7.Quiz Application 3

Handling:

i) add questions
ii) getQuestionByCategory
ii) update questions by id
iii) delete questions id

Controller Layer without ResponseEntity and Exception Handling

The controller layer handles the endpoints and calls the service methods. In this version, the controller directly returns the objects or messages without using ResponseEntity. If any error occurs, the response would be a basic error message, and there is no fine control over HTTP response codes.

```
@RestController
@RequestMapping("question")
public class QuestionController {

    @Autowired
    QuestionService questionService;

    // Get all questions
    @GetMapping("allQuestions")
    public List<Question> getAllQuestions() {
        // Returning the list of questions directly
        return questionService.getAllQuestions();
    }

    // Get questions by category
    @GetMapping("category/{category}")
    public List<Question> getQuestionsByCategory(@PathVariable String category) {
        // Returning the list of questions by category directly
        return questionService.getQuestionsByCategory(category);
    }

    // Add a new question
    @PostMapping("add")
    public String addQuestion(@RequestBody Question question) {
        // Returning the success or failure message directly
        return questionService.addQuestion(question);
    }

    // Update a question by ID
    @PutMapping("update/{id}")
    public String updateQuestion(@PathVariable Integer id, @RequestBody Question question)
{
        // Directly returning the update message (success or failure)
        return questionService.updateQuestion(id, question);
    }

    // Delete a question by name
    @DeleteMapping("delete/name/{name}")
    public String deleteQuestionByName(@PathVariable String name) {
        // Returning the deletion result (success or failure)
        return questionService.deleteQuestionByName(name);
    }

    // Delete a question by ID
    @DeleteMapping("delete/{id}")
    public String deleteQuestionById(@PathVariable Integer id) {
        // Returning the deletion result (success or failure)
        return questionService.deleteQuestionById(id);
    }
}
```

Service Layer without ResponseEntity and Exception Handling.

```
@Service
public class QuestionService {

    @Autowired
    QuestionRepository questionRepository;

    // Get all questions without response entity
    public List<Question> getAllQuestions() {
        // Direct return of all questions from the repository
        return questionRepository.findAll();
    }

    // Get questions by category
    public List<Question> getQuestionsByCategory(String category) {
        // Returning the result without error checking or exception handling
        return questionRepository.findByCategory(category);
    }

    // Add a question without response entity
    public String addQuestion(Question question) {
        Question out = questionRepository.save(question);
        // Directly returning a success message or null check without detailed status codes
        return out != null ? "Question added successfully with id: " + out.getId() : "Failed to add 
question";
    }

    // Update question without response entity
    public String updateQuestion(Integer id, Question updatedQuestion) {
        Question existingQuestion = questionRepository.findById(id).orElse(null);
        if (existingQuestion != null) {
            // Updating and saving the existing question
            existingQuestion.setQuestionTitle(updatedQuestion.getQuestionTitle());
            existingQuestion.setOption1(updatedQuestion.getOption1());
            existingQuestion.setOption2(updatedQuestion.getOption2());
            existingQuestion.setOption3(updatedQuestion.getOption3());
            existingQuestion.setOption4(updatedQuestion.getOption4());
            existingQuestion.setCategory(updatedQuestion.getCategory());
            existingQuestion.setRightAnswer(updatedQuestion.getRightAnswer());
            existingQuestion.setDifficultyLevel(updatedQuestion.getDifficultyLevel());
            questionRepository.save(existingQuestion);
            return "Question updated successfully";
        } else {
            return "Question not found";
        }
    }

    // Delete question by name
    public String deleteQuestionByName(String name) {
        List<Question> questions = questionRepository.findByQuestionTitle(name);
        if (!questions.isEmpty()) {
            questionRepository.deleteAll(questions);
            return "Questions with name '" + name + "' deleted successfully";
        } else {
            return "Question not found";
        }
    }

    // Delete question by ID
    public String deleteQuestionById(Integer id) {
        if (questionRepository.existsById(id)) {
            questionRepository.deleteById(id);
            return "Question deleted successfully";
        } else {
            return "Question not found";
```

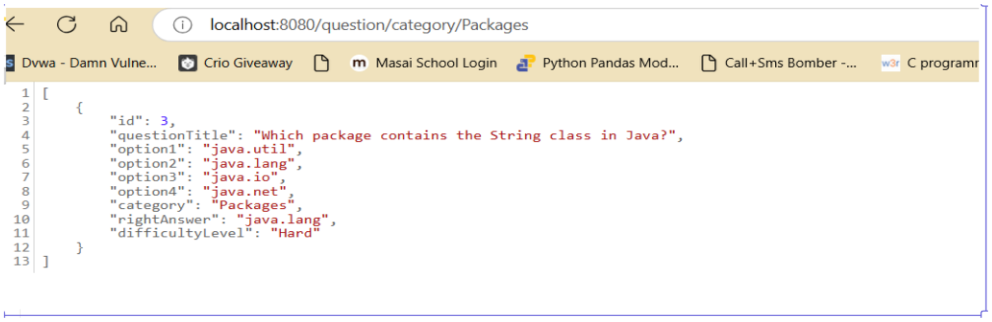
```
      }
    }
}
```

DAO Layer

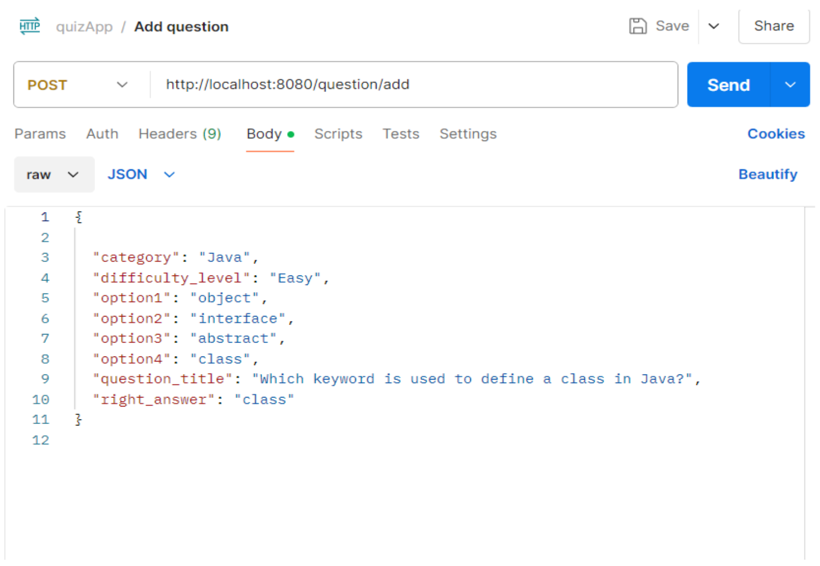
```
@Repository
public interface QuestionRepository extends JpaRepository<Question, Integer> {
   public List<Question> findByCategory(String category);

   public List<Question> findByQuestionTitle(String questionTitle);

}
```

Postman:

Question by Category:



```
[
    {
        "id": 3,
        "questionTitle": "Which package contains the String class in Java?",
        "option1": "java.util",
        "option2": "java.lang",
        "option3": "java.io",
        "option4": "java.net",
        "category": "Packages",
        "rightAnswer": "java.lang",
        "difficultyLevel": "Hard"
    }
]
```



POST http://localhost:8080/question/add

```
{

  "category": "Java",
  "difficulty_level": "Easy",
  "option1": "object",
  "option2": "interface",
  "option3": "abstract",
  "option4": "class",
  "question_title": "Which keyword is used to define a class in Java?",
  "right_answer": "class"
}
```