**M06\_U2\_E2(JPA-BDO4)**

**Cosas a tener en cuenta:**

JPA trabaja con el paquete “javax.persistence”.

Hay que crear (NetBeans) en el IDE Persistence Unit. Esto creará el paquete “META-INF” con una unidad de persistencia (persistence.xml) el cual en su interior deberemos configurar un mínimo de:

**<properties>**

**<property name="javax.persistence.jdbc.url" value="targetesgrafiques.odb"/>** 🡪 Nombre y ruta de la BD.

**<property name="javax.persistence.jdbc.user" value="ioc"/>** 🡪 Usuario de la BD.

**<property name="javax.persistence.jdbc.password" value="ioc"/>** 🡪 Password D.

**<property name="javax.persistence.schema-generation.database.action" value="drop-and-create"/>**

**</properties>**

**Anotaciones en la clase:**

Cuando se trabaja con JPA es necesario poner anotaciones al principio de la clase y al menos en el campo de clase o atributo que consideremos el ID de la tabla. Ponemos el ejemplo de la clase “TargetaGrafica”:

**@Entity** 🡪Tenemos que indicar que es una entidad al principio de la clase

public class **TargetaGrafica** {

**@Id** 🡪 Tenemos que indicar que será el campo “id” de la tabla.

private int **nombreSerie**;

**Conexión BD:**

Para la conexión a la BD necesitamos hacer uso de los objetos:

* EntityManagerFactory
* EntityManager 🡪 Este último hará uso de la instancia del primero para trabajar.

Veamos el funcionamiento:

**EntityManagerFactory emf = Persistence.createEntityManagerFactory(“Nombre\_Unidad\_persis”);**

**EntityManager em = emf. createEntityManager();**

Ejemplo de métodos para abrir y cerrar la BD:

private final String **PU =** "ObjectDB"**;**

**private void obrir() throws SQLException{**

**gfxFactory = Persistence.createEntityManagerFactory(PU);**

**gfx = gfxFactory.createEntityManager();**

**gestor = new GestorGrafica(gfx);** 🡪Se crea una instancia de otra clase y se le pasa la conexión

**}**

**private void tancar() throws SQLException{**

**gfx.close();** 🡪 Cerramos EntityManager

**gfxFactory.close();**🡪 Cerramos EntityManagerFactory

**}**

A partir de aquí, haremos uso del objeto “em” para trabajar las transacciones (insert, update, remove…).

**Insert:**

Lo primero es iniciar la transacción:

**gfx.** **getTransaction().begin();**

Continuamos con la instrucción de “insert”, en este caso se denomina “persist”:

**gfx.persist(Objeto\_hacer\_persistente);**

Finalmente, certificamos la transacción:

**gfx.getTransaction().commit();**

**NOTA**: Para actualizar un registro, cambiaremos “persist” por “merge” 🡪 gfx.merge(Objeto);

Vemos como queda la secuencia completa:

En el caso de que no recibamos un objeto en un método por parámetro lo crearemos:

**TargetaGrafica tarjeta = new TargetaGrafica(numSerie, memram, ….)**

**gfx. getTransaction().begin();**

**gfx.persist(tarjeta);**

**gfx.getTransaction().commit();**

**Delete:**

Como en el caso del “Insert” hay que iniciar la transacción (**getTransaction().begin()**) y finalizar con un commit (**getTransaction().begin()**). La diferencia es que en vez de “persist” o “merge”, utilizaremos la instrucción “remove”. Vamos a ver como queda:

**gfx. getTransaction().begin();**

**gfx.remove(object);**

**gfx.getTransaction().commit();**

**Nota:** el objeto lo podemos obtener mediante su ID con la instrucción “find” 🡪 Objeto obj = gfx.find(Clase.class, id)

**Obtener un objeto:**

Para obtener un objeto lo hacemos con la instrucción “find”:

**TargetaGrafica tarjeta = gfx.find(TargetaGrafica.class, 1);**

**Otros comandos:**

Podemos borrar toda la base de datos de la siguiente manera:

Debemos hacer uso del objeto “Query” y el objeto “EntityManager”:

**Query query = gfx.createQuery(“DELETE FROM TargetaGrafica”);**

**gfx.getTransaction().begin();**

**query.executeUpdate();**

**gfx.getTransaction().commit();**