**Demonstrate implementation of O/R Mapping**

@ManyToOne, @JoinColumn, @OneToMany, FetchType.EAGER, FetchType.LAZY, @ManyToMany, @JoinTable, mappedBy

Relationships reference - <https://www.baeldung.com/spring-data-rest-relationships>

**STRUCTURE:**



**AuthorController.java**

package com.example.library.controller;

import com.example.library.repository.AuthorRepository;

import com.example.library.entity.Author;

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

import java.util.List;

@RestController

@RequestMapping("/authors")

public class AuthorController {

    private final AuthorRepository repo;

    public AuthorController(AuthorRepository repo) {

        this.repo = repo;

    }

    @GetMapping

    public List<Author> getAll() {

        return repo.findAll();

    }

}

**Author.java**

package com.example.library.entity;

import jakarta.persistence.\*;

import lombok.\*;

@Entity

@Data

@NoArgsConstructor

@AllArgsConstructor

public class Author {

    @Id

    @GeneratedValue(strategy = GenerationType.IDENTITY)

    private Long id;

    private String name;

    public Author(String name) {

        this.name = name;

    }

}

**Book.java**

package com.example.library.entity;

import jakarta.persistence.\*;

import java.util.List;

@Entity

public class Book {

    @Id

    @GeneratedValue(strategy = GenerationType.IDENTITY)

    private Long id;

    private String title;

    @ManyToOne(fetch = FetchType.EAGER)

    @JoinColumn(name = "author\_id")

    private Author author;

    @ManyToMany

    @JoinTable(

        name = "book\_category",

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

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

    )

    private List<Category> categories;

    // Getters and setters

}

**Category.java**

package com.example.library.entity;

import jakarta.persistence.\*;

import java.util.List;

@Entity

public class Category {

    @Id

    @GeneratedValue(strategy = GenerationType.IDENTITY)

    private Long id;

    private String name;

    @ManyToMany(mappedBy = "categories")

    private List<Book> books;

    // Getters and setters

}

**AuthorRepository.java**

package com.example.library.repository;

import com.example.library.entity.Author;

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

public interface AuthorRepository extends JpaRepository<Author, Long> {

}

**BookRepository.java**

package com.example.library.repository;

import com.example.library.entity.Book;

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

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

CategoryRepository.java

package com.example.library.repository;

import com.example.library.entity.Category;

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

public interface CategoryRepository extends JpaRepository<Category, Long> {}

**LibraryApplication.java**

package com.example.library;

import com.example.library.entity.Author;

import com.example.library.repository.AuthorRepository;

import org.springframework.boot.CommandLineRunner;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

import org.springframework.context.annotation.Bean;

@SpringBootApplication

public class LibraryApplication {

    public static void main(String[] args) {

        SpringApplication.run(LibraryApplication.class, args);

    }

    @Bean

    public CommandLineRunner loadData(AuthorRepository authorRepository) {

        return args -> {

           authorRepository.save(new Author("J.K. Rowling"));

authorRepository.save(new Author("George R.R. Martin"));

        };

    }

}

Application.properties

spring.datasource.url=jdbc:h2:mem:testdb

spring.datasource.driver-class-name=org.h2.Driver

spring.datasource.username=sa

spring.datasource.password=

spring.jpa.hibernate.ddl-auto=update

spring.h2.console.enabled=true

server.port=8084

**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

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

    <modelVersion>4.0.0</modelVersion>

    <groupId>com.example</groupId>

    <artifactId>springdatajpa</artifactId>

    <version>1.0.0</version>

    <packaging>jar</packaging>

    <name>Spring Data JPA</name>

    <description>Country Code Demo</description>

    <parent>

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

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

        <version>3.2.0</version>

        <relativePath/> <!-- lookup parent from repository -->

    </parent>

    <properties>

        <java.version>17</java.version>

    </properties>

    <dependencies>

        <!-- Spring Web -->

        <dependency>

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

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

        </dependency>

        <!-- Spring Data JPA -->

        <dependency>

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

            <artifactId>spring-boot-starter-data-jpa</artifactId>

        </dependency>

        <!-- H2 Database -->

        <dependency>

            <groupId>com.h2database</groupId>

            <artifactId>h2</artifactId>

            <scope>runtime</scope>

        </dependency>

        <!-- Lombok -->

        <dependency>

    <groupId>org.projectlombok</groupId>

    <artifactId>lombok</artifactId>

    <version>1.18.32</version>

    <scope>provided</scope>

</dependency>

        <!-- DevTools (Hot Reloading) -->

        <dependency>

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

            <artifactId>spring-boot-devtools</artifactId>

            <scope>runtime</scope>

        </dependency>

        <!-- Spring Boot Test -->

        <dependency>

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

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

            <scope>test</scope>

        </dependency>

    </dependencies>

    <build>

        <plugins>

            <!-- Spring Boot Maven Plugin -->

            <plugin>

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

                <artifactId>spring-boot-maven-plugin</artifactId>

            </plugin>

        </plugins>

    </build>

</project>

**OUTPUT:**

****