

# Quiz Questions

## (1) Implementing the Delete Method

← Classroom

REST API - PUT and DELETE  
Fill the missing annotation

?

V

Problem Submissions Doubts

Implementing the DELETE Method

Easy • Score 20/20

Send feedback

Problem statement

You are building a RESTful API for a Blog application using Spring Boot. You want to allow users to update existing blog content. You have written the following handler method to handle the update request:

```
----- MISSING STATEMENT -----
public ResponseEntity<TodoItem> updateTodoItem(@PathVariable long id,
@RequestBody TodoItem updatedItem) {
    // update the to-do item with the given ID
    // return the updated to-do item
}
```

What annotation will come in the place of MISSING STATEMENT?

Options: Pick one correct answer from below

☐ @PostMapping("/{todo/{id}}")

☒ @PutMapping("/{todo/{id}}")

☐ @GetMapping("/{todo/{id}}")

☐ @PutMapping("/{todo}")

Solution description

@PutMapping is used to update existing data, and we can get the id from the URI by using @PathVariable.

← Classroom

REST API - PUT and DELETE  
Fill the missing annotation

?

V

Problem Submissions Doubts

Implementing the DELETE Method

Easy • Score 20/20

Send feedback

Problem statement

You are building a RESTful API for a Blog application using Spring Boot. You want to allow users to update existing blog content. You have written the following handler method to handle the update request:

```
----- MISSING STATEMENT -----
public ResponseEntity<TodoItem> updateTodoItem(@PathVariable long id,
@RequestBody TodoItem updatedItem) {
    // update the to-do item with the given ID
    // return the updated to-do item
}
```

What annotation will come in the place of MISSING STATEMENT?

Options: Pick one correct answer from below

☐ @PostMapping("/{todo/{id}}")

☒ @PutMapping("/{todo/{id}}")

☐ @GetMapping("/{todo/{id}}")

☐ @PutMapping("/{todo}")

Solution description

@PutMapping is used to update existing data, and we can get the id from the URI by using @PathVariable.

## (2) Exception Handling

← Classroom

REST API - PUT and DELETE  
Exception Handling

?

V

Problem Submissions Doubts

Exception Handling

Easy • Score 20/20

Send feedback

Problem statement

Match the Error codes with their respective meanings.

A. 500	1. Not Found
B. 404	2. Bad Request
C. 200	3. Internal Server Error
D. 400	4. OK

Options: Pick one correct answer from below

☐ (A-3)(B-2)(C-4)(D-1)

☐ (A-2)(B-3)(C-4)(D-1)

☐ (A-2)(B-3)(C-1)(D-4)

☒ (A-3)(B-1)(C-4)(D-2)

Solution description

HTTP status codes indicate whether the request was successful or not. These are the meanings of some status codes - 404 - Requested data is not found, 200 - The request was successful, 500 - The server was unable to perform the request due to an internal error, 400 - Indicates that the request was not sent in the correct format or syntax.

← Classroom

REST API - PUT and DELETE  
Exception Handling

?

V

Problem Submissions Doubts

Exception Handling

Easy • Score 20/20

Send feedback

Problem statement

Match the Error codes with their respective meanings.

A. 500	1. Not Found
B. 404	2. Bad Request
C. 200	3. Internal Server Error
D. 400	4. OK

Options: Pick one correct answer from below

☐ (A-3)(B-2)(C-4)(D-1)

☐ (A-2)(B-3)(C-4)(D-1)

☐ (A-2)(B-3)(C-1)(D-4)

☒ (A-3)(B-1)(C-4)(D-2)

Solution description

HTTP status codes indicate whether the request was successful or not. These are the meanings of some status codes - 404 - Requested data is not found, 200 - The request was successful, 500 - The server was unable to perform the request due to an internal error, 400 - Indicates that the request was not sent in the correct format or syntax.

### (3) Exception Handling

A.

```
if(ObjectUtils.isEmpty(blogMap.get(id))) {  
    // throw Exception  
}
```

C.

```
if(blogMap.get(id) == null){  
    // throw Exception  
}
```

B.

```
if(blogMap.get(id) == null){  
    // throw Exception  
}
```

D.

```
if(ObjectUtils.isEmpty(blogMap.get(id))) {  
    // throw Exception  
}
```

← Classroom

REST API - PUT and DELETE  
Exception Handling

?

V

:

Problem

Submissions

Doubts

✓ Exception Handling

Easy • Score 20/20

Send feedback

Problem statement

You are building a REST API to get the Blog details from a GET request using the blog ID.  
You have made the Service Class as follows. You noticed that it works as expected when the user provides a valid blog ID.  
Now you want to provide a check to ensure the blog by the given id exists and should not be EMPTY or NULL before returning the response.  
What is the best way to check from the given options?

```
public class BlogService{  
    private List<Blog> blogList = new ArrayList<>();  
    Map<Long, Blog> blogMap = new HashMap<>();  
    public void getBlog(Long id){  
        //Insert the code LINE 1  
        return this.blogMap.get(id);  
        // Insert the code LINE 2  
    }  
  
    public void addBlog(Blog blog){  
        this.blogList.add(blog);  
        this.blogMap.put(blog.getId(), blog);  
    }  
}
```

Options: Pick one correct answer from below

☐ Add (A) on LINE 2

☐ Add (B) on LINE 1

☐ Add (C) on LINE 2

☒ Add (D) on LINE 1

Solution description

isEmpty() is a static method that belongs to the ObjectUtils class, and it provides the functionality to check if the passed object is null or empty. But in the case of blogMap.get(id) == null, it only checks if the object is null but doesn't check if the object is empty.

## (4) Exception Handling

← Classroom

REST API - PUT and DELETE  
Exception Handling

?

V

Problem Submissions Doubts

✓ Exception Handling

Easy • Score 20/20

Problem statement

Send feedback

You are creating a custom exception class for your REST API to send a response status of 404 when an object is not found. The exception class is given below.  
Custom Exception Class:

```
public class BlogNotFoundException extends RuntimeException {
    BlogNotFoundException(String message){
        super(message);
    }
}
```

Which annotation should be used for the class?

Options: Pick one correct answer from below

☐ @ResponseStatus(HttpStatus.NOT\_FOUND)

☐ @ResponseStatus(HttpStatus.BAD\_REQUEST)

☒ @ResponseStatus(HttpStatus.NOT\_FOUND) ✓

☐ @ResponseStatus(HttpStatus.BAD\_REQUEST)

Solution description

@ResponseStatus is the annotation that can be used on a method or an Exception class to mark the error code and the response message to be sent for the specified exception to the user.

## (5) Validation for RESTful Services

← Classroom

REST API - PUT and DELETE  
Choose the correct option

2 V

Problem Submissions Doubts

Validations for RESTful Services

Easy • Score 20/20

Send feedback

Problem statement

You are building a Student Model class for a RESTful web service with fields given below

FIELD NAME	DATA TYPE	RESTRICTION
Name	STRING	Length of name should be greater than 2 & less than 21
Roll Number	NUMBER	Should be greater than 0
Mobile Number	NUMBER	Mobile number should have 10 digits
Class	Number	Class should be greater than 0 and less than 13

Select the best option from the below implementations of Student Class.

Options: Pick one correct answer from below

☐

```
***
public class Student {
    @Size(min=7, max=70)
    private String name;
    @Min(1)
    private Integer rollNumber;
    @Min(10)
    @Max(10)
    private Long mobileNumber;
    @Min(1)
    @Max(12)
    private Integer class;
    //getter and setter
}
```

☐

```
***
public class Student {
    @Size(min=3, max=20)
    private String name;
    @Size(min=1)
    private Integer rollNumber;
    @Size(min=10, max=10)
    private Long mobileNumber;
    @Size(min=1, max=12)
    private Integer class;
    //getter and setter
}
```

☒

```
***
public class Student {
    @Size(min=3, max=20)
    private String name;
    @Min(1)
    private Integer rollNumber;
    @Min(1000000000)
    @Max(9999999999)
    private Long mobileNumber;
    @Min(1)
    @Max(12)
    private Integer class;
    //getter and setter
}
```

☐

```
***
public class Student {
    @Min(3)
    @Max(20)
    private String name;
    @Min(1)
    private Integer rollNumber;
    @Size(min=10, max=10)
    private Long mobileNumber;
    @Min(1)
    @Max(12)
    private Integer class;
    //getter and setter
}
```

Solution description

@Size annotation is used to constrain the length of the field, and the @Min is used on a number (byte, short, long, int and their respective wrappers) to restrict their values greater than or equal to the specified value. Similarly, @Max is used to restrict the number value should be less than or equal to the specified value.