UTN.BA
UNIVERSIDAD TECNOLÓGICA NACIONAL
FACULTAD REGIONAL BUENOS AIRES

Persistencia en Java

JPA: Java Persistence API

UTN DDS 2021

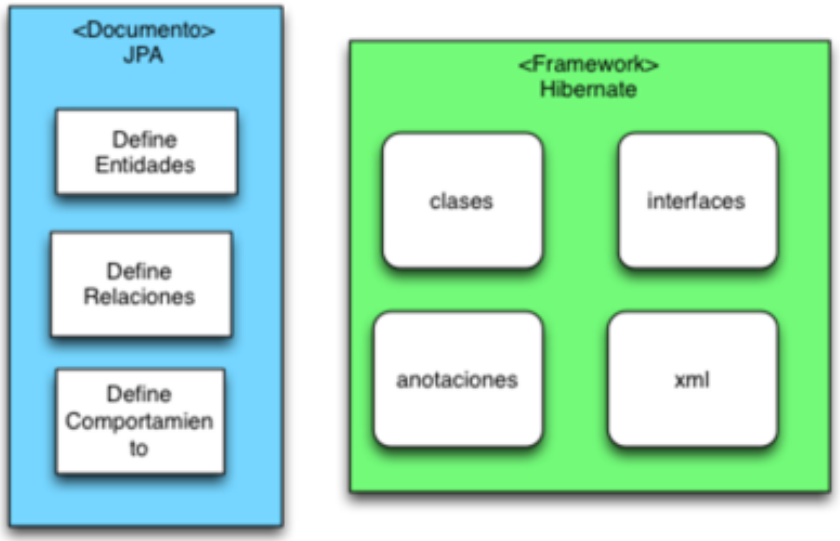
Temas

1. Qué es qué
2. JPA
3. Hibernate
4. Ejemplo

Qué es qué

JPA: Es la especificación de Java (*API*) para acceder a la persistencia de objetos

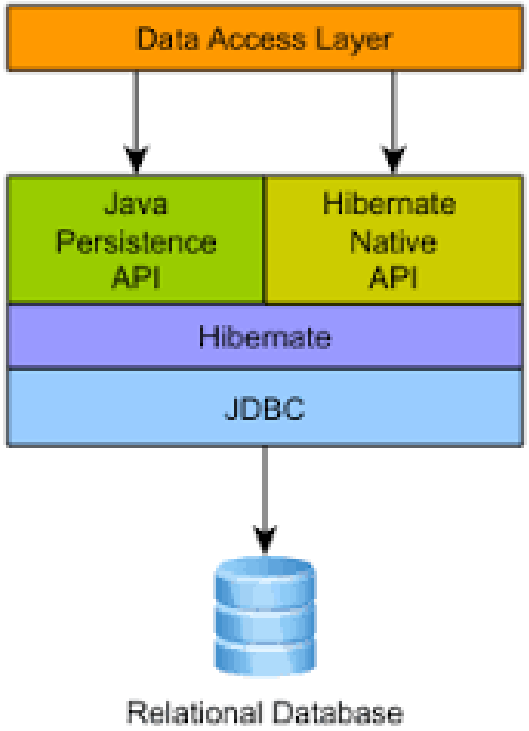
Hibernate: Es la implementación de JPA más usada. Hay otras como Eclipselink o Toplink...



Qué es qué

JDBC: Es el conector entre una aplicación Java y un motor de BD particular (MySQL, Oracle, SqlServer, etc).

Hibernate genera las queries por nosotros y se comunica con el driver JDBC (una librería), que recibe queries como String y las traslada a la BD.



Diseño de JPA

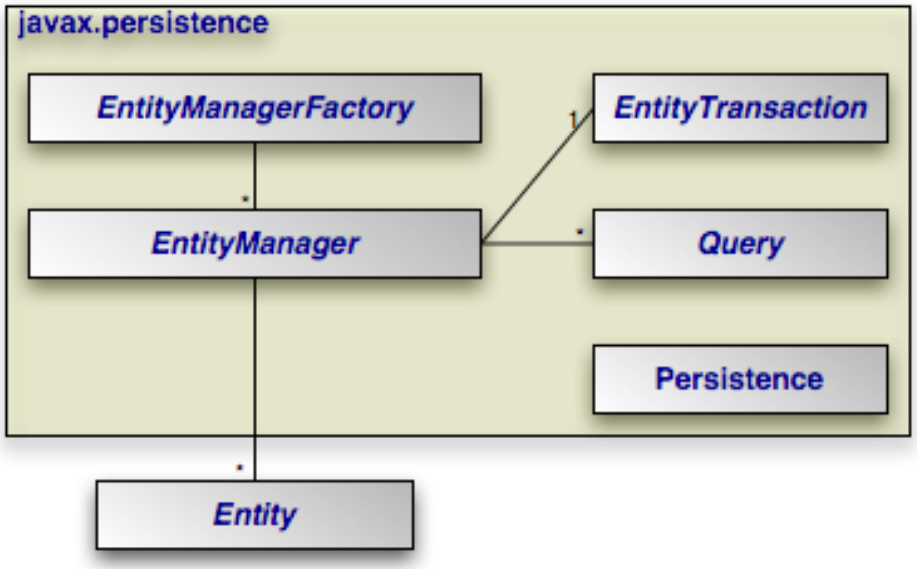
Entidades: Son los objetos de nuestro negocio que se persisten.

EntityManager: Interfaz que gestiona la persistencia de entidades (objetos) en el contexto.

EntityManagerFactory: Crea EntityManagers.

Persistence: Esta clase contiene métodos para obtener un EntityManagerFactory.

Query: Esta interfaz se usa para obtener objetos que cumplan ciertos criterios más complejos.



persistence.xml

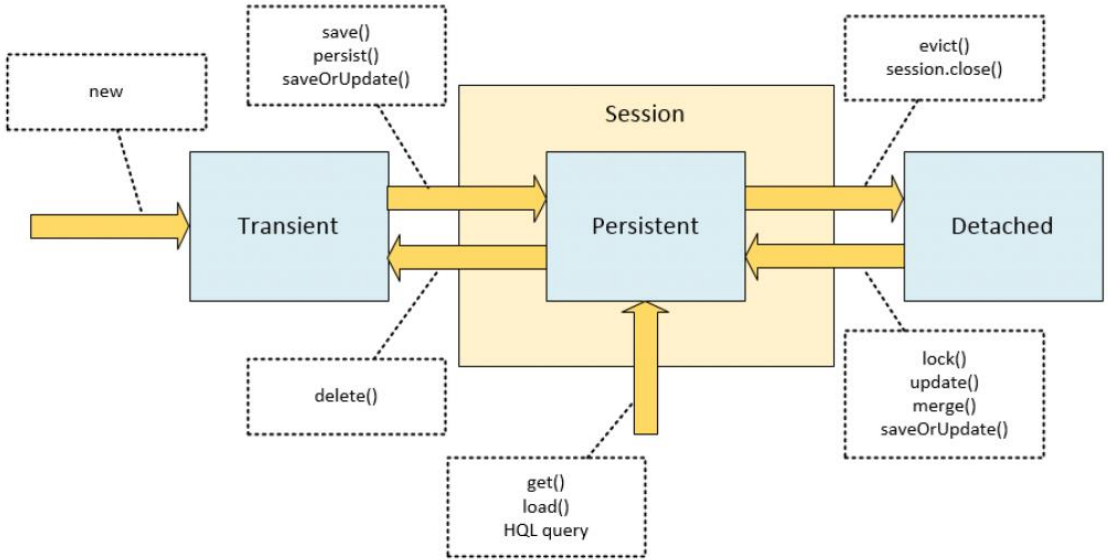
```
<persistence xmlns="http://java.sun.com/xml/ns/persistence"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://java.sun.com/xml/ns/persistence
http://java.sun.com/xml/ns/persistence/persistence_2_0.xsd"
version="2.0">
  <persistence-unit name="UnidadPersonas">
   <class>com.example.dominio.Persona</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="mypass" />
     <property name="javax.persistence.jdbc.url" value="jdbc:mysql://localhost/jpa" />
   </properties>

  </persistence-unit>
</persistence>
```

Hibernate: Estados de una entidad

Las **entidades son clases POJO** (sólo atributos con getters y setters). Al crear una instancia, Hibernate no la conoce y se dice que está en estado Transient. Cuando la **persistimos** o la **obtenemos** con el EntityManager, pasa a estar **administrada por Hibernate**.

¿Qué significa *administrada*?

Que **mantiene una relación entre el objeto en memoria y el registro** en la BD. Cada vez que lo modifiquemos se va a disparar una query.



Ejemplo: Proyecto

https://github.com/jonybuzz/demo-hibernate

1. Agregar en pom.xml las dependencias de hibernate y el Driver
2. Configurar conexión en persistence.xml
3. Agregar anotaciones de JPA en las entidades

pom.xml

```
<dependency>
    <groupId>mysql</groupId>
    <artifactId>mysql-connector-java</artifactId>
    <version>8.0.25</version>
</dependency>
<dependency>
    <groupId>org.hibernate</groupId>
    <artifactId>hibernate-entitymanager</artifactId>
    <version>5.4.18.Final</version>
</dependency>
```

Ejemplo: Entidad

@Entity: Obligatorio

@Table: Opcional

@Id @GeneratedValue: Obligatorio. Indica cuál es la PK y que debe autogenerarse cuando insertamos

@ManyToOne: Indica que la relación es muchos a uno respecto de Curso. Puede ir acompañado por **@JoinColumn** para indicar la FK

@ManyToMany: Indica que la relación es muchos. Puede ir con **@JoinTable** para especificar la tabla intermedia.

@OneToMany: Indica que la relación es uno a muchos respecto de Curso. Debe haber un ManyToOne del otro lado.

Profesor y **Alumno** también deben ser entidades.

```
@Entity
@Table(name = "curso")
public class Curso {

    @Id
    @GeneratedValue
    private Long codigo;

    private String nombre;

    @ManyToOne
    @JoinColumn(name = "id_profesor")
    private Profesor profesor;

    @OneToMany(mappedBy = "curso")
    private List<Alumno> alumnos;

    // getters, setters
}
```

Ejemplo: Ejecución

EntityManagerFactory: Debe crearse una sola vez durante la vida de la aplicación.

EntityManager: Puede crearse una sola vez o cada vez que realizo una acción (request, schedule)

EntityTransaction: Debe abrirse una transacción cada vez que se quieren ejecutar un conjunto de operaciones atómicas contra la BD. La transacción puede confirmarse (*commit*) o revertirse (*rollback*).

```
EntityManagerFactory factory =
Persistence.createEntityManagerFactory
("demo-hibernate-PU");

EntityManager em =
factory.createEntityManager();

EntityTransaction tx =
em.getTransaction();

tx.begin();
tx.commit();
tx.rollback();
```

Links

¿Qué es JPA?: http://oraclejuniors.blogspot.com/2014/11/que-es-jpa-java-persistence-api.html

Interfaces principales de JPA: https://www.tutorialspoint.com/es/jpa/jpa_architecture.htm

Hibernate: https://www.baeldung.com/hibernate-save-persist-update-merge-saveorupdate

Más sobre relaciones OneToMany y ManyToOne: https://thoughts-on-java.org/best-practices-many-one-one-many-associations-mappings/

Proyecto de ejemplo: https://github.com/jonybuzz/demo-hibernate

Herencia: https://www.baeldung.com/hibernate-inheritance