Creating Nested Resource API

[Category and Course]

Handling a many-to-many relationship in JPA can be done by defining a join table between the two entities. In this example, I'll show you how to set up a many-to-many relationship between `Category` and `Course`, and then create REST APIs to manage this relationship.

1. Entity Definitions

First, let's define the `Category` and `Course` entities with a many-to-many relationship.

`Category` Entity

```
import javax.persistence.*;
import java.util.HashSet;
import java.util.Set;

@Entity
public class Category {
```

```
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;

    private String name;

    @ManyToMany(cascade = CascadeType.ALL)
    @JoinTable(
        name = "category_course",
        joinColumns = @JoinColumn(name = "category_id"),
        inverseJoinColumns = @JoinColumn(name = "course_i
d")
    )
    private Set<Course> courses = new HashSet<>();

    // Getters and Setters

    public void addCourse(Course course) {
        this.courses.add(course);
        course.getCategories().add(this);
    }

    public void removeCourse(Course course) {
        this.courses.remove(course);
        course.getCategories().remove(this);
    }
}
```

`Course` Entity

```
import javax.persistence.*;
import java.util.HashSet;
import java.util.Set;

@Entity
```

```
public class Course {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;

    private String title;

    @ManyToMany(mappedBy = "courses")
    private Set<Category> categories = new HashSet<>();

    // Getters and Setters

    public void addCategory(Category category) {
        this.categories.add(category);
        category.getCourses().add(this);
    }

    public void removeCategory(Category category) {
        this.categories.remove(category);
        category.getCourses().remove(this);
    }
}
```

2. Repository Interfaces

Now, create repository interfaces for `Category` and `Course`.

`CategoryRepository`

```
import org.springframework.data.jpa.repository.JpaRepositor
y;

public interface CategoryRepository extends JpaRepository<C
ategory, Long> {
```

```
}
```

`CourseRepository`

```
import org.springframework.data.jpa.repository.JpaRepositor
y;

public interface CourseRepository extends JpaRepository<Cou
rse, Long> {
}
```

3. Service Layer

Implement the service layer to handle the business logic.

```
javaCopy code
import org.springframework.beans.factory.annotation.Autowir
ed;
import org.springframework.stereotype.Service;
import org.springframework.transaction.annotation.Transacti
onal;

import java.util.Optional;

@Service
public class CategoryService {

    @Autowired
    private CategoryRepository categoryRepository;

    @Autowired
    private CourseRepository courseRepository;

    @Transactional
    public Category addCategory(Category category) {
```

```
        return categoryRepository.save(category);
    }

    @Transactional
    public Course addCourse(Course course) {
        return courseRepository.save(course);
    }

    @Transactional
    public void addCourseToCategory(Long categoryId, Long c
ourseId) {
        Category category = categoryRepository.findById(cat
egoryId).orElseThrow(() -> new EntityNotFoundException("Cat
egory not found"));
        Course course = courseRepository.findById(courseI
d).orElseThrow(() -> new EntityNotFoundException("Course no
t found"));

        category.addCourse(course);
        categoryRepository.save(category);
    }

    @Transactional
    public void removeCourseFromCategory(Long categoryId, L
ong courseId) {
        Category category = categoryRepository.findById(cat
egoryId).orElseThrow(() -> new EntityNotFoundException("Cat
egory not found"));
        Course course = courseRepository.findById(courseI
d).orElseThrow(() -> new EntityNotFoundException("Course no
t found"));

        category.removeCourse(course);
        categoryRepository.save(category);
    }

    @Transactional(readOnly = true)
    public Category getCategory(Long id) {
```

```
        return categoryRepository.findById(id).orElseThrow
(() -> new EntityNotFoundException("Category not found"));
    }

    @Transactional(readOnly = true)
    public Course getCourse(Long id) {
        return courseRepository.findById(id).orElseThrow(()
-> new EntityNotFoundException("Course not found"));
    }
}
```

4. Controller Layer

Create a controller to expose the APIs.

```
import org.springframework.beans.factory.annotation.Autowir
ed;
import org.springframework.web.bind.annotation.*;

@RestController
@RequestMapping("/api/categories")
public class CategoryController {

    @Autowired
    private CategoryService categoryService;

    @PostMapping
    public Category createCategory(@RequestBody Category ca
tegory) {
        return categoryService.addCategory(category);
    }

    @PostMapping("/courses")
    public Course createCourse(@RequestBody Course course)
{
        return categoryService.addCourse(course);
    }
```

```
    @PostMapping("/{categoryId}/courses/{courseId}")
    public void addCourseToCategory(@PathVariable Long cate
goryId, @PathVariable Long courseId) {
        categoryService.addCourseToCategory(categoryId, cou
rseId);
    }

    @DeleteMapping("/{categoryId}/courses/{courseId}")
    public void removeCourseFromCategory(@PathVariable Long
categoryId, @PathVariable Long courseId) {
        categoryService.removeCourseFromCategory(categoryI
d, courseId);
    }

    @GetMapping("/{id}")
    public Category getCategory(@PathVariable Long id) {
        return categoryService.getCategory(id);
    }

    @GetMapping("/courses/{id}")
    public Course getCourse(@PathVariable Long id) {
        return categoryService.getCourse(id);
    }
}
```

5. API Endpoints Overview

- **Create a Category:**

  `POST /api/categories`

  Body: `{"name": "Technology"}`

- **Create a Course:**

  `POST /api/categories/courses`

  Body: `{"title": "Spring Boot"}`

- **Add a Course to a Category:**

  `POST /api/categories/{categoryId}/courses/{courseId}`

- **Remove a Course from a Category:**
  `DELETE /api/categories/{categoryId}/courses/{courseId}`
- **Get a Category by ID:**
  `GET /api/categories/{id}`
- **Get a Course by ID:**
  `GET /api/categories/courses/{id}`

6. Handling Bidirectional Relationships

- The `Category` and `Course` entities are bidirectional, meaning both sides are aware of the relationship.
- When adding or removing a course from a category, you must ensure the relationship is managed on both sides by adding or removing the course from the category and vice versa.

This setup allows you to effectively manage the many-to-many relationship between categories and courses and expose it through REST APIs.

If you want to map the many-to-many relationship on the `Course` side rather than on the `Category` side, you'll need to adjust the entity mappings accordingly. Here's how you can do it:

1. Entity Definitions

`Category` Entity

The `Category` entity will not directly manage the relationship. It will be mapped by the `Course` entity.

```
javaCopy code
import javax.persistence.*;
import java.util.HashSet;
import java.util.Set;

@Entity
public class Category {
```

```
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;

    private String name;

    @ManyToMany(mappedBy = "categories")
    private Set<Course> courses = new HashSet<>();

    // Getters and Setters
}
```

`Course` Entity

The `Course` entity will own the relationship and manage the join table.

```
javaCopy code
import javax.persistence.*;
import java.util.HashSet;
import java.util.Set;

@Entity
public class Course {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;

    private String title;

    @ManyToMany(cascade = CascadeType.ALL)
    @JoinTable(
        name = "course_category",
        joinColumns = @JoinColumn(name = "course_id"),
        inverseJoinColumns = @JoinColumn(name = "category_i
d")
    )
```

```
    private Set<Category> categories = new HashSet<>();

    // Getters and Setters

    public void addCategory(Category category) {
        this.categories.add(category);
        category.getCourses().add(this);
    }

    public void removeCategory(Category category) {
        this.categories.remove(category);
        category.getCourses().remove(this);
    }
}
```

2. Service Layer

Here's how the service layer might look:

```
javaCopy code
import org.springframework.beans.factory.annotation.Autowir
ed;
import org.springframework.stereotype.Service;
import org.springframework.transaction.annotation.Transacti
onal;

@Service
public class CourseService {

    @Autowired
    private CourseRepository courseRepository;

    @Autowired
    private CategoryRepository categoryRepository;

    @Transactional
    public Course addCourse(Course course) {
        return courseRepository.save(course);
```

```
    }

    @Transactional
    public Category addCategory(Category category) {
        return categoryRepository.save(category);
    }

    @Transactional
    public void addCategoryToCourse(Long courseId, Long cat
egoryId) {
        Course course = courseRepository.findById(courseI
d).orElseThrow(() -> new EntityNotFoundException("Course no
t found"));
        Category category = categoryRepository.findById(cat
egoryId).orElseThrow(() -> new EntityNotFoundException("Cat
egory not found"));

        course.addCategory(category);
        courseRepository.save(course);
    }

    @Transactional
    public void removeCategoryFromCourse(Long courseId, Lon
g categoryId) {
        Course course = courseRepository.findById(courseI
d).orElseThrow(() -> new EntityNotFoundException("Course no
t found"));
        Category category = categoryRepository.findById(cat
egoryId).orElseThrow(() -> new EntityNotFoundException("Cat
egory not found"));

        course.removeCategory(category);
        courseRepository.save(course);
    }

    @Transactional(readOnly = true)
    public Course getCourse(Long id) {
        return courseRepository.findById(id).orElseThrow(()
```

```
-> new EntityNotFoundException("Course not found"));
    }

    @Transactional(readOnly = true)
    public Category getCategory(Long id) {
        return categoryRepository.findById(id).orElseThrow
(() -> new EntityNotFoundException("Category not found"));
    }
}
```

3. Controller Layer

You can create the following REST controller to handle the operations:

```
javaCopy code
import org.springframework.beans.factory.annotation.Autowir
ed;
import org.springframework.web.bind.annotation.*;

@RestController
@RequestMapping("/api/courses")
public class CourseController {

    @Autowired
    private CourseService courseService;

    @PostMapping
    public Course createCourse(@RequestBody Course course)
{
        return courseService.addCourse(course);
    }

    @PostMapping("/categories")
    public Category createCategory(@RequestBody Category ca
tegory) {
        return courseService.addCategory(category);
    }
```

```
    @PostMapping("/{courseId}/categories/{categoryId}")
    public void addCategoryToCourse(@PathVariable Long cour
seId, @PathVariable Long categoryId) {
        courseService.addCategoryToCourse(courseId, categor
yId);
    }

    @DeleteMapping("/{courseId}/categories/{categoryId}")
    public void removeCategoryFromCourse(@PathVariable Long
courseId, @PathVariable Long categoryId) {
        courseService.removeCategoryFromCourse(courseId, ca
tegoryId);
    }

    @GetMapping("/{id}")
    public Course getCourse(@PathVariable Long id) {
        return courseService.getCourse(id);
    }

    @GetMapping("/categories/{id}")
    public Category getCategory(@PathVariable Long id) {
        return courseService.getCategory(id);
    }
}
```

4. API Endpoints Overview

- **Create a Course:**

  `POST /api/courses`

  Body: `{"title": "Spring Boot"}`

- **Create a Category:**

  `POST /api/courses/categories`

  Body: `{"name": "Technology"}`

- **Add a Category to a Course:**

  `POST /api/courses/{courseId}/categories/{categoryId}`

- **Remove a Category from a Course:**

  `DELETE /api/courses/{courseId}/categories/{categoryId}`

- **Get a Course by ID:**

  `GET /api/courses/{id}`

- **Get a Category by ID:**

  `GET /api/courses/categories/{id}`