ТОАД
Практична робота
Spring Data JPA. Зв’язки між таблицями

В цій роботі продовжується розробка проєкту із попередньої практичної роботи.

Частина А.

1. Створіть сутність Teacher (викладач). Зверніть увагу на використання класу Address:

```
import jakarta.persistence.*;
import lombok.EqualsAndHashCode;
import lombok.Getter;
import lombok.NoArgsConstructor;
import lombok.experimental.FieldDefaults;
import org.hibernate.annotations.GenericGenerator;

import java.util.Calendar;

import static lombok.AccessLevel.PRIVATE;

@Entity
@Table(name = "teacher")
@Getter
@NoArgsConstructor
@EqualsAndHashCode(of = "id")
@FieldDefaults(level = PRIVATE)
public class Teacher {

    @Id
    @GeneratedValue(strategy = GenerationType.AUTO, generator = "native")
    @GenericGenerator(name = "native", strategy = "native")
    @Column(updatable = false)
    Long id;

    @Column(name="teacher_name")
    String name;

    @Embedded
    Address address;

    @Column(name="birth_date", updatable = false)
    Calendar birthDate;
}
```

2. Створіть сутність Curator (куратор). В даному випадку куратор — це не окрема особа, а функція, обов’язок чи посада викладача.

```java
import jakarta.persistence.*;
import lombok.EqualsAndHashCode;
import lombok.Getter;
import lombok.NoArgsConstructor;
import lombok.experimental.FieldDefaults;
import org.hibernate.annotations.GenericGenerator;

import static lombok.AccessLevel.PRIVATE;

@Entity
@Table(name = "curator")
@Getter
@NoArgsConstructor
@EqualsAndHashCode(of = "id")
@FieldDefaults(level = PRIVATE)
public class Curator {

    @Id
    @GeneratedValue(strategy = GenerationType.AUTO, generator = "native")
    @GenericGenerator(name = "native", strategy = "native")
    @Column(updatable = false)
    Long id;

    @Column(name = "work_experience")
    Integer workExperience;

}
```

Задача: потрібно зв'язати сутності викладач та куратор з такою умовою, що кожен куратор є викладачем, але не кожен викладач є куратором.

В даному випадку доцільно використати зв'язок **один-до-одного**.

3. В клас Teacher додайте поле:

```java
@OneToOne(cascade = CascadeType.ALL)
@JoinColumn(name = "curator_id", referencedColumnName = "id")
Curator curator;
```

4. В клас Curator додайте поле:

```java
@OneToOne(mappedBy = "curator")
Teacher teacher;
```

5. Запустити проєкт, подивитися структуру таблиць в базі даних.

Задача: отримати ім'я куратора (викладача) по id куратора.

6. В клас Teacher додайте конструктор та сетер:

```
public Teacher(String name, Address address, Calendar birthDate) {
    this.name = name;
    this.address = address;
    this.birthDate = birthDate;
}

public void setCurator(Curator curator) {
    this.curator = curator;
}
```

7. В клас Curator додайте конструктор:

```
public Curator(Integer workExperience) {
    this.workExperience = workExperience;
}
```

8. Для нових сутностей створіть репозиторії без додавання власних методів.

9. Створіть клас TeacherService:

```
import com.example.academicservice.entity.Teacher;
import com.example.academicservice.repository.TeacherRepository;
import lombok.RequiredArgsConstructor;
import org.springframework.stereotype.Service;

@Service
@RequiredArgsConstructor
public class TeacherService {
    private final TeacherRepository teacherRepository;

    public Teacher save(Teacher teacher){
        return teacherRepository.save(teacher);
    }
}
```

10. Створіть клас CuratorService:

```
import com.example.academicservice.entity.Curator;
import com.example.academicservice.repository.CuratorRepository;
import lombok.RequiredArgsConstructor;
import org.springframework.stereotype.Service;

import java.util.Optional;

@Service
@RequiredArgsConstructor
public class CuratorService {
    private final CuratorRepository curatorRepository;
```

```
    public Curator save(Curator curator){
        return curatorRepository.save(curator);
    }

    public String getNameById(Long id){
        Optional<Curator> curator = curatorRepository.findById(id);
        if(curator.isPresent()){
            return curator.get().getTeacher().getName();
        }else{
            throw new RuntimeException("Curator doesn't exist, id="+id);
        }
    }
}
```

11. Створіть клас CuratorController в пакеті API:

```
@RestController
@RequestMapping("api/v1/curators")
@RequiredArgsConstructor
public class CuratorController {

    private final CuratorService curatorService;

    @GetMapping("/{id}/name")
    public ResponseEntity<String> getCuratorName(@PathVariable Long id){
        try{
            String name=curatorService.getNameById(id);
            return new ResponseEntity<>(name, HttpStatus.OK);
        }catch (Exception e){
            return new
ResponseEntity(e.getMessage(),HttpStatus.INTERNAL_SERVER_ERROR);
        }
    }

}
```

12. Для внесення тестових даних в базу в метод loadData (клас DemoConfiguration) додайте ще два параметри:

```
public CommandLineRunner loadData(
        StudentService studentService,
        CuratorService curatorService,
        TeacherService teacherService
){
```

13. В тілі метода додайте викладача, куратора та зв’яжіть їх:

```
Teacher teacher01 = new Teacher(
        "Альберт Айнштайн",
        new Address(
            "01004",
```

```
            "Main street",
            "London",
              Country.UK
        ),
        new GregorianCalendar(1879, Calendar.MARCH, 14)
);
teacher01=teacherService.save(teacher01);

Curator curator01=new Curator(10);
curator01 = curatorService.save(curator01);

teacher01.setCurator(curator01);
teacher01=teacherService.save(teacher01);
```

14. Додайте викладача та куратора іншим способом:

```
Teacher teacher02 = new Teacher(
        "Ервін Шредінгер",
        new Address(
                "22007",
                "First street",
                "Bristol",
                Country.UK
        ),
        new GregorianCalendar(1887, Calendar.AUGUST, 12)
);
Curator curator02=new Curator(7);
teacher02.setCurator(curator02);
teacher02=teacherService.save(teacher02);
```

Завдання: Подумайте, який спосіб кращий з точки зору поставленої задачі?

16. Запустіть проєкт. Перевірте його роботу. Перевірте структуру та вміст бази даних.

Задача: В базу даних додати поняття Курс (в значенні дисципліна, предмет) та зберігати для кожного студента список курсів, які він закінчив.
Оскільки кожен студент може закінчити кілька курсів, а кожен курс може закінчити кілька студентів, то доцільно використати зв’язок **багато-до багатьох**.

17. Створіть сутність Course:

```
import jakarta.persistence.*;
import lombok.EqualsAndHashCode;
import lombok.Getter;
import lombok.NoArgsConstructor;
import lombok.experimental.FieldDefaults;
import org.hibernate.annotations.GenericGenerator;
```

```
import static lombok.AccessLevel.PRIVATE;

@Entity
@Table(name = "course")
@Getter
@NoArgsConstructor
@EqualsAndHashCode(of = "id")
@FieldDefaults(level = PRIVATE)
public class Course {
    @Id
    @GeneratedValue(strategy = GenerationType.AUTO, generator = "native")
    @GenericGenerator(name = "native", strategy = "native")
    @Column(updatable = false)
    Long id;

    @Column(name="course_name")
    String name;

    @Column(name = "credits")
    Double credits;
}
```

18. В клас Student додайте поле:

```
@ManyToMany
@JoinTable(
        name = "finish_course",
        joinColumns = @JoinColumn(name = "student_id"),
        inverseJoinColumns = @JoinColumn(name = "course_id"))
List<Course> finishedCourses;
```

19. В клас Course додайте поле:

```
@ManyToMany(mappedBy = "finishedCourses")
List<Student> finishedStudents;
```

20. Запустіть проєкт, подивіться на структуру бази даних.

Задача: Для курсу вивести список студентів, що його закінчили.

21. В клас Student додайте метод:

```
public void addCourse(Course course){
    finishedCourses.add(course);
}
```

22. В конструктор класу Student додайте рядок:

```
this.finishedCourses=new ArrayList<>();
```

23. Додайте репозиторій без власних методів.

24. Додайте клас:

```
@Service
@RequiredArgsConstructor
public class CourseService {
    private final CourseRepository courseRepository;

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

    public List<Student> getStudentsByCourseId(Long courseId){
        Optional<Course> course=courseRepository.findById(courseId);
        if(course.isPresent()){
            return course.get().getFinishedStudents();
        }else{
            throw new RuntimeException("Course doesn't exist, id="+courseId);
        }
    }
}
```

25. В пакеті api створіть клас CourseController:

```
@RestController
@RequestMapping("api/v1/courses")
@RequiredArgsConstructor
public class CourseController {
    private final CourseService courseService;

    @GetMapping("{id}/students")
    public ResponseEntity<List<Student>>
getStudentsByCourse(@PathVariable Long id){
        try{
            List <Student> students=courseService.getStudentsByCourseId(id);
            return new ResponseEntity<>(students, HttpStatus.OK);
        }catch (Exception e){
            return new
ResponseEntity(e.getMessage(),HttpStatus.INTERNAL_SERVER_ERROR);
        }
    }
}
```

26. В класі DemoConfiguration додайте необхідні параметри в метод, а також кілька курсів, наприклад:

```
Course course01=new Course("Java", 8.0);
course01=courseService.save(course01);
Course course02=new Course("Python",5.0);
course02=courseService.save(course02);
Course course03=new Course("C#",6.5);
course03=courseService.save(course03);
```

27. Додайте курси до існуючих студентів в , наприклад:

```
student01.addCourse(course01);
student01.addCourse(course02);
student01.addCourse(course03);
student01=studentService.save(student01);

student02.addCourse(course01);
student02=studentService.save(student02);

student03.addCourse(course01);
student03.addCourse(course02);
student03=studentService.save(student03);
```

28. Запустіть проєкт, дослідіть вміст бази, перевірте роботу метода.

Завдання: Вивід результату в браузері буде несподівано великим. Спробуйте зрозуміти, в чому проблема. Ця проблема буде вирішена на наступних заняттях.

Задача: Дозволити студентам призначати куратора.

Оскільки кожен студент має одного куратора, а кожен куратор може курувати багатьма студентами, то доцільно використати зв’язок **один-до-багатьох**.

28. В клас Student додайте поле:

```
@ManyToOne
@JoinColumn(name="curator_id")
private Curator curator;
```

29. Додайте сетер для цього поля.

30. В клас Curator додайте поле:

```
@OneToMany(mappedBy = "curator")
List<Student>students;
```

Задача: Вивести ім’я куратора по id студента.

31. В клас DemoConfiguration додайте тестові дані:

```
student01.setCurator(teacher01.getCurator());
student01=studentService.save(student01);

student02.setCurator(teacher01.getCurator());
student02=studentService.save(student02);

student03.setCurator(teacher02.getCurator());
student03=studentService.save(student03);
```

32. В клас StudentService додайте метод:

```
public String getCuratorName(Long studentId){
    Optional<Student> student=studentRepository.findById(studentId);
    if(student.isPresent()){
        Curator curator=student.get().getCurator();
        if(curator!=null){
            return curator.getTeacher().getName();
        }else{
            return "This student doesn't have a curator";
        }
    }else{
        throw new RuntimeException("Student doesn't exist, id="+studentId);
    }
}
```

33. В клас StudentController додайте метод:

```
@GetMapping("{id}/curator/name")
public ResponseEntity<String> getCuratorNameByStudentId(@PathVariable
Long id){
    try{
        String name=studentService.getCuratorName(id);
        return new ResponseEntity<>(name,HttpStatus.OK);
    }catch (Exception e){
        return new
ResponseEntity(e.getMessage(),HttpStatus.INTERNAL_SERVER_ERROR);
    }
}
```

34. Запустіть проєкт, перевірте роботу методів та вміст бази даних.

Частина В.

35. Для класу свого web-сервісу реалізуйте зв’язок один-до-багатьох (багато-до-одного) з деяким новим класом.