La estructura de JPA para la persistencia y la recuperación de objetos

EntityTransaction

Query

EntityManager

EntityManagerFactory

EntityManagerFactory: Está disponible en memoria durante el tiempo que el programa este en ejecución. Gestiona los entityManager. Es específico para cada base de datos.

EntityManagerFactory emf = Persistence.createEntityManagerFactory (“myDbFile.odb”);

myDbFile.odb 🡪 nombre de la unidad de persistencia, la encontramos en el persistence.xml/name, esto debemos proveerle al EntityManagerFactory.

EntityManager: representa nuestra conexión a la base de datos, podemos tener más de uno con sus respectivas transacciones.

EntityManager em= emf.create

Con este creamos la conexión a la basde de datos, permite realizar las transacciones y las consultas.

ESTADOS DE OBJETOS EN EL PERSISTENCE CONTEXT

Cuando instanciamos un objeto pasa a estar en estado NEW.

Cuando hacemos entityManager.Persist (objeto), el objeto pasa a estar en el estado MANAGED, pasa a estar dentro del contexto del mapeador.

Cuando vamos a guardar el objeto físicamente en la base de datos con el commit.

Cuando hacemos consultas, trae de la base de datos objetos siguen en estado MANAGED.

Tener en cuenta que nos comunicamos en un espacio de objetos denominado persistence context, el entityManager trae objetos de este espacio.

Cuando cerramos el entityMaganerFactory, los objetos quedan “Flotando” en memoria.

Operaciones CRUD con JPA

Todo lo que hacemos debe ser dentro del begin y commit, para comenzar y cerrar la transacción.

Creación

Em.getTransaction().begin();

Em.persist(objeto);

Em.getTransaction().commit();

En caso de que una clase tenga un atributo tipo de dato definido por usuario se debe realizar em.persist de ambos objetos, con la estrategia cascada previamente configurada.

El modo cascada global en todo el proyecto, se establece en el persistence.xml

Consultas

Employee e= em.find (Emplyee.class, 1)🡪 trae el empleado con el id 1, es decir trae en base al id.

Diferencia entre las estrategias de búsqueda

LAZY (Perezoso) 🡪 No trae toda la cascada de referencias, ahorrando memoria según las necesidades.

Si necesito todo, utilizamos EAGER 🡪 Requiere menos transacciones pero ocupa mucha memoria.

Actualización

Tenemos modo cascada tanto en la creación como en la actualización, para que tenga en cuenta las asociaciones con los demás objetos.

Ejemplo:

Despues de obtener el empleado, con:

Employee employee = em.find(Employee.class, id);

Em.getTransaction().begin():

Employee.setName(“Carlixxx”);

Em.getTransaction().commit();

Borrado

Despues de obtener el empleado correspondiente.

Em.getTransaction().begin():

Em.remove(objeto);

Em.getTransaction().begin();

Tener en cuenta el modo cascada para tener en cuenta las asociaciones.

**Estas transacciones se pueden hacer con consultas mediante JPQL.**