**Demonstrate implementation of O/R Mapping**

**Book.java**

package com.example.librarydemo.entity;

import com.fasterxml.jackson.annotation.JsonIgnoreProperties;

import jakarta.persistence.\*;

import java.util.\*;

@Entity

public class Book {

@Id @GeneratedValue(strategy = GenerationType.IDENTITY)

private Long id;

private String title;

private Integer year;

@ManyToOne(fetch = FetchType.EAGER)

@JsonIgnoreProperties("books")

@JoinColumn(name = "publisher\_id", nullable = false)

private Publisher publisher;

@ManyToMany(fetch = FetchType.LAZY)

@JoinTable(name = "book\_author",

joinColumns = @JoinColumn(name = "book\_id"),

inverseJoinColumns = @JoinColumn(name = "author\_id"))

private Set<Author> authors = new HashSet<>();

public Book() {}

public Book(String title, Integer year, Publisher publisher, Set<Author> authors) {

this.title = title;

this.year = year;

this.publisher = publisher;

this.authors = authors;

}

public Long getId() { return id; }

public String getTitle() { return title; }

public Integer getYear() { return year; }

public Publisher getPublisher() { return publisher; }

public Set<Author> getAuthors() { return authors; }

}

**Author.java**

package com.example.librarydemo.entity;

import com.fasterxml.jackson.annotation.JsonIgnoreProperties;

import jakarta.persistence.\*;

import java.util.\*;

@Entity

public class Author {

@Id @GeneratedValue(strategy = GenerationType.IDENTITY)

private Long id;

private String fullName;

@ManyToMany(mappedBy = "authors", fetch = FetchType.LAZY)

@JsonIgnoreProperties("authors")

private Set<Book> books = new HashSet<>();

public Author() {}

public Author(String fullName) { this.fullName = fullName; }

public Long getId() { return id; }

public String getFullName() { return fullName; }

public Set<Book> getBooks() { return books; }

}

**Publisher.java**

package com.example.librarydemo.entity;

import com.fasterxml.jackson.annotation.JsonIgnoreProperties;

import jakarta.persistence.\*;

import java.util.\*;

@JsonIgnoreProperties("books")

@Entity

public class Publisher {

@Id @GeneratedValue(strategy = GenerationType.IDENTITY)

private Long id;

private String name;

@OneToMany(mappedBy = "publisher",

fetch = FetchType.LAZY,

cascade = CascadeType.ALL,

orphanRemoval = true)

private List<Book> books = new ArrayList<>();

public Publisher() {}

public Publisher(String name) { this.name = name; }

public Long getId() { return id; }

public String getName() { return name; }

public List<Book> getBooks() { return books; }

}

**BookRepository.java**

package com.example.librarydemo.repository;

import com.example.librarydemo.entity.Book;

import org.springframework.data.jpa.repository.JpaRepository;

public interface BookRepository extends JpaRepository<Book, Long> {}

**AuthorRepository.java**

package com.example.librarydemo.repository;

import com.example.librarydemo.entity.Author;

import org.springframework.data.jpa.repository.JpaRepository;

public interface AuthorRepository extends JpaRepository<Author, Long> {}

**PublisherRepository.java**

package com.example.librarydemo.repository;

import com.example.librarydemo.entity.Publisher;

import org.springframework.data.jpa.repository.JpaRepository;

public interface PublisherRepository extends JpaRepository<Publisher, Long> {}

**BookController.java**

package com.example.librarydemo.controller;

import com.example.librarydemo.entity.Book;

import com.example.librarydemo.repository.BookRepository;

import org.springframework.web.bind.annotation.\*;

import java.util.List;

@RestController

@RequestMapping("/books")

public class BookController {

private final BookRepository bookRepository;

public BookController(BookRepository bookRepository) { this.bookRepository = bookRepository; }

@GetMapping

public List<Book> getAllBooks() { return bookRepository.findAll(); }

}

**LibrarydemoApplication.java**

package com.example.librarydemo;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication

public class LibrarydemoApplication {

public static void main(String[] args) { SpringApplication.run(LibrarydemoApplication.class, args); }

}

**DataLoader.java**

package com.example.librarydemo.config;

import com.example.librarydemo.entity.\*;

import com.example.librarydemo.repository.\*;

import org.springframework.boot.ApplicationRunner;

import org.springframework.context.annotation.Bean;

import org.springframework.context.annotation.Configuration;

import java.util.Set;

@Configuration

public class DataLoader {

@Bean

ApplicationRunner load(BookRepository books,

AuthorRepository authors,

PublisherRepository publishers) {

return args -> {

Publisher p = publishers.save(new Publisher("Penguin"));

Author a = authors.save(new Author("George Orwell"));

Book b = new Book("1984", 1949, p, Set.of(a));

books.save(b);

};

}

}

**application.properties**

spring.datasource.url=jdbc:mysql://localhost:3306/librarybd

spring.datasource.username=root

spring.datasource.password=root

spring.jpa.hibernate.ddl-auto=update

spring.jpa.open-in-view=false

server.port=8081

**pom.xml**

<project xmlns="http://maven.apache.org/POM/4.0.0"

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:schemaLocation="http://maven.apache.org/POM/4.0.0

https://maven.apache.org/xsd/maven-4.0.0.xsd">

<modelVersion>4.0.0</modelVersion>

<groupId>com.example</groupId>

<artifactId>librarydemo</artifactId>

<version>0.0.1-SNAPSHOT</version>

<parent>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-parent</artifactId>

<version>3.3.0</version>

<relativePath/>

</parent>

<properties><java.version>21</java.version></properties>

<dependencies>

<dependency><groupId>org.springframework.boot</groupId><artifactId>spring-boot-starter-data-jpa</artifactId></dependency>

<dependency><groupId>org.springframework.boot</groupId><artifactId>spring-boot-starter-web</artifactId></dependency>

<dependency><groupId>com.mysql</groupId><artifactId>mysql-connector-j</artifactId><scope>runtime</scope></dependency>

</dependencies>

<build>

<plugins><plugin><groupId>org.springframework.boot</groupId><artifactId>spring-boot-maven-plugin</artifactId></plugin></plugins>

</build>

</project>

**Output:**





