1. **Bật @ManyToMany ở cả 2 entity**

@Entity

public class Book {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private Long id;

private String name;

@ManyToMany

@JoinTable(

name = "Book\_TaiKhoan",

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

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

)

private Set<TaiKhoan> taiKhoanTheoDoi = new HashSet<>();

// Getters and Setters

}

@Entity

public class TaiKhoan {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private Long id;

private String username;

private String password;

private String name;

@ManyToMany(mappedBy = "taiKhoanTheoDoi")

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

// Getters and Setters

}

Sẽ giúp hcho truy vấn và quản lí dễ dàng hơn

1. **Hàm add sách vào danh sách theo dõi**

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.stereotype.Service;

@Service

public class TaiKhoanService {

@Autowired

private TaiKhoanRepository taiKhoanRepository;

@Autowired

private BookRepository bookRepository;

public void addBookToTaiKhoan(Long taiKhoanId, Long bookId) {

// Tìm kiếm tài khoản theo ID

TaiKhoan taiKhoan = taiKhoanRepository.findById(taiKhoanId)

.orElseThrow(() -> new EntityNotFoundException("Tài khoản không tồn tại"));

// Tìm kiếm sách theo ID

Book book = bookRepository.findById(bookId)

.orElseThrow(() -> new EntityNotFoundException("Sách không tồn tại"));

// Thêm sách vào danh sách theo dõi của tài khoản

taiKhoan.getBooksTheoDoi().add(book);

book.getTaiKhoanTheoDoi().add(taiKhoan); // Cập nhật ngược lại để đồng bộ

// Lưu lại tài khoản (không cần thiết phải lưu sách nếu chỉ thay đổi tập hợp)

taiKhoanRepository.save(taiKhoan);

bookRepository.save(book); // Cũng có thể cần nếu có thay đổi trong tập hợp

}

}

1. **Thuộc tính oneToMany và ManyToOne**

import javax.persistence.\*;

import java.util.HashSet;

import java.util.Set;

@Entity

public class TacGia {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private Long id;

private String name;

@OneToMany(mappedBy = "tacGia")

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

// Getters and Setters

}

import javax.persistence.\*;

@Entity

public class Book {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private Long id;

private String name;

@ManyToOne

@JoinColumn(name = "tacgia\_id") // Khóa ngoại trong bảng Book

private TacGia tacGia;

// Getters and Setters

}

Giải Thích Cấu Hình

TacGia:

@OneToMany(mappedBy = "tacGia"): Thiết lập rằng một tác giả có thể có nhiều quyển sách. mappedBy chỉ ra rằng mối quan hệ này được quản lý bởi thuộc tính tacGia trong thực thể Book.

Book:

@ManyToOne: Thiết lập rằng mỗi quyển sách thuộc về một tác giả.

@JoinColumn(name = "tacgia\_id"): Định nghĩa cột khóa ngoại trong bảng Book để tham chiếu đến TacGia.

1. **Thêm book vào tacgia**

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.stereotype.Service;

@Service

public class BookService {

@Autowired

private BookRepository bookRepository;

@Autowired

private TacGiaRepository tacGiaRepository;

public void addBookToTacGia(Long tacGiaId, Book book) {

// Tìm kiếm tác giả theo ID

TacGia tacGia = tacGiaRepository.findById(tacGiaId)

.orElseThrow(() -> new EntityNotFoundException("Tác giả không tồn tại"));

// Gán tác giả cho sách

book.setTacGia(tacGia);

// Lưu sách

bookRepository.save(book);

}

}

1. **Create Book**

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.stereotype.Service;

import javax.persistence.EntityNotFoundException;

import java.util.List;

@Service

public class BookService {

@Autowired

private BookRepository bookRepository;

@Autowired

private TacGiaRepository tacGiaRepository;

@Autowired

private TheLoaiRepository theLoaiRepository;

public Book createBook(Book book, Long tacGiaId, List<Long> theLoaiIds) {

// Tìm kiếm tác giả

TacGia tacGia = tacGiaRepository.findById(tacGiaId)

.orElseThrow(() -> new EntityNotFoundException("Tác giả không tồn tại"));

book.setTacGia(tacGia);

// Tìm kiếm các thể loại và gán vào sách

if (theLoaiIds != null) {

Set<TheLoai> theLoais = new HashSet<>();

for (Long id : theLoaiIds) {

TheLoai theLoai = theLoaiRepository.findById(id)

.orElseThrow(() -> new EntityNotFoundException("Thể loại không tồn tại"));

theLoais.add(theLoai);

}

book.setTheLoai(theLoais);

}

return bookRepository.save(book);

}

public List<Book> getAllBooks() {

return bookRepository.findAll();

}

public Book getBookById(Long id) {

return bookRepository.findById(id)

.orElseThrow(() -> new EntityNotFoundException("Sách không tồn tại"));

}

public Book updateBook(Long id, Book updatedBook, Long tacGiaId, List<Long> theLoaiIds) {

// Kiểm tra xem sách có tồn tại không

Book existingBook = getBookById(id);

// Cập nhật thông tin

existingBook.setName(updatedBook.getName());

// Cập nhật tác giả

TacGia tacGia = tacGiaRepository.findById(tacGiaId)

.orElseThrow(() -> new EntityNotFoundException("Tác giả không tồn tại"));

existingBook.setTacGia(tacGia);

// Cập nhật thể loại

if (theLoaiIds != null) {

Set<TheLoai> theLoais = new HashSet<>();

for (Long theLoaiId : theLoaiIds) {

TheLoai theLoai = theLoaiRepository.findById(theLoaiId)

.orElseThrow(() -> new EntityNotFoundException("Thể loại không tồn tại"));

theLoais.add(theLoai);

}

existingBook.setTheLoai(theLoais);

}

return bookRepository.save(existingBook);

}

public void deleteBook(Long id) {

if (!bookRepository.existsById(id)) {

throw new EntityNotFoundException("Sách không tồn tại");

}

bookRepository.deleteById(id);

}

}

1. **CRUD thể loại**

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.stereotype.Service;

import javax.persistence.EntityNotFoundException;

import java.util.List;

@Service

public class TheLoaiService {

@Autowired

private TheLoaiRepository theLoaiRepository;

@Autowired

private BookRepository bookRepository;

public TheLoai createTheLoai(TheLoai theLoai) {

return theLoaiRepository.save(theLoai);

}

public List<TheLoai> getAllTheLoais() {

return theLoaiRepository.findAll();

}

public TheLoai getTheLoaiById(Long id) {

return theLoaiRepository.findById(id)

.orElseThrow(() -> new EntityNotFoundException("Thể loại không tồn tại"));

}

public TheLoai updateTheLoai(Long id, TheLoai updatedTheLoai) {

TheLoai existingTheLoai = getTheLoaiById(id);

existingTheLoai.setName(updatedTheLoai.getName()); // Giả sử bạn chỉ có thuộc tính name

return theLoaiRepository.save(existingTheLoai);

}

public void deleteTheLoai(Long id) {

if (!theLoaiRepository.existsById(id)) {

throw new EntityNotFoundException("Thể loại không tồn tại");

}

// Xóa thể loại chỉ khi không có sách nào tham chiếu đến nó

if (!bookRepository.existsByTheLoaiId(id)) {

theLoaiRepository.deleteById(id);

} else {

throw new IllegalArgumentException("Không thể xóa thể loại đang được sách tham chiếu");

}

}

}

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

public interface BookRepository extends JpaRepository<Book, Long> {

boolean existsByTheLoaiId(Long theLoaiId);

}

1. **Form thêm sách**

import React, { useState, useEffect } from 'react';

import axios from 'axios';

const AddBook = () => {

const [bookName, setBookName] = useState('');

const [tacGias, setTacGias] = useState([]);

const [theLoais, setTheLoais] = useState([]);

const [selectedTacGia, setSelectedTacGia] = useState('');

const [selectedTheLoais, setSelectedTheLoais] = useState([]);

useEffect(() => {

// Lấy danh sách tác giả

axios.get('/api/tacgias')

.then(response => {

setTacGias(response.data);

})

.catch(error => {

console.error("Có lỗi xảy ra khi lấy danh sách tác giả!", error);

});

// Lấy danh sách thể loại

axios.get('/api/theloais')

.then(response => {

setTheLoais(response.data);

})

.catch(error => {

console.error("Có lỗi xảy ra khi lấy danh sách thể loại!", error);

});

}, []);

const handleSubmit = (e) => {

e.preventDefault();

const bookData = {

name: bookName,

tacGiaId: selectedTacGia,

theLoaiIds: selectedTheLoais

};

axios.post('/api/books', bookData)

.then(response => {

console.log("Sách đã được thêm thành công!", response.data);

// Reset form sau khi thêm thành công

setBookName('');

setSelectedTacGia('');

setSelectedTheLoais([]);

})

.catch(error => {

console.error("Có lỗi xảy ra khi thêm sách!", error);

});

};

return (

<div>

<h1>Thêm Sách</h1>

<form onSubmit={handleSubmit}>

<div>

<label>Tên sách:</label>

<input

type="text"

value={bookName}

onChange={(e) => setBookName(e.target.value)}

required

/>

</div>

<div>

<label>Tác giả:</label>

<select

value={selectedTacGia}

onChange={(e) => setSelectedTacGia(e.target.value)}

required

>

<option value="">Chọn tác giả</option>

{tacGias.map((tacGia) => (

<option key={tacGia.id} value={tacGia.id}>

{tacGia.name}

</option>

))}

</select>

</div>

<div>

<label>Thể loại:</label>

<select

multiple

value={selectedTheLoais}

onChange={(e) => setSelectedTheLoais([...e.target.selectedOptions].map(o => o.value))}

required

>

{theLoais.map((theLoai) => (

<option key={theLoai.id} value={theLoai.id}>

{theLoai.name}

</option>

))}

</select>

</div>

<button type="submit">Thêm sách</button>

</form>

</div>

);

};

export default AddBook;