

Points possible: 75

URL to GitHub Repository: <https://github.com/MiloGonz1990/WebAPIandSpringBoot>

URL to Public Link of your Video: <https://www.youtube.com/watch?v=Wp2GcsPmMG4>

Instructions :

1. Follow the **Coding Steps** below to complete this assignment.

1. In Spring Tool Suite (STS), or an IDE of your choice, write the code that accomplishes the objectives listed below. Ensure that the code compiles and runs as directed.

1. Create a new repository on GitHub for this week's assignment and push your completed code to this dedicated repo, including your entire Maven Project Directory (e.g., jeep-sales) and any additional files (e.g. .sql files) that you create. In addition, screenshot your ERD and push the screenshot to your GitHub repo.

2. Include the screenshots into this Assignment Document indicated by:

1. Create a video showcasing your work:

1. In this video: record and present your project verbally while showing the results of the working project.
2. Easy way to Create a video: Start a meeting in Zoom, share your screen, open Eclipse with the code and your Console window, start recording & record yourself describing and running the program showing the results.
3. Your video should be a maximum of 5 minutes.
4. Upload your video with a public link.
5. Easy way to Create a Public Video Link: Upload your video recording to YouTube with a public link.

2. In addition, please include the following in your Coding Assignment Document:

2. The requested screenshots, indicated by:
3. The URL for this week's GitHub repository.
4. The URL of the public link of your video.

3. Save the Coding Assignment Document as a .pdf and do the following:

1. Push the .pdf to the GitHub repo for this week.
2. Upload the .pdf to the LMS in your Coding Assignment Submission.

```

1 package com.promineotech.jeeptest.controller;
2
3 import java.util.List;
4
5 import org.springframework.http.HttpStatus;
6 import org.springframework.web.bind.annotation.GetMapping;
7 import org.springframework.web.bind.annotation.RequestMapping;
8 import org.springframework.web.bind.annotation.RequestParam;
9 import org.springframework.web.bind.annotation.ResponseStatus;
10
11 import com.promineotech.jeeptest.entity.Jeeptest;
12 import com.promineotech.jeeptest.entity.JeeptestModel;
13
14 import io.swagger.v3.oas.annotations.OpenAPIDefinition;
15 import io.swagger.v3.oas.annotations.Operation;
16 import io.swagger.v3.oas.annotations.Parameter;
17 import io.swagger.v3.oas.annotations.info.Info;
18 import io.swagger.v3.oas.annotations.media.Content;
19 import io.swagger.v3.oas.annotations.media.Schema;
20 import io.swagger.v3.oas.annotations.responses.ApiResponse;
21 import io.swagger.v3.oas.annotations.servers.Server;
22
23 @RequestMapping("/jeeptest")
24
25 @OpenAPIDefinition(info = @Info(title = "Jeeptest Sales Service", servers = {
26     @Server(url = "http://localhost:8080", description = "Local server.") })
27 public interface JeeptestSalesController {
28     // @formatter:off
29     @Operation(
30         summary = "Returns a list of Jeeptest",
31         description = "Returns a list of Jeeptest given an optional model and/or trim",
32         responses = {
33             @ApiResponse(
34                 responseCode = "200",
35                 description = "A list of Jeeptest is returned.",
36                 content = @Content(
37                     mediaType = "application/json",
38                     schema = @Schema(implementation = Jeeptest.class))),
39             @ApiResponse(
40                 responseCode = "400",
41                 description = "The request parameters are invalid.",
42                 content = @Content(
43                     mediaType = "application/json")),
44             @ApiResponse(
45                 responseCode = "404",
46                 description = "No Jeeptest were found with the input criteria.",
47                 content = @Content(
48                     mediaType = "application/json")),
49             @ApiResponse(
50                 responseCode = "500",
51                 description = "An unplanned error occurred.",
52                 content = @Content(
53                     mediaType = "application/json"))
54         },
55         parameters = {
56             @Parameter(
57                 name = "model",
58                 description = "The model name (e.g. WRANGLER)",
59                 required = false,
60                 allowEmptyValue = false),
61             @Parameter(
62                 name = "trim",
63                 description = "The trim level (e.g. Sport)",
64                 required = false,
65                 allowEmptyValue = false)
66         }
67     )
68     List<Jeeptest> fetchJeeptest(
69         @RequestParam JeeptestModel model,
70         @RequestParam String trim);
71
72 // @formatter: on

```

Jeeptest Sales Service

View Source

Servers

http://localhost:8080 - Local server

basic-jeeptest-sales-controller

GET /jeeptest Returns a list of Jeeptest		
Returns a list of Jeeptest given an optional model and/or trim		
Parameters		
Name	Description	
model <small>required</small>	The model name (e.g. WRANGLER)	
string (query)	Available values: WRANGLER, GRAND_CHEROKEE, CHEROKEE, COMPASS, RENEGADE, GLADIATOR, WRANGLER_4XE	
	<input type="text" value="WRANGLER"/>	
trim <small>required</small>	The trim level (e.g. Sport)	
string (query)	<input type="text" value="trim"/>	
Responses		
Code	Description	Links
200	A list of Jeeptest is returned.	No links

Responses		
Code	Description	Links
200	A list of Jeeps is returned. Media type application/json Control: Accept header Example Value Schema <pre>{ "modelID": 1, "makeID": "MINIMAL", "trimLevel": "string", "makeID": 1, "makeID": 1, "makeID": 1 }</pre>	No links
400	The request parameters are invalid.	No links
404	No jeeps were found with the input criteria. Media type application/json	No links
500	An unplanned error occurred. Media type application/json	No links

Schemas	
Jeep	<pre>{ "modelID": integer(int64), "makeID": string, "trimLevel": string, "makeID": integer(int64), "makeID": integer(int64), "makeID": integer(int64) }</pre>

```

3
4
5 import static org.assertj.core.api.Assertions.assertThat;
6 import static org.junit.jupiter.api.Assertions.*;
7 import java.util.List;
8 import org.junit.jupiter.api.Test;
9 import org.springframework.beans.factory.annotation.Autowired;
10 import org.springframework.boot.test.context.SpringBootTest;
11 import org.springframework.boot.test.web.client.TestRestTemplate;
12 import org.springframework.boot.test.web.server.LocalServerPort;
13 import org.springframework.core.ParameterizedTypeReference;
14 import org.springframework.http.HttpStatus;
15 import org.springframework.http.ResponseEntity;
16 import org.springframework.test.context.ActiveProfiles;
17 import org.springframework.test.context.jdbc.Sql;
18 import org.springframework.test.context.jdbc.SqlConfig;
19 import com.promineotech.jeeptest.support.FetchJeepTestSupport;
20 import com.promineotech.jeeptest.entity.Jeep;
21 import org.springframework.http.HttpMethod;
22
23
24 @SpringBootTest(webEnvironment = WebEnvironment.RANDOM_PORT)
25 @ActiveProfiles("test")
26 @Sql(scripts = {
27     "classpath:way/migrations/V1.0_Jeep_Schema.sql",
28     "classpath:way/migrations/V1.1_Jeep_Data.sql"
29 }, config = @SqlConfig(encoding = "utf-8"))
30 class FetchJeepTest extends FetchJeepTestSupport {
31     @Autowired
32     private TestRestTemplate restTemplate;
33
34     @LocalServerPort
35     private int serverPort;
36
37     @Test
38     void testThatJeepsAreReturnedWhenValidModelAndTrimAreSupplied() {
39         //Given: a valid model, trim, and URI
40         JeepModel model = JeepModel.MINIMAL;
41         String trim = "Sport";
42         String uri = String.format("http://localhost:%d/jeeps?model=%s&trim=%s", serverPort, model, trim);
43
44         //When: a connection is made to the URI
45         ResponseEntity<List<Jeep>> response = restTemplate.exchange(uri, HttpMethod.GET, null, new ParameterizedTypeReference<>() {});
46
47         //Then: a success (OK = 200) status code is returned
48         assertThat(response.getStatusCode()).isEqualTo(HttpStatus.OK);
49     }
50 }
51
52 }

```