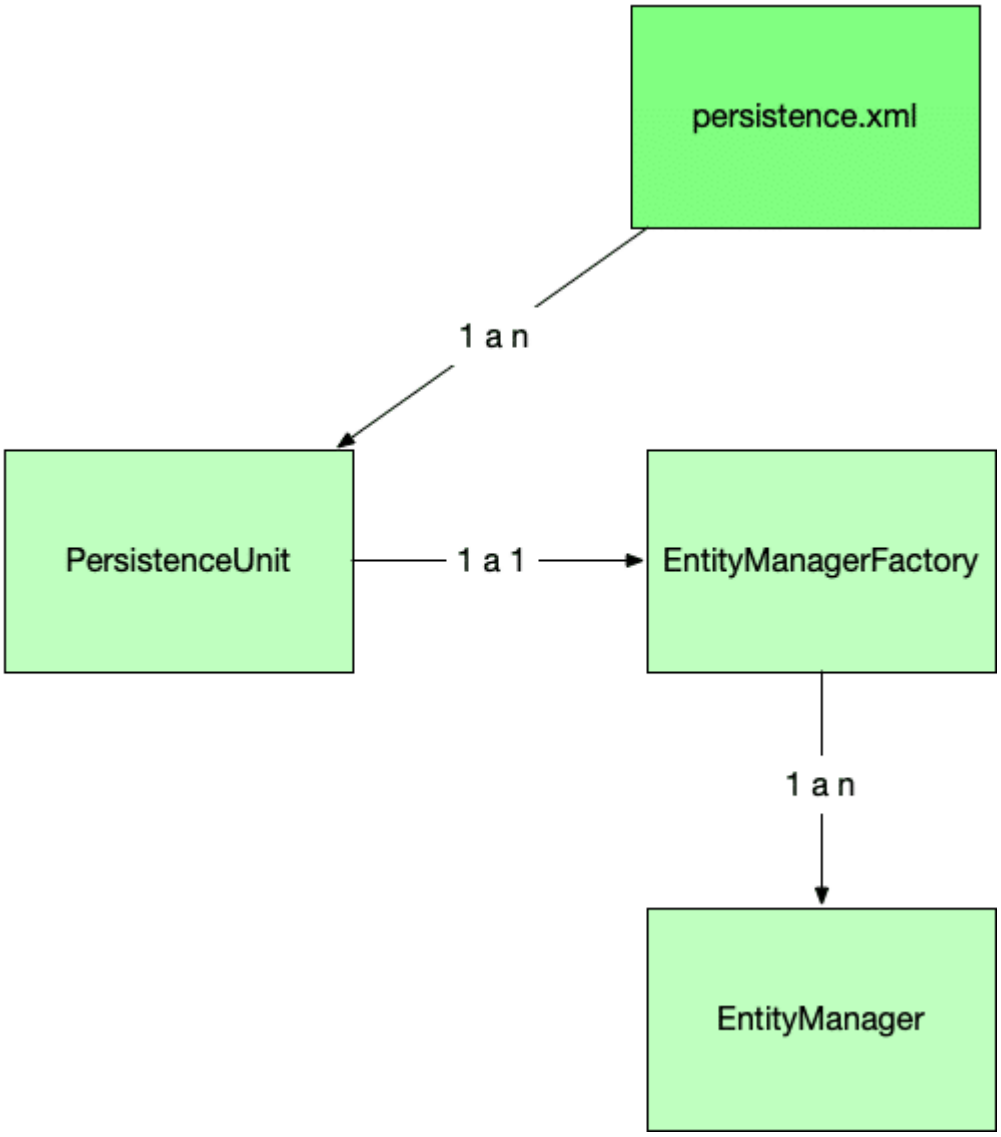
Ejemplo de JPA y cómo usar Java Persistence API para persistir objetos en nuestra base de datos . Al comienzo esto siempre genera dudas muchas dudas . Por lo tanto vamos a construir una serie de diagramas que nos ayuden a entenderlo mejor.



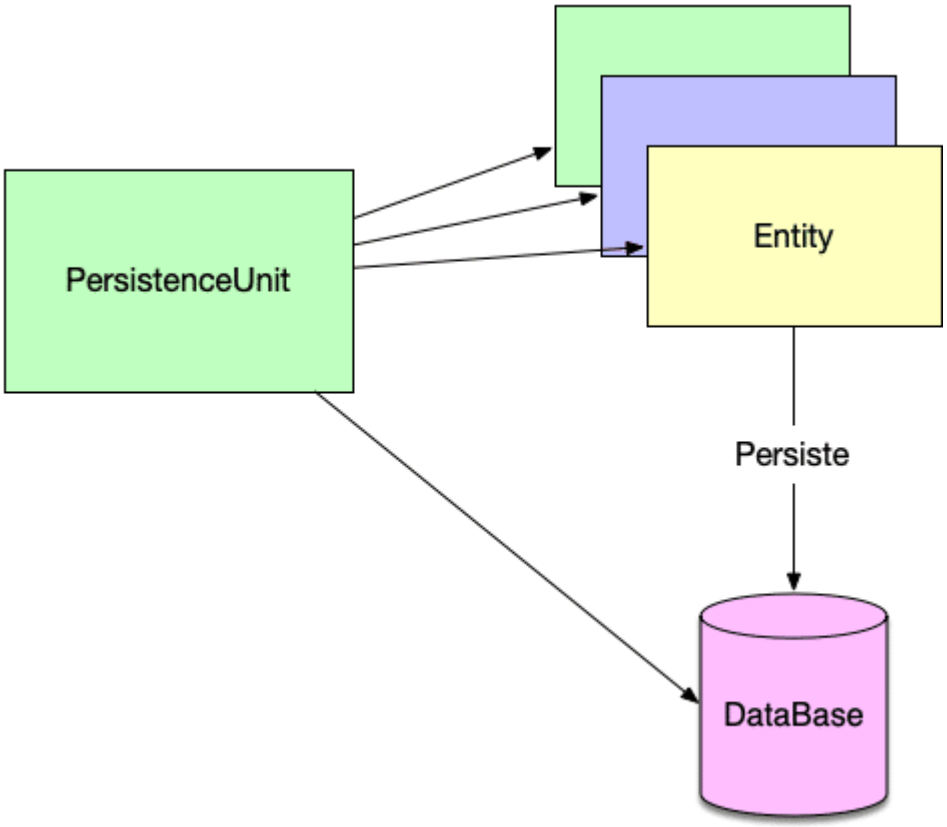
Java Persistence API y Persistence.xml

En primer lugar vamos a hablar es del fichero persistence.xml que se encuentra ubicado en la carpeta META-INF de un proyecto Java EE . Puesto que este fichero se encarga de conectarnos a la base de datos y define el conjunto de entidades que vamos a gestionar. Por suerte el fichero es parte del standard y existirá en cualquier implementación de JPA que se utilice.

```
<persistence xmlns="http://java.sun.com/xml/ns/persistence"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
```

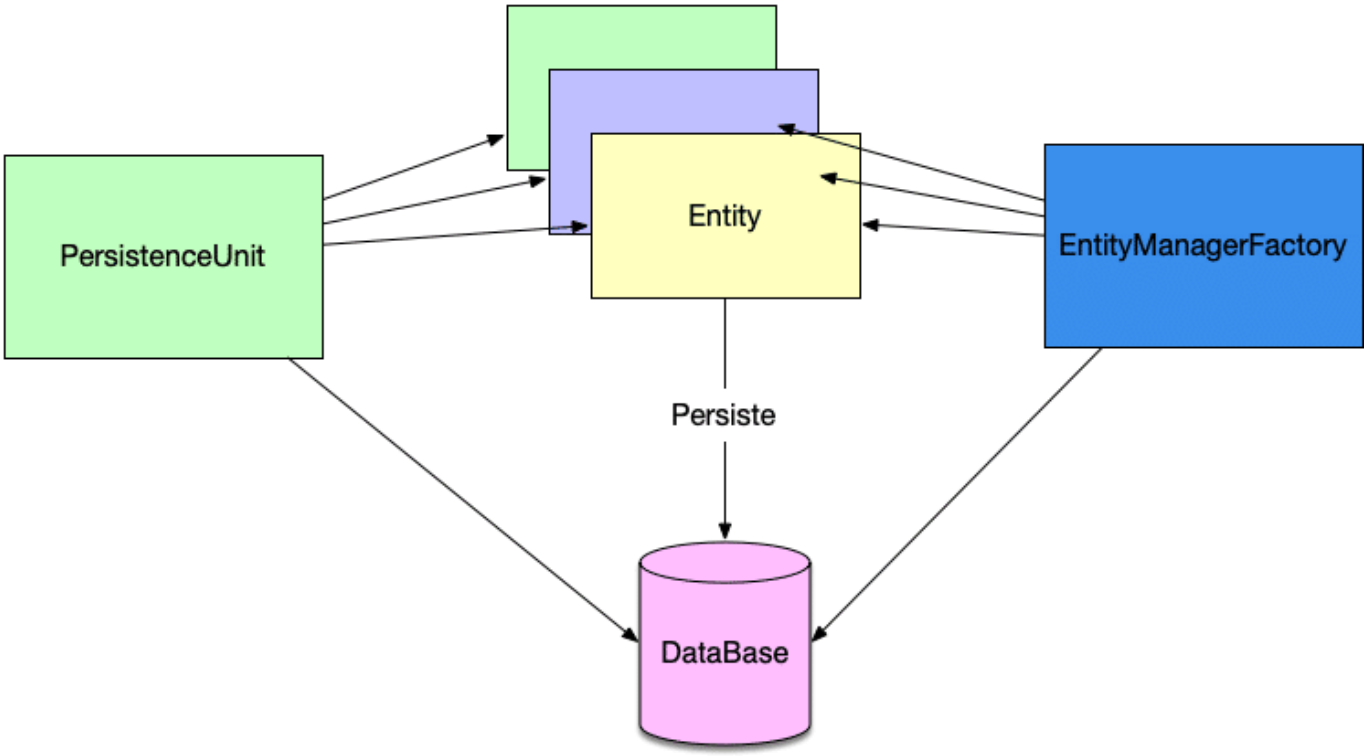
```
 xsi:schemaLocation="http://java.sun.com/xml/ns/persistence
 http://java.sun.com/xml/ns/persistence/persistence_2_0.xsd"
 version="2.0">
 <persistence-unit name="UnidadPersonas">
 <class>es.curso.bo.Persona</class>
 <properties>
 <property name= "hibernate.show_sql" value="true" />
 <property name="hibernate.dialect"
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>
```

Ademas en el documento XML podemos ver tanto nuestra clase de persistencia (Persona) como las propiedades de conexión a la base de datos.

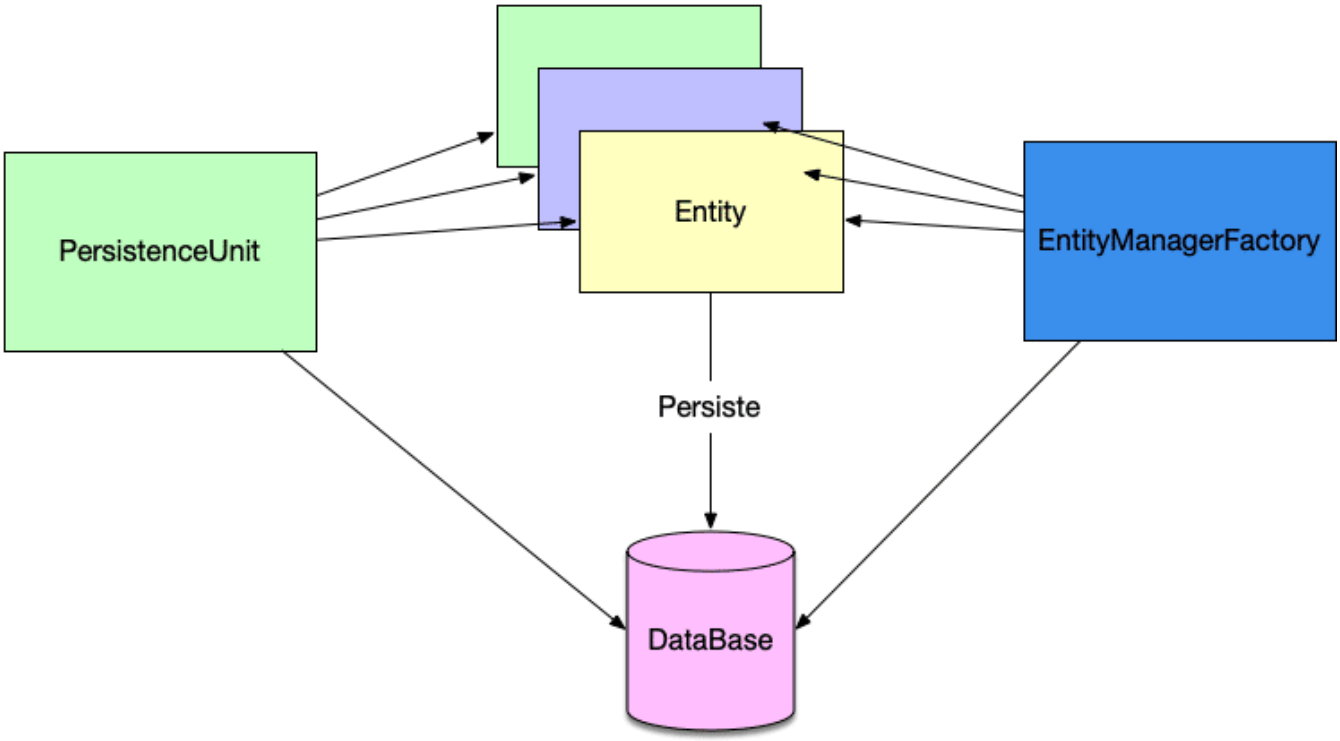


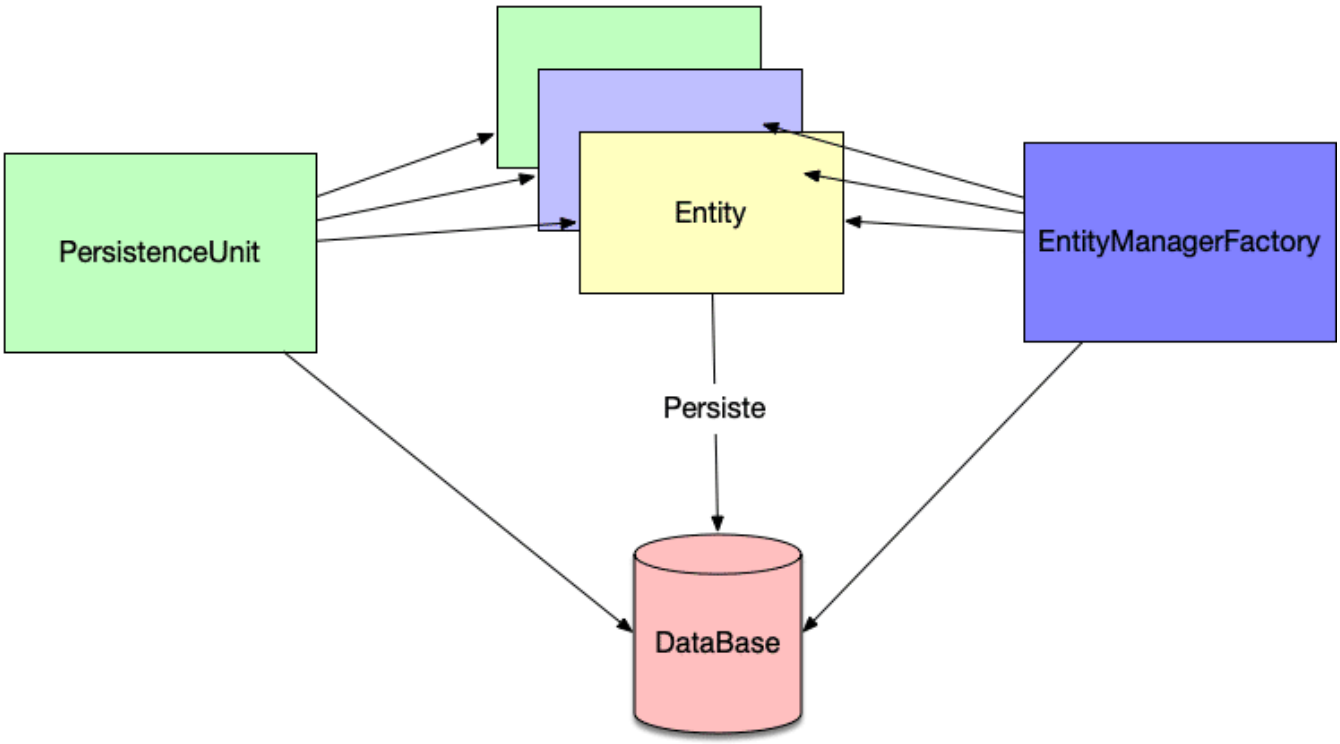
JPA y EntityManagerFactory

Por lo tanto hay que tener en cuenta que en Java Persistence API el fichero persistence.xml define la conectividad a la base de datos y las entidades que vamos a considerar persistibles en ella. Como resultado de 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. Eso si, muchas aplicaciones JEE conectan a varias bases de datos y tendrán diferentes EntityManagerFactorys . Cada uno estará ligado un PersistenceUnit diferente.


PersistenceUnit
Entity
EntityManagerFactory
Persiste
DataBase

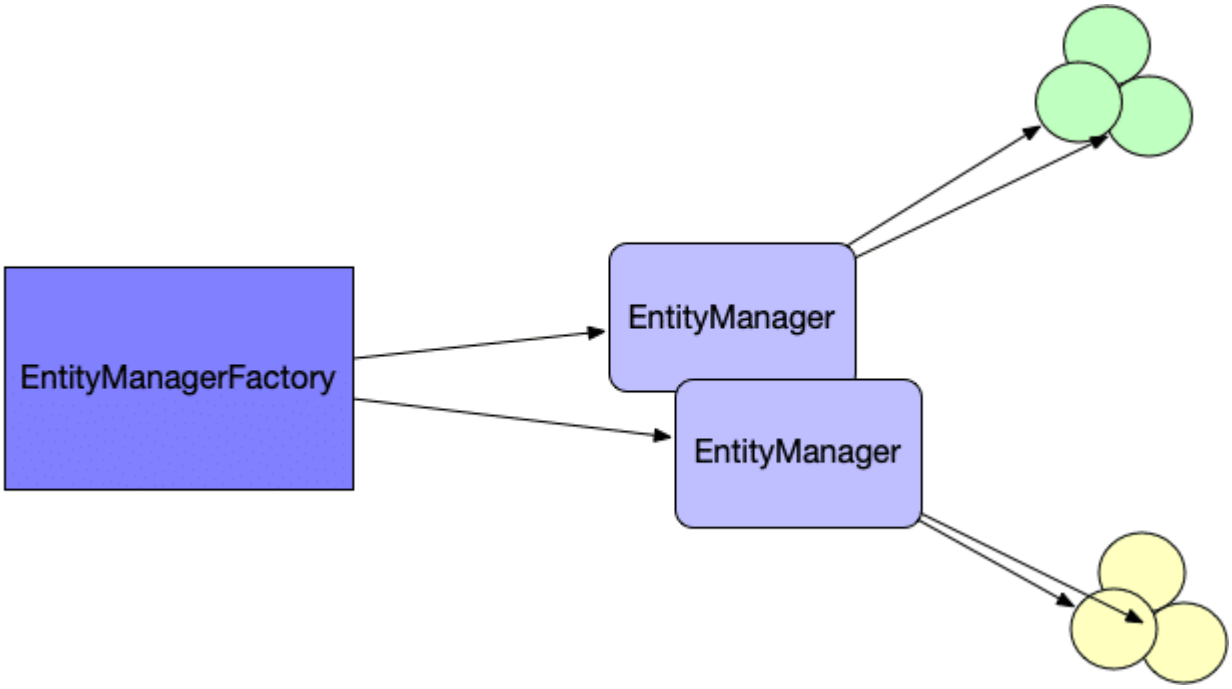

PersistenceUnit
Entity
EntityManagerFactory
Persiste
DataBase

Java Persistence API y EntityManagerFactory

Lo habitual es disponer de un único EntityManagerFactory con el que nosotros gestionamos todas las entidades. Por lo tanto cada fichero Persistence.xml, EntityManagerFactory y PersistenceUnit tiene sus propias responsabilidades.

Manejo del EntityManager

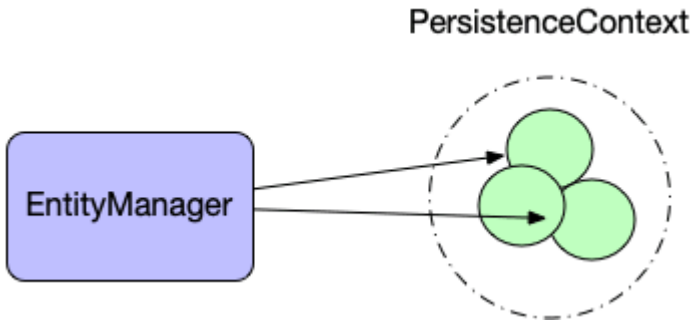
Una vez disponemos de un EntityManagerFactory , este será capaz de construir un objeto EntityManager que automatizará la persistencia de los objetos.



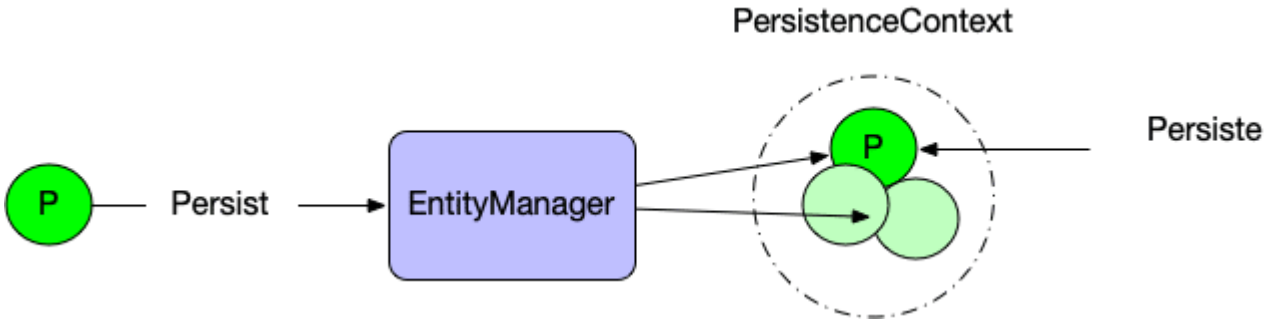
Hay que recordar que estas entidades son objetos POJO (Plain Old Java Object) con los que estamos trabajando en nuestro programa . Por consiguiente , el EntityManager será el encargado de realizar todas las operaciones de tipo CRUD (insertar , borrar ,seleccionar y actualizar etc) sobre ellos.

JPA y PersistenceContext

¿Qué es un PersistenceContext? . En primer lugar hay que tener en cuenta que un EntityManager persistirá un “conjunto de objetos” ¿Pero qué objetos? . Claro esta que aquellos que hayan sufrido modificaciones a nivel de sus propiedades y no estén sincronizados . Esto es a lo que comunmente se le denomina PersistenceContext.



Por ello para conseguir que alguno de nuestros objetos pase a ubicarse dentro del PersistenceContext bastará con invocar los métodos persist , merge , sobre él.



Usando Java Persistence API

Una vez un objeto se encuentra dentro del PersistenceContext el EntityManager este será capaz de controlar todos los cambios que se han realizado en él y ejecutar las consultas adecuadas contra 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 confirmar de la transacción) . Los datos serán salvamos en la base de datos.

Ejemplo de Java Persistence API (Video)

Finalmente vamos a ver un ejemplo complementario en video .Para ello usaremos 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. EntityManager métodos
3. Relaciones OneToMany
4. Usando @ ManyToOne
5. Persistencia y Merge
6. JPA Remove y objetos Gestionados

Cursos gratuitos relacionados

1. Introducción a JPA