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 :

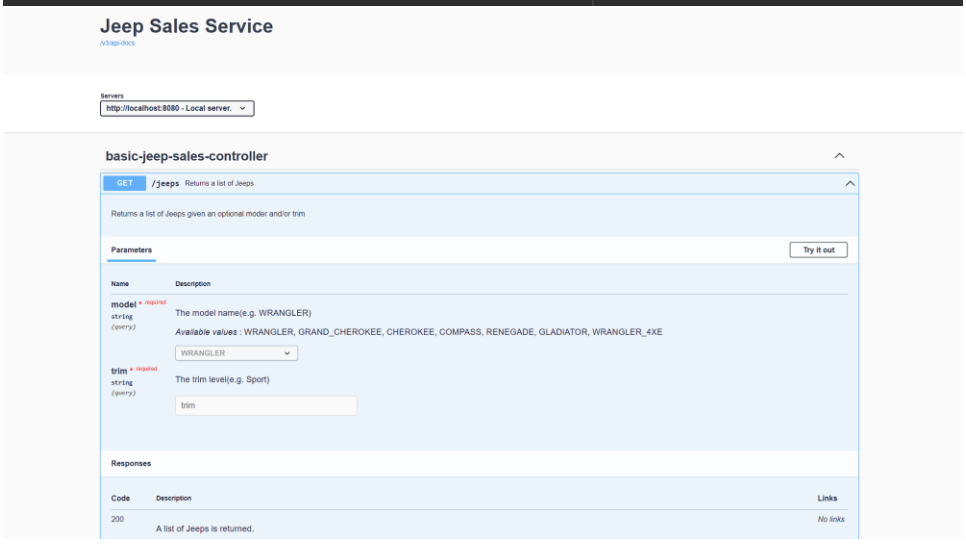
1. Follow the **Coding Steps** below to complete this assignment.

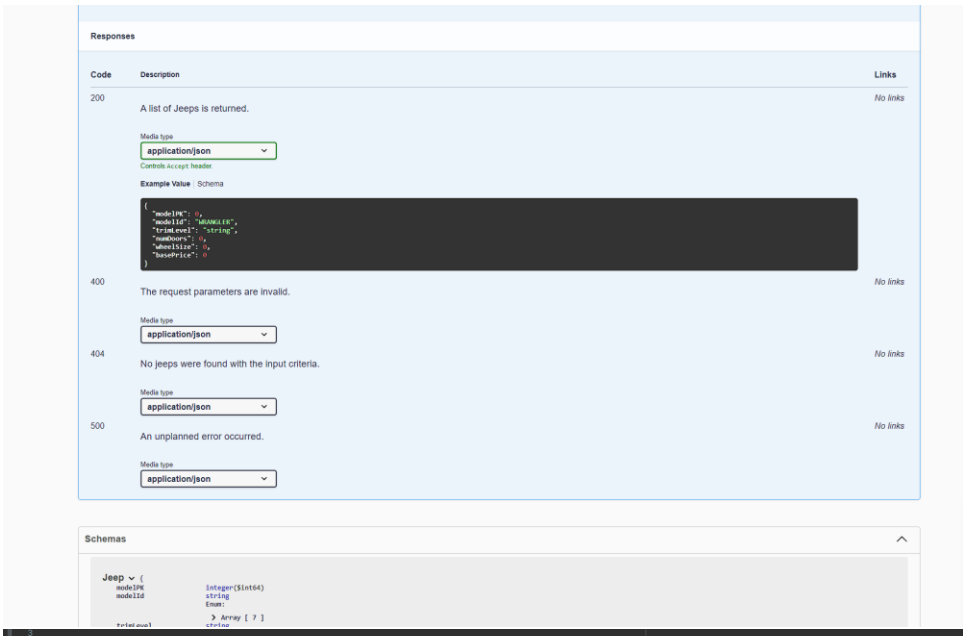
1. In Spring Tool Suite (STS), or an IDE of your choice, write the code that accomplishes the objectives listed below. Ensure that the code compiles and runs as directed.
1. Create a new repository on GitHub for this week's assignment and push your completed code to this dedicated repo, including your 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.

```java
package com.promineotech.jeep.controller;

import java.util.List;

import org.springframework.http.HttpStatus;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestParam;
import org.springframework.web.bind.annotation.ResponseStatus;

import com.promineotech.jeep.entity.Jeep;
import com.promineotech.jeep.entity.JeepModel;

import io.swagger.v3.oas.annotations.OpenAPIDefinition;
import io.swagger.v3.oas.annotations.Operation;
import io.swagger.v3.oas.annotations.Parameter;
import io.swagger.v3.oas.annotations.info.Info;
import io.swagger.v3.oas.annotations.media.Content;
import io.swagger.v3.oas.annotations.media.Schema;
import io.swagger.v3.oas.annotations.responses.ApiResponse;
import io.swagger.v3.oas.annotations.servers.Server;

@RequestMapping("/jeeps")

@OpenAPIDefinition(info = @Info(title = "Jeep Sales Service"), servers = {
    @Server(url = "http://localhost:8080", description = "Local server.") })
public interface JeepSalesController {
    //@formatter:off
    @Operation(
        summary = "Returns a list of Jeeps",
        description = "Returns a list of Jeeps given an optional moder and/or trim",
        responses = {
            @ApiResponse(
                responseCode = "200",
                description = "A list of Jeeps is returned.",
                content = @Content(
                    mediaType = "application/json",
                    schema = @Schema(implementation = Jeep.class))),
            @ApiResponse(
                responseCode = "400",
                description = "The request parameters are invalid.",
                content = @Content(
                    mediaType = "application/json")),
            @ApiResponse(
                responseCode = "404",
                description = "No jeeps were found with the input criteria.",
                content = @Content(
                    mediaType = "application/json")),
            @ApiResponse(
                responseCode = "500",
                description = "An unplanned error occurred.",
                content = @Content(
                    mediaType = "application/json"))
        },
        parameters = {
            @Parameter(
                name = "model",
    ),
    parameters = {
        @Parameter(
            name = "model",
            allowEmptyValue = false,
            required = false,
            description = "The model name(e.g. WRANGLER)"),
        @Parameter(
            name = "trim",
            allowEmptyValue = false,
            required = false,
            description = "The trim level(e.g. Sport)")

    }
        )


@GetMapping
@ResponseStatus(code = HttpStatus.OK)

List<Jeep> fetchJeeps(
    @RequestParam JeepModel model,
    @RequestParam String trim);

//@formatter:on
```






```java
import static org.assertj.core.api.Assertions.assertThat;
import static org.junit.jupiter.api.Assertions.*;
import java.util.List;
import org.junit.jupiter.api.Test;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.boot.test.context.SpringBootTest;
import org.springframework.boot.test.context.SpringBootTest.WebEnvironment;
import org.springframework.boot.test.web.client.TestRestTemplate;
import org.springframework.boot.test.web.server.LocalServerPort;
import org.springframework.core.ParameterizedTypeReference;
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.test.context.ActiveProfiles;
import org.springframework.test.context.jdbc.Sql;
import org.springframework.test.context.jdbc.SqlConfig;
import com.promineotech.jeep.controller.support.FetchJeepTestSupport;
import com.promineotech.jeep.entity.Jeep;
import com.promineotech.jeep.entity.JeepModel;
import org.springframework.http.HttpMethod;

@SpringBootTest(webEnvironment = WebEnvironment.RANDOM_PORT)
@ActiveProfiles("test")
@Sql(scripts = {
    "classpath:flyway/migrations/V1.0__Jeep_Schema.sql",
    "classpath:flyway/migrations/V1.1__Jeep_Data.sql"},
    config = @SqlConfig(encoding = "utf-8"))
class FetchJeepTest extends FetchJeepTestSupport{
  @Autowired
  private TestRestTemplate restTemplate;

  @LocalServerPort
  private int serverPort;

  @Test
  void testThatJeepsAreReturnedWhenAValidModelAndTrimAreSupplied() {
    //Given: a valid model, trim, and URI
    JeepModel model = JeepModel.WRANGLER;
    String trim = "Sport";
    String uri = String.format("http://localhost:%d/jeeps?model=%s&trim=%s", serverPort, model, trim);


    //When: a connection is made to the URI
    ResponseEntity<List<Jeep>> response = restTemplate.exchange(uri, HttpMethod.GET, null, new ParameterizedTypeReference<>() {});

    //Then: a success (OK - 200) status code is returned
    assertThat(response.getStatusCode()).isEqualTo(HttpStatus.OK);
  }

}
```