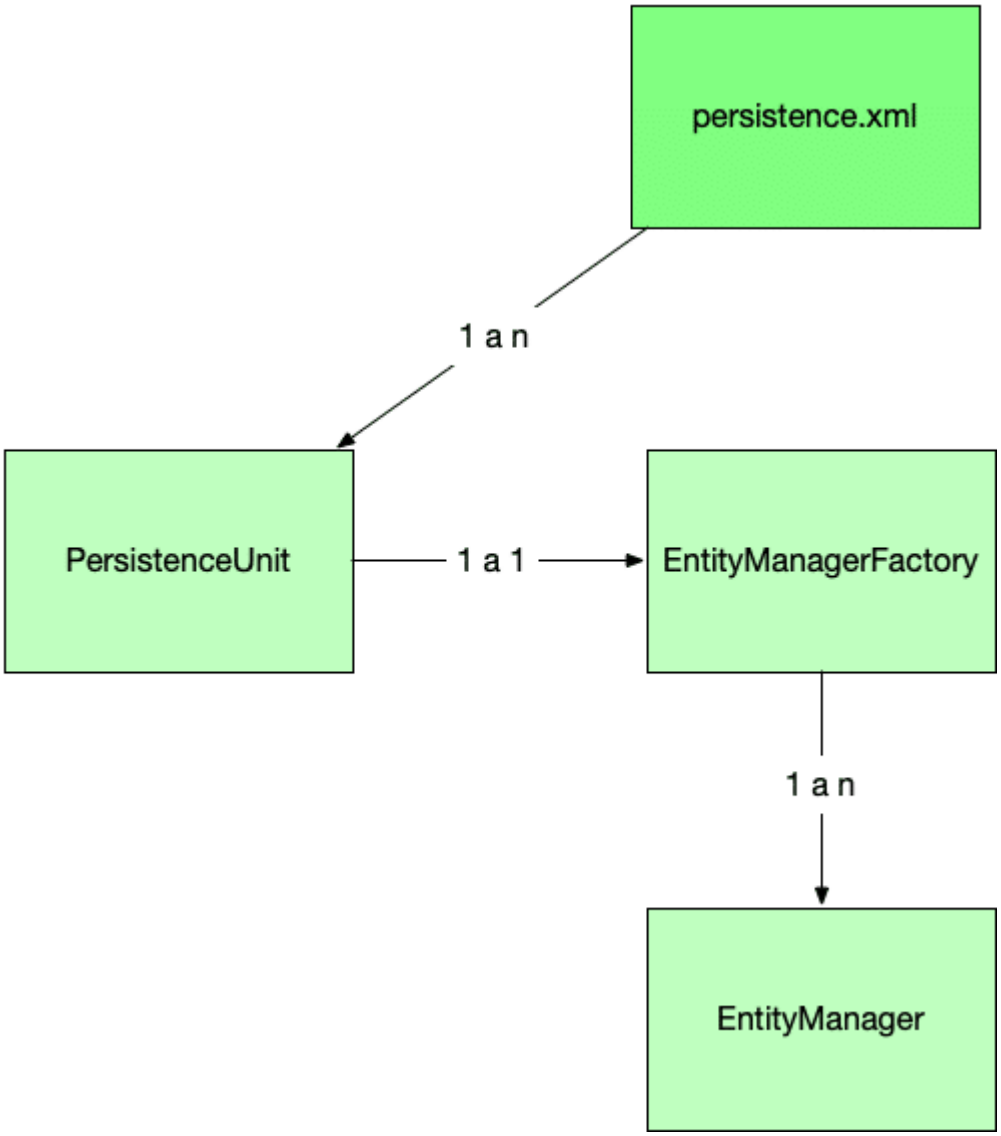
Tabla de Contenidos

Ejemplo de JPA o Java Persistence API . JPA es el standard de Java encargado de automatizar dentro de lo posible la persistencia de nuestros objetos en base de datos .Sin embargo incluso a nivel básico genera dudas a los desarrolladores . Así pues vamos a dedicar algunas entradas del blog a hablar de los conceptos mas importantes.Para ello nos apoyaremos en el siguiente diagrama UML.



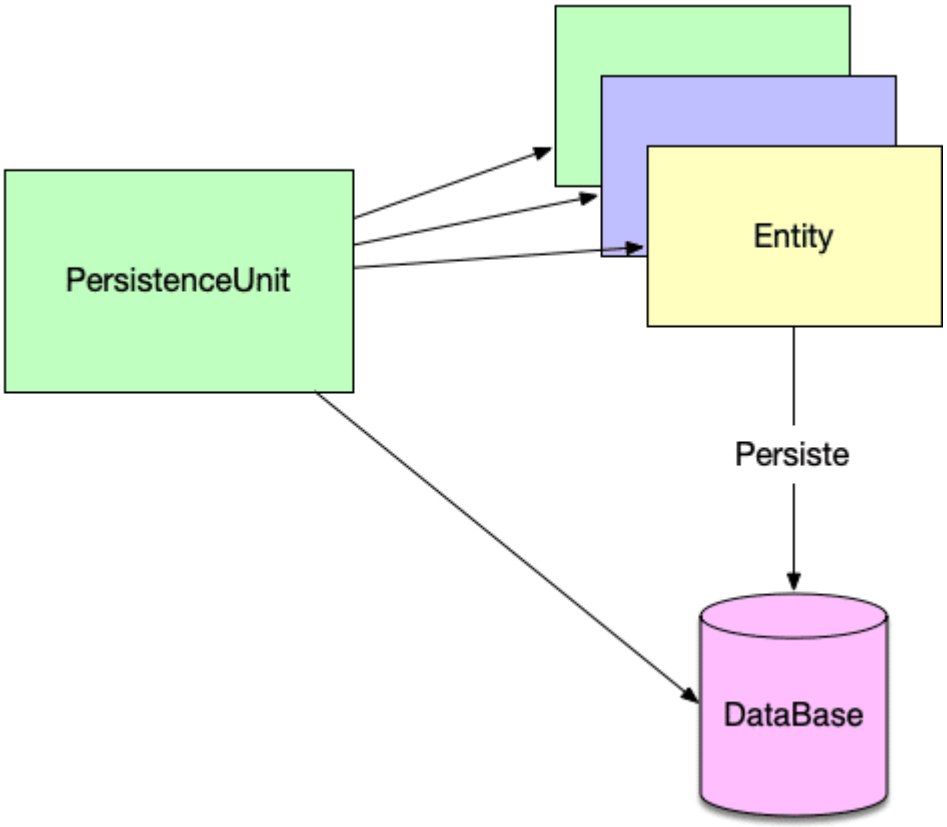
Persistence.xml

El primer concepto del que vamos a hablar es del fichero persistence.xml que se encuentra ubicado en la carpeta META-INF de un proyecto Java EE clásico . Este fichero se encarga de conectarnos a la base de datos y define el conjunto de entidades que vamos a gestionar. Es un fichero que pertenece a los estandares y por lo tanto dará lo mismo que framework usemos o como lo manejemos que siempre podremos recurrir a él o a una configuración muy similar.

```
<persistence xmlns="http://java.sun.com/xml/ns/persistence"
```

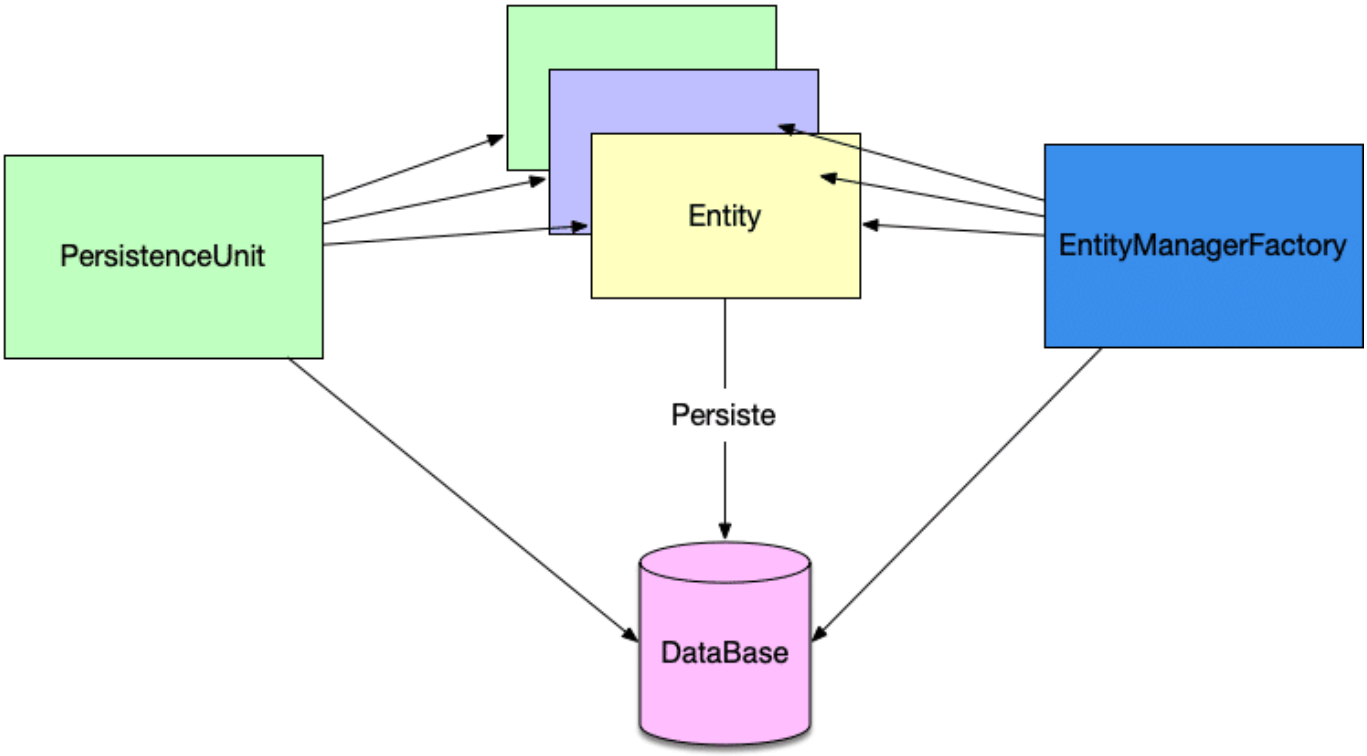
```
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="http://java.sun.com/xml/ns/persistence
 http://java.sun.com/xml/ns/persistence/persistence_2_0.xsd"
 version="2.0">
 <persistence-unit name="UnidadPersonas">
 <class>es.curso.bo.Persona</class>
 <properties>
 <property name= "hibernate.show_sql" value="true" />
 <property name="hibernate.dialect"
value="org.hibernate.dialect.MySQLDialect" />
 <property name="javax.persistence.jdbc.driver"
value="com.mysql.jdbc.Driver" />
 <property name="javax.persistence.jdbc.user" value="root" />
 <property name="javax.persistence.jdbc.password" value="jboss" />
 <property name="javax.persistence.jdbc.url"
value="jdbc:mysql://localhost/jpa" />

</properties>

</persistence-unit>
```

En nuestro caso unicamente tenemos una entidad «Persona» y luego la parte que se encarga de definir el acceso a la base de datos generando un pool de conexiones etc.

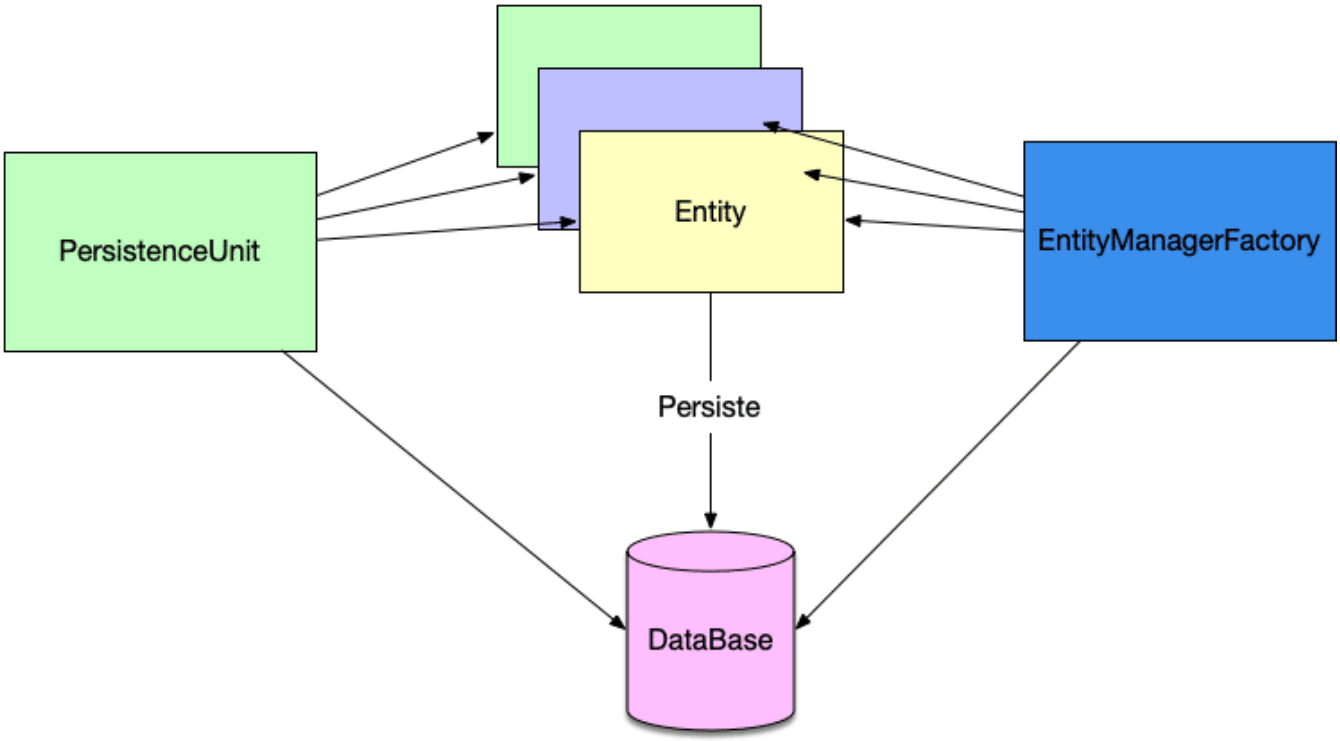


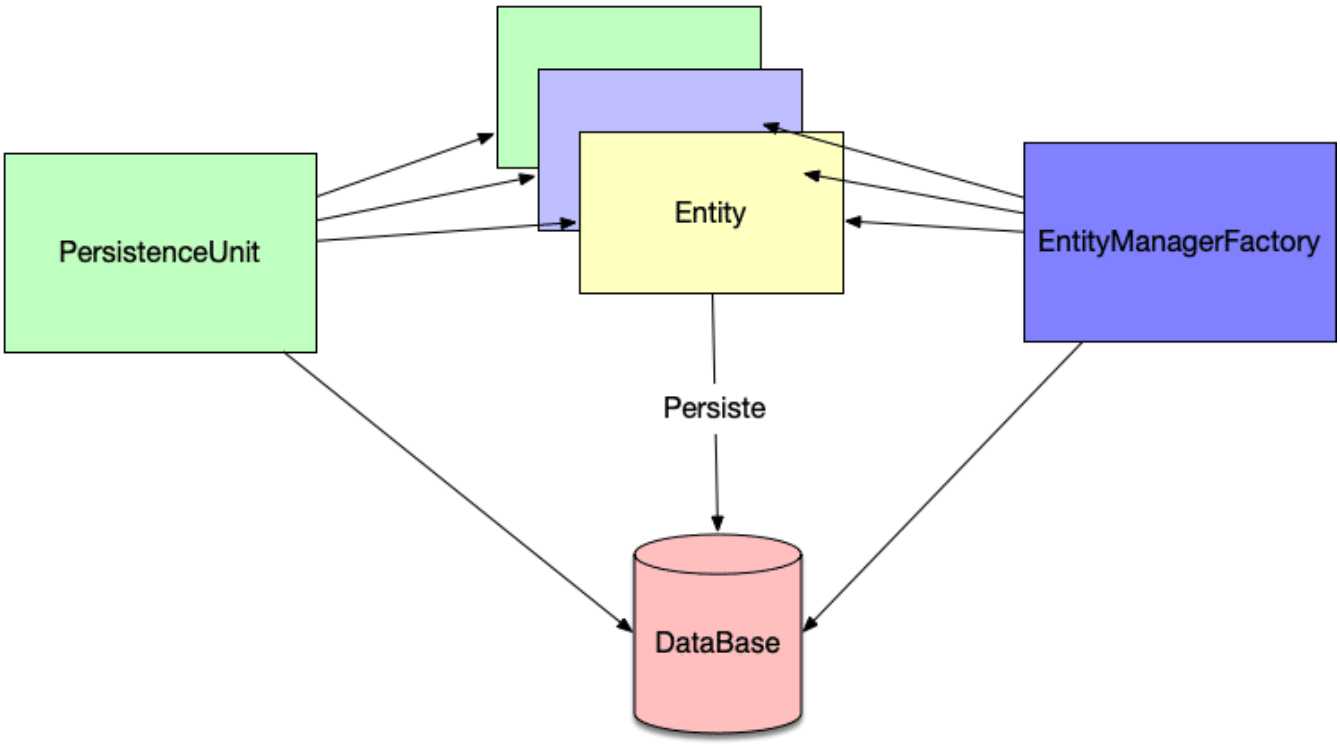
EntityManagerFactory

El persistence.xml define la conectividad a la base de datos y las entidades que vamos a considerar persistibles en ella. Con ambos conceptos se genera un objeto EntityManagerFactory que se encargará de gestionar todas estas entidades para la base de datos.



De esta forma tendremos a nuestra disposición un EntityManagerFactory con el que empezar a gestionar las entidades que se encuentran definidas a nivel del fichero persistence.xml. Ahora bien muchas aplicaciones JEE se conectan a varias bases de datos y generan distintos EntityManagerFactorys cada uno asociados a diferente Persistence Unit


PersistenceUnit
Entity
EntityManagerFactory
Persiste
DataBase

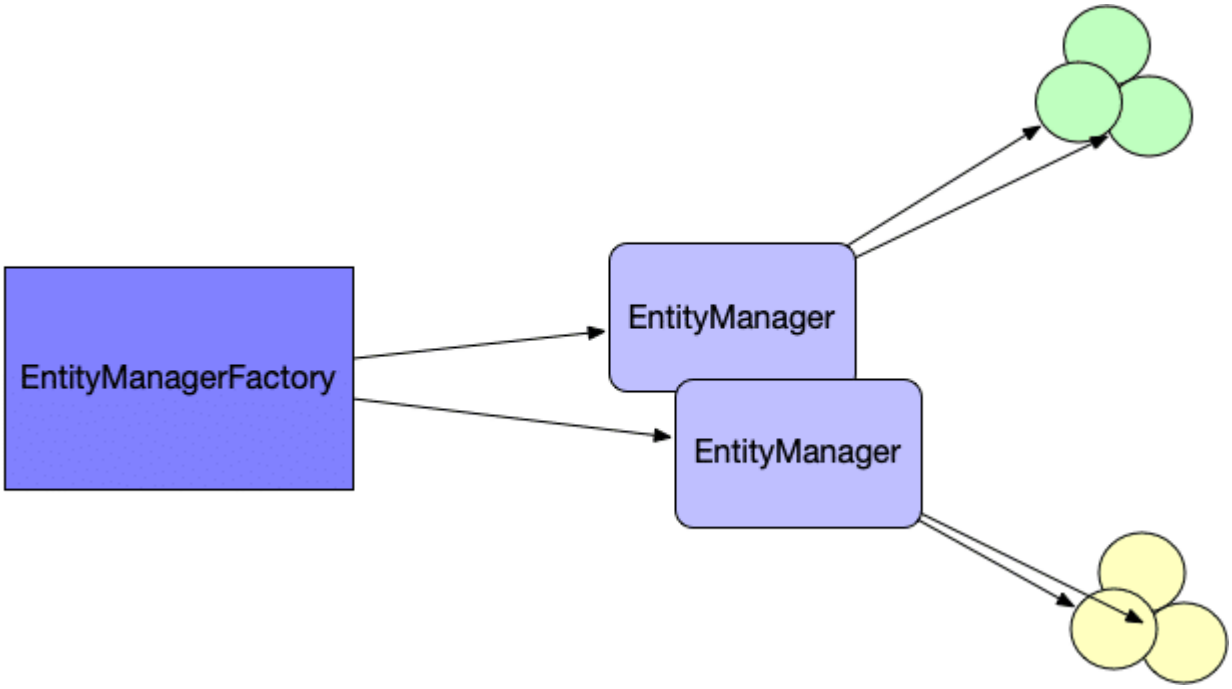

PersistenceUnit
Entity
EntityManagerFactory
Persiste
DataBase

Ejemplo de JPA y EntityManagerFactory

Lo habitual es disponer de un único EntityManagerFactory con el que nosotros gestionamos todas las entidades. De esta forma queda mas claras las diferencias entre persistence.xml, EntityManagerFactory y PersistenceUnit.

EntityManager

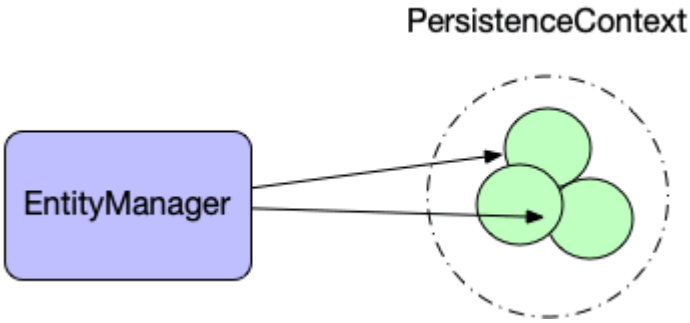
Una vez disponemos de un EntityManagerFactory este será capaz de construir un objeto de tipo EntityManager que como su nombre indica gestiona un conjunto de entitades o objetos.



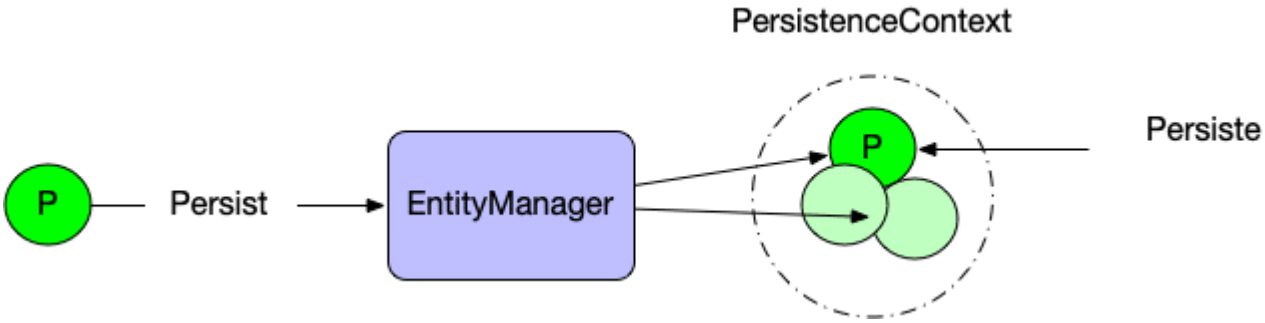
En principio estas entidades son objetos POJO (Plain Old Java Object) normales con los cuales estamos trabajando en nuestro programa Java .El EntityManager será el encargado de realizar todas las operaciones CRUD sobre estos objetos como es insertar , borrar ,seleccionar y actualizar.

PersistenceContext

Para ello se define otro concepto adicional denominado «PersistenceContext» . Este concepto hace referencia a los objetos que han sido manipulados por el EntityManager y se encuentran bajo su control



Para conseguir que alguno de nuestros objetos pase a ubicarse dentro del PersistenceContext bastará con invocar a alguno de los métodos típicos del EntityManager como persist , merge , find etc.



Ejemplo de JPA

Una vez un objeto se encuentra dentro del PersistenceContext el EntityManager será capaz de controlar todos los cambios que se han realizado en él y ejecutar las consultas adecuadas contra la base de datos. A continuación se muestra un ejemplo de JPA de hola mundo que persiste un objeto en la base de datos,

```
package com.arquitecturajava;
```

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

import es.curso.bo.Persona;

public class Principal01Add {

    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();

        }
    }
}
```

En este ejemplo hemos usado el método persist() el EntityManager para almacenar la información en base de datos. Hemos tenido ademas que gestionar esta persistencia bajo en entorno transaccional . De tal forma que cuando se ejecute el método merge automáticamente pasemos a realizar un commit (confirmación de la transacción) y los datos

serán salvamos en la base de datos.

Ejemplo de JPA (Video)

Vamos a ver ahora un ejemplo complementario con la clase Libro en mi curso gratuito de introducción a JPA que nos sirva de complemento y apoyo al código que acabamos de ver:

Otros artículos relacionados

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

Cursos gratuitos relacionados

1. Introducción a JPA