Asignación 13 – Consultas con JPA

Uso de los elementos de CriteriaQuery API:

-Query: Es la interfaz base para todas las consultas en JPA (Java Persistence API). Define los métodos básicos para ejecutar una consulta y obtener resultados.

-Root: Representa la entidad de nivel superior en una consulta. Es similar a la cláusula FROM en SQL. A partir de la raíz, puedes navegar a través de las relaciones de entidades para especificar condiciones de búsqueda.

-QueryBuilder: Es una clase utilizada para crear consultas de Criteria. Proporciona métodos para construir una consulta paso a paso, agregando criterios de selección, condiciones, ordenación, etc.

-CriteriaQuery: Es una interfaz que representa una consulta de Criteria. Define métodos para configurar la consulta, como seleccionar los atributos que deseas recuperar, agregar restricciones (cláusula WHERE), ordenar los resultados, etc.

-TypedQuery: Es una subinterfaz de Query que representa una consulta tipada. Proporciona métodos para establecer el tipo de resultado esperado.

-Select: Define los atributos que deseas seleccionar en una consulta. Puedes usar métodos como criteriaQuery.select() para especificar qué atributos de la entidad deseas recuperar en los resultados.

-From: Especifica las entidades involucradas en la consulta. Se utiliza para definir las entidades sobre las que se está realizando la consulta.

-Where: Define las condiciones de filtrado de una consulta. Puedes agregar restricciones utilizando métodos como criteriaBuilder.equal() para comparar valores de atributos con ciertos criterios.

-Order: Se utiliza para especificar el orden en el que deseas que se devuelvan los resultados de la consulta. Puedes ordenar los resultados ascendente o descendentemente en función de los valores de un atributo específico.

Ejemplo de 3 consultas utilizando CriteriaQuery API explicando que hacen

Ejemplo 1:

```
CriteriaBuilder criteriaBuilder = entityManager.getCriteriaBuilder();

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

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


criteriaQuery.select(root);

List<Employee> employees = entityManager.createQuery(criteriaQuery).getResultList();
```

Selecciona todos los registros de la entidad Employee

Ejemplo 2:

```
CriteriaBuilder criteriaBuilder = entityManager.getCriteriaBuilder();
CriteriaQuery<Employee> criteriaQuery = criteriaBuilder.createQuery(Employee.class);
Root<Employee> root = criteriaQuery.from(Employee.class);

criteriaQuery.select(root)
        .where(criteriaBuilder.equal(root.get("department"), "IT"));
List<Employee> employees = entityManager.createQuery(criteriaQuery).getResultList();
```

Selecciona todos los empleados que pertenecen al departamento "IT".

Ejemplo 3:

```
CriteriaBuilder criteriaBuilder = entityManager.getCriteriaBuilder();
CriteriaQuery<Employee> criteriaQuery = criteriaBuilder.createQuery(Employee.class);
Root<Employee> root = criteriaQuery.from(Employee.class);

criteriaQuery.select(root)
        .orderBy(criteriaBuilder.asc(root.get("name")));
List<Employee> employees = entityManager.createQuery(criteriaQuery).getResultList();
```

Selecciona todos los empleados y los ordena alfabéticamente por el nombre.

Definición de JPQL

JPQL (Java Persistence Query Language) es un lenguaje de consulta orientado a objetos utilizado en el marco de Java Persistence API (JPA). Proporciona una forma de realizar consultas a bases de datos relacionales utilizando entidades y atributos de objetos en lugar de tablas y columnas de bases de datos.

Diferencias entre JPQL y SQL

-Orientación a objetos vs. Orientación a tablas: JPQL está orientado a objetos, lo que significa que trabaja con entidades y atributos de objetos en lugar de tablas y columnas de bases de datos, como lo hace SQL.

-Portabilidad: Las consultas JPQL son portables entre diferentes bases de datos porque están formuladas en términos de entidades y atributos de objetos, lo que permite que las consultas se adapten a diferentes esquemas de bases de datos subyacentes. SQL, por otro lado, puede variar considerablemente entre diferentes sistemas de gestión de bases de datos (DBMS).

-Sintaxis: Aunque comparten algunas similitudes en la sintaxis, JPQL tiene algunas diferencias en comparación con SQL. Por ejemplo, JPQL utiliza entidades y atributos de objetos en lugar de tablas y columnas, y no soporta algunas operaciones específicas de SQL como JOINs complejos o funciones de agregación avanzadas.

Ejemplo de 3 consultas utilizando JPQL explicando que hacen

Ejemplo 1:

```
String jpql = "SELECT e FROM Employee e";

List<Employee> employees = entityManager.createQuery(jpql, Employee.class).getResultList();
```

selecciona todas las instancias de la entidad Employee.

Ejemplo 2:

```
String jpql = "SELECT e FROM Employee e WHERE e.department = :dept";

List<Employee> employees = entityManager.createQuery(jpql, Employee.class)
                        .setParameter("dept", "IT")
                        .getResultList();
```

Selecciona todas las instancias de la entidad Employee donde el atributo department es igual a "IT".

Ejemplo 3:

```
String jpql = "SELECT e FROM Employee e ORDER BY e.name ASC";

List<Employee> employees = entityManager.createQuery(jpql, Employee.class).getResultList();
```

Selecciona todas las instancias de la entidad Employee y las ordena alfabéticamente por el atributo name.

Referencias:

~ danielme.com. (2023, 13 mayo). Curso Jakarta EE 9 (31). JPA con Hibernate (14): el lenguaje JPQL\HQL (1). Consultas básicas y modificaciones. danielme.com. https://danielme.com/2022/01/19/curso-jakarta-ee-jpa-con-hibernate-el-lenguaje-jpql-hql-consultas-basicas-y-modificaciones/

Baeldung, & Baeldung. (2024, 8 enero). JPA criteria queries | Baeldung. Baeldung. https://www.baeldung.com/hibernate-criteria-queries

JPA - JPQL. (s. f.). https://www.tutorialspoint.com/es/jpa/jpa_jpql.htm