

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".
- b) Check "Create a simple project (skip archetype selection)". Click "Next".
- c) Enter the following:

Group Id	com.promineotech
Artifact Id	jeep-sales

Click "Finish".

- 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
 - c) Click "Explore" at the bottom of the page.
 - d) Click "Copy" to copy the pom.xml generated by the Initializr to the clipboard.

- 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.
- 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.
- 5) Create a package in src/**main**/java named com.promineotech.jeep. In this package:
 - a) Create a Java class with a main method named JeepSales.
 - 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.jeep;

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".
 - 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".
- 7) Using the MySQL Workbench or MySQL command line client (CLI), create a database named "jeep".
- 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**.
- 9) Create a new package in src/**test**/java named com.promineotech.jeep.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 { }
```

- 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;
```

- 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`.
- 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;
}

```

- 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.**
- 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
<code>JeepModel</code>	<code>model</code>	<code>JeepModel.WRANGLER</code>
<code>String</code>	<code>trim</code>	<code>"Sport"</code>
<code>String</code>	<code>uri</code>	<code>String.format("http://localhost:%d/jeeps?model=%s&trim=%s", serverPort, model, trim);</code>

- 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. 

```

1 package com.promineotech.jeep.controller;
2
3 import static org.assertj.core.api.Assertions.assertThat;
4 import static org.junit.jupiter.api.Assertions.*;
5
6 import java.util.List;
7 import org.junit.jupiter.api.Test;
8 import org.springframework.beans.factory.annotation.Autowired;
9 import org.springframework.boot.test.context.SpringBootTest;
10 import org.springframework.boot.test.context.SpringBootTest.WebEnvironment;
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.HttpMethod;
15 import org.springframework.http.HttpStatus;
16 import org.springframework.http.ResponseEntity;
17 import org.springframework.test.context.ActiveProfiles;
18 import org.springframework.test.context.jdbc.Sql;
19 import org.springframework.test.context.jdbc.SqlConfig;
20 import com.promineotech.jeep.entity.Jeep;
21 import com.promineotech.jeep.entity.JeepModel;
22 import lombok.Getter;
23
24 @SpringBootTest(webEnvironment = WebEnvironment.RANDOM_PORT)
25 @ActiveProfiles("test")
26 @Sql(scripts = {
27     "classpath:flyway/migrations/V1.0__Jeep_Schema.sql",
28     "classpath:flyway/migrations/V1.1__Jeep_Data.sql"},
29     config = @SqlConfig(encoding = "utf-8"))
30 class FetchJeepsTest {
31     @Autowired
32     @Getter
33     private TestRestTemplate restTemplate;
34
35     @LocalServerPort
36     private int serverPort;
37
38     @Test
39     void testThatJeepsAreReturnedWhenValidModelAndTrimAreSupplied() {
40         //Given: a valid model, trim and URI
41         JeepModel model = JeepModel.WRANGLER;
42         String trim = "Sport";
43         String uri =
44             String.format("http://localhost:%d/jeeps?model=%s&trim=%s", serverPort, model, trim);
45
46         //When: a connection is made to the URI
47         ResponseEntity<List<Jeep>> response = restTemplate.exchange(uri, HttpMethod.GET, null, new ParameterizedTypeReference<>() {});
48
49         //Then: a success(OK - 200) status code is returned
50         assertThat(response.getStatusCode()).isEqualTo(HttpStatus.OK);
51     }
52 }
53

```

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. 

```
20  @RequestMapping("/jeeps")
21  @OpenAPIDefinition(info = @Info(title = "Jeep Sales Service"), servers = {
22      @Server(url = "http://localhost:8080", description = "Local server."))}
23  public interface JeepSalesController {
24      // @formatter:off
25      @Operation(
26          summary = "Returns a list of Jeeps",
27          description = "Returns a list of Jeeps given an optional model and/or trim",
28          responses = {
29              @ApiResponse(
30                  responseCode = "200",
31                  description = "A list of Jeeps is returned.",
32                  content = @Content(mediaType = "application/json",
33                  schema = @Schema(implementation = Jeep.class))),
34              @ApiResponse(
35                  responseCode = "400",
36                  description = "The request parameters are invalid.",
37                  content = @Content(mediaType = "application/json")),
38              @ApiResponse(
39                  responseCode = "404",
40                  description = "No Jeeps were found with the input criteria.",
41                  content = @Content(mediaType = "application/json")),
42              @ApiResponse(
43                  responseCode = "500",
44                  description = "An unplanned error occurred.",
45                  content = @Content(mediaType = "application/json"))
46          },
47          parameters = {
48              @Parameter(name = "model",
49                  allowEmptyValue = false,
50                  required = false,
51                  description = "The model name (i.e., 'WRANGLER'"),
52              @Parameter(name = "trim",
53                  allowEmptyValue = false,
54                  required = false,
55                  description = "The trim level (i.e., 'Sport')")
56          }
57      )
58
59      @GetMapping
60      @ResponseStatus(code = HttpStatus.OK)
61      List<Jeep> fetchJeeps(
62          @RequestParam(required = false)
63          JeepModel model,
64          @RequestParam(required = false)
65          String trim);
66      // @formatter:on
67  }
68 }
```

- 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.

The screenshot shows a Java development environment with the following details:

- IDE:** Eclipse (version 2022-07, build 20220515-1416)
- Project:** JeepSales (Spring Boot App)
- Console Output:**

```
jeep-sales - JeepSales Spring Boot App [Applications/TestingToolSuite4.app/Contents/Eclipse/plugins/org.eclipse.jdt.core/eclipse.jdt.core_20220515-1416/reboot.java] (Jul 30, 2022, 7:38:58 AM) [pid: 2817]
07:36:58.856 [Thread-0] DEBUG org.springframework.boot.devtools.restart.classloader.RestartClassLoader - Created RestartClassLoader org.springframework.boot.devtools.restart.classloader.RestartClassLoader@6bd652e7
:: Spring Boot ::
(v2.7.2)

2022-07-30 07:36:59.116 INFO 2817 --- [ restartedMain] com.prominotech.jeep.JeepSales      : Starting JeepSales using Java 17.0.8 on Olivias-MacBook-Pro.local with PID 2817 (/Users/oliviaguzman/Promineo/SpringBoot/JEEP-Sales)
2022-07-30 07:36:59.116 INFO 2817 --- [ restartedMain] com.prominotech.jeep.JeepSales      : No active profile set, falling back to 1 default profile: 'default'
2022-07-30 07:36:59.156 INFO 2817 --- [ restartedMain] e.DevToolsPropertyDefaultsPostProcessor : Devtools property defaults active! Set 'spring.devtools.add-properties' to 'false' to disable
2022-07-30 07:36:59.157 INFO 2817 --- [ restartedMain] e.DevToolsPropertyDefaultsPostProcessor : For additional web related logging consider setting the 'logging.level.web' property to 'DEBUG'
2022-07-30 07:36:59.177 INFO 2817 --- [ restartedMain] o.s.boot.SpringApplication : Starting service [Tomcat]
2022-07-30 07:36:59.177 INFO 2817 --- [ restartedMain] o.s.boot.SpringApplication : Starting service [Tomcat]
2022-07-30 07:36:59.875 INFO 2817 --- [ restartedMain] o.apache.catalina.core.StandardEngine : Starting Servlet engine: [Apache Tomcat/9.0.65]
2022-07-30 07:36:59.928 INFO 2817 --- [ restartedMain] o.s.c.c.[Tomcat].localhost.[] : Initializing Spring embedded WebApplicationContext
2022-07-30 07:36:59.928 INFO 2817 --- [ restartedMain] o.s.web.context.ContextLoader : Root WebApplicationContext: initialization completed in 762 ms
2022-07-30 07:37:00.432 INFO 2817 --- [ restartedMain] o.s.boot.SpringApplication : LiveReload server is running on port 3572
2022-07-30 07:37:00.432 INFO 2817 --- [ restartedMain] o.s.boot.SpringApplication : Context started on port(s): 8080 with context path ''
2022-07-30 07:37:00.451 INFO 2817 --- [ restartedMain] com.prominotech.jeep.JeepSales      : Started JeepSales in 1.588 seconds (JVM running for 2.143)
```
- Browser:** A screenshot of the generated OpenAPI documentation for the `/jeeps` endpoint.

default-jeep-sales-controller

GET /jeeps Returns a list of Jeeps

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

Parameters

Name	Description
model <small>(query)</small>	The model name (i.e., 'WRANGLER') <small>Available values : WRANGLER, GRAND_CHEROKEE, CHEROKEE, COMPASS, RENEGADE, GLADIATOR, WRANGLER_4XE</small>
trim <small>(query)</small>	The trim level (i.e., 'Sport') <small>trim</small>

Responses

Code	Description	Links
200	A list of Jeeps is returned.	No links
400	The request parameters are invalid.	No links
404	No Jeeps were found with the input criteria.	No links
500	An unplanned error occurred.	No links

Example Value | **Schema**

```
{
  "modelPK": 0,
  "modelId": "WRANGLER",
  "trimLevel": "string",
  "numDoors": 0,
  "wheelsize": 0,
  "basePrice": 0
}
```

Screenshots of Code:

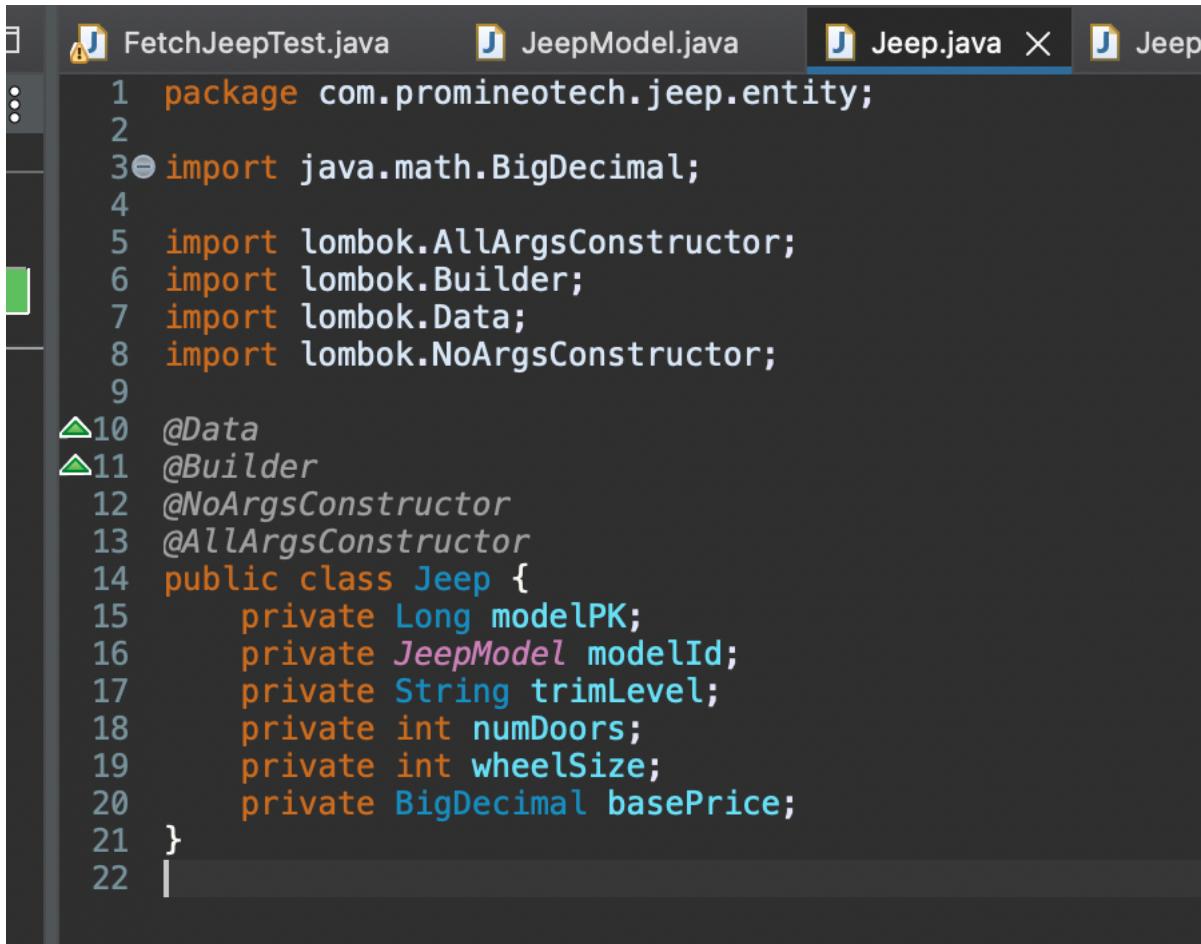
The screenshot shows a Java IDE interface with the pom.xml file open. The window title is "jeep-sales/pom.xml". The code is a Maven POM file for a Spring Boot application named "jeep-sales".

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <project xmlns="http://maven.apache.org/POM/4.0.0"
3   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
4   xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 https://maven.apache.org/xsd/maven-4.0.0.xsd">
5   <modelVersion>4.0.0</modelVersion>
6
7   <parent>
8     <groupId>org.springframework.boot</groupId>
9     <artifactId>spring-boot-starter-parent</artifactId>
10    <version>2.7.2</version>
11    <relativePath /> <!-- lookup parent from repository -->
12  </parent>
13
14  <groupId>com.promineotech</groupId>
15  <artifactId>jeep-sales</artifactId>
16  <version>1.0.0.1-SNAPSHOT</version>
17
18  <name>jeep-sales</name>
19  <description>Jeep Sales</description>
20
21  <properties>
22    <java.version>11</java.version>
23  </properties>
24
25  <dependencies>
26    <dependency>
27      <groupId>org.springframework.boot</groupId>
28      <artifactId>spring-boot-starter-web</artifactId>
29    </dependency>
30
31    <dependency>
32      <groupId>org.springframework.boot</groupId>
33      <artifactId>spring-boot-devtools</artifactId>
34      <scope>runtime</scope>
35      <optional>true</optional>
36    </dependency>
37
38    <!-- OpenAPI dependency -->
39    <dependency>
40      <groupId>org.springdoc</groupId>
41      <artifactId>springdoc-openapi-ui</artifactId>
42      <version>1.6.9</version>
43    </dependency>
44
45    <dependency>
46      <groupId>org.projectlombok</groupId>
47      <artifactId>lombok</artifactId>
48      <optional>true</optional>
49    </dependency>
50
51    <dependency>
52      <groupId>org.springframework.boot</groupId>
53      <artifactId>spring-boot-starter-test</artifactId>
54      <scope>test</scope>
55    </dependency>
56  </dependencies>
57
58  <build>
59    <plugins>
60      <plugin>
61        <groupId>org.springframework.boot</groupId>
62        <artifactId>spring-boot-maven-plugin</artifactId>
63      <configuration>
64        <excludes>
65          <exclude>
66            <groupId>org.projectlombok</groupId>
67            <artifactId>lombok</artifactId>
68          </exclude>
69        </excludes>
70      </configuration>
71    </plugin>
72  </plugins>
73</build>
74
75</project>
```

```
FetchJeepTest.java X JeepModel.java X Jeep.java X JeepSalesController.java M jeep-sales/pom.xml D DefaultJeepSalesController.java
1 package com.promineotech.jeep.controller;
2
3 import static org.assertj.core.api.Assertions.assertThat;
4 import static org.junit.jupiter.api.Assertions.*;
5
6 import java.util.List;
7 import org.junit.jupiter.api.Test;
8 import org.springframework.beans.factory.annotation.Autowired;
9 import org.springframework.boot.test.context.SpringBootTest;
10 import org.springframework.boot.test.context.SpringBootTest.WebEnvironment;
11 import org.springframework.boot.test.web.client.TestRestTemplate;
12 import org.springframework.boot.web.server.LocalServerPort;
13 import org.springframework.core.ParameterizedTypeReference;
14 import org.springframework.http.HttpMethod;
15 import org.springframework.http.HttpStatus;
16 import org.springframework.http.ResponseEntity;
17 import org.springframework.test.context.ActiveProfiles;
18 import org.springframework.test.context.jdbc.Sql;
19 import org.springframework.test.context.jdbc.SqlConfig;
20 import com.promineotech.jeep.entity.Jeep;
21 import com.promineotech.jeep.entity.JeepModel;
22 import lombok.Getter;
23
24 @SpringBootTest(webEnvironment = WebEnvironment.RANDOM_PORT)
25 @ActiveProfiles("test")
26 @Sql(scripts = {
27     "classpath:flyway/migrations/V1.0__Jeep_Schema.sql",
28     "classpath:flyway/migrations/V1.1__Jeep_Data.sql"},
29     config = @SqlConfig(encoding = "utf-8"))
30 class FetchJeepTest {
31     @Autowired
32     @Getter
33     private TestRestTemplate restTemplate;
34
35     @LocalServerPort
36     private int serverPort;
37
38     @Test
39     void testThatJeepsAreReturnedWhenAValidModelAndTrimAreSupplied() {
40         //Given: a valid model, trim and URI
41         JeepModel model = JeepModel.WRANGLER;
42         String trim = "Sport";
43         String uri =
44             String.format("http://localhost:%d/jeeps?model=%s&trim=%s", serverPort, model, trim);
45
46         //When: a connection is made to the URI
47         ResponseEntity<List<Jeep>> response = restTemplate.exchange(uri, HttpMethod.GET, null, new ParameterizedTypeReference<>() {});
48
49         //Then: a success(OK - 200) status code is returned
50         assertThat(response.getStatusCode()).isEqualTo(HttpStatus.OK);
51     }
52 }
53
```

```
FetchJeepTest.java X JeepModel.java X Jeep.java X JeepSalesController.java M jeep-sales/pom.xml D DefaultJeepSale...
1 package com.promineotech.jeep.entity;
2
3 public enum JeepModel {
4     WRANGLER, GRAND_CHEROKEE, CHEROKEE, COMPASS, RENEGADE, GLADIATOR, WRANGLER_4XE
5 }
6
```

```
FetchJeepTest.java... X JeepModel.java X Jeep.java X JeepSalesController.java... M jeep-sales/pom... D DefaultJeepSale... X JeepSales.java X
1 package com.promineotech.jeep;
2
3 import org.springframework.boot.SpringApplication;
4 import org.springframework.boot.autoconfigure.SpringBootApplication;
5
6 @SpringBootApplication
7 public class JeepSales {
8
9     public static void main(String[] args) {
10         SpringApplication.run(JeepSales.class, args);
11     }
12 }
13
14
```

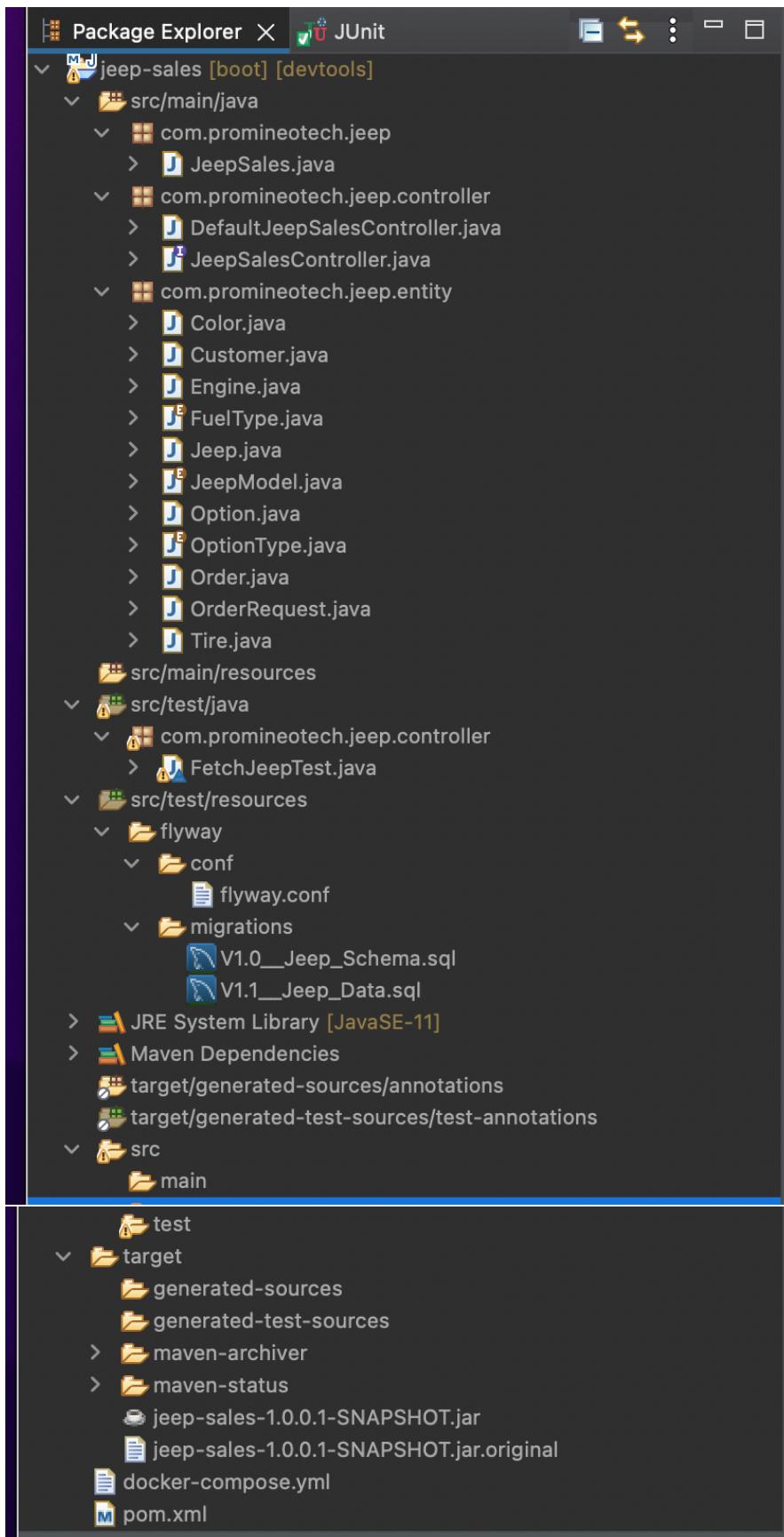


```
FetchJeepTest.java  JeepModel.java  Jeep.java  Jeep.java X  Jeep.java X
1 package com.promineotech.jeep.entity;
2
3 import java.math.BigDecimal;
4
5 import lombok.AllArgsConstructor;
6 import lombok.Builder;
7 import lombok.Data;
8 import lombok.NoArgsConstructor;
9
10 @Data
11 @Builder
12 @NoArgsConstructor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

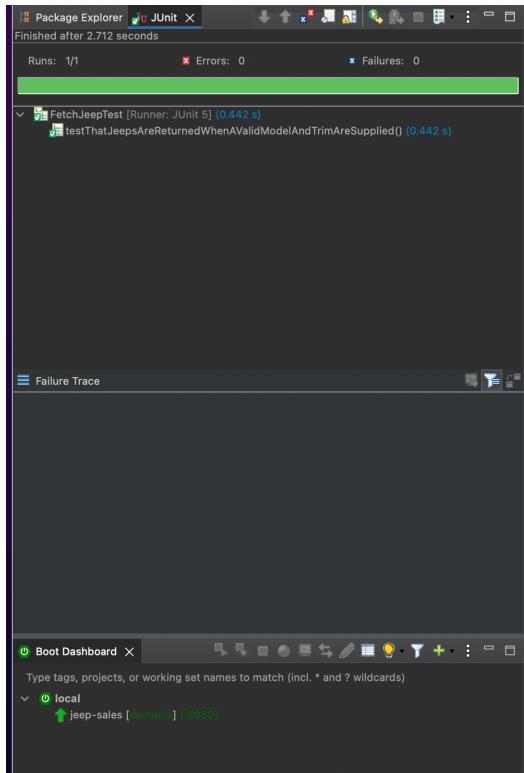


```
FetchJeepTest.java  JeepModel.java  Jeep.java  JeepSalesController.java  jeep-sales/pom.xml  DefaultJeepSalesController.java X
1 package com.promineotech.jeep.controller;
2
3 import java.util.List;
4 import org.springframework.web.bind.annotation.RestController;
5 import com.promineotech.jeep.entity.Jeep;
6 import com.promineotech.jeep.entity.JeepModel;
7
8 @RestController
9 public class DefaultJeepSalesController implements JeepSalesController {
10
11     @Override
12     public List<Jeep> fetchJeeps(JeepModel model, String trim) {
13         return null;
14     }
15 }
16 |
```

```
FetchJeepTest.java      JeepModel.java      Jeep.java      JeepSalesController.java  X  jeep-sales/pom.xml
1 package com.promineotech.jeep.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.jeep.entity.Jeep;
10 import com.promineotech.jeep.entity.JeepModel;
11 import io.swagger.v3.oas.annotations.OpenAPIDefinition;
12 import io.swagger.v3.oas.annotations.Operation;
13 import io.swagger.v3.oas.annotations.info.Info;
14 import io.swagger.v3.oas.annotations.servers.Server;
15 import io.swagger.v3.oas.annotations.responses.ApiResponse;
16 import io.swagger.v3.oas.annotations.media.Schema;
17 import io.swagger.v3.oas.annotations.media.Content;
18 import io.swagger.v3.oas.annotations.Parameter;
19
20 @RequestMapping("/jeeps")
21 @OpenAPIDefinition(info = @Info(title = "Jeep Sales Service"), servers = {
22     @Server(url = "http://localhost:8080", description = "Local server.")))
23 public interface JeepSalesController {
24     // @formatter:off
25     @Operation(
26         summary = "Returns a list of Jeeps",
27         description = "Returns a list of Jeeps given an optional model and/or trim",
28         responses = {
29             @ApiResponse(
30                 responseCode = "200",
31                 description = "A list of Jeeps is returned.",
32                 content = @Content(mediaType = "application/json",
33                     schema = @Schema(implementation = Jeep.class))),
34             @ApiResponse(
35                 responseCode = "400",
36                 description = "The request parameters are invalid.",
37                 content = @Content(mediaType = "application/json")),
38             @ApiResponse(
39                 responseCode = "404",
40                 description = "No Jeeps were found with the input criteria.",
41                 content = @Content(mediaType = "application/json")),
42             @ApiResponse(
43                 responseCode = "500",
44                 description = "An unplanned error occurred.",
45                 content = @Content(mediaType = "application/json"))
46         },
47         parameters = {
48             @Parameter(name = "model",
49                         allowEmptyValue = false,
50                         required = false,
51                         description = "The model name (i.e., 'WRANGLER'"),
52             @Parameter(name = "trim",
53                         allowEmptyValue = false,
54                         required = false,
55                         description = "The trim level (i.e., 'Sport')")
56         }
57     )
58
59     @GetMapping
60     @ResponseStatus(code = HttpStatus.OK)
61     List<Jeep> fetchJeeps(
62         @RequestParam(required = false)
63         JeepModel model,
64         @RequestParam(required = false)
65         String trim);
66     // @formatter:on
67 }
```



Screenshots of Running Application:



```
jeep-sales - JeepSales [Spring Boot App] [/Applications/SpringToolSuite4.app/Contents/Eclipse/plugins/org.eclipse.justjavajdk.hotspot.jre.full.macosx.x86_64_17.0.3.v20220515-1416/re/bin/java] (Jul 30, 2022, 10:19:54 AM) [pid: 6672]
10:19:54.927 [Thread-8] DEBUG org.springframework.boot.devtools.restart.classloader.RestartClassLoader - Created RestartClassLoader org.springframework.boot.devtools.restart.classloader.RestartClassLoader@7f2fd6

 (v2.7.2)

:: Spring Boot ::

2022-07-30 10:19:55.157 INFO 6672 --- [ restartedMain] com.prominotech.jeep.JeepSales : Starting JeepSales using Java 17.0.3 on Olivias-MacBook-Pro.local with PID 6672 (/Users/oliviaguzman/Promineo/SpringToolSuite4/applications/JEEP-SALES.jar)
2022-07-30 10:19:55.158 INFO 6672 --- [ restartedMain] com.prominotech.jeep.JeepSales : No active profile set, falling back to 1 default profile: "default"
2022-07-30 10:19:55.284 INFO 6672 --- [ restartedMain] e.DevToolsPropertiesDefaultsPostProcessor : Devtools property defaults active! Set 'spring.devtools.add-properties' to 'false' to disable
2022-07-30 10:19:55.284 INFO 6672 --- [ restartedMain] e.DevToolsPropertiesDefaultsPostProcessor : For additional web related logging consider setting the 'logging.level.web' property to 'DEBUG'
2022-07-30 10:19:55.285 INFO 6672 --- [ restartedMain] o.s.boot.SpringApplication : detected auto configuration resources in META-INF/resources
2022-07-30 10:19:55.889 INFO 6672 --- [ restartedMain] o.s.boot.SpringApplication : detected auto configuration resources in META-INF/resources
2022-07-30 10:19:55.889 INFO 6672 --- [ restartedMain] o.apache.catalina.core.StandardService : Starting service [Tomcat]
2022-07-30 10:19:55.889 INFO 6672 --- [ restartedMain] o.apache.catalina.core.StandardEngine : Starting Servlet engine: [Apache Tomcat/9.0.65]
2022-07-30 10:19:55.938 INFO 6672 --- [ restartedMain] o.a.c.c.C.[Tomcat].[localhost].[/] : Initializing Spring embedded WebApplicationContext
2022-07-30 10:19:55.938 INFO 6672 --- [ restartedMain] w.s.c.ServletWebServerApplicationContext : Root WebApplicationContext: initialization completed in 734 ms
2022-07-30 10:19:56.498 INFO 6672 --- [ restartedMain] o.s.w.d.a.OptionalLiveReloadServer : LiveReload server is running on port 2022
2022-07-30 10:19:56.526 INFO 6672 --- [ restartedMain] o.s.w.d.a.TomcatReactiveWebServer : Tomcat started on port(s): 8080 (http) with context path ''
2022-07-30 10:19:56.435 INFO 6672 --- [ restartedMain] com.prominotech.jeep.JeepSales : Started JeepSales in 1.502 seconds (JVM running for 2.159)
```

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

Name	Description
model <small>string (query)</small>	The model name (i.e., 'WRRANGER') Available values : WRANGLER, GRAND_CHEROKEE, CHEROKEE, COMPASS, RENEGADE, GLADIATOR, WRANGLER_4XE
trim <small>string (query)</small>	The trim level (i.e., Sport) trim

Responses

Code	Description	Links
200	A list of Jeeps is returned.	No links
400	The request parameters are invalid.	No links
404	No Jeeps were found with the input criteria.	No links
500	An unplanned error occurred.	No links

URL to GitHub Repository:

<https://github.com/OliGuzman/Spring-Boot-Week-1>