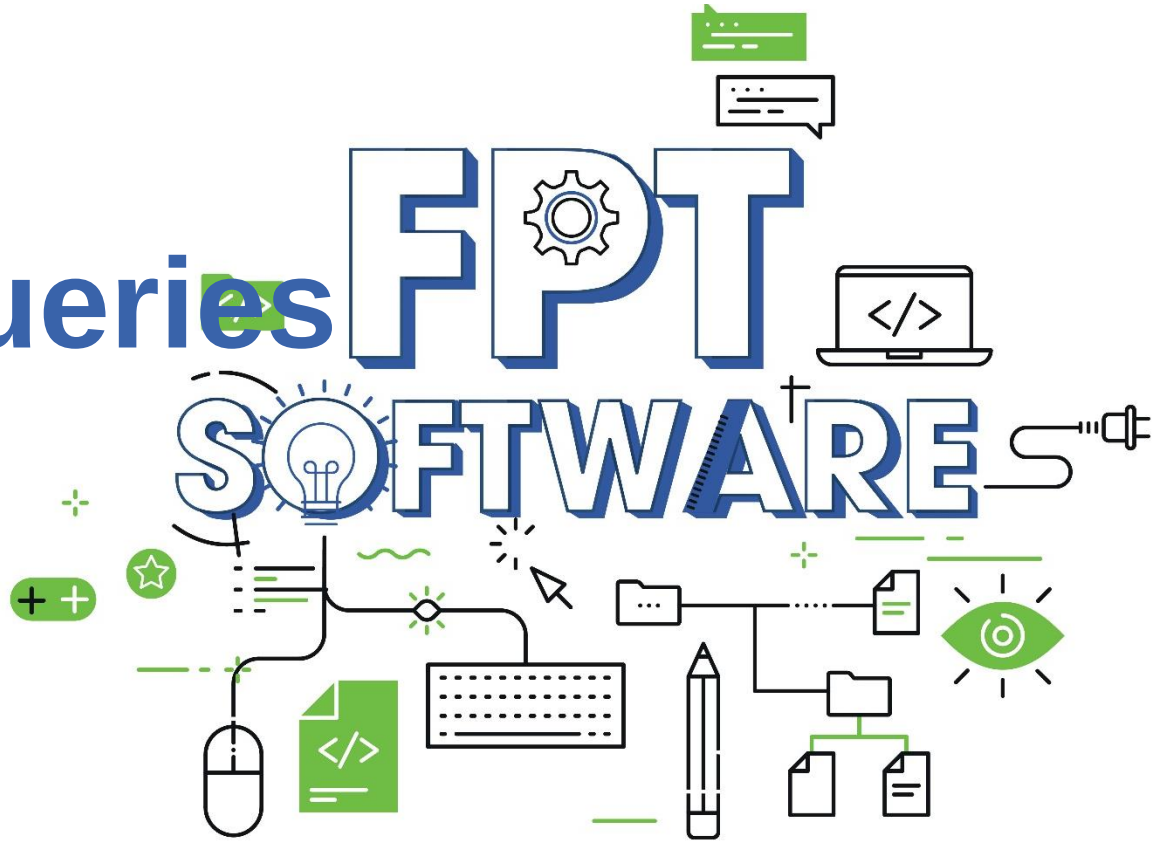
Hibernate Criteria Queries

Design by: DieuNT1



Agenda

1. JPA Criteria Queries Introduction
2. Create Criteria in Hibernate
3. HCQL: JOIN, GROUP BY, SORT, etc
4. CriteriaUpdate and CriteriaDelete
5. Question and Answer

Lesson Objectives

2. Understand the HCQL be used in Hibernate.
3. Understand the **basic steps** to create a Criteria query.
4. Able to use Hibernate Criteria Query Language to Join, Aggregation Functions, Pagination.

Section 01

JPA Criteria Queries Introduction

JPA Criteria Queries

Criteria Queries enables us to write queries without doing raw SQL as well as gives us some object-oriented control over the queries

The Criteria API allows us to build up a criteria query object programmatically, where we can apply different kinds of *filtration rules* and *logical conditions*.

- **Since Hibernate 5.2, the Hibernate Criteria API is deprecated, and new development is focused on the JPA Criteria API.**
- We'll explore how to use Hibernate and JPA to build Criteria Queries.

JPA Criteria Queries

Three types:

- ✓CriteriaQuery
- ✓CriteriaUpdate
- ✓CriteriaDelete

Section 2

Create Criteria in Hibernate

Create Criteria in Hibernate

The basic steps to create a Criteria query are:

1 - Create a CriteriaBuilder instance by calling the Session.getCriteriaBuilder() method.

```
CriteriaBuilder builder = session.getCriteriaBuilder();
```

2 - Create a query object by creating an instance of the CriteriaQuery interface.

```
CriteriaQuery<T> query = builder.createQuery(T.class);
```

3 - Set the query Root by calling the from() method on the CriteriaQuery object to define a range variable in FROM clause.

```
Root<T> root = query.from(T.class);
```

4 - Specify what the type of the query result will be by calling the select() method of the CriteriaQuery object.

```
query.select(root);
```

5 - Prepare the query for execution by creating a org.hibernate.query.Query instance by calling the Session.createQuery() method, specifying the type of the query result.

```
Query<T> q = session.createQuery(query);
```

6 - Execute the query by calling the getResultList() or getSingleResult() method on the org.hibernate.query.Query object.

```
List<T> list = q.getResultList();
```

Create Criteria in Hibernate

- **Example 1**: Selecting an entity

```
CriteriaBuilder builder = session.getCriteriaBuilder();

CriteriaQuery<Departments> criteria = builder
                    .createQuery(Departments.class);

Root<Departments> root = criteria.from(Departments.class);

criteria.select(root);
List<Departments> departments = session.createQuery(criteria)
                    .getResultList();
```

Create Criteria in Hibernate

- **Example 2**: Selecting an expression

```
CriteriaBuilder builder = session.getCriteriaBuilder();

CriteriaQuery<Departments> criteria = builder
                    .createQuery(Departments.class);

Root<Departments> root = criteria.from(Departments.class);

criteria.multiselect(root.get("deptId"), root.get("deptName"));

List<Departments> departments = session.createQuery(criteria)
                    .getResultList();
```

Create Criteria in Hibernate

- **Example 3**: Selecting multiple values

```
CriteriaBuilder builder = session.getCriteriaBuilder();

CriteriaQuery<Object[]> criteria = builder.
                                        createQuery( Object[].class );
Root<Person> root = criteria.from( Person.class );

Path<Long> idPath = root.get("id");
Path<String> nickNamePath = root.get("nickName");

criteria.select( builder.array( idPath, nickNamePath ) );
criteria.where( builder.
                                    equal(root.get("nickName"), "John Doe" ) );

List<Object[]> idAndNickNames = session.
                                        createQuery( criteria ).getResultList();
```

Create Criteria in Hibernate

- **Example 3**: Selecting multiple values

```
CriteriaBuilder builder = session.getCriteriaBuilder();

CriteriaQuery<Object[]> criteria = builder.
                                        createQuery( Object[].class );
Root<Person> root = criteria.from( Person.class );

Path<Long> idPath = root.get("id");
Path<String> nickNamePath = root.get("nickName");

criteria.select( builder.array( idPath, nickNamePath ) );
criteria.where( builder.equal(root.get("nickName"), "John Doe" ) );

List<Object[]> idAndNickNames = session.createQuery( criteria ).getResultList();
```

Create Criteria in Hibernate

- **Using *Expressions***
  - ✓ The *CriteriaBuilder* can be used to restrict query results based on specific conditions.
  - ✓ By using *CriteriaQuery* *where()* method and provide *Expressions* created by *CriteriaBuilder.*
- **Common Examples:**
  - ✓ *To get items having a price more than 1000:*

```
criteria.select(root).where(builder.gt(root.get("itemPrice"), 1000));
```

  - ✓ *Getting items having itemPrice less than 1000:*

```
criteria.select(root).where(builder.lt(root.get("itemPrice"), 1000));
```

  - ✓ *Items having itemNames contain Chair:*

```
criteria.select(root).where(builder.like(root.get("itemName"), "%chair%"));
```

  - ✓ *Records having itemPrice in between 100 and 200:*

```
criteria.select(root).where(builder.between(root.get("itemPrice"), 100, 200));
```

Create Criteria in Hibernate

- **Common Examples:**
  - ✓ *To check if the given property is null:*

```
criteria.select(root).where(builder.isNull(root.get("itemDescription")));
```

  - ✓ *To check if the given property is not null:*

```
criteria.select(root).where(builder.isNotNull(root.get("itemDescription")));
```

- **Criteria API allows us to easily chain expressions**:

```
Predicate greaterThanPrice = builder.gt(root.get("itemPrice"), 1000);
Predicate chairItems = builder.like(root.get("itemName"), "Chair%");
criteria.where(builder.and(greaterThanPrice, chairItems));
```

Create Criteria in Hibernate

- **Sorting**

```
criteria.orderBy(

builder.asc(root.get("itemName")),

builder.desc(root.get("itemPrice")));
```

Create Criteria in Hibernate

- **GROUP BY and HAVING example**

```
List<Departments> departments = session.createQuery(criteria).getResultList();

CriteriaBuilder builder = session.getCriteriaBuilder();

CriteriaQuery<Object[]> criteriaQuery = builder.createQuery(Object[].class);
Root<Employee> root = criteriaQuery.from(Employee.class);
criteriaQuery.multiselect(builder.count(root.get("name")),
                                root.get("salary"), root.get("department"));
criteriaQuery.groupBy(root.get("salary"), root.get("department"));
criteriaQuery.having(builder.greaterThan(root.get("salary"), 30000));

Query<Object[]> query = session.createQuery(criteriaQuery);
List<Object[]> list = query.getResultList();
```

Create Criteria in Hibernate

- **FROM and JOIN example:**

```
public List<CustomerContactVo> findCustomerContact() {
    try (Session session = HibernateUtil.getSessionFactory().openSession()) {
        // SELECT * FROM Contacts
        CriteriaBuilder builder = session.getCriteriaBuilder();
        // --> conditions: <, <=, >, >=
        CriteriaQuery<CustomerContactVo> criteriaQuery =
        builder.createQuery(CustomerContactVo.class);
        // --> select, from, where, group by, having clauses
        Root<Contacts> contRoot = criteriaQuery.from(Contacts.class);
        Join<Contacts, Customers> custRoot = contRoot.join("customer", JoinType.LEFT);
        criteriaQuery.multiselect(contRoot.get("email"),
                                  contRoot.get("phone"), custRoot.get("customerId"),
                                  custRoot.get("customerName"), custRoot.get("address"));
        SelectionQuery<CustomerContactVo> query = session.createQuery(criteriaQuery);
        return query.list();
    }
}
```

Create Criteria in Hibernate

- **FROM and JOIN example:** Create **CustomerContactVo** class

```
public class CustomerContactVo {
    private String email;
    private String phone;
    private Integer customerId;
    private String customerName;
    private String address;

    // getter, setter and constructor methods

}
```

Create Criteria in Hibernate

- **FROM and JOIN example 2**

```
session = HibernateUtils.getSessionFactory().openSession();
CriteriaBuilder builder = session.getCriteriaBuilder();
// Using FROM and JOIN
CriteriaQuery<Employees> criteriaQuery = builder
                    .createQuery(Employees.class);
Root<Employees> empRoot = criteriaQuery.from(Employees.class);
Root<Departments> deptRoot = criteriaQuery.from(Departments.class);
criteriaQuery.select(empRoot);

criteriaQuery.where(builder.equal(empRoot.get("department"),
                    deptRoot.get("deptId")));

Query<Employees> query = session.createQuery(criteriaQuery);
List<Employees> list = query.getResultList();

return list;
```

Create Criteria in Hibernate

- HCQL Pagination

```
public List<UserInfor> search(int pageNumber, int pageSize) {
   Session session = sessionFactory.getCurrentSession();

   CriteriaBuilder criteriaBuilder = session.getCriteriaBuilder();
   CriteriaQuery<UserInfor> criteriaQuery = criteriaBuilder.createQuery(UserInfor.class);

   Root<UserInfor> root = criteriaQuery.from(UserInfor.class);
   criteriaQuery.select(root);

   Query<UserInfor> query = session.createQuery(criteriaQuery);
   query.setFirstResult((pageNumber - 1) * pageSize);
   query.setMaxResults(pageSize);

   List<UserInfor> listOfUser = query.getResultList();
   sessionFactory.close();

   return listOfUser;
}
```

Create Criteria in Hibernate

- **Aggregate functions examples**
  - ✓ ***Count number of employees:***

```
CriteriaQuery<Long> criteriaQuery = builder.createQuery(Long.class);
Root<Employees> root = criteriaQuery.from(Employees.class);
criteriaQuery.select(builder.count(root));
Query<Long> query = session.createQuery(criteriaQuery);
long count = query.getSingleResult();
System.out.println("Count = " + count);
```

  - ✓ ***Get max salary***

```
CriteriaQuery<Integer> criteriaQuery = builder.createQuery(Integer.class);
Root<Employees> root = criteriaQuery.from(Employees.class);
criteriaQuery.select(builder.max(root.get("salary")));
Query<Integer>      query =    session.createQuery(criteriaQuery);
int maxSalary =     query.getSingleResult();
System.out.println("Max Salary = " + maxSalary);
```

Create Criteria in Hibernate

- **Aggregate functions examples**
  - ✓ ***Get Average Salary***

```
CriteriaQuery<Double> criteriaQuery = builder.createQuery(Double.class);
Root<Employees> root = criteriaQuery.from(Employees.class);
criteriaQuery.select(builder.avg(root.get("salary")));
Query<Double> query = session.createQuery(criteriaQuery);
double avgSalary = query.getSingleResult();
System.out.println("Average Salary = " + avgSalary);
```

  - ✓ ***Count distinct employees***

```
CriteriaQuery<Long> criteriaQuery = builder.createQuery(Long.class);
Root<Employees> root = criteriaQuery.from(Employees.class);
criteriaQuery.select(builder.countDistinct(root));
Query<Long> query = session.createQuery(criteriaQuery);
long distinct = query.getSingleResult();
System.out.println("Distinct count = " + distinct);
```

CriteriaUpdate

Starting from JPA 2.1, there's support for performing database updates using the *Criteria* API.

The *CriteriaUpdate* interface can be used to implement bulk update operations.

- These operations are directly mapped to database update operations.
- Therefore the persistence context is not synchronized with the result and there is no optimistic locking of the involved entities.
- **Methods:**

  ✓ *set()* method that can be used to provide new values for database records:

If you use optimistic locking, you need to update the version column as part of your update statement.

CriteriaUpdate

- **Example**:

```
CriteriaUpdate<Item> criteriaUpdate = cb.createCriteriaUpdate(Item.class);

Root<Item> root = criteriaUpdate.from(Item.class);

criteriaUpdate.set("itemPrice", newPrice);

criteriaUpdate.where(cb.equal(root.get("itemPrice"), oldPrice));

Transaction transaction = session.beginTransaction();

session.createQuery(criteriaUpdate).executeUpdate();

transaction.commit();
```

CriteriaDelete

- *CriteriaDelete* enables a delete operation using the *Criteria* API.
- We just need to create an instance of *CriteriaDelete* and use the *where()* method to apply restrictions:

```
CriteriaDelete<Item> criteriaDelete = cb.createCriteriaDelete(Item.class);

Root<Item> root = criteriaDelete.from(Item.class);

criteriaDelete.where(cb.greaterThan(root.get("itemPrice"), targetPrice));

Transaction transaction = session.beginTransaction();

session.createQuery(criteriaDelete).executeUpdate();

transaction.commit();
```

Summary

- JPA Criteria Queries Introduction
- Create Criteria in Hibernate
- HCQL: JOIN, GROUP BY, SORT, etc
- CriteriaUpdate and CriteriaDelete

THANK YOU!