


Web API Design with Spring Boot Week 1 Coding Assignment

Points possible: 70

Category	Criteria	% of Grade
Functionality	Does the code work?	25
Organization	Is the code clean and organized? Proper use of white space, syntax, and consistency are utilized. Names and comments are concise and clear.	25
Creativity	Student solved the problems presented in the assignment using creativity and out of the box thinking.	25
Completeness	All requirements of the assignment are complete.	25

Instructions: In Eclipse, or an IDE of your choice, write the code that accomplishes the objectives listed below. Ensure that the code compiles and runs as directed. Take screenshots of the code and of the running program (make sure to get screenshots of all required functionality) and paste them in this document where instructed below. Create a new repository on GitHub for this week's assignments and push this document, with your Java project code, to the repository. Add the URL for this week's repository to this document where instructed and submit this document to your instructor when complete.

Here's a friendly tip: as you watch the videos, code along with the videos. This will help you with the homework. When a screenshot is required, look for the icon:  You will keep adding to this project throughout this part of the course. When it comes time for the final project, use this project as a starter.

Here's a hint: make sure you are running a version of Java that is 11+. To get the version, open a Windows command window or a Mac Terminal window and type `java -version`. If you need to upgrade, go here:

<https://docs.aws.amazon.com/corretto/latest/corretto-11-ug/downloads-list.html>. Pick the .msi installer version (Windows) or the .pkg version (Mac).

Project Resources:

<https://github.com/promineotech/Spring-Boot-Course-Student-Resources>

Coding Steps:

- 1) Create a Maven project named `JeepSales` as described in the video.

- a) In Spring Tool Suite, click the "File" menu. Select "New/Project...". In the popup, expand "Maven" and select "Maven Project". Click "Next".(done)
- b) Check "Create a simple project (skip archetype selection)". Click "Next". (done)
- c) Enter the following: (done)

Group Id	com.promineotech
Artifact Id	jeep-sales

Click "Finish".(done)

- 2) Navigate to the Spring Initializr (<https://start.spring.io/>).

- a) Confirm the following settings:

Project	Maven Project
Language	Java
Spring Boot	Select the latest stable version (not SNAPSHOT or RC)
Group	com.promineotech
Artifact	jeep-sales
Name	jeep-sales
Description	Jeep Sales
Package name	com.promineotech
Packaging	Jar
Java	11

- b) Add the dependencies from the Initializr:
 - i) Web
 - ii) Devtools
 - iii) Lombok(done)
- c) Click "Explore" at the bottom of the page.(done)
- d) Click "Copy" to copy the pom.xml generated by the Initializr to the clipboard.(done)

- 3) In Spring Tool Suite, open pom.xml (in the project root directory). Select all the text in the editor and replace it with the XML copied to the clipboard in the prior step. (done)
- 4) Navigate to <https://mvnrepository.com/>. Search for springdoc-openapi-ui. Select the latest version and add the entry to the POM file in the <dependencies> section. (done~)
- 5) Create a package in src/main/java named com.promineotech.jeepp. In this package: (done)
 - a) Create a Java class with a main method named JeepSales. (done)
 - b) Add a class-level annotation: @SpringBootApplication and the import statement.
 - c) In the main() method, add a call to SpringApplication.run();. Use JeepSales.class as the first parameter, and the args parameter that was passed into the main() method as the second. The entire class should look like this:

```
package com.promineotech.jeepp;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication
public class JeepSales {

    public static void main(String[] args) {
        SpringApplication.run(JeepSales.class, args);
    }
}
```

- 6) Refer to README.docx in the supplied project resources. Copy all files in the Files folder in the resources to your project as described in the README. **Do not copy the files in the Entity or Source folders at this time.**
 - a) Load the files that were added: right-click on the project in Package Explorer and select "Refresh". (yup)
 - b) Update the project with the new POM dependencies: right-click on the project in Package Explorer, select "Maven/Update Project". When the "Update Maven Project" panel appears, click "OK". (ok)
- 7) Using the MySQL Workbench or MySQL command line client (CLI), create a database named "jeepp".(done video 2)
- 8) Using dBeaver, or the MySQL client of choice, load the supplied .sql files (v1.0__Jeep_Schema.sql, and v1.1__Jeep_Data.sql) into the MySQL database to create the tables and populate them with data. These files are found in the project folder src/test/resources/flyway/migrations. (done video 3)
- 9) Create a new package in src/test/java named com.promineotech.jeepp.controller. Create a Spring Boot integration test named FetchJeepTest using the techniques shown in the video.

- a) Add the `@SpringBootTest`, `@ActiveProfiles`, and `@Sql` annotations as described in the video.
- b) The class must not be public. It should have package-level access (i.e., not public, private, or protected).
- c) The video extended `FetchJeepTestSupport`, but you don't need to do that for the homework. Just put everything in `FetchJeepTest`. It should look like this:

```
@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 {
}(done)
```

- d) Create a test method in `FetchJeepTest`. The method must have the following method signature:

```
void testThatJeepsAreReturnedWhenAValidModelAndTrimAreSupplied()
```

- e) Inject a `TestRestTemplate` in the test class. Name the variable `restTemplate`. Inject the port used in the test using the `@LocalServerPort` annotation. Name the variable `serverPort`. The variables and annotations should look like this:

```
@Autowired
private TestRestTemplate restTemplate;

@LocalServerPort
private int serverPort;(done)
```

- 10) Create a new package in `src/main/java` named `com.promineotech.jeep.entity`. In that package, create an enum named `JeepModel`. Add all the jeep models from the `model_id` column in the `models` table in the database. You can use this query in dBeaver: `SELECT DISTINCT model_id FROM models`. (done but in a weird way ask for correct way)
- 11) Create a `Jeep` class in the `com.promineotech.jeep.entity` package. Add the columns from the `models` table into this class as instance variables. Annotate the class with the Lombok annotations `@Data`, `@Builder` (and optionally both `@NoArgsConstructor` and `@AllArgsConstructor`). Note that `modelId` should be of type `JeepModel` and `basePrice` should be of type `BigDecimal`. The class should look like this (remember to add the appropriate import statements):

```
@Data
@Builder
@NoArgsConstructor
@AllArgsConstructor
public class Jeep {
    private Long modelPK;
```

```

private JeepModel modelId;
private String trimLevel;
private int numDoors;
private int wheelSize;
private BigDecimal basePrice;
} (done)

```

12) In the supplied resources, copy all files in the Entities folder to the src/main/java/com/-promineotech/jeep/entity folder. **Do not copy anything from the Source folder at this time. (done)**

13) Back in the test method that you were writing, create local variables for JeepModel, trim, and uri. Set them appropriately like this:

Variable Type	Variable Name	Variable Value
JeepModel	model	JeepModel.WRANGLER
String	trim	"Sport"
String	uri	String.format("http://localhost/%d/jeeps?model=%s&trim=%s", serverPort, model, trim);

a) Send an HTTP request to the REST service that passes a JeepModel and trim level as URI parameters (as shown in the video). Use this method call:

```

ResponseEntity<List<Jeep>> response = restTemplate.exchange(uri,
    HttpMethod.GET, null, new ParameterizedTypeReference<>() {});

```

Make sure to use the import java.util.List and org.springframework.http.HttpMethod.

b) Using [AssertJ](#), test that the response that comes back from the server is 200 (success) – or as is shown in the video: HttpStatus.OK. The code should look like this:

```

assertThat(response.getStatusCode()).isEqualTo(HttpStatus.OK);


```

Use the import statements:

```

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

```

c) Produce a screenshot showing the completed test class. 

```

JeepSales.java BaseTest.java FetchJeepTest.java x JeepModel.java Jeep.java
1 package com.promineotech.jeep.controller;
2
3 import static org.junit.jupiter.api.Assertions.*;
4 import org.junit.jupiter.api.Test;
5 import org.springframework.beans.factory.annotation.Autowired;
6 import org.springframework.boot.test.context.SpringBootTest;
7 import org.springframework.boot.test.context.SpringBootTest.WebEnvironment;
8 import org.springframework.boot.test.web.client.TestRestTemplate;
9 import org.springframework.boot.web.server.LocalServerPort;
10 import org.springframework.core.ParameterizedTypeReference;
11 import org.springframework.http.HttpMethod;
12 import org.springframework.http.HttpStatus;
13 import org.springframework.http.ResponseEntity;
14 import org.springframework.test.context.ActiveProfiles;
15 import org.springframework.test.context.jdbc.Sql;
16 import org.springframework.test.context.jdbc.SqlConfig;
17 import com.promineotech.jeep.entity.Jeep;
18 import com.promineotech.jeep.entity.JeepModel;
19 //import com.promineotech.controller.support.FetchJeepTestSupport; //from video and but not needed
20 import lombok.Getter;
21 import java.util.List;
22 import static org.assertj.core.api.Assertions.assertThat;
23
24
25 //Below is the test annotation
26 //we are telling it that we want the tests to be run in a web environment AND to ensure the tests will not
27 //run on top of each other we specify the .random_port for each test class
28 //host class is always local host means it will stay within the local machine environment
29 //class FetchJeepTest extends FetchJeepTestSupport {
30     @SpringBootTest(webEnvironment = WebEnvironment.RANDOM_PORT)
31     @ActiveProfiles("test")
32     @Sql(scripts = {
33         "classpath:flyway/migrations/V1.0__Jeep_Schema.sql",
34         "classpath:flyway/migrations/V1.1__Jeep_Data.sql"},
35         config = @SqlConfig(encoding = "utf-8"))
36     class FetchJeepTest {
37         @LocalServerPort
38         private int serverPort;
39

```


```

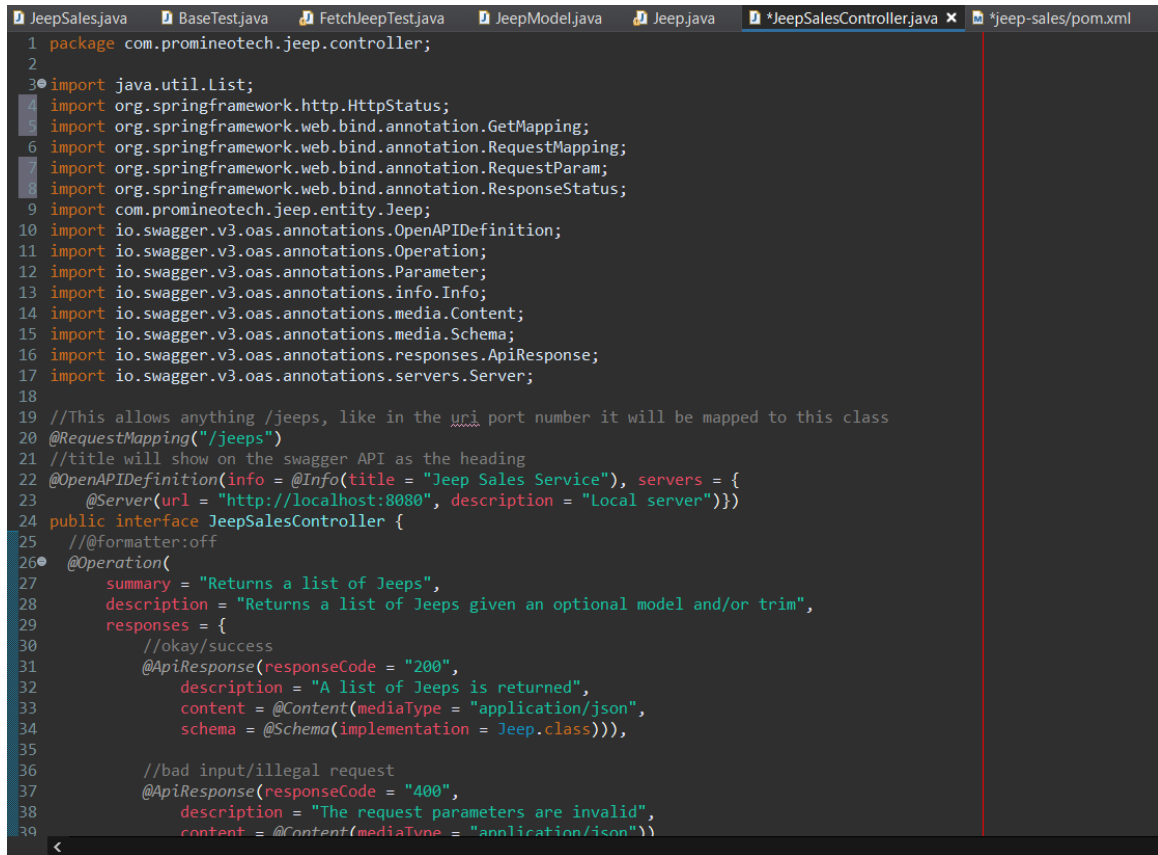
JeepSales.java BaseTest.java FetchJeepTest.java x JeepModel.java Jeep.java
26 //we are telling it that we want the tests to be run in a web environment AND to ensure the tests will not
27 //run on top of each other we specify the .random_port for each test class
28 //host class is always local host means it will stay within the local machine environment
29 //class FetchJeepTest extends FetchJeepTestSupport {
30 @SpringBootTest(webEnvironment = WebEnvironment.RANDOM_PORT)
31 @ActiveProfiles("test")
32 @Sql(scripts = {
33     "classpath:flyway/migrations/V1.0__Jeep_Schema.sql",
34     "classpath:flyway/migrations/V1.1__Jeep_Data.sql"},
35     config = @SqlConfig(encoding = "utf-8"))
36 class FetchJeepTest {
37     @LocalServerPort
38     private int serverPort;
39
40     //TEST REST TEMPLATE TO SEND THE HTTP REQUESTS
41     //This one allows a test rest template to be created for us
42     @Autowired
43     @Getter
44     private TestRestTemplate restTemplate;
45
46     @Test
47     //a test should be self describing so should have a name to what they exactly do
48     void testThatJeepsAreReturnedWhenAValidModelAndTrimAreSupplied() {
49         //Given: a valid model, trim URI
50         JeepModel model = JeepModel.WRANGLER;
51         String trim = "Sport";
52         //this makes it so it can have two parameters
53         String uri = String.format("http://localhost:%d/jeeps?model=%s&trim=%s", serverPort, model, trim);
54         System.out.println(uri);
55
56         //When: a connection is made to the URI
57         ResponseEntity<List<Jeep>> response =
58             restTemplate.exchange(uri, HttpMethod.GET, null, new ParameterizedTypeReference<>() {});
59
60         //Then: a success (OK - 200) status code is returned
61         assertThat(response.getStatusCode()).isEqualTo(HttpStatus.OK);
62     }
63 }
64 }

```

- 14) In src/main/java, create a new package com.promineotech.jeep.controller. In this package, create an interface named JeepSalesController.
 - a) Add the class-level annotation @RequestMapping("/jeeps").
 - b) Add the fetchJeeps method in a controller interface with the following signature:
`List<Jeep> fetchJeeps(JeepModel model, String trim);`
 Make sure you use the List from java.util.List.
 - c) Add OpenAPI documentation to document the four possible outcomes: 200 (success), 400 (bad input), 404 (not found) and 500 (unplanned error) as shown in the video.
 - d) Add the parameter annotations in the OpenAPI documentation to describe the model and trim parameters.
 - e) Add the @GetMapping annotation and the @ResponseStatus(code = HttpStatus.OK) annotation as method-level annotations to the fetchJeeps method.
 - f) Add the @RequestParam annotations to the parameters as described in the video. The interface should look like this (omitting the OpenAPI annotations):

```
@RequestMapping("/jeeps")
public interface JeepSalesController {
    @GetMapping
    @ResponseStatus(code = HttpStatus.OK)
    List<Jeep> fetchJeeps(@RequestParam JeepModel model,
        @RequestParam String trim);
}
```


g) Produce a screenshot showing the interface and OpenAPI documentation. 




```
1 package com.promineotech.jeepp.controller;
2
3 import java.util.List;
4 import org.springframework.http.HttpStatus;
5 import org.springframework.web.bind.annotation.GetMapping;
6 import org.springframework.web.bind.annotation.RequestMapping;
7 import org.springframework.web.bind.annotation.RequestParam;
8 import org.springframework.web.bind.annotation.ResponseStatus;
9 import com.promineotech.jeepp.entity.Jeep;
10 import io.swagger.v3.oas.annotations.OpenAPIDefinition;
11 import io.swagger.v3.oas.annotations.Operation;
12 import io.swagger.v3.oas.annotations.Parameter;
13 import io.swagger.v3.oas.annotations.info.Info;
14 import io.swagger.v3.oas.annotations.media.Content;
15 import io.swagger.v3.oas.annotations.media.Schema;
16 import io.swagger.v3.oas.annotations.responses.ApiResponse;
17 import io.swagger.v3.oas.annotations.servers.Server;
18
19 //This allows anything /jeeps, like in the url port number it will be mapped to this class
20 @RequestMapping("/jeeps")
21 //title will show on the swagger API as the heading
22 @OpenAPIDefinition(info = @Info(title = "Jeep Sales Service"), servers = {
23     @Server(url = "http://localhost:8080", description = "Local server")})
24 public interface JeepSalesController {
25     //formatter:off
26     @Operation(
27         summary = "Returns a list of Jeeps",
28         description = "Returns a list of Jeeps given an optional model and/or trim",
29         responses = {
30             //okay/success
31             @ApiResponse(responseCode = "200",
32                 description = "A list of Jeeps is returned",
33                 content = @Content(mediaType = "application/json",
34                     schema = @Schema(implementation = Jeep.class))),
35             //bad input/illegal request
36             @ApiResponse(responseCode = "400",
37                 description = "The request parameters are invalid",
38                 content = @Content(mediaType = "application/json"))
39         }
40     )
41 }
```

```

39         content = @Content(mediaType = "application/json")),
40
41         //not found
42         @ApiResponse(responseCode = "404",
43             description = "No Jeeps were found with the input criteria",
44             content = @Content(mediaType = "application/json")),
45
46         //unplanned exception
47         @ApiResponse(responseCode = "500",
48             description = "An unplanned error occurred",
49             content = @Content(mediaType = "application/json"))
50     },
51
52     parameters = {
53         //if you pass bunch of empty characters it will return it as null
54         @Parameter(name = "model",
55             allowEmptyValue = false,
56             required = false,
57             description = "The model name (i.e., 'WRANGLER')"),
58         @Parameter(name = "trim",
59             allowEmptyValue = false,
60             required = false,
61             description = "The trim level (i.e., 'Sport')")
62     }
63 )
64
65
66 //Spring will map the get request at /jeeps to the fetchjeeps method
67 @GetMapping
68 //then it will return the status 200, 400, 404, or 500
69 @ResponseStatus(code = HttpStatus.OK)
70 //Spring needs annotations to read it will map the parameter model based on name of the parameter within the method call
71 List<Jeep> fetchJeeps(
72     @RequestParam(required = false) |
73     String model,
74     @RequestParam(required = false)
75     String trim);
76 //formatter:on
77 }
78 }


```

- 15) Add the controller implementation class named DefaultJeepSalesController. Don't forget the @RestController annotation.
- 16) 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. 

← → ↻

localhost:8080/swagger-ui/index.html?configUrl=/v3/api-docs/swagger-config#/

☆ ⚙️ 👤

 **Swagger**
OpenAPI 3.0

/v3/api-docs

Explore

Jeep Sales Service

/v3/api-docs

Servers

http://localhost:8080 - Local server

default-jeep-sales-controller

GET /jeeps Returns a list of Jeeps

Schemas

Jeep >

Responses

Code	Description	Links
200	A list of Jeeps is returned <div>Media type application/json Controls Accept header: Example Value Schema <pre>{}</pre></div>	No links
400	The request parameters are invalid <div>Media type application/json</div>	No links
404	No Jeeps were found with the input criteria <div>Media type application/json</div>	No links
500	An unplanned error occurred <div>Media type application/json</div>	No links

Screenshots of Code:

```

JeepSales.java x BaseTest.java FetchJeepTestj... JeepModel.java Jeep.java JeepSalesContr...
1 package com.promineotech.jeepp;
2
3 import org.springframework.boot.SpringApplication;
4
5
6 //annotations are very important
7 //tomcat is how it runs - handles all its requests and such
8 //snapshot is a maven convention
9 //ctrl c stops it on cmd
10 @SpringBootApplication
11 public class JeepSales {
12
13     public static void main(String[] args) {
14         SpringApplication.run(JeepSales.class, args);
15     }
16 }
17
18

```

```

JeepSales.java BaseTest.java FetchJeepTestj... JeepModel.java Jeep.java JeepSalesContr... jeep-sales/po... DefaultJeepSal...
1 package com.promineotech.jeepp.controller;
2
3 import java.util.List;
4 import org.springframework.web.bind.annotation.RestController;
5 import com.promineotech.jeepp.entity.Jeep;
6
7 //We need to tell Spring that this is a restcontroller and we can only do that here
8 //This restcontroller tells spring boot that this class is special, its a controller for the jeepsalescontroller so it
9 //will look into that interface to find out that the class is mapped to /jeeps, get verb needs two parameters
10 @RestController
11 public class DefaultJeepSalesController implements JeepSalesController {
12
13     @Override
14     public List<Jeep> fetchJeeps(String model, String trim) {
15         return null;
16     }
17
18 }
19

```

```

JeepSales.java BaseTest.java FetchJeepTestj... JeepModel.java Jeep.java JeepSalesContr... x jeep-s
1 package com.promineotech.jeepp.controller;
2
3 import java.util.List;
4 import org.springframework.http.HttpStatus;
5 import org.springframework.web.bind.annotation.GetMapping;
6 import org.springframework.web.bind.annotation.RequestMapping;
7 import org.springframework.web.bind.annotation.RequestParam;
8 import org.springframework.web.bind.annotation.ResponseStatus;
9 import com.promineotech.jeepp.entity.Jeep;
10 import io.swagger.v3.oas.annotations.OpenAPIDefinition;
11 import io.swagger.v3.oas.annotations.Operation;
12 import io.swagger.v3.oas.annotations.Parameter;
13 import io.swagger.v3.oas.annotations.info.Info;
14 import io.swagger.v3.oas.annotations.media.Content;
15 import io.swagger.v3.oas.annotations.media.Schema;
16 import io.swagger.v3.oas.annotations.responses.ApiResponse;
17 import io.swagger.v3.oas.annotations.servers.Server;
18
19 //This allows anything /jeeps, like in the uri port number it will be mapped to this class
20 @RequestMapping("/jeeps")
21 //title will show on the swagger API as the heading
22 @OpenAPIDefinition(info = @Info(title = "Jeep Sales Service"), servers = {
23     @Server(url = "http://localhost:8080", description = "Local server")})
24 public interface JeepSalesController {
25     //@formatter:off
26     @Operation(
27         summary = "Returns a list of Jeeps",
28         description = "Returns a list of Jeeps given an optional model and/or trim",
29         responses = {
30             //okay/success
31             @ApiResponse(responseCode = "200",
32                 description = "A list of Jeeps is returned",
33                 content = @Content(mediaType = "application/json",
34                     schema = @Schema(implementation = Jeep.class))),
35

```

```

JeepSales.java BaseTest.java FetchJeepTestj... JeepModel.java Jeep.java JeepSalesContr... x jeep-sa
34         schema = @Schema(implementation = Jeep.class))),
35
36         //bad input/illegal request
37         @ApiResponse(responseCode = "400",
38             description = "The request parameters are invalid",
39             content = @Content(mediaType = "application/json")),
40
41         //not found
42         @ApiResponse(responseCode = "404",
43             description = "No Jeeps were found with the input criteria",
44             content = @Content(mediaType = "application/json")),
45
46         //unplanned exception
47         @ApiResponse(responseCode = "500",
48             description = "An unplanned error occurred",
49             content = @Content(mediaType = "application/json"))
50     },
51
52     parameters = {
53         //if you pass bunch of empty characters it will return it as null
54         @Parameter(name = "model",
55             allowEmptyValue = false,
56             required = false,
57             description = "The model name (i.e., 'WRANGLER')"),
58         @Parameter(name = "trim",
59             allowEmptyValue = false,
60             required = false,
61             description = "The trim level (i.e., 'Sport')")
62     }
63
64 )
65
66 //Spring will map the get request at /jeeps to the fetchjeeps method
67 @GetMapping
68 //then it will return the status 200, 400, 404, or 500
69

```

```

JeepSales.java BaseTest.java FetchJeepTestj... JeepModel.java Jeep.java JeepSalesContr... x jeep-sales/po... DefaultJeepSal...
44         content = @Content(mediaType = "application/json")),
45
46         //unplanned exception
47         @ApiResponse(responseCode = "500",
48             description = "An unplanned error occurred",
49             content = @Content(mediaType = "application/json"))
50     },
51
52     parameters = {
53         //if you pass bunch of empty characters it will return it as null
54         @Parameter(name = "model",
55             allowEmptyValue = false,
56             required = false,
57             description = "The model name (i.e., 'WRANGLER')"),
58         @Parameter(name = "trim",
59             allowEmptyValue = false,
60             required = false,
61             description = "The trim level (i.e., 'Sport')")
62     }
63
64 )
65
66 //Spring will map the get request at /jeeps to the fetchjeeps method
67 @GetMapping
68 //then it will return the status 200, 400, 404, or 500
69 @ResponseStatus(code = HttpStatus.OK)
70 //Spring needs annotations to read it will map the parameter model based on name of the parameter within the method call
71 List<Jeep> fetchJeeps(
72     @RequestParam(required = false)
73     String model,
74     @RequestParam(required = false)
75     String trim);
76 //formatter:on
77 }
78

```

```

JeepSales.java BaseTest.java FetchJeepTe...
1 package com.promineotech.jeep.entity;
2
3 import java.math.BigDecimal;
4
5
6
7 @Data
8 @Builder
9 public class Color {
10     private Long colorPK;
11     private String colorId;
12     private String color;
13     private BigDecimal price;
14     private boolean isExterior;
15 }
16

```

```
JeepSales.java  FetchJeepTe...  Jeep.java  J
1 package com.promineotech.jeep.entity;
2
3+ import lombok.Builder;
4
5
6 @Data
7 @Builder
8 public class Customer {
9     private Long customerPK;
10    private String customerId;
11    private String firstName;
12    private String lastName;
13    private String phone;
14 }
15
```

```
JeepSales.java  Jeep.java  JeepSalesCo...
1 package com.promineotech.jeep.entity;
2
3+ import java.math.BigDecimal;
4
5
6
7 @Data
8 @Builder
9 public class Engine {
10    private Long enginePK;
11    private String engineId;
12    private Float sizeInLiters;
13    private String name;
14    private FuelType fuelType;
15    private Float mpgCity;
16    private Float mpgHwy;
17    private boolean hasStartStop;
18    private String description;
19    private BigDecimal price;
20 }
21
```



```
1 package com.promineotech.jeep.entity;
2
3 public enum FuelType {
4     GASOLINE, DIESEL, HYBRID
5 }
6
```

```
JeepSales.java Jeep.java x JeepSalesCo...
1 package com.promineotech.jeep.entity;
2
3 import java.math.BigDecimal;
4 import lombok.AllArgsConstructor;
5 import lombok.Builder;
6 import lombok.Data;
7 import lombok.NoArgsConstructor;
8
9 @Data
10 @Builder
11 @NoArgsConstructor
12 @AllArgsConstructor
13 public class Jeep {
14     private Long modelPK;
15     private JeepModel modelId;
16     private String trimLevel;
17     private int numDoors;
18     private int wheelSize;
19     private BigDecimal basePrice;
20 }
21
22
```

```

JeepModel.java x Jeep.java JeepSalesCo...
1 package com.promineotech.jeep.entity;
2
3 public enum JeepModel {
4     GRAND_CHEROKEE,
5     CHEROKEE,
6     COMPASS,
7     RENEGADE,
8     WRANGLER,
9     GLADIATOR,
10    WRANGLER_4XE
11 }
12

```

```

JeepModel.java Jeep.java JeepSalesCo...
1 package com.promineotech.jeep.entity;
2
3+ import java.math.BigDecimal;
6
7 @Data
8 @Builder
9 public class Option {
10     private Long optionPK;
11     private String optionId;
12     private OptionType category;
13     private String manufacturer;
14     private String name;
15     private BigDecimal price;
16 }
17

```

```

JeepModel.java Jeep.java Color.java Customer.java
1 package com.promineotech.jeep.entity;
2
3 public enum OptionType {
4     DOOR, EXTERIOR, INTERIOR, STORAGE, TOP, WHEEL
5 }
6

```

JeepModel.java Jeep.java Customer.java

```
1 package com.promineotech.jeep.entity;
2
3 import java.math.BigDecimal;
4
5
6
7
8
9 @Data
10 @Builder
11 public class Order {
12     private Long orderPK;
13     private Customer customer;
14     private Jeep model;
15     private Color color;
16     private Engine engine;
17     private Tire tire;
18     private List<Option> options;
19     private BigDecimal price;
20
21     @JsonIgnore
22     public Long getOrderPK() {
23         return orderPK;
24     }
25 }
26
```

```
JeepModel.java Jeep.java Engine.java
1 package com.promineotech.jeep.entity;
2
3 import java.util.List;
4
5
6 @Data
7 public class OrderRequest {
8     private String customer;
9     private JeepModel model;
10    private String trim;
11    private int doors;
12    private String color;
13    private String engine;
14    private String tire;
15
16    private List<String> options;
17 }
18
```

```
JeepModel.java Jeep.java FuelType.java
1 package com.promineotech.jeep.entity;
2
3 import java.math.BigDecimal;
4
5
6 @Data
7 @Builder
8 public class Tire {
9     private Long tirePK;
10    private String tireId;
11    private String tireSize;
12    private String manufacturer;
13    private BigDecimal price;
14    private int warrantyMiles;
15 }
16
17
```

```
FetchJeepTe... x JeepModel.java Jeep.java *Option.java OptionType.java Order.java OrderReques... Tire.java
1 package com.promineotech.jeepp.controller;
2
3 import static org.junit.jupiter.api.Assertions.*;
4 import org.junit.jupiter.api.Test;
5 import org.springframework.beans.factory.annotation.Autowired;
6 import org.springframework.boot.test.context.SpringBootTest;
7 import org.springframework.boot.test.context.SpringBootTest.WebEnvironment;
8 import org.springframework.boot.test.web.client.TestRestTemplate;
9 import org.springframework.boot.web.server.LocalServerPort;
10 import org.springframework.core.ParameterizedTypeReference;
11 import org.springframework.http.HttpMethod;
12 import org.springframework.http.HttpStatus;
13 import org.springframework.http.ResponseEntity;
14 import org.springframework.test.context.ActiveProfiles;
15 import org.springframework.test.context.jdbc.Sql;
16 import org.springframework.test.context.jdbc.SqlConfig;
17 import com.promineotech.jeepp.entity.Jeep;
18 import com.promineotech.jeepp.entity.JeepModel;
19 //import com.promineotech.controller.support.FetchJeepTestSupport; //from video and but not needed
20 import lombok.Getter;
21 import java.util.List;
22 import static org.assertj.core.api.Assertions.assertThat;
23
24
25 //Below is the test annotation
26 //we are telling it that we want the tests to be run in a web environment AND to ensure the tests will not
27 //run on top of each other we specify the .random_port for each test class
28 //host class is always local host means it will stay within the local machine environment
29 //class FetchJeepTest extends FetchJeepTestSupport {
30     @SpringBootTest(webEnvironment = WebEnvironment.RANDOM_PORT)
31     @ActiveProfiles("test")
32     @Sql(scripts = {
33         "classpath:flyway/migrations/V1.0__Jeep_Schema.sql",
34         "classpath:flyway/migrations/V1.1__Jeep_Data.sql"},
35         config = @SqlConfig(encoding = "utf-8"))
36 }
```

```

FetchJeepTe... x JeepModel.java Jeep.java *Option.java OptionType.java Order.java OrderReques... Tirej
22 //class FetchJeepTest extends FetchJeepTestSupport {
30 @SpringBootTest(webEnvironment = WebEnvironment.RANDOM_PORT)
31 @ActiveProfiles("test")
32 @Sql(scripts = {
33     "classpath:flyway/migrations/V1.0__Jeep_Schema.sql",
34     "classpath:flyway/migrations/V1.1__Jeep_Data.sql"},
35     config = @SqlConfig(encoding = "utf-8"))
36 class FetchJeepTest {
37     @LocalServerPort
38     private int serverPort;
39
40     //TEST REST TEMPLATE TO SEND THE HTTP REQUESTS
41     //This one allows a test rest template to be created for us
42     @Autowired
43     @Getter
44     private TestRestTemplate restTemplate;
45
46     @Test
47     //a test should be self describing so should have a name to what they exactly do
48     void testThatJeepsAreReturnedWhenAValidModelAndTrimAreSupplied() {
49         //Given: a valid model, trim URI
50         JeepModel model = JeepModel.WRANGLER;
51         String trim = "Sport";
52         //this makes it so it can have two parameters
53         String uri = String.format("http://localhost:%d/jeeps?model=%s&trim=%s", serverPort, model, trim);
54         System.out.println(uri);
55
56         //When: a connection is made to the URI
57         ResponseEntity<List<Jeep>> response =
58             restTemplate.exchange(uri, HttpMethod.GET, null, new ParameterizedTypeReference<>() {});
59
60         //Then: a success (OK - 200) status code is returned
61         assertThat(response.getStatusCode()).isEqualTo(HttpStatus.OK);
62     }
63 }
64 }

```

```
FetchJeepTe... JeepModel.java jeep-sales/p... *Option.java OptionType.java Order.java OrderReques...
1 <?xml version="1.0" encoding="UTF-8"?>
2 <project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
3   xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 https://maven.apache.org/xsd/maven-4.0.0.xsd">
4   <modelVersion>4.0.0</modelVersion>
5   <parent>
6     <groupId>org.springframework.boot</groupId>
7     <artifactId>spring-boot-starter-parent</artifactId>
8     <version>2.5.4</version>
9     <relativePath/> <!-- lookup parent from repository -->
10  </parent>
11  <groupId>com.promineotech</groupId>
12  <artifactId>jeep-sales</artifactId>
13  <version>1.0.0.1-SNAPSHOT</version>
14  <name>jeep-sales</name>
15  <description>Jeep Sales</description>
16  <properties>
17    <java.version>11</java.version>
18  </properties>
19  <dependencies>
20    <dependency>
21      <groupId>org.springframework.boot</groupId>
22      <artifactId>spring-boot-starter-web</artifactId>
23    </dependency>
24
25    <dependency>
26      <groupId>org.springframework.boot</groupId>
27      <artifactId>spring-boot-devtools</artifactId>
28      <scope>runtime</scope>
29      <optional>true</optional>
30    </dependency>
31
32    <dependency>
33      <groupId>org.projectlombok</groupId>
34      <artifactId>lombok</artifactId>
```

FetchJeepTe... JeepModel.java jeep-sales/p... *Option.java OptionType.java Order.java OrderReques

```
37
38     <!--OpenAPI Dependency ===== -->
39
40     <!-- https://mvnrepository.com/artifact/org.springdoc/springdoc-openapi-ui -->
41     <dependency>
42         <groupId>org.springdoc</groupId>
43         <artifactId>springdoc-openapi-ui</artifactId>
44         <version>1.5.10</version>
45     </dependency>
46
47     <dependency>
48         <groupId>org.springframework.boot</groupId>
49         <artifactId>spring-boot-starter-test</artifactId>
50         <scope>test</scope>
51     </dependency>
52     <!-- https://mvnrepository.com/artifact/org.springdoc/springdoc-openapi-ui -->
53     <dependency>
54         <groupId>org.springdoc</groupId>
55         <artifactId>springdoc-openapi-ui</artifactId>
56         <version>1.5.10</version>
57     </dependency>
58
59 </dependencies>
60
61 <build>
62     <plugins>
63         <plugin>
64             <groupId>org.springframework.boot</groupId>
65             <artifactId>spring-boot-maven-plugin</artifactId>
66             <configuration>
67                 <excludes>
68                     <exclude>
69                         <groupId>org.projectlombok</groupId>
70                         <artifactId>lombok</artifactId>
```



```

45     </dependency>
46
47     <dependency>
48       <groupId>org.springframework.boot</groupId>
49       <artifactId>spring-boot-starter-test</artifactId>
50       <scope>test</scope>
51     </dependency>
52     <!-- https://mvnrepository.com/artifact/org.springdoc/springdoc-openapi-ui -->
53     <dependency>
54       <groupId>org.springdoc</groupId>
55       <artifactId>springdoc-openapi-ui</artifactId>
56       <version>1.5.10</version>
57     </dependency>
58
59   </dependencies>
60
61   <build>
62     <plugins>
63       <plugin>
64         <groupId>org.springframework.boot</groupId>
65         <artifactId>spring-boot-maven-plugin</artifactId>
66         <configuration>
67           <excludes>
68             <exclude>
69               <groupId>org.projectlombok</groupId>
70               <artifactId>lombok</artifactId>
71             </exclude>
72           </excludes>
73         </configuration>
74       </plugin>
75     </plugins>
76   </build>
77
78 </project>

```

Screenshots of Running Application:

OverviewDependenciesDependency HierarchyEffective POMpom.xml

jeep-sales - JeepSales [Spring Boot App] C:\Users\Admin\workspace\sts-4.11.0.RELEASE\plugins\org.eclipse.justi.openjdk.hotspot.jre.full.win32.x86_64.16.0.1.v20210528-1205\jre\bin\java.exe (Aug 24, 2021, 9:56:07 PM)

ProblemsJavadocDeclarationConsole

2021-08-24 21:56:09.024 INFO 15392 --- [restartedMain] com.promineotech.jee... : Starting JeepSales using Java 16.0.1 on DESKTOP-IL9T190 with PID 15392

2021-08-24 21:56:09.026 INFO 15392 --- [restartedMain] com.promineotech.jee... : No active profile set, falling back to default profiles: default

2021-08-24 21:56:09.082 INFO 15392 --- [restartedMain] .e.DevToolsPropertyDefaultsPostProcessor : Devtools property defaults active! Set 'spring.devtools.add-properties' to 'true' to enable additional web related logging consider setting the 'logging.level.org.apache.catalina.core.StandardService' to 'debug' to see more details.

2021-08-24 21:56:09.920 INFO 15392 --- [restartedMain] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat initialized with port(s): 8080 (http)

2021-08-24 21:56:09.928 INFO 15392 --- [restartedMain] o.apache.catalina.core.StandardService : Starting service [Tomcat]

2021-08-24 21:56:09.928 INFO 15392 --- [restartedMain] org.apache.catalina.core.StandardEngine : Starting Servlet engine: [Apache Tomcat/9.0.52]

2021-08-24 21:56:10.000 INFO 15392 --- [restartedMain] o.a.c.c.C.[Tomcat].[localhost].[/] : Initializing Spring embedded WebApplicationContext

2021-08-24 21:56:10.000 INFO 15392 --- [restartedMain] w.s.c.ServletWebServerApplicationContext : Root WebApplicationContext: initialization completed in 918 ms

2021-08-24 21:56:10.612 INFO 15392 --- [restartedMain] o.s.b.d.a.OptionalLiveReloadServer : LiveReload server is running on port 35729

2021-08-24 21:56:10.661 INFO 15392 --- [restartedMain] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat started on port(s): 8080 (http) with context path ''


2021-08-24 21:56:10.670 INFO 15392 --- [restartedMain] com.promineotech.jee... : Started JeepSales in 1.949 seconds (JVM running for 2.68)

2021-08-24 22:00:08.121 INFO 15392 --- [nio-8080-exec-1] o.a.c.c.C.[Tomcat].[localhost].[/] : Initializing Spring DispatcherServlet 'dispatcherServlet'

2021-08-24 22:00:08.121 INFO 15392 --- [nio-8080-exec-1] o.s.web.servlet.DispatcherServlet : Initializing Servlet 'dispatcherServlet'

2021-08-24 22:00:08.122 INFO 15392 --- [nio-8080-exec-1] o.s.web.servlet.DispatcherServlet : Completed initialization in 1 ms

2021-08-24 22:00:08.851 INFO 15392 --- [nio-8080-exec-8] o.springdoc.api.AbstractOpenApiResource : Init duration for springdoc-openapi is: 71 ms



Spring Boot
(v2.5.4)

2021-08-24 22:11:41.397 INFO 15392 --- [restartedMain] com.promineotech.jee... : Starting JeepSales using Java 16.0.1 on DESKTOP-IL9T190 with PID 15392

2021-08-24 22:11:41.397 INFO 15392 --- [restartedMain] com.promineotech.jee... : No active profile set, falling back to default profiles: default

2021-08-24 22:11:41.604 INFO 15392 --- [restartedMain] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat initialized with port(s): 8080 (http)

2021-08-24 22:11:41.605 INFO 15392 --- [restartedMain] o.apache.catalina.core.StandardService : Starting service [Tomcat]

2021-08-24 22:11:41.605 INFO 15392 --- [restartedMain] org.apache.catalina.core.StandardEngine : Starting Servlet engine: [Apache Tomcat/9.0.52]

2021-08-24 22:11:41.673 INFO 15392 --- [restartedMain] o.a.c.c.C.[Tomcat].[localhost].[/] : Initializing Spring embedded WebApplicationContext

2021-08-24 22:11:41.673 INFO 15392 --- [restartedMain] w.s.c.ServletWebServerApplicationContext : Root WebApplicationContext: initialization completed in 223 ms

2021-08-24 22:11:41.844 INFO 15392 --- [restartedMain] o.s.b.d.a.OptionalLiveReloadServer : LiveReload server is running on port 35729

2021-08-24 22:11:41.852 INFO 15392 --- [restartedMain] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat started on port(s): 8080 (http) with context path ''

2021-08-24 22:11:41.857 INFO 15392 --- [restartedMain] com.promineotech.jee... : Started JeepSales in 0.484 seconds (JVM running for 933.837)

2021-08-24 22:11:41.858 INFO 15392 --- [restartedMain] .ConditionEvaluationDeltaLoggingListener : Condition evaluation unchanged

localhost:8080/swagger-ui/index.html?configUrl=/v3/api-docs/swagger-config/#/

Swagger

v3/api-docs

Explore

Jeep Sales Service

v3/api-docs

Servers

http://localhost:8080 - Local server

default-jeep-sales-controller

GET /jeeps Returns a list of Jeeps

Schemas

Jeep

default-jeep-sales-controller

GET /jeeps Returns a list of Jeeps

Returns a list of Jeeps given an optional model and/or trim

Parameters

Try it out

Name	Description
model string (query)	The model name (i.e., 'WRANGLER') <input type="text" value="model"/>
trim string (query)	The trim level (i.e., 'Sport') <input type="text" value="trim"/>

Responses

Code	Description	Links
200		No links

localhost:8080/swagger-ui/index.html?configUrl=/v3/api-docs/swagger-config#/default-jeep-sales-controller/fetchJeeps

application/json

Controls Accept header.

Example Value Schema

400

The request parameters are invalid

No links

Media type

application/json

404

No Jeeps were found with the input criteria

No links

Media type

application/json

500

An unplanned error occurred

No links

Media type

application/json

Schemas

Jeep >

URL to GitHub Repository:

<https://github.com/JoleneMel/Jeep-SalesW1>