JPA (Jakarta) e PostgreSQL

1. Creazione del progetto in IntelliJ

- Apri IntelliJ IDEA
- Crea un nuovo progetto
- Scegli **Java** come linguaggio
- Scegli **Maven** come sistema di build
- Inserisci nome del progetto, gruppo e artefatto
- Clicca su **Finish**

2. Struttura del progetto

```
progetto-libro/
├── pom.xml
└── src/
    ├── main/
    │   ├── java/
    │   │   └── it/
    │   │       └── epicode/
    │   │           ├── Main.java
    │   │           ├── dao/
    │   │           │   └── LibroDao.java
    │   │           └── model/
    │   │               ├── Categoria.java
    │   │               └── Libro.java
    │   └── resources/
    │       └── META-INF/
    │           └── persistence.xml
    └── test/
```

> ⚠️ Se `META-INF` non esiste, crealo manualmente dentro `resources`.

3. Entity: `Libro.java`

```java
package it.epicode.model;

// Enum Categoria
public enum Categoria {
    ROMANZO, SAGGIO, FANTASCIENZA, STORICO
}
```

```java
package it.epicode.model;

import jakarta.persistence.*;
import java.util.Objects;

@Entity // Indica che la classe è un'entità JPA
@Table(name = "libro") // Specifica il nome della tabella nel database (in minuscolo)
public class Libro {

    @Id // Indica il campo chiave primaria
    @GeneratedValue(strategy = GenerationType.IDENTITY) // Generazione automatica dell
    private Long id;

    @Column(length = 100, nullable = false, unique = true) // Opzioni: lunghezza, null
    private String titolo;

    @Column(length = 50)
    private String autore;

    @Column(length = 50)
    private String casaEditrice;

    @Enumerated(EnumType.STRING) // Salva l'enum come stringa nel DB (non come indice
    private Categoria categoria;

    public Libro() {}

    public Libro(String titolo, String autore, String casaEditrice, Categoria categori
        this.titolo = titolo;
        this.autore = autore;
        this.casaEditrice = casaEditrice;
        this.categoria = categoria;
    }

    // Getters e Setters
    public Long getId() { return id; }
    public String getTitolo() { return titolo; }
    public void setTitolo(String titolo) { this.titolo = titolo; }
    public String getAutore() { return autore; }
    public void setAutore(String autore) { this.autore = autore; }
    public String getCasaEditrice() { return casaEditrice; }
    public void setCasaEditrice(String casaEditrice) { this.casaEditrice = casaEditric
    public Categoria getCategoria() { return categoria; }
    public void setCategoria(Categoria categoria) { this.categoria = categoria; }

    // equals, hashCode, toString
    @Override
    public boolean equals(Object o) {
        if (this == o) return true;
        if (o == null || getClass() != o.getClass()) return false;
        Libro libro = (Libro) o;
        return Objects.equals(id, libro.id);
```

```
    }

    @Override
    public int hashCode() { return Objects.hash(id); }

    @Override
    public String toString() {
        return "Libro{" +
                "id=" + id +
                ", titolo='" + titolo + '\'' +
                ", autore='" + autore + '\'' +
                ", casaEditrice='" + casaEditrice + '\'' +
                ", categoria=" + categoria +
                '}';
    }
```

4. Creazione del database in pgAdmin (GUI)

1. Apri **pgAdmin**
2. Clicca con il destro su **Databases > Create > Database**
3. Inserisci un nome a piacere (es. `epicode_libri`)
4. Clicca su **Save**

5. persistence.xml

Percorso: `src/main/resources/META-INF/persistence.xml`

```
<?xml version="1.0" encoding="UTF-8"?>
<persistence xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" version="2.2"
             xmlns="http://xmlns.jcp.org/xml/ns/persistence"
             xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/persistence
             http://xmlns.jcp.org/xml/ns/persistence/persistence_2_2.xsd">
    <persistence-unit name="epicode">
        <properties>
            <property name="javax.persistence.jdbc.url" value="jdbc:postgresql://local
            <property name="javax.persistence.jdbc.user" value="postgres"/>
            <property name="javax.persistence.jdbc.password" value="XXXXXX"/>
            <property name="javax.persistence.jdbc.driver" value="org.postgresql.Drive
            <property name="hibernate.default_schema" value="public"/>
            <property name="hibernate.dialect" value="org.hibernate.dialect.PostgreSQL
            <property name="hibernate.hbm2ddl.auto" value="create"/>
        </properties>
```

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

> ✅ Sostituisci `XXXXXX` con la tua password e `YYYYYYYYYYYY` con il nome del DB

hibernate.hbm2ddl.auto:

- `create` : crea il DB da zero ad ogni avvio (cancella i dati!)
- `update` : aggiorna senza cancellare
- `validate` : verifica se lo schema è corretto
- `none` : disabilita ogni azione

6. DAO: `LibroDao.java`

```
package it.epicode.dao;

import it.epicode.model.Libro;
import jakarta.persistence.EntityManager;

public class LibroDao {

    private EntityManager em;

    public LibroDao(EntityManager em) {
        this.em = em;
    }

    public void save(Libro libro) {
        em.getTransaction().begin();
        em.persist(libro);
        em.getTransaction().commit();
    }

    public Libro getById(Long id) {
        return em.find(Libro.class, id);
    }

    public void delete(Long id) {
        em.getTransaction().begin();
        Libro libro = em.find(Libro.class, id);
        if (libro != null) {
            em.remove(libro);
        }
        em.getTransaction().commit();
    }
}
```

7. Main.java

```
package it.epicode;

import it.epicode.dao.LibroDao;
import it.epicode.model.*;
import jakarta.persistence.*;

public class Main {
    public static void main(String[] args) {
        EntityManagerFactory emf = Persistence.createEntityManagerFactory("epicode");
        EntityManager em = emf.createEntityManager();

        LibroDao dao = new LibroDao(em);

        // Inserimento libri
        dao.save(new Libro("Il Nome della Rosa", "Eco", "Bompiani", Categoria.ROMANZO)
        dao.save(new Libro("1984", "Orwell", "Mondadori", Categoria.FANTASCIENZA));
        dao.save(new Libro("Sapiens", "Harari", "Bompiani", Categoria.SAGGIO));
        dao.save(new Libro("Il Gattopardo", "Tomasi", "Feltrinelli", Categoria.STORICO
        dao.save(new Libro("Il barone rampante", "Calvino", "Einaudi", Categoria.ROMAN

        // Recupero by ID
        System.out.println(dao.getById(1L));
        System.out.println(dao.getById(2L));
        System.out.println(dao.getById(3L));

        // Cancellazione
        dao.delete(4L);

        // Chiusura
        em.close();
        emf.close();
    }
}
```

> ⚠️ È importante chiudere `EntityManager` e `Factory` per liberare risorse.

8. pom.xml (dipendenze principali)

```
<dependencies>
    <dependency>
        <groupId>org.hibernate</groupId>
        <artifactId>hibernate-core</artifactId>
        <version>6.2.5.Final</version>
    </dependency>
```

```
    <dependency>
        <groupId>org.postgresql</groupId>
        <artifactId>postgresql</artifactId>
        <version>42.7.1</version>
    </dependency>


</dependencies>
```