Java Persistence API

- Sesión 7: Consultas JPA.

Índice

- Java Persistence Query Language (JPQL)
- Definición de consultas

JPQL

- Tiene una sintaxis muy similar a SQL
- La clausula básica es SELECT, que puede devolver una lista de valores o un único valor
- Los valores devueltos pueden ser datos básicos (tipos de los atributos) o entidades (resultantes de relaciones definidas entre las entidades)
- Podemos filtrar los datos devueltos para que complan las condiciones definidas con la clausula WHERE
- Es posible proyectar el SELECT y devolver tuplas como resultados
- Soporta JOINS entre las entidades para formular las condiciones del SELECT

SELECT

- Seleccionamos todas las instancias de una entidad

```
SELECT e
FROM Empleado e
```

- Seleccionamos todos los posibles valores de un atributo de la entidad

```
SELECT e.nombre
FROM Empleado e
```

- Seleccionamos todas las entidades asociadas con una instancia

```
SELECT e.departamento
FROM Empleado e
```

WHERE

- La mayoría de operadores de SQL están en JPQL, incluyendo los operadores `IN`, `LIKE` y `BETWEEN`, llamadas a funciones como `SUBSTRING` o `LENGTH` y subqueries

```
SELECT e
FROM Empleado e
WHERE e.departamento.name = 'NA42' AND
      e.direccion.provincia IN ('ALC','VAL')
```

```
SELECT x FROM Magazine x WHERE x.title LIKE 'J%'
```

Proyecciones

- El resultado de una consulta pueden ser tuplas
- Cada n-tupla se implementa como un array de n componentes

```
SELECT e.nombre, e.salario
FROM Empleado e

result[0] - nombre
result[1] - salario
```

Joins entre entidades

- Es posible definir las condiciones sobre el resultado de unir entidades entre las que hay una relación
- La consulta se mapeará en una consulta similar en SQL
- Devolvemos todas las direcciones de correo del departamento ‘NA42’

```
SELECT c.correo
FROM Empleado e, CuentaCorreo c
WHERE e = c.empleado AND
      e.departamento.nombre = 'NA42'
```

```
SELECT c.correo
FROM Empleado e JOIN e.cuentasCorreo c
WHERE e.departamento.nombre = 'NA42'
```

Más ejemplos

- Selecciona todos los departamentos distintos asociados a empleados

```
SELECT DISTINCT e.departamento
FROM Empleado e
```

```
SELECT DISTINCT d
FROM Empleado e JOIN e.departamento d
```

- Selecciona todos los departamentos que trabajan en ‘Alicante’ y que participan en el proyecto ‘BlueBook’

```
SELECT DISTINCT e.departamento
FROM Proyecto p JOIN p.empleados e
WHERE p.nombre = 'BlueBook' AND
      e.direccion.localidad = 'ALC'
```

Más ejemplos (2)

- Selecciona todos los proyectos distintos de empleados de un departamento

```
SELECT DISTINCT p
FROM Departamento d JOIN d.empleados JOIN e.proyectos p
```

- Selecciona todos los empleados y recupera la entidad `Direccion` con la que está relacionado cada uno

```
SELECT e
FROM Empleado e JOIN FETCH e.direccion
```

Parámetros de las consultas

- En JPQL (igual que en JDBC) es posible definir las conultas de forma estática y parametrizar ciertos valores
- El acceso a esos valores se puede hacer por posición o por identificador

```
SELECT  e
FROM Empleado e
WHERE e.departamento = ?1 AND
      e.salario > ?2
```

```
SELECT  e
FROM Empleado e
WHERE e.departamento = :dept AND
  e.salario > :sal
```

Definiendo y ejecutando las consultas

- Dos formas de definir consultas:
  - Dinámicas con el método `createQuery()` del entity manager
  - Estáticas, definiéndolas en la entidad y asociándoles un nombre
- Para ejecutar un consulta hay que llamar `setParameter()` para definir los parámetros y a `getSingleResult()` o `getResultList()` dependiendo de si se devuelve un único valor o una lista
- Hay que hacer casting de los objetos resultantes

Ejemplos consultas dinámicas

```
public class EmpleadoDAO {
   private static final String QUERY =
      "SELECT e.salary " +
      "FROM Empleado e " +
      "WHERE e.departamento.nombre = : deptNombre AND " +
      "      e.nombre = : empNombre";

   // ...
   public long queryEmpleadoSalario(String empNombre, String deptNombre) {
      return (Long) em.createQuery(QUERY)
                      .setParameter("deptNombre", deptNombre)
                      .setParameter("empNombre", empNombre)
                      .getSingleResult();
   }
}
```

Ejemplo consultas con nombre

```
@NamedQueries({
   @NamedQuery(name="Empleado.findAll",
               query="SELECT e FROM Empleado e"),
   @NamedQuery(name="Empleado.findById",
               query="SELECT e FROM Empleado e WHERE e.id = :id"),
   @NamedQuery(name="Empleado.findByNombre",
               query="SELECT e FROM Empleado e WHERE e.nombre = :nombre")
})
@Entity
public class Empleado {...}
```

```
public class EmpleadoDAO {
   // ...
   public Empleado findEmpleadoByNombre(String nombre) {
      return (Empleado) em.createNamedQuery("Empleado.findByNombre")
                          .setParameter("nombre", nombre)
                          .getSingleResult();
   }
   public List<Empleado> findAll() {
      return (List<Empleado>) em.createNamedQuery("Empleado.findAll")
                                .getResultList();
   }
}
```

Un último ejemplo

```
public void muestraEmpleadosProyecto(String nombreProyecto) {
   List<Empleado> result = em.createQuery(
                              "SELECT e " +
                              "FROM Proyecto p JOIN p.empleados e " +
                              "WHERE p.nombre = ?1 " +
                              "ORDER BY e.name")
                              .setParameter(1, nombreProyecto)
                              .getResultList();
   for (empleado : result) {
      System.out.println(empleado.nombre);
   }
}
```

¿Preguntas?