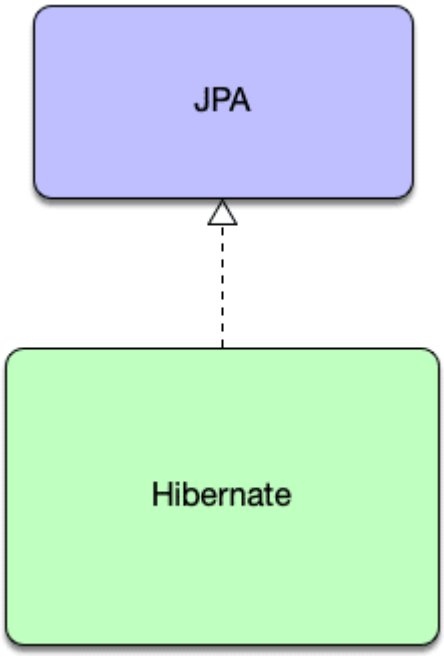
¿Qué cambios implica usar JPA y Jakarta EE? . Poco a poco Jarkarta EE va entrando entre las tecnologías que usan los proyectos.



Recordemos que en muchos casos Jakarta EE implica por ahora simplemente un cambio de nombre de paquetización . Por ejemplo cuando usamos Hibernate 6 podemos comenzar a usar las capacidades que tiene Jakarta EE a nivel de anotaciones etc . Tendremos un proyecto de Maven que use las siguientes dependencias.

```
<dependencies>

		<!--
https://mvnrepository.com/artifact/com.mysql/mysql-connector-j -->
		<dependency>
			<groupId>com.mysql</groupId>
			<artifactId>mysql-connector-j</artifactId>
			<version>8.0.32</version>
		</dependency>
```

```
		<!--
https://mvnrepository.com/artifact/org.junit.jupiter/junit-jupiter-api
-->
		<dependency>
			<groupId>org.junit.jupiter</groupId>
			<artifactId>junit-jupiter-api</artifactId>
			<version>5.9.2</version>
			<scope>test</scope>
		</dependency>
		<!--
https://mvnrepository.com/artifact/org.mockito/mockito-junit-jupiter -
->
		<dependency>
			<groupId>org.mockito</groupId>
			<artifactId>mockito-junit-jupiter</artifactId>
			<version>5.1.1</version>
			<scope>test</scope>
		</dependency>

		<!--
https://mvnrepository.com/artifact/org.mockito/mockito-core -->
		<dependency>
			<groupId>org.mockito</groupId>
			<artifactId>mockito-core</artifactId>
			<version>5.1.1</version>
			<scope>test</scope>
		</dependency>
		<!--
https://mvnrepository.com/artifact/org.hibernate.javax.persistence/hib
ernate-jpa-2.1-api -->
		<!--
```

```
https://mvnrepository.com/artifact/org.hibernate.orm/hibernate-core -
->
                <dependency>
                        <groupId>org.hibernate.orm</groupId>
                        <artifactId>hibernate-core</artifactId>
                        <version>6.1.7.Final</version>
                </dependency>

        </dependencies>
```

Estas dependencias nos permiten usar las capacidades de Hibernate 6 que implementan las anotaciones de JPA. Usando fundamentalmente la dependencia de Jakarta EE.

```
package com.arquitecturajava.jpa;

import java.util.Objects;

import jakarta.persistence.Entity;
import jakarta.persistence.Id;
import jakarta.persistence.Table;

@Entity
@Table (name="personas")
public class Persona {

        @Id
        private String nombre;
        private String apellidos;
        private int edad;
        public String getNombre() {
                return nombre;
        }
```

```
public void setNombre(String nombre) {
        this.nombre = nombre;
}
public String getApellidos() {
        return apellidos;
}
public void setApellidos(String apellidos) {
        this.apellidos = apellidos;
}
public int getEdad() {
        return edad;
}
public void setEdad(int edad) {
        this.edad = edad;
}
public Persona(String nombre, String apellidos, int edad) {
        super();
        this.nombre = nombre;
        this.apellidos = apellidos;
        this.edad = edad;
}
public Persona() {
        super();
}
@Override
public int hashCode() {
        return Objects.hash(nombre);
}
@Override
public boolean equals(Object obj) {
        if (this == obj)
```

```
                return true;
            if (obj == null)
                return false;
            if (getClass() != obj.getClass())
                return false;
            Persona other = (Persona) obj;
            return Objects.equals(nombre, other.nombre);
    }
}
```

JPA y Jakarta EE

Aquí podemos ver como aunque la clase parece que usa las anotaciones más clásicas de Java Persistence API como son @Entity @id y @Table cada una de estas anotaciones en los imports ya no aparece como javax. persistence sino que aparece como jakarta.persistence.

```
import jakarta.persistence.Entity;
import jakarta.persistence.Id;
import jakarta.persistence.Table;
```

Poco a poco con las últimas versiones de los framemworks vamos asumiendo los nuevos cambios de los standares. Si por ejemplo intentamos sacar con JPA una lista de Personas de la base de datos esta lista vendrá con una consulta muy sencilla. Tanto el EntityManager como el EntityManagerFactory pertenecen a Jakarta.

```
package com.arquitecturajava.jpa;

import java.util.List;

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

```
import jakarta.persistence.TypedQuery;

public class Principal4 {

	public static void main(String[] args) {

		EntityManagerFactory emf =
Persistence.createEntityManagerFactory("curso");

		EntityManager em = emf.createEntityManager();

		TypedQuery<Persona> consulta = em.createQuery("select
p from Persona p",Persona.class);
		List<Persona> personas= consulta.getResultList();
		for (Persona p : personas) {
			System.out.println(p.getNombre());
			System.out.println(p.getApellidos());
		}

	}

}
```

aqui podemos ver las dependencias de JPA y Jakarta EE

```
import jakarta.persistence.EntityManager;
import jakarta.persistence.EntityManagerFactory;
import jakarta.persistence.Persistence;
import jakarta.persistence.TypedQuery;
```

Poco a poco los cambios van llegando a nuestras nuevas aplicaciones

Otros artículos relacionados

- JPA Persist y buenas prácticas
- JPA Query Language Objetos vs Tablas
- JPA vs Spring Data y sus diferencias
- JPA Single Table Inheritance
- JPA