**AE-4. JPA**

**Ivan Nuñez Rodriguez**

2º DAM

Acceso a Datos

**Índice:**

1. Código.
2. Ejecución y resultado.
3. Enlace a GitHub.

**Código:**

**Entrada**import controller.LibreriaController;  
import model.Autor;  
import model.Editorial;  
import model.Libreria;  
import model.Libro;  
  
import java.util.Arrays;  
  
public class Entrada {  
  
 public static void main(String[] args) {  
 LibreriaController libreriaController = new LibreriaController();  
  
 // Agregar autores  
 Autor autor1 = new Autor("Gabriel", "García Márquez", "1927-03-06");  
 Autor autor2 = new Autor("Isabel", "Allende", "1942-08-02");  
 libreriaController.agregarAutor(autor1);  
 libreriaController.agregarAutor(autor2);  
  
 // Agregar editoriales  
 Editorial editorial1 = new Editorial("RBA", "Madrid, España");  
 Editorial editorial2 = new Editorial("Planeta", "Barcelona, España");  
 libreriaController.agregarEditorial(editorial1);  
 libreriaController.agregarEditorial(editorial2);  
  
 // Agregar libros  
 libreriaController.agregarLibro(new Libro("Cien años de soledad", 25.99), 1, 1);  
 libreriaController.agregarLibro(new Libro("La casa de los espíritus", 20.50), 2, 2);  
  
 // Agregar librerías  
 Libreria libreria1 = new Libreria("Librería Central", "Juan Pérez", "Calle Mayor, 10");  
 libreriaController.agregarLibreria(libreria1, Arrays.*asList*(1, 2));  
  
  
 // Mostrar información  
 libreriaController.mostrarLibrosConEditorialYAutor();  
 libreriaController.mostrarAutoresConLibros();  
 libreriaController.mostrarLibreriasConLibros();  
 }  
}

**Model.libro**

package model;  
  
import jakarta.persistence.\*;  
import lombok.AllArgsConstructor;  
import lombok.Getter;  
import lombok.NoArgsConstructor;  
import lombok.Setter;  
  
@Getter  
@Setter  
@NoArgsConstructor  
@AllArgsConstructor  
@Entity  
@NamedQueries({@NamedQuery(name = "Libro.buscaPrimerosCuatro", query = "SELECT l FROM Libro l ORDER BY l.id ASC"), @NamedQuery(name = "Libro.buscaSuientesCuatro", query = "SELECT l FROM Libro l ORDER BY l.id ASC")})  
public class Libro {  
 @Id  
 @GeneratedValue(strategy = GenerationType.*IDENTITY*)  
 private Long id;  
 @Column  
 private String titulo;  
 @Column  
 private double precio;  
  
 @ManyToOne  
 @JoinColumn(name = "editorial\_id")  
 private Editorial editorial;  
  
 @ManyToOne  
 @JoinColumn(name = "autor\_id")  
 private Autor autor;  
  
 public Libro(String titulo, double precio, Autor autor, Editorial editorial) {  
 this.titulo = titulo;  
 this.precio = precio;  
 this.autor = autor;  
 this.editorial = editorial;  
 }  
  
 public Libro(String titulo, double precio) {  
 this.titulo = titulo;  
 this.precio = precio;  
  
  
 }  
}

**Model.libreria**

package model;  
  
import jakarta.persistence.\*;  
import lombok.AllArgsConstructor;  
import lombok.Getter;  
import lombok.NoArgsConstructor;  
import lombok.Setter;  
  
import java.util.ArrayList;  
import java.util.List;  
  
@Getter  
@Setter  
@NoArgsConstructor  
@AllArgsConstructor  
@Entity  
public class Libreria {  
 @Id  
 @GeneratedValue(strategy = GenerationType.*IDENTITY*)  
 private Long id;  
 @Column  
 private String nombre;  
 @Column  
 private String nombreDueno;  
 @Column  
 private String direccion;  
  
 @ManyToMany  
 @JoinTable(name = "libreria\_libro", joinColumns = @JoinColumn(name = "libreria\_id"), inverseJoinColumns = @JoinColumn(name = "libro\_id"))  
 private List<Libro> libros = new ArrayList<>();  
  
 public Libreria(String nombre, String nombreDueno, String direccion, List<Libro> libros) {  
 this.nombre = nombre;  
 this.nombreDueno = nombreDueno;  
 this.direccion = direccion;  
 this.libros = libros;  
 }  
  
 public Libreria(String nombre, String nombreDueno, String direccion) {  
 this.nombre = nombre;  
 this.nombreDueno = nombreDueno;  
 this.direccion = direccion;  
  
 }  
}

**Model.editorial**

package model;  
  
import jakarta.persistence.\*;  
import lombok.AllArgsConstructor;  
import lombok.Getter;  
import lombok.NoArgsConstructor;  
import lombok.Setter;  
  
import java.util.ArrayList;  
import java.util.List;  
  
@Getter  
@Setter  
@NoArgsConstructor  
@AllArgsConstructor  
@Entity  
public class Editorial {  
 @Id  
 @GeneratedValue(strategy = GenerationType.*IDENTITY*)  
 private Long id;  
 @Column  
 private String nombre;  
 @Column  
 private String direccion;  
  
 @OneToMany(mappedBy = "editorial", cascade = CascadeType.*ALL*)  
 private List<Libro> libros = new ArrayList<>();  
  
 public Editorial(String nombre, String direccion) {  
 this.nombre = nombre;  
 this.direccion = direccion;  
 }  
  
 public void addLibro(Libro libro) {  
 this.libros.add(libro);  
 libro.setEditorial(this);  
 }  
}

**Model.autor**

package model;  
  
import jakarta.persistence.\*;  
import lombok.AllArgsConstructor;  
import lombok.Getter;  
import lombok.NoArgsConstructor;  
import lombok.Setter;  
  
import java.text.ParseException;  
import java.text.SimpleDateFormat;  
import java.util.ArrayList;  
import java.util.Date;  
import java.util.List;  
  
@Getter  
@Setter  
@NoArgsConstructor  
@AllArgsConstructor  
@Entity  
public class Autor {  
 @Id  
 @GeneratedValue(strategy = GenerationType.*IDENTITY*)  
 private Long id;  
@Column  
 private String nombre;  
@Column  
 private String apellidos;  
@Column  
 private Date fechaNacimiento;  
  
 @OneToMany(mappedBy = "autor", cascade = CascadeType.*ALL*, fetch = FetchType.*LAZY*)  
 private List<Libro> libros = new ArrayList<>();  
  
 public Autor(String nombre, String apellidos, String fechaNacimiento) {  
 try {  
 SimpleDateFormat sdf = new SimpleDateFormat("yyyy-MM-dd");  
 this.fechaNacimiento = sdf.parse(fechaNacimiento);  
 } catch (ParseException e) {  
 throw new IllegalArgumentException("Formato de fecha incorrecto, use yyyy-MM-dd", e);  
 }  
 this.nombre = nombre;  
 this.apellidos = apellidos;  
 }  
  
 public List<Libro> getLibros() {  
 return libros;  
 }  
  
 public void setLibros(List<Libro> libros) {  
 this.libros = libros != null ? libros : new ArrayList<>();  
 }  
  
 public void addLibro(Libro libro) {  
 this.libros.add(libro);  
 libro.setAutor(this);  
 }  
}

**Database.hibernateutil**

package database;  
  
import org.hibernate.SessionFactory;  
import org.hibernate.cfg.Configuration;  
  
public class HibernateUtil {  
  
 private static SessionFactory *sessionFactory*;  
  
 public SessionFactory getSessionFactory() {  
 if (*sessionFactory* == null) {  
 createSessionFactory();  
 }  
 return *sessionFactory*;  
 }  
  
 private void createSessionFactory() {  
 *sessionFactory* = new Configuration().configure().buildSessionFactory();  
 }  
}

**DAO.autorDAO**

package dao;  
  
import database.HibernateUtil;  
import model.Autor;  
import org.hibernate.Session;  
import org.hibernate.query.Query;  
  
import java.util.List;  
  
public class AutorDAO {  
  
 public void agregarAutor(Autor autor) {  
 Session session = new HibernateUtil().getSessionFactory().getCurrentSession();  
 session.beginTransaction();  
 session.persist(autor);  
 session.getTransaction().commit();  
 session.close();  
 }  
  
 public List<Autor> obtenerAutoresConLibros() {  
 Session session = new HibernateUtil().getSessionFactory().getCurrentSession();  
 session.beginTransaction();  
 String querySTR = "SELECT DISTINCT a FROM Autor a LEFT JOIN FETCH a.libros";  
 Query<Autor> query = session.createQuery(querySTR, Autor.class);  
 List<Autor> autores = query.getResultList();  
 session.getTransaction().commit();  
 session.close();  
 return autores;  
 }  
}

**DAO.editorialDAO**

package dao;  
  
import database.HibernateUtil;  
import model.Editorial;  
import org.hibernate.Session;  
  
public class EditorialDAO {  
  
 public void agregarEditorial(Editorial editorial) {  
 Session session = new HibernateUtil().getSessionFactory().getCurrentSession();  
 session.beginTransaction();  
 session.persist(editorial);  
 session.getTransaction().commit();  
 session.close();  
 }  
}

**DAO.libreriaDAO**

package dao;  
  
import database.HibernateUtil;  
import model.Libreria;  
import org.hibernate.Session;  
import org.hibernate.query.Query;  
  
import java.util.List;  
  
public class LibreriaDAO {  
  
 public void agregarLibreria(Libreria libreria) {  
 Session session = new HibernateUtil().getSessionFactory().getCurrentSession();  
 session.beginTransaction();  
 session.persist(libreria);  
 session.getTransaction().commit();  
 session.close();  
 }  
  
 public List<Libreria> obtenerLibreriasConLibros() {  
 Session session = new HibernateUtil().getSessionFactory().getCurrentSession();  
 session.beginTransaction();  
 String querySTR = "SELECT DISTINCT l FROM Libreria l LEFT JOIN FETCH l.libros WHERE SIZE(l.libros) > 0";  
 Query<Libreria> query = session.createQuery(querySTR, Libreria.class);  
  
 List<Libreria> librerias = query.getResultList();  
 session.getTransaction().commit();  
 session.close();  
  
 return librerias;  
 }  
}

**DAO.LibroDAO**

package dao;  
  
import database.HibernateUtil;  
import model.Libro;  
import org.hibernate.Session;  
import org.hibernate.query.Query;  
  
import java.util.List;  
  
public class LibroDAO {  
  
 public void agregarLibro(Libro libro) {  
 Session session = new HibernateUtil().getSessionFactory().getCurrentSession();  
 session.beginTransaction();  
 session.persist(libro);  
 session.getTransaction().commit();  
 session.close();  
 }  
  
 public List<Libro> obtenerLibrosConEditorialYAutor() {  
 Session session = new HibernateUtil().getSessionFactory().getCurrentSession();  
 session.beginTransaction();  
 String querySTR = "SELECT l FROM Libro l JOIN FETCH l.editorial JOIN FETCH l.autor";  
 Query<Libro> query = session.createQuery(querySTR, Libro.class);  
 List<Libro> libros = query.getResultList();  
 session.getTransaction().commit();  
 session.close();  
 return libros;  
 }  
}

**Controller.libreria.controller**

package controller;  
  
import dao.AutorDAO;  
import dao.EditorialDAO;  
import dao.LibreriaDAO;  
import dao.LibroDAO;  
import database.HibernateUtil;  
import model.Autor;  
import model.Editorial;  
import model.Libreria;  
import model.Libro;  
import org.hibernate.Session;  
  
import java.util.List;  
  
public class LibreriaController {  
 private final AutorDAO autorDAO;  
 private final EditorialDAO editorialDAO;  
 private final LibroDAO libroDAO;  
 private final LibreriaDAO libreriaDAO;  
  
 public LibreriaController() {  
 autorDAO = new AutorDAO();  
 editorialDAO = new EditorialDAO();  
 libroDAO = new LibroDAO();  
 libreriaDAO = new LibreriaDAO();  
 }  
  
 public void agregarAutor(Autor autor) {  
 if (autor.getFechaNacimiento() != null) {  
 autorDAO.agregarAutor(autor);  
 } else {  
 System.*out*.println("Fecha de nacimiento no válida");  
 }  
 }  
  
 public void agregarEditorial(Editorial editorial) {  
 editorialDAO.agregarEditorial(editorial);  
 }  
  
 public void agregarLibro(Libro libro, int idAutor, int idEditorial) {  
 Session session = new HibernateUtil().getSessionFactory().getCurrentSession();  
 session.beginTransaction();  
 Autor autor = session.get(Autor.class, idAutor);  
 Editorial editorial = session.get(Editorial.class, idEditorial);  
 session.getTransaction().commit();  
 session.close();  
 if (autor == null) {  
 System.*out*.println("Error: No se encontró el autor con ID " + idAutor);  
 return;  
 }  
 if (editorial == null) {  
 System.*out*.println("Error: No se encontró la editorial con ID " + idEditorial);  
 return;  
 }  
 libro.setAutor(autor);  
 libro.setEditorial(editorial);  
 libroDAO.agregarLibro(libro);  
 }  
  
 public void agregarLibreria(Libreria libreria, List<Integer> idsLibros) {  
 Session session = new HibernateUtil().getSessionFactory().getCurrentSession();  
 session.beginTransaction();  
 for (Integer id : idsLibros) {  
 Libro libro = session.get(Libro.class, id);  
 libreria.getLibros().add(libro);  
 }  
 session.getTransaction().commit();  
 session.close();  
  
 libreriaDAO.agregarLibreria(libreria);  
 }  
  
 public void mostrarLibrosConEditorialYAutor() {  
 List<Libro> libros = libroDAO.obtenerLibrosConEditorialYAutor();  
 for (Libro libro : libros) {  
 System.*out*.println(libro.getTitulo() + " - " + libro.getAutor().getNombre() + " - " + libro.getEditorial().getNombre());  
 }  
 }  
  
 public void mostrarAutoresConLibros() {  
 List<Autor> autores = autorDAO.obtenerAutoresConLibros();  
 for (Autor autor : autores) {  
 System.*out*.println(autor.getNombre() + " " + autor.getApellidos() + " - " + autor.getLibros().size() + " libros");  
 }  
 }  
  
 public void mostrarLibreriasConLibros() {  
 List<Libreria> librerias = libreriaDAO.obtenerLibreriasConLibros();  
 for (Libreria libreria : librerias) {  
 System.*out*.println(libreria.getNombre() + " - " + libreria.getLibros().size() + " libros");  
 }  
 }  
}

**Resources.hibernate.cfg.xml**

<?xml version="1.0" encoding="UTF-8" ?>  
<hibernate-configuration xmlns="http://www.hibernate.org/xsd/orm/cfg">  
 <session-factory>  
 <property name="current\_session\_context\_class">thread</property>  
 <property name="connection.url">jdbc:mysql://localhost/libreria</property>  
 <property name="connection.username">root</property>  
 <property name="connection.password"></property>  
 <property name="connection.driver\_class">com.mysql.cj.jdbc.Driver</property>  
 <property name="dialect">org.hibernate.dialect.MySQLDialect</property>  
 <mapping class="model.Autor"/>  
 <mapping class="model.Editorial"/>  
 <mapping class="model.Libro"/>  
 <mapping class="model.Libreria"/>  
 </session-factory>  
</hibernate-configuration>

**Ejecución y resultado.**

Interfaz de usuario gráfica, Texto, Aplicación

El contenido generado por IA puede ser incorrecto.Interfaz de usuario gráfica, Texto, Aplicación

El contenido generado por IA puede ser incorrecto.

**Interfaz de usuario gráfica, Aplicación

El contenido generado por IA puede ser incorrecto.Captura de pantalla de un celular

El contenido generado por IA puede ser incorrecto.**

**Texto

El contenido generado por IA puede ser incorrecto.**

**Enlace a GitHub:**

<https://github.com/Ivannunezrodriguez/Acceso_a_Datos_24-25/tree/6ad6d0f9574196996fb7a0eda07e6a9350454205/Acitvidades/AE-4.%20JPA>