METODOLOGÍA Y PROGRAMACIÓN ORIENTADA A OBJETOS I

Wiki JPA (Java Persistence API)

Andrea Bojorge

Gabriel Gómez

Gabriel Lacayo

Ervin Perez

Oscar Ramirez

04/10/2025

Prof. Jose Duran

1. Introducción a JPA

Java Persistence API (JPA) es una especificación de Java que permite gestionar datos relacionales mediante objetos Java. Su objetivo principal es simplificar el acceso, almacenamiento y administración de información en bases de datos sin depender directamente de código SQL.

En lugar de escribir consultas SQL manuales, JPA utiliza un enfoque orientado a objetos, donde las **clases Java representan tablas** y los **atributos representan columnas**.

Ejemplo conceptual:

```java
@Entity
public class Estudiante {
    @Id
    private int id;
    private String nombre;
}
```

2. Entidades

Una **entidad** es una clase Java que representa una tabla en la base de datos.

Cada instancia de una entidad equivale a una fila en esa tabla.

🔹 **Características principales:**

- Debe tener la anotación `@Entity`.
- Debe tener un campo marcado con `@Id` (clave primaria).
- Debe tener un constructor vacío público o protegido.
- Puede implementar `Serializable` (recomendado).

ejemplo en codigo:

```java
@Entity
public class Profesor implements Serializable {
    @Id
    private Long id;
    private String nombre;
    private String correo;
}
```

3. El proceso de Persistencia

La **persistencia** es el proceso mediante el cual los objetos Java se almacenan en la base de datos.

JPA maneja este proceso mediante un **EntityManager**, que se encarga de realizar las operaciones CRUD (Crear, Leer, Actualizar, Eliminar).

Ejemplo conceptual:

```java
EntityManagerFactory emf = Persistence.createEntityManagerFactory("MiUnidadPersistencia");
EntityManager em = emf.createEntityManager();

em.getTransaction().begin();
em.persist(nuevoProfesor);
em.getTransaction().commit();
```

4. Anotaciones mas utilizadas en JPA

Anotación	Descripción
@Entity	Declara una clase como entidad (tabla en la BD)
@Table(name=”nombre_tabla”)	Permite personalizar el nombre de la tabla

@Id	Indica la clave primaria
@GeneratedValue	Especifica como se genera el ID (AUTO, IDENTITY, SEQUENCE, TABLE).
@Column(name=”nombre_columna)	Permite personalizar el nombre de la columna.
@OneToOne, @OneToMany, @ManyToOne, @ManyToMany	Definen relaciones entre entidades.
@JoinColumn	Define la columna usada como clave foránea en relaciones.

Ejemplo de uso:

```java
@Entity
@Table(name = "estudiantes")
public class Estudiante {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;

    @Column(name = "nombre_completo")
    private String nombre;
}
```

5. Mapeo de Relaciones

   En base de datos relacionales, las tablas se relacionan entre sí. JPA permite representar dichas relaciones usando anotaciones.

   Tipos de Relaciones

   - **@OneToOne (uno a uno)**

     Un objeto está vinculado a un solo objeto de otra entidad.

     Ejemplo: un profesor tiene una dirección.

```java
@OneToOne
@JoinColumn(name = "direccion_id")
private Direccion direccion;
```

- **@OneToMany / @ManyToOne (uno a muchos / muchos a uno)**

  Ejemplo: un profesor puede tener muchos estudiantes.

```java
@OneToMany(mappedBy = "profesor")
private List<Estudiante> estudiantes;
```

  Punto de vista del estudiante:

```java
@ManyToOne
@JoinColumn(name = "profesor_id")
private Profesor profesor;
```

- **@ManyToMany (muchos a muchos)**

  Ejemplo: un estudiante puede estar en muchos cursos y un curso puede tener muchos estudiantes.

```java
@ManyToMany
@JoinTable(
    name = "estudiante_curso",
    joinColumns = @JoinColumn(name = "estudiante_id"),
    inverseJoinColumns = @JoinColumn(name = "curso_id")
)
private List<Curso> cursos;
```

6. Ciclo de Vida de una Entidad en JPA

Una entendidad para por varios estados durante su vida

Estado	Descripción

New	El objeto acaba de crearse se ha guardado en la base de datos
Managed	La entidad está siendo administrada por el EntityManager
Detached	La entidad fue persistida, pero ahora está fuera del contexto del EntityManager
Removed	La entidad está marcada para ser eliminada

Ejemplos:

```
Estudiante e = new Estudiante(); // New
em.persist(e);                   // Managed
em.detach(e);                    // Detached
em.remove(e);                    // Removed
```

7. Operaciones CRUD En JPA

   JPA Facilita las operaciones basicas sobre la base de datos:

Operacion	Metodo de Entity Manager	Descripcion
Crear	persist(objeto)	Inserta un nuevo registro
Leer	find(Clase.class, id)	Busca un registro por ID
Actualizar	merge(objeto)	Actualiza los datos de un registro
Eliminar	remove(objeto)	Elimina un registro existente

8. Conclusión

JPA es una herramienta poderosa que permite a los desarrolladores **interactuar con bases de datos de forma orientada a objetos**, reduciendo la complejidad del SQL manual.

Con conceptos como entidades, relaciones y el ciclo de vida, se logra una abstracción limpia y modular del manejo de datos.

Su integración con frameworks como **Hibernate** o **Spring Data JPA** la convierte en un estándar clave en el desarrollo empresarial en Java.

Link de Repositorio: https://github.com/Gigomez23/WikiJPA