Web API Design with Spring Boot Week 1 Coding Assignment

a) Produce a screenshot showing the completed test class.

JeepSales.java | FetchJeepTest.java | JeepModel.java | Jeep.java | JeepSalesController.java | Bas

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

import static org.assertj.core.api.Assertions.assertThat;

/**
 * @author D
 *
 */
@SpringBootTest(webEnvironment = WebEnvironment.RANDOM_PORT)
@ActiveProfiles("test")
@Sql(scripts = {
    "classpath:flyway/migration/V1.0__Jeep_Schema.sql",
    "classpath:flyway/migration/V1.1__Jeep_Data.sql"},
    config=@SqlConfig(encoding = "utf-8")
    )

class FetchJeepTest {
  @Autowired
  @Getter
  private TestRestTemplate restTemplate;

  @LocalServerPort
  private int serverPort;
  //
  // Here for learning purposes
  //   String getBaseUri() {
  //    return String.format("http://localhost:%d/jeeps", serverPort);
  //  }

  @Test
  void testThatJeepsAreReturnedWhenAValidModelAndTrimAreSupplied() {
    //    fail("Not yet Implemented");
    //    System.out.println(getBaseUri());
    JeepModel model = JeepModel.WRANGLER;
    String trim = "Sport";
    String uri = String.format("http://localhost:%d/jeeps?model=%s&trim=%s",
        serverPort, model , trim);

//      ResponseEntity<Jeep> response =  getRestTemplate().getForEntity(uri, Jeep.class);
//      assertThat(response.getStatusCode() == HttpStatus.OK);
```

```
51
52      //Whoa even after reading the Super Type Token blog this was a bit mind bending
53      //But to not use an anonymous try this
54      ParameterizedTypeReference<List<Jeep>> typeRef =
55          new ParameterizedTypeReference<List<Jeep>>() {};
56
57          ResponseEntity<List<Jeep>> response
58          = restTemplate.exchange(uri, HttpMethod.GET, null, typeRef);
59  //
60  //    ResponseEntity<List<Jeep>> response
61  //        = restTemplate.exchange(uri, HttpMethod.GET, null,
62  //            new ParameterizedTypeReference<>() {});
63
64          assertThat(response.getStatusCode() == HttpStatus.OK);
65
66
67    }
68
69  }
70
```

Produce a screenshot showing the interface and OpenAPI documentation.

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

/**
 * @author D
 *
 */
@RequestMapping("/jeeps")

@OpenAPIDefinition(info = @Info(title = "Jeep Sales Service"), servers = {
    @Server(url = "http://localhost:8080", description = "Local server.")})


public interface JeepSalesController {
  @Operation(
      summary = "Returns a List of Jeeps",
      description = "Returns a list of Jeeps , optional params  model , trim",
      responses = {
          @ApiResponse(responseCode = "200", description = "Success, A list of Jeeps",
              content = @Content(
                  mediaType = "application/json" ,
                  schema  = @Schema(implementation = Jeep.class))),
          @ApiResponse(responseCode = "400", description = "Request parameters invlaid",
              content = @Content(mediaType = "application/json")),
          @ApiResponse(responseCode = "404", description = "No Jeeps found with given model/trim",
              content = @Content(mediaType = "application/json")),
          @ApiResponse(responseCode = "500", description = "Error 500 ",
              content = @Content(mediaType = "application/json"))},
```

```java
      parameters = {
          @Parameter(name = "model" , allowEmptyValue = false , description = "The model Name"),
          @Parameter(name = "trim" , allowEmptyValue = false , description = "The vehicle trim level")
      }
)

@GetMapping
@ResponseStatus(code=HttpStatus.OK)
List<Jeep> fetchJeeps(
    @RequestParam JeepModel model,
//    @RequestParam(required = false) String model,
    @RequestParam String trim
    );
}
```

Run the application within the IDE and show the resulting OpenAPI (Swagger) documentation produced in the browser. Produce a screenshot of the documentation showing all four possible outcomes. 🖥️

Jeep Sales Service

/v3/api-docs

Servers

http://localhost:8080 - Local server.

basic-jeep-sales-controller

GET /jeeps Returns a List of Jeeps

Returns a list of Jeeps , optional params model , trim

Parameters Try it out

Name	Description
model * required string (query)	The model Name *Available values* : CHEROKEE, COMPASS, GLADIATOR, GRAND_CHEROKEE, RENEGADE, WRANGLER, WRANGLER_4XE CHEROKEE
trim * required string (query)	The vehicle trim level trim

Responses

Code	Description	Links

200	Success, A list of Jeeps
	Media type
	application/json
	Controls Accept header.
	Example Value | Schema
	{
	"modelPk": 0,
	"modelID": "CHEROKEE",
	"trimLevel": "string",
	"numDoors": 0,
	"wheelSize": 0,
	"basePrice": 0
	}
		No links

400	Request parameters invlaid Media type application/json	No links
404	No Jeeps found with given model/trim Media type application/json	No links
500	Error 500 Media type application/json	No links

Schemas

Jeep ›

URL to GitHub Repository:

https://github.com/david2joh/springweek1.git