Spring Criteria API

Helps us to build criteria query object programmatically and where we can apply different kind of filtrations rules and logical conditions.

CriteriaBuilder criteriaBuilder = new CriteriaBuilder();

CriteriaQuery<Employee> criteriaQuery =
criteriaBuilder.createQuery(Employee.class);

// select * from employee

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

//prepare where clause
Predicate firstNamePredicate = criteriaBuilder.like(root.get(“firstName”), “%” + firstName + “%”);
Predicate lastNamePredicate = criteriaBuilder.like(root.get(“lastName”), “%” +lastName + “%”);
Predicate emailPredicate = criteriaBuilder.like(root.get(“email”), “%” +email + “%”);

//select * from Employee where firstName like “%pratik%” or lastName like “%pratik%”;
Predicate mergedPredicate = criteriaBuilder.or(firstNamePredicate, lastNamePredicate);
Predicate lastMergedPredicate = criteriaBuilder.and(mergedPredicate, emailPredicate);

//final query
criteriaQuery.where(lastMergedPredicate);

TypedQuery<Employee> q = entityManager.createQuery(criteriaQuery);
return q.getResultSet();

/**

In JPA, when you use `criteriaQuery.from(Employee.class)`, it indeed retrieves all columns (`SELECT *`) from the `Employee` entity. If you want to retrieve specific columns instead of all columns, you can specify the columns you want in the result set using the `multiselect()` method.

Here's how you can modify your code to retrieve specific parts only:

```java
CriteriaQuery<Tuple> criteriaQuery = criteriaBuilder.createTupleQuery();
Root<Employee> root = criteriaQuery.from(Employee.class);
```

```
// Specify the columns you want to retrieve
criteriaQuery.multiselect(
    root.get("id"),       // Assuming "id" is the attribute name of the ID column
    root.get("name"),     // Assuming "name" is the attribute name of the name column
    root.get("age")       // Assuming "age" is the attribute name of the age column
);

// Execute the query and retrieve the result set
List<Tuple> results = entityManager.createQuery(criteriaQuery).getResultList();

// Process the results
for (Tuple tuple : results) {
    Long id = tuple.get(root.get("id"));
    String name = tuple.get(root.get("name"));
    Integer age = tuple.get(root.get("age"));

    // Do something with the retrieved data
}
```

In this code:

- `multiselect()` method is used to specify the specific columns you want to retrieve from the `Employee` entity.
- Each `root.get("attributeName")` call retrieves the value of a specific attribute/column from the `Employee` entity. You can replace `"attributeName"` with the actual names of the attributes in your `Employee` entity.
- The result set is retrieved as a list of `Tuple` objects, where each `Tuple` represents a row of the result set with the specified columns.
- You can then access the values of the specific columns using the `get()` method on the `Tuple` object.

*/