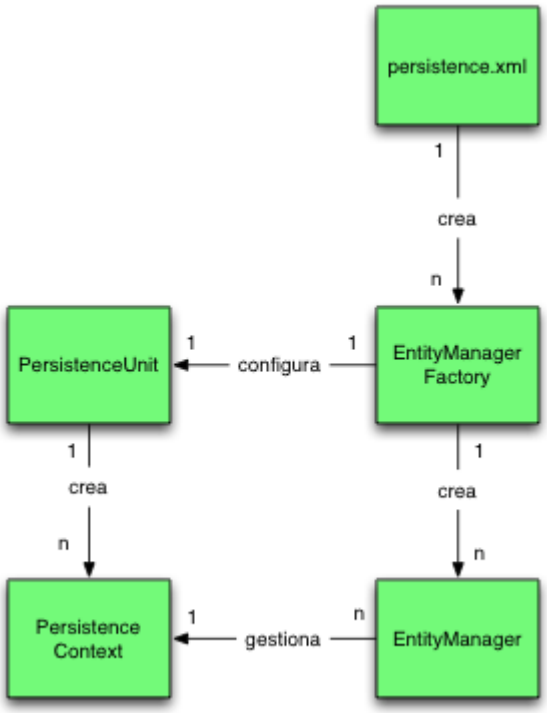
Tabla de Contenidos

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.



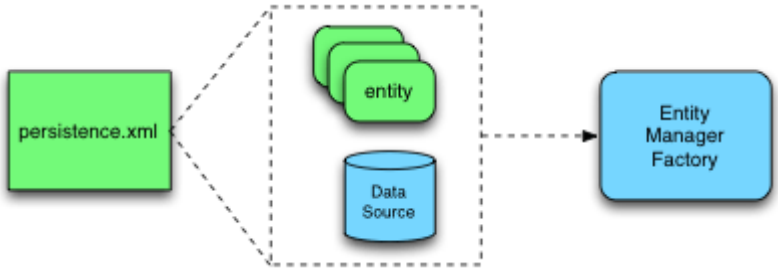
Persistence.xml

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.

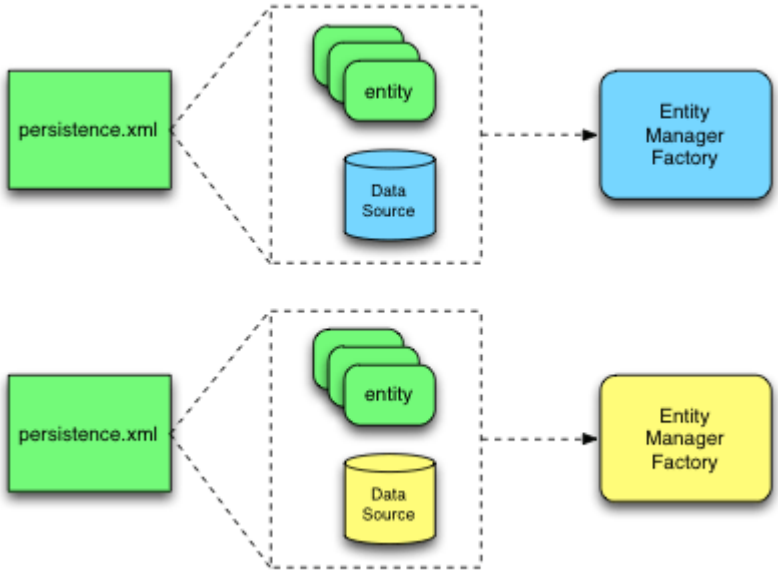
```
<persistence xmlns="http://java.sun.com/xml/ns/persistence"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="http://java.sun.com/xml/ns/persistence
 http://java.sun.com/xml/ns/persistence/persistence_2_0.xsd"
 version="2.0">
 <persistence-unit name="UnidadPersonas">
 <class>es.curso.bo.Persona</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.

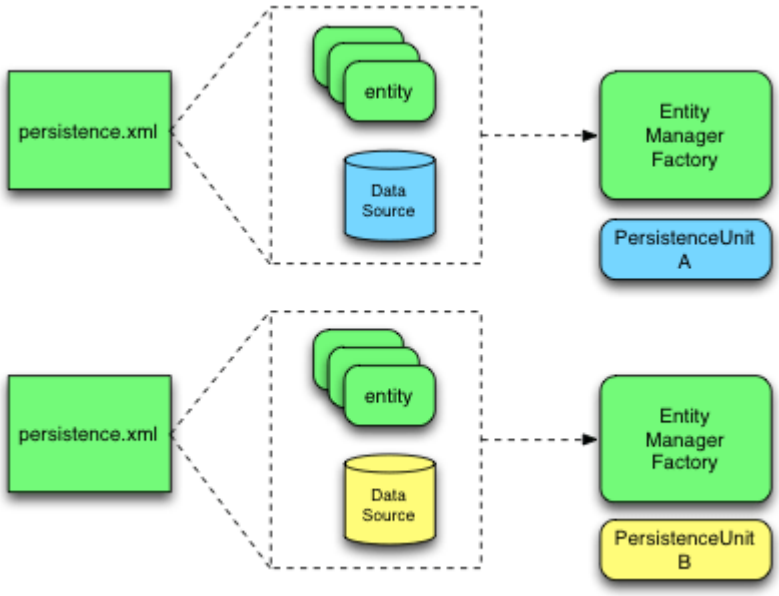


De esta forma tendremos a nuestra disposición un EntityManagerFactory con el que empezar a gestionar las entidades que se encuentran definidas a nivel del fichero persistence.xml. Ahora bien muchas aplicaciones JEE se conectan a varias bases de datos y generan distintos EntityManagerFactorys



EntityManagerFactory

En un primer lugar un EntityManagerFactory es único y es con el que nosotros gestionamos todas las entidades. Ahora bien si tenemos varias conexiones a base de datos deberemos definir un nuevo concepto que nos permite clarificar que tenemos dos EntityManagerFactories distintos. Este concepto es el que se conoce como PersistenceUnit o unidad de persistencia. Cada PersistenceUnit tiene asociado un EntityManagerFactory diferente que gestiona un conjunto de entidades distinto.

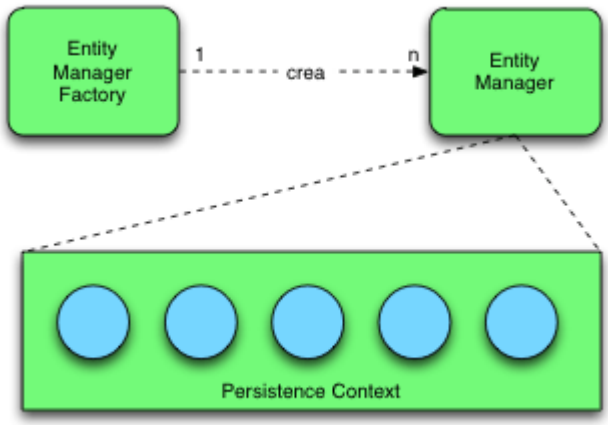


De esta forma queda mas claras las diferencias entre persistence.xml, EntityManagerFactory y PersistenceUnit.

EntityManager

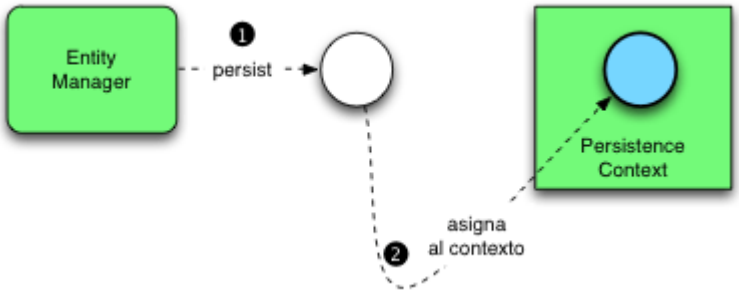
Una vez disponemos de un EntityManagerFactory este será capaz de construir un objeto de tipo EntityManager que como su nombre indica gestiona un conjunto de entitades o objetos.

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 salvarlos a la base de datos , eliminarlos de la base de datos etc . Para ello define otro concepto adicional “PersistenceContext” . Este concepto hace referencia a los objetos que han sido manipulados por el EntityManager y se encuentran controlados por él.



PersistenceContext

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 etc.



.

Ejemplo de JPA Conclusiones

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.

Cursos Asociados

- Programación Orientada a Objeto
- Java APIs

- Desarrollo Web Java
- Java 8 Stream y Lambdas

```
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.

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