En el post anterior hemos hablado sobre los distintos estados que una entidad puede tener dentro de JPA .En este artículo mostraremos el código de cada una de las operaciones básicas. Para ello nos apoyaremos en la clase Persona. En el siguiente bloque de código hemos usado las anotaciones de @Entity para identificar el bean como una entidad a guardar en la base de datos y @Id para identificar el campo de clave primaria. Por último hemos sobrecargado los métodos hashCode y equals a la hora de comparar objetos

```
package es.curso.bo;

import javax.persistence.Entity;
import javax.persistence.Id;

@Entity
public class Persona {

@Id
private String nombre;
private int edad;
public Persona() {
super();

}

public Persona(String nombre, int edad) {
super();
setEdad(edad);
setNombre(nombre);
}
public String getNombre() {
```

```
return nombre;
}
public void setNombre(String nombre) {
this.nombre = nombre;

}

public int getEdad() {

return edad;
}
public void setEdad(int edad) {
this.edad = edad;

}

@Override
public int hashCode() {
//simplificacion
return nombre.hashCode();
}

@Override
public boolean equals(Object obj) {
//simplificacion
Persona nueva= (Persona)obj;
return nueva.getNombre().equals(this.getNombre());

}
}
```

Persist()

Ya vimos anteriormente el método persist pero vamos a volverlo a mostrar ya que es el encargado de almacenar nuevas entidades en base de datos.

```
public static void main(String[] args) {

Persona yo = new Persona("pedro",25);
EntityManagerFactory emf =
Persistence.createEntityManagerFactory("UnidadPersonas");
EntityManager em = emf.createEntityManager();
try {
em.getTransaction().begin();
em.persist(yo);
em.getTransaction().commit();
} catch (Exception e) {

e.printStackTrace();
}finally {
em.close();

}

}
```

Contains()

Como código complementario podemos apoyarnos en el método contains que comprueba si una entidad esta gestionada por el EntityManager devolviendo true o false según

correspanda.

```
public static void main(String[] args) {

Persona yo = new Persona("pedro",25);
EntityManagerFactory emf =
Persistence.createEntityManagerFactory("UnidadPersonas");
EntityManager em = emf.createEntityManager();
try {
em.getTransaction().begin();

System.out.println(em.contains(yo));

em.persist(yo);

System.out.println(em.contains(yo));

em.getTransaction().commit();
} catch (Exception e) {

e.printStackTrace();
}finally {
em.close();

}

}
```

Find()

Este método se encarga de localizar una Entidad a traves de su clave primaria. Para ello necesita que le pasemos la clave y el tipo de Entidad a buscar. Eso si recordemos que para que nos encuentre la entidad debemos haber sobrecargado los métodos equals y hashcode de forma correcta . En nuestro ejemplo hemos utilizado una implementación básica.

```
public static void main(String[] args) {

EntityManagerFactory emf = Persistence
.createEntityManagerFactory("UnidadPersonas");
EntityManager em = emf.createEntityManager();
Persona yo = em.find(Persona.class, "cecilio");
System.out.println(yo.getNombre());
System.out.println(yo.getEdad());

}
```

Remove()

Este método se encarga de eliminar una entidad de la base de datos.

```
public static void main(String[] args) {

EntityManagerFactory emf = Persistence
.createEntityManagerFactory("UnidadPersonas");
EntityManager em = emf.createEntityManager();
```

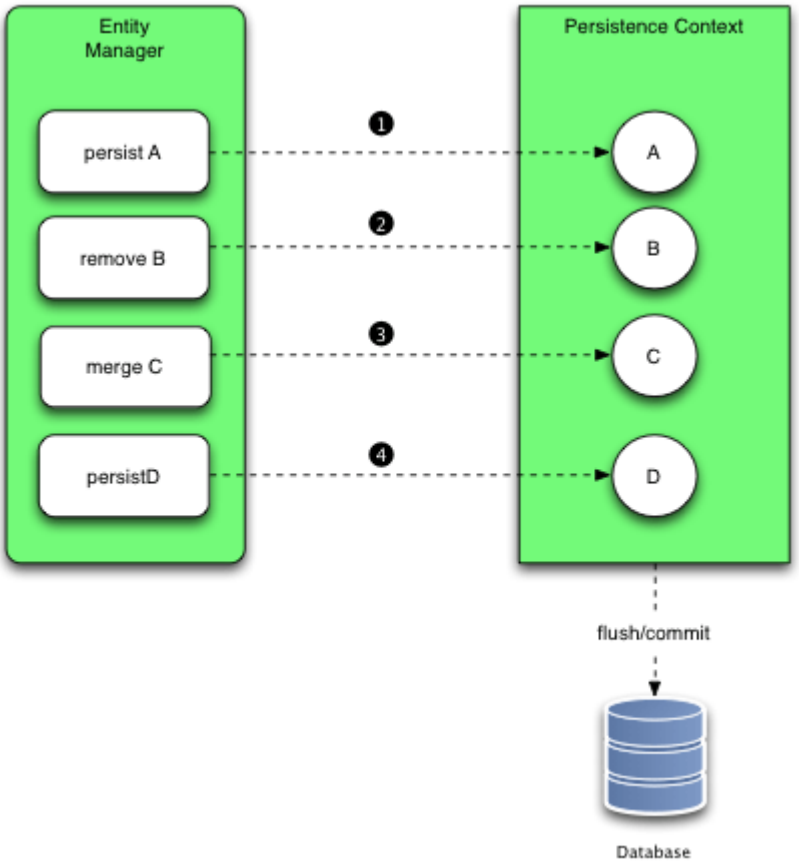
```
try {

Persona yo = em.find(Persona.class, "cecilio");
em.getTransaction().begin();
em.remove(yo);
em.getTransaction().commit();
} catch (Exception e) {
em.close();
}

}
```

Flush()

Este método es un poco mas curioso y es el encargado de sincronizar el PersistenceContext contra la base de datos .Normalmente todo el mundo piensa que cuando nosotros invocamos el método persist() o remove() se ejecutan automaticamente las consultas contra la base de datos .Esto no es así ya que el EntityManager irá almacenando las modificaciones que nosotros realizamos sobre las entidades .Para mas adelante persistirlas contra la base de datos todas de golpe ya sea invocando flush o realizando un commit a una transacción.


Entity Manager
persist A
remove B
merge C
persistD
Persistence Context
A
B
C
D
1
2
3
4
flush/commit
Database

Otros artículos relacionados

1. Un ejemplo de JPA Entity Graph
2. JPA (III) EntityManager métodos
3. JPA @ OneToMany
4. JPA @ ManyToOne
5. Ejemplo JPA

Cursos gratuitos relacionados

1. Introducción a JPA