Advanced Java Report

Doujana Alhabib

11/12/2019

This project is an (API, Web-based) that helps in dealing with quiz assessments. Now we have two scenarios, the first one is the teacher/professor who will make the quiz inside it ,I have created the MCQ question, of course the professor he has the ability to filter, search for the questions and delete them and the other scenario is that the student he is going to have the quiz and on the real time the final result.

BACK End Work:

For now, I have created these tables for the Rest API:

-Student

-Professor

-Option

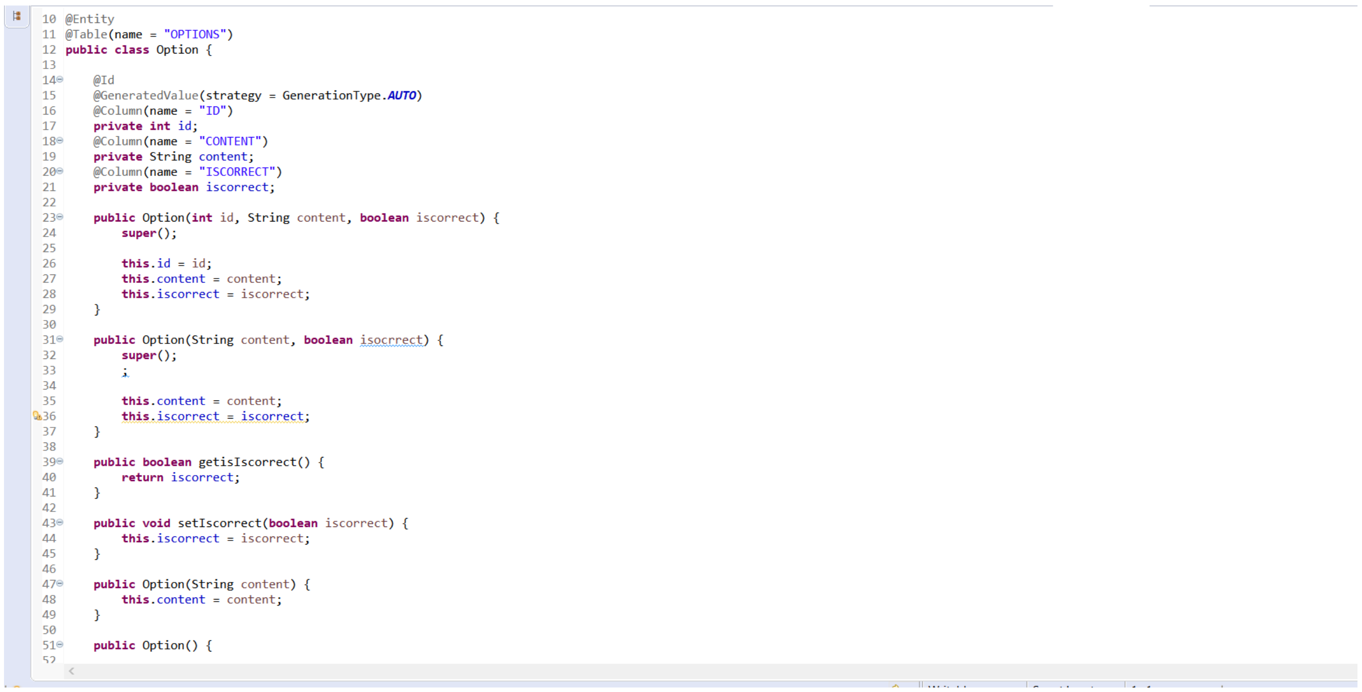
- Answer

-Question

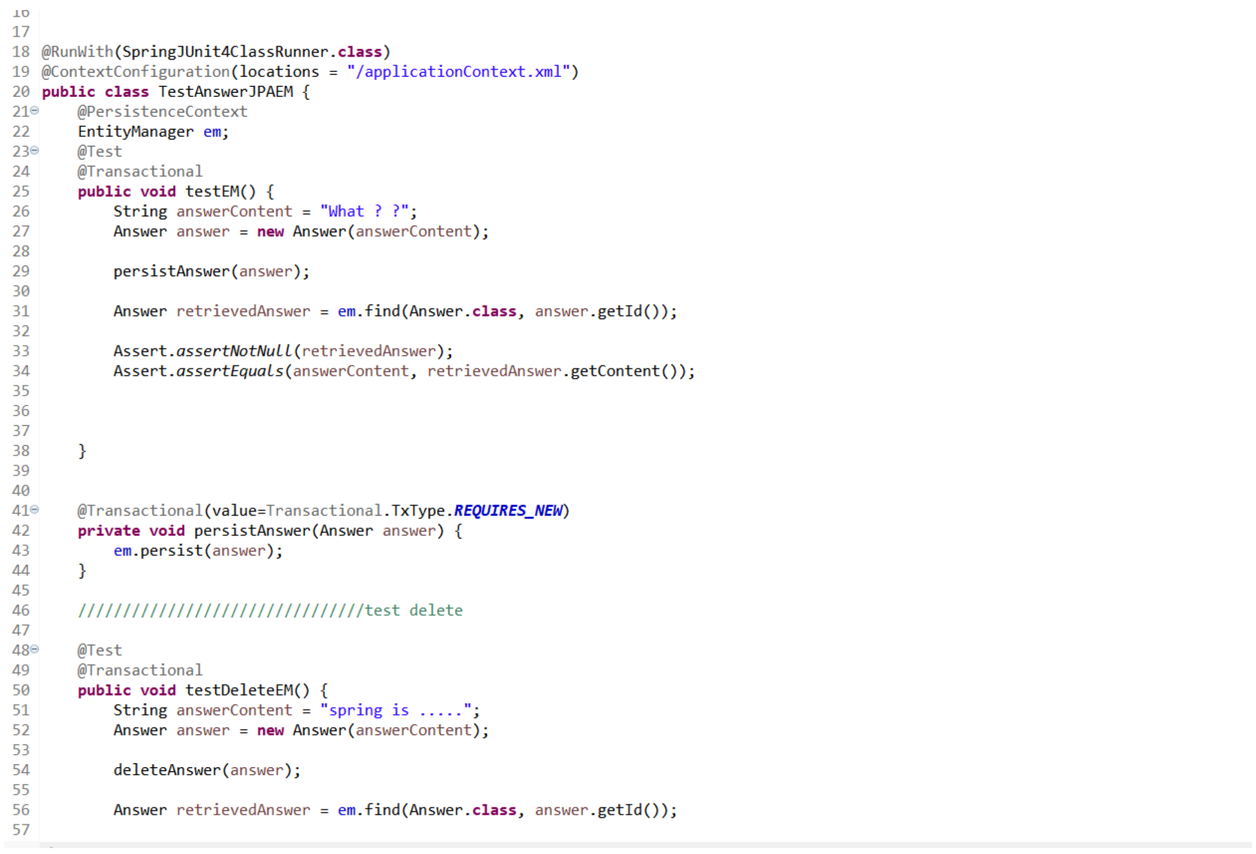
-Quiz

For each table I have provided the attributes ,the methods and I made the relationship between the tables using Hibernate.

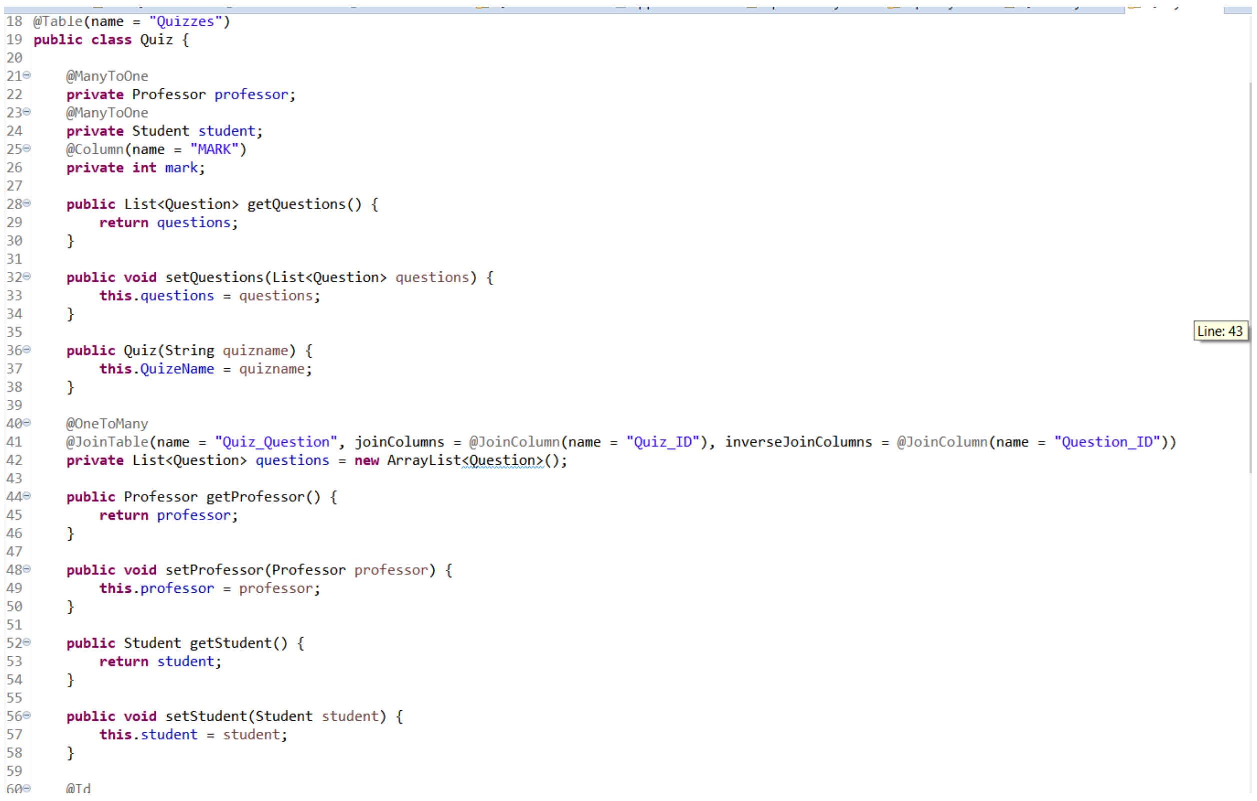
And that's is a picture showing the code for some of the tables as an example I attached the picture of the quiz and the option:



```java
@Entity
@Table(name = "OPTIONS")
public class Option {

    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    @Column(name = "ID")
    private int id;
    @Column(name = "CONTENT")
    private String content;
    @Column(name = "ISCORRECT")
    private boolean iscorrect;

    public Option(int id, String content, boolean iscorrect) {
        super();

        this.id = id;
        this.content = content;
        this.iscorrect = iscorrect;
    }

    public Option(String content, boolean isocrrect) {
        super();
        ;

        this.content = content;
        this.iscorrect = iscorrect;
    }

    public boolean getisIscorrect() {
        return iscorrect;
    }

    public void setIscorrect(boolean iscorrect) {
        this.iscorrect = iscorrect;
    }

    public Option(String content) {
        this.content = content;
    }

    public Option() {
```

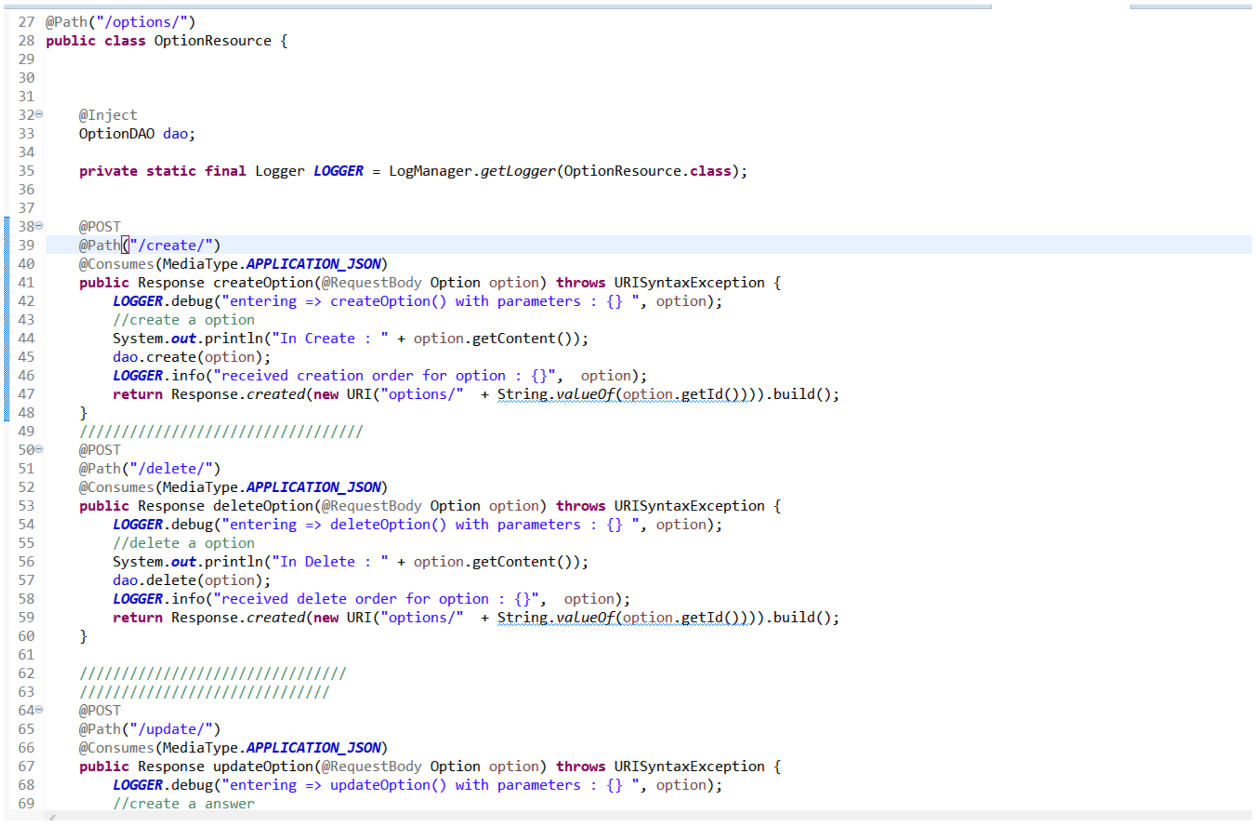

```
@RunWith(SpringJUnit4ClassRunner.class)
@ContextConfiguration(locations = "/applicationContext.xml")
public class TestAnswerJPAEM {
    @PersistenceContext
    EntityManager em;
    @Test
    @Transactional
    public void testEM() {
        String answerContent = "What ? ?";
        Answer answer = new Answer(answerContent);

        persistAnswer(answer);

        Answer retrievedAnswer = em.find(Answer.class, answer.getId());

        Assert.assertNotNull(retrievedAnswer);
        Assert.assertEquals(answerContent, retrievedAnswer.getContent());



    }


    @Transactional(value=Transactional.TxType.REQUIRES_NEW)
    private void persistAnswer(Answer answer) {
        em.persist(answer);
    }

    ////////////////////////////////test delete

    @Test
    @Transactional
    public void testDeleteEM() {
        String answerContent = "spring is .....";
        Answer answer = new Answer(answerContent);

        deleteAnswer(answer);

        Answer retrievedAnswer = em.find(Answer.class, answer.getId());

```

I made the test files also to test the EM for my tables, and here is how I made it as an example I have the answer table



```java
@Table(name = "Quizzes")
public class Quiz {

    @ManyToOne
    private Professor professor;
    @ManyToOne
    private Student student;
    @Column(name = "MARK")
    private int mark;

    public List<Question> getQuestions() {
        return questions;
    }

    public void setQuestions(List<Question> questions) {
        this.questions = questions;
    }

    public Quiz(String quizname) {
        this.QuizeName = quizname;
    }

    @OneToMany
    @JoinTable(name = "Quiz_Question", joinColumns = @JoinColumn(name = "Quiz_ID"), inverseJoinColumns = @JoinColumn(name = "Question_ID"))
    private List<Question> questions = new ArrayList<Question>();

    public Professor getProfessor() {
        return professor;
    }

    public void setProfessor(Professor professor) {
        this.professor = professor;
    }

    public Student getStudent() {
        return student;
    }

    public void setStudent(Student student) {
        this.student = student;
    }

    @Id
```

I have the listed Resources for the REST API in and this is a screenshot of it:

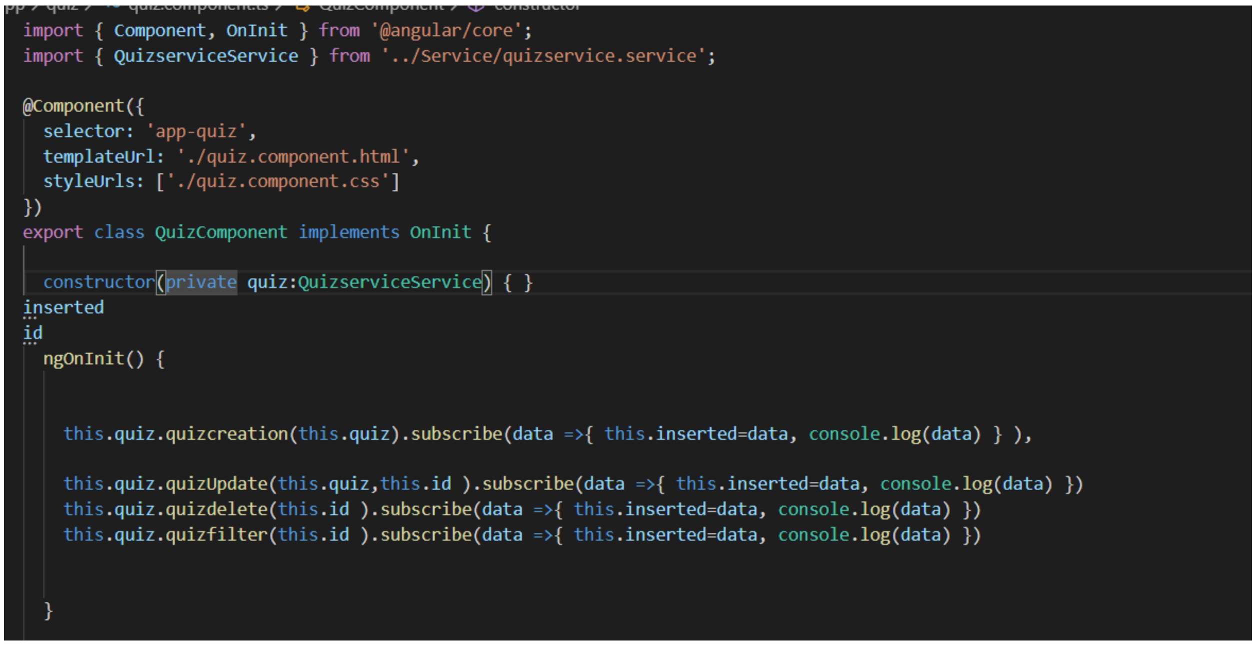


```java
@Path("/options/")
public class OptionResource {



    @Inject
    OptionDAO dao;

    private static final Logger LOGGER = LogManager.getLogger(OptionResource.class);


    @POST
    @Path("/create/")
    @Consumes(MediaType.APPLICATION_JSON)
    public Response createOption(@RequestBody Option option) throws URISyntaxException {
        LOGGER.debug("entering => createOption() with parameters : {} ", option);
        //create a option
        System.out.println("In Create : " + option.getContent());
        dao.create(option);
        LOGGER.info("received creation order for option : {}",  option);
        return Response.created(new URI("options/"  + String.valueOf(option.getId()))).build();
    }
    /////////////////////////////////
    @POST
    @Path("/delete/")
    @Consumes(MediaType.APPLICATION_JSON)
    public Response deleteOption(@RequestBody Option option) throws URISyntaxException {
        LOGGER.debug("entering => deleteOption() with parameters : {} ", option);
        //delete a option
        System.out.println("In Delete : " + option.getContent());
        dao.delete(option);
        LOGGER.info("received delete order for option : {}",  option);
        return Response.created(new URI("options/"  + String.valueOf(option.getId()))).build();
    }

    ///////////////////////////////
    /////////////////////////////
    @POST
    @Path("/update/")
    @Consumes(MediaType.APPLICATION_JSON)
    public Response updateOption(@RequestBody Option option) throws URISyntaxException {
        LOGGER.debug("entering => updateOption() with parameters : {} ", option);
        //create a answer
```

Front End

For the front end We have the two pages one for the teacher and the other one for the student who will get the quiz

I have created all the services and components for the project, and this is a screenshot for one component.



```
import { Component, OnInit } from '@angular/core';
import { QuizserviceService } from '../Service/quizservice.service';

@Component({
  selector: 'app-quiz',
  templateUrl: './quiz.component.html',
  styleUrls: ['./quiz.component.css']
})
export class QuizComponent implements OnInit {

  constructor(private quiz:QuizserviceService) { }
inserted
id
  ngOnInit() {


    this.quiz.quizcreation(this.quiz).subscribe(data =>{ this.inserted=data, console.log(data) } ),

    this.quiz.quizUpdate(this.quiz,this.id ).subscribe(data =>{ this.inserted=data, console.log(data) })
    this.quiz.quizdelete(this.id ).subscribe(data =>{ this.inserted=data, console.log(data) })
    this.quiz.quizfilter(this.id ).subscribe(data =>{ this.inserted=data, console.log(data) })


  }
```

The service which I have made for the resources and we have the CRUD for them