CURSO JPA
GRATIS
APUNTATE!!

El uso de JPA Merge a veces genera dudas entre los desarrolladores que entienden de una forma más natural métodos como persist() .Vamos a explicar un poco como funciona este método y cual es su utilidad. Imaginemonos que tenemos una entidad que deseamos persistir en la base de datos . Utilizando JPA vamos a crear la clase Persona:

```
package com.arquitecturajava;

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

@Entity
public class Persona {
  @Id
  private String nombre;
  private String apellidos;
  private int edad;
  public String getNombre() {
    return nombre;
  }
  public void setNombre(String nombre) {
    this.nombre = nombre;
  }
  public int getEdad() {
    return edad;
  }
```

```
  public String getApellidos() {
    return apellidos;
  }
  public void setApellidos(String apellidos) {
    this.apellidos = apellidos;
  }
  public void setEdad(int edad) {
    this.edad = edad;
  }
  public Persona(String nombre, String apellidos, int edad) {
    super();
    this.nombre = nombre;
    this.apellidos = apellidos;
    this.edad = edad;
  }
}
```

Una vez hecho esto podemos usar un programa main para insertar la Persona en la base de datos. Para ello usaremos el EntityManager y el EntityManagerFactory.

```
package com.arquitecturajava;

import javax.persistence.EntityManager;
import javax.persistence.EntityManagerFactory;
import javax.persistence.Persistence;

public class Principal {

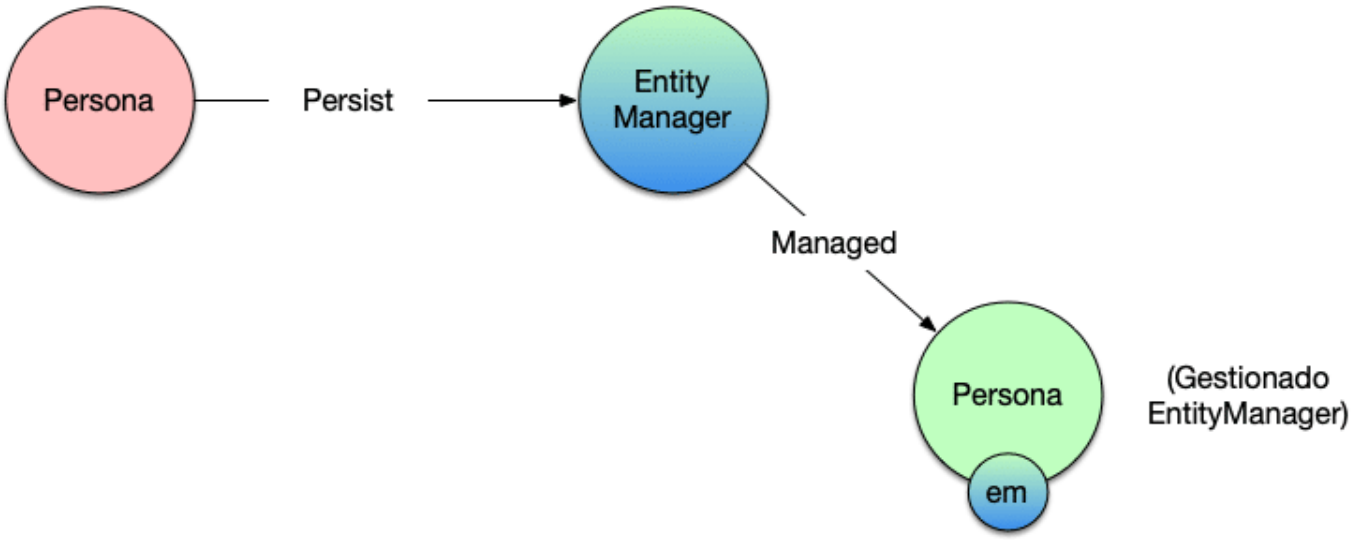
  public static void main(String[] args) {

    Persona yo = new Persona("pedro","perez",25);
    EntityManagerFactory emf =
```

```
Persistence.createEntityManagerFactory("curso");
    EntityManager em = emf.createEntityManager();
    try {
      em.getTransaction().begin();
      em.persist(yo);
      em.getTransaction().commit();
    } catch (Exception e) {
      e.printStackTrace();
    } finally {
      em.close();
    }
  }

}
```

Al utilizar el método persist lo que conseguimos es que un objeto Java que no esta controlado por ningún gestor de persistencia pase automáticamente a estarlo . Es decir el método persist hace que se inserte un registro en la tabla Persona y a la vez pasa el objeto a estar bajo el control del EntityManager.



JPA Merge

El problema suele venir cuando nosotros queremos ejecutar una funcionalidad de eliminar con el método remove(). En este caso muchas personas piensan que sería suficiente con un código de este estilo.

```
package com.arquitecturajava;

import javax.persistence.EntityManager;
import javax.persistence.EntityManagerFactory;
import javax.persistence.Persistence;

public class Principal2 {

  public static void main(String[] args) {

    Persona yo = new Persona("pedro","perez",25);
    EntityManagerFactory emf =
Persistence.createEntityManagerFactory("curso");
    EntityManager em = emf.createEntityManager();
    try {
      em.getTransaction().begin();
      em.remove(yo);
      em.getTransaction().commit();
    } catch (Exception e) {
      e.printStackTrace();
    } finally {
      em.close();
    }
  }
```

```
}
```

Podemos invocar al método remove y esto eliminaría la Entidad . Lamentablemente el metodo genera una Excepción y el programa falla al ejecutarse. ¿Qué es lo que esta ocurriendo? Parece algo muy sencillo y sin embargo la consola imprime el siguiente error.

java.lang.IllegalArgumentException: Removing a detached instance com.arquitecturajava.Persona#pedro

Esto se debe a que no entendemos realmente cómo funciona el EntityManager . El EntityManager es capaz de trabajar con Entidades Gestionadas (Managed) . Algo que en el segundo ejemplo no tenemos ya que el objeto Persona no ha sido insertado ni se ha realizado operación alguna con él, por lo tanto no podremos eliminarle . Para que esta operación pueda funcionar debemos primero usar JPA Merge . Esta funcionalidad obliga al EntityManager a convertir una Entidad Detached (No gestionada) en una Entidad Managed (Gestionada) y así podremos borrarlo . Por lo tanto el método remove que tenemos deberá modificarse de la siguiente forma.

```
package com.arquitecturajava;

import javax.persistence.EntityManager;
import javax.persistence.EntityManagerFactory;
import javax.persistence.Persistence;

public class Principal2 {

  public static void main(String[] args) {

    Persona yo = new Persona("pedro","perez",25);
    EntityManagerFactory emf =
Persistence.createEntityManagerFactory("curso");
    EntityManager em = emf.createEntityManager();
```

```
    try {
      em.getTransaction().begin();
      Persona yo2=em.merge(yo);
      em.remove(yo2);
      em.getTransaction().commit();
    } catch (Exception e) {
      e.printStackTrace();
    } finally {
      em.close();
    }
  }

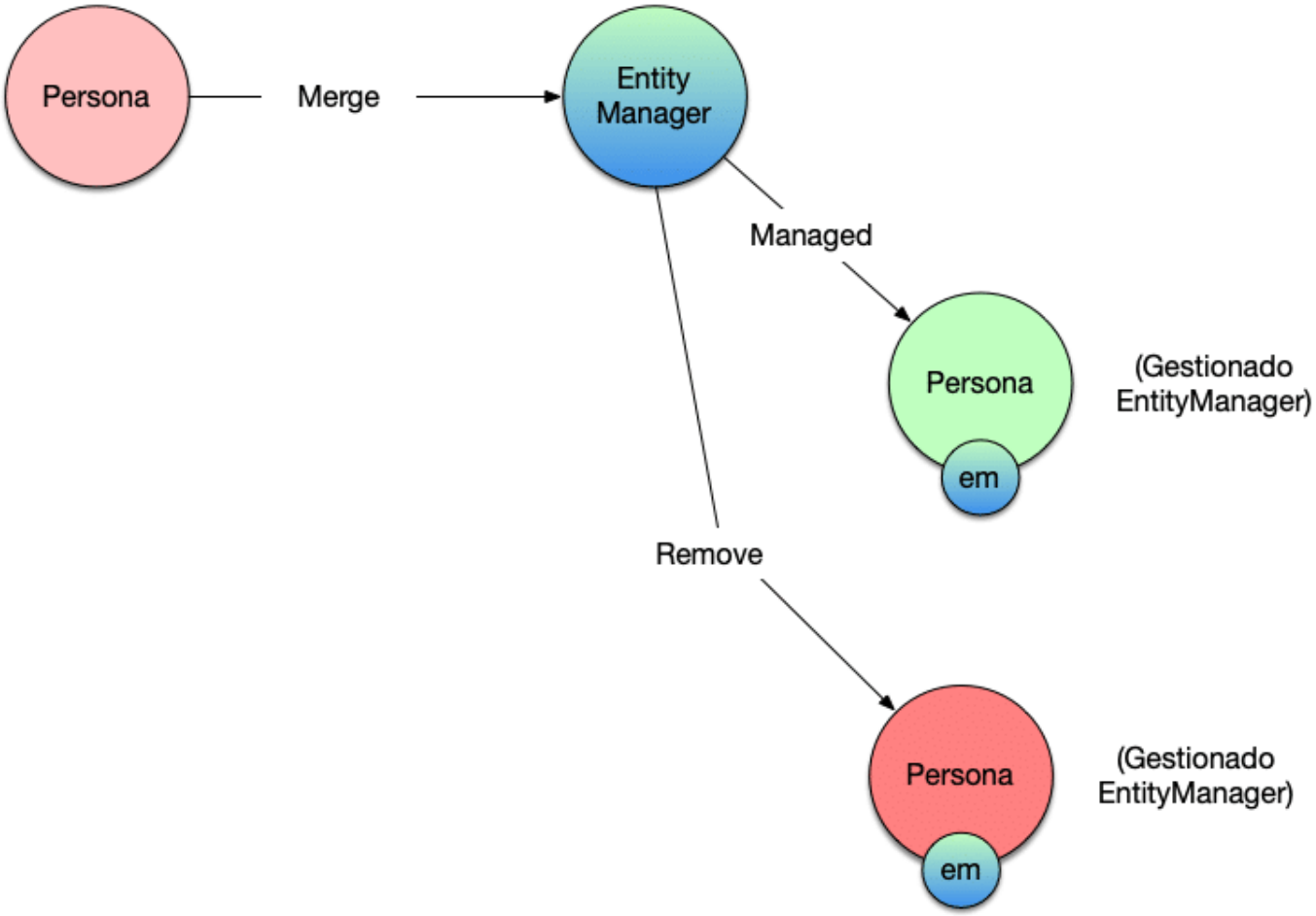
}
```

Como se puede observar primero se ejecuta JPA Merge y una vez que la entidad esta bajo el control del EntityManager se procede a removerla. Si vemos los mensajes de la consola nos quedará algo más claro.

```
Hibernate: select persona0_.nombre as nombre1_0_0_,
persona0_.apellidos as apellido2_0_0_, persona0_.edad as edad3_0_0_
from Persona persona0_ where persona0_.nombre=?

Hibernate: delete from Persona where nombre=?
```

Hibernate primero selecciona la Persona de la base de datos , convierte la Persona en Managed y luego procede a borrarla.



Tengamos siempre en cuenta este funcionamiento para gestionar el código con JPA

Otros artículos relacionados

1. EntityManager, EntityManagerFactory y Singletons
2. JPA Object States y como usarlos
3. JPA @Basic , optimizando los fechings
4. EntityManager
5. Ejemplo JPA