JPA

📘 ¿Qué es JPA?

JPA (Java Persistence API) es una **especificación de Java** para el mapeo objeto-relacional (ORM). Permite gestionar la persistencia de datos en una base de datos relacional a través de clases Java.

🔧 Tecnologías relacionadas

- **Hibernate**: Implementación más popular de JPA.
- **EclipseLink**: Implementación oficial de Oracle.
- **Spring Data JPA**: Extensión de Spring que facilita el uso de JPA.

🧱 Entidades

Una entidad es una clase Java anotada que se mapea a una tabla de base de datos.

```
import javax.persistence.*;
@Entity
@Table(name = "usuarios")
public class Usuario {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;
    @Column(nullable = false)
    private String nombre;
    private String email;
    // Getters y Setters
}
```

🔑 Anotaciones principales

Anotación	Descripción
`@Entity`	Marca la clase como entidad persistente
`@Table(name="...")`	Define el nombre de la tabla
`@Id`	Indica la clave primaria
`@GeneratedValue`	Estrategia para generar la clave primaria (`IDENTITY`, `SEQUENCE`, `AUTO`)

Anotación	Descripción
`@Column`	Personaliza columnas (nombre, nullable, longitud, etc.)
`@ManyToOne`, `@OneToMany`, `@OneToOne`, `@ManyToMany`	Relaciones entre entidades

🔁 Relaciones entre entidades

@ManyToOne

`@ManyToOne @JoinColumn(name = "departamento_id") private Departamento departamento;`

@OneToMany

`@OneToMany(mappedBy = "departamento") private List<Usuario> usuarios;`

@OneToOne

`@OneToOne @JoinColumn(name = "detalle_id") private DetalleUsuario detalle;`

@ManyToMany

`@ManyToMany @JoinTable( name = "usuario_rol", joinColumns = @JoinColumn(name = "usuario_id"), inverseJoinColumns = @JoinColumn(name = "rol_id")) private Set<Rol> roles;`

💾 EntityManager

```
@PersistenceContext
private EntityManager em;
public Usuario buscar(Long id) {
    return em.find(Usuario.class, id);
}
public void guardar(Usuario u) {
    em.persist(u);
}
```

🔍 Consultas JPA

JPQL (Java Persistence Query Language)

```
TypedQuery<Usuario> query = em.createQuery("SELECT u FROM Usuario u WHERE u.email = :email", Usuario.class);
```

```
query.setParameter("email", "correo@ejemplo.com");
Usuario usuario = query.getSingleResult();
```

Named Queries

`@Entity @NamedQuery(name = "Usuario.findByEmail", query = "SELECT u FROM Usuario u WHERE u.email = :email") public class Usuario { ... }`

Native Query

```
Query q = em.createNativeQuery("SELECT * FROM usuarios WHERE email = ?", Usuario.class);
q.setParameter(1, "correo@ejemplo.com");
```

📦 Repositorios con Spring Data JPA

```
public interface UsuarioRepository extends JpaRepository<Usuario, Long> {
    List<Usuario> findByNombreContaining(String nombre);
    Usuario findByEmail(String email);
}
```

📁 persistence.xml (solo en proyectos Java EE)

```
<persistence-unit name="miUnidad">
    <class>com.ejemplo.Usuario</class>
    <properties>
        <property
            name="javax.persistence.jdbc.url"
            value="jdbc:mysql://localhost/db"
        />
        <property name="javax.persistence.jdbc.user" value="root"/>
        <property
            name="javax.persistence.jdbc.driver"
            value="com.mysql.cj.jdbc.Driver"
        />
        <property name="hibernate.hbm2ddl.auto" value="update"/>
    </properties>
</persistence-unit>
```

🛠 Estados de una entidad

1. **New**: aún no está gestionada.
2. **Managed**: está siendo gestionada por el `EntityManager`.
3. **Detached**: se ha desconectado del contexto.
4. **Removed**: marcada para eliminar.