8.ResponseEntity and Exception

In the Spring Boot application, ResponseEntity and HttpStatus are used to define the HTTP response structure, providing greater control over the returned status codes and content.

1. ResponseEntity:

- ResponseEntity<T> represents the entire HTTP response: status code, headers, and body.
- It is an extension of HttpEntity that adds the ability to define an HTTP status code explicitly.
- **Advantage**: Gives full control over the HTTP response and is widely used for building RESTful APIs.

2. HttpStatus:

- An enumeration in Spring used to represent HTTP status codes (e.g., 200 OK, 404 NOT FOUND, 500 INTERNAL SERVER ERROR).
- It is used in combination with ResponseEntity to return specific status codes based on the outcome of the operation.

Code Update in Service Layer with ResponseEntity:

```
@Service
public class QuestionService {

    @Autowired
    QuestionRepository questionRepository;

    // Get all questions
    public ResponseEntity<List<Question>> getAllQuestions() {
        try {
            return new ResponseEntity<>(questionRepository.findAll(), HttpStatus.OK);
        } catch (Exception e) {
            e.printStackTrace();
        }
        return new ResponseEntity<>(new ArrayList<>(), HttpStatus.BAD_REQUEST);
    }
```

```
    // Get questions by category
    public ResponseEntity<List<Question>> getQuestionsByCategory(String category) {
        try {
            return new ResponseEntity<>(questionRepository.findByCategory(category),  HttpStatus.OK);
        } catch (Exception e) {
            e.printStackTrace();
        }
        return new ResponseEntity<>(new ArrayList<>(), HttpStatus.BAD_REQUEST);
    }

    // Add a new question
    public ResponseEntity<String> addQuestion(Question question) {
        Question out = questionRepository.save(question);
        if (out != null)
            return new ResponseEntity<>("Question added successfully with id: " + out.getId(),
HttpStatus.CREATED);
        else
            return new ResponseEntity<>("Something went wrong", HttpStatus.NO_CONTENT);
    }

    // Update an existing question by ID
    public ResponseEntity<String> updateQuestion(Integer id, Question updatedQuestion) {
        try {
            Question existingQuestion = questionRepository.findById(id).orElse(null);
            if (existingQuestion != null) {
                // Update question details
                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 new ResponseEntity<>("Question updated successfully", HttpStatus.OK);
            } else {
                return new ResponseEntity<>("Question not found", HttpStatus.NOT_FOUND);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
        return new ResponseEntity<>("Something went wrong", HttpStatus.BAD_REQUEST);
    }
```

```
    // Delete a question by name
    public ResponseEntity<String> deleteQuestionByName(String name) {
        try {
            List<Question> questions = questionRepository.findByQuestionTitle(name);
            if (!questions.isEmpty()) {
                questionRepository.deleteAll(questions);
                return new ResponseEntity<>("Questions with name '" + name + "' deleted successfully",
HttpStatus.OK);
            } else {
                return new ResponseEntity<>("Question with name '" + name + "' not found",
HttpStatus.NOT_FOUND);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
        return new ResponseEntity<>("Something went wrong", HttpStatus.BAD_REQUEST);
    }

    // Delete a question by ID
    public ResponseEntity<String> deleteQuestionById(Integer id) {
        try {
            if (questionRepository.existsById(id)) {
                questionRepository.deleteById(id);
                return new ResponseEntity<>("Question deleted successfully", HttpStatus.OK);
            } else {
                return new ResponseEntity<>("Question with ID " + id + " not found",
HttpStatus.NOT_FOUND);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
        return new ResponseEntity<>("Something went wrong", HttpStatus.BAD_REQUEST);
    }
}
```

Code Update in Controller Layer with ResponseEntity:

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

    @Autowired
    QuestionService questionService;

    // Get all questions
    @GetMapping("allQuestions")
    public ResponseEntity<List<Question>> getAllQuestions() {
        return questionService.getAllQuestions();
    }

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

    // Add a new question
    @PostMapping("add")
    public ResponseEntity<String> addQuestion(@RequestBody Question question) {
        return questionService.addQuestion(question);
    }

    // Update a question by ID using PUT
    @PutMapping("update/{id}")
    public ResponseEntity<String> updateQuestion(@PathVariable Integer id, @RequestBody
Question question) {
        return questionService.updateQuestion(id, question);
    }

    // Delete a question by name using DELETE
    @DeleteMapping("delete/name/{name}")
    public ResponseEntity<String> deleteQuestionByName(@PathVariable String name) {
        return questionService.deleteQuestionByName(name);
    }

    // Delete a question by ID using DELETE
    @DeleteMapping("delete/{id}")
    public ResponseEntity<String> deleteQuestionById(@PathVariable Integer id) {
        return questionService.deleteQuestionById(id);
    }
}
```

Explanation of HTTP Status Codes:

1. **Informational responses (100 - 199)**: Rarely used, they indicate an interim response.
2. **Successful responses (200 - 299)**: Indicate that the client's request was successfully received, understood, and accepted.
    - **200 OK**: The request was successful.
    - **201 Created**: The request was successful, and a new resource was created (used in POST requests).
    - **204 No Content**: The server successfully processed the request, but there's no content to return.
3. **Redirection messages (300 - 399)**: Indicates the client must take additional action to complete the request.
4. **Client error responses (400 - 499)**: Indicates a bad request from the client.
    - **400 Bad Request**: The server could not understand the request due to invalid syntax.
    - **404 Not Found**: The server can't find the requested resource.
5. **Server error responses (500 - 599)**: Indicates that the server encountered an error.
    - **500 Internal Server Error**: A generic server error occurred.
    - **502 Bad Gateway**: The server was acting as a gateway and got an invalid response.