


# 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:

<b>Group Id</b>	com.promineotech
<b>Artifact Id</b>	jeep-sales

Click "Finish".

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

- a) Confirm the following settings:

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

```

1 package com.promineotech.jeep.controller;
2
3 import static org.assertj.core.api.Assertions.assertThat;
4
5 @SpringBootTest(webEnvironment = WebEnvironment.RANDOM_PORT)
6 @ActiveProfiles("test")
7 @Sql(scripts = {
8     "classpath:flyway/migrations/V1.0__Jeep_Schema.sql",
9     "classpath:flyway/migrations/V1.1__Jeep_Data.sql"},
10     config = @SqlConfig(encoding = "utf-8"))
11 class FetchJeepTest {
12
13     @Autowired
14     @Getter
15     private TestRestTemplate restTemplate;
16
17     @LocalServerPort
18     private int serverPort;
19
20     @Test
21     void testThatJeepsAreReturnedWhenAValidModelAndTrimAreSupplied() {
22         // Given: a valid model, trim and URI
23         JeepModel model = JeepModel.WRANGLER;
24         String trim = "Sport";
25         String uri = String.format("http://localhost:%d/jeeps?model=%s&trim=%s", serverPort, model, trim);
26
27         // When: a connection is made to the URI
28
29         // ResponseEntity<List<Jeep>> response = restTemplate.exchange(uri, HttpMethod.GET, null, new ParameterizedTypeReference<>());
30         ResponseEntity<Jeep> response = getRestTemplate().getForEntity(uri, Jeep.class);
31
32         // Then: a success (OK - 200) status code is returned
33         assertThat(response.getStatusCode()).isEqualTo(HttpStatus.OK);
34     }
35 }

```

- 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(responseCode = "200",
30                 description = "A list of Jeeps is returned",
31                 content = @Content(
32                     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         },

```

```

        content = @Content(mediaType = "application/Json"));
    },
    parameters = {
        @Parameter(
            name = "model",
            allowEmptyValue = false,
            required = false,
            description = "The model name (i.e., 'WRANGLER')"),
        @Parameter(
            name = "trim",
            allowEmptyValue = false,
            required = false,
            description = "The trim level (i.e., 'Sport')"),
    }
)

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

```

- 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. 🖥️

## Jeep Sales Service

/v3/api-docs

Servers

http://localhost:8080 - Local server. ▾

### 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                                                                                                                                                                            |
|----------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>model</b> * required<br>string<br>(query) | The model name (i.e., 'WRANGLER')<br><i>Available values</i> : WRANGLER, GRAND_CHEROKEE, CHEROKEE, COMPASS, RENEGADE, GLADIATOR, WRANGLER_4XE<br><input type="text" value="WRANGLER"/> |
| <b>trim</b> * required<br>string<br>(query)  | The trim level (i.e., 'Sport')<br><input type="text" value="trim"/>                                                                                                                    |

### Responses

| Code | Description                                                                                                                                                                                                                                                                                | Links    |
|------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------|
| 200  | A list of Jeeps is returned<br><br>Media type<br><input type="text" value="application/Json"/><br>Controls Accept header<br>Example Value   Schema<br><pre>{   "modelPK": 0,   "modelId": "WRANGLER",   "trimLevel": "string",   "numDoors": 0,   "wheelSize": 0,   "basePrice": 0 }</pre> | No links |
| 400  | The request parameters are invalid<br><br>Media type<br><input type="text" value="application/Json"/>                                                                                                                                                                                      | No links |
| 404  | No Jeeps were found with the input criteria<br><br>Media type<br><input type="text" value="application/Json"/>                                                                                                                                                                             | No links |
| 500  | An unplanned error occurred<br><br>Media type<br><input type="text" value="application/Json"/>                                                                                                                                                                                             | No links |

**Screenshots of Code:**

```

FetchJeepTest.java x JeepModel.java Jeep.java JeepSales.java JeepSalesController.java DefaultJeepSalesController.java
21
22 @SpringBootTest(webEnvironment = WebEnvironment.RANDOM_PORT)
23 @ActiveProfiles("test")
24 @Sql(scripts = {
25     "classpath:flyway/migrations/V1.0__Jeep_Schema.sql",
26     "classpath:flyway/migrations/V1.1__Jeep_Data.sql"},
27     config = @SqlConfig(encoding = "utf-8"))
28 class FetchJeepTest {
29
30     @Autowired
31     @Getter
32     private TestRestTemplate restTemplate;
33
34     @LocalServerPort
35     private int serverPort;
36
37     @Test
38     void testThatJeepsAreReturnedWhenAValidModelAndTrimAreSupplied() {
39         // Given: a valid model, trim and URI
40         JeepModel model = JeepModel.WRANGLER;
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
46         ResponseEntity<List<Jeep>> response = restTemplate.exchange(uri, HttpMethod.GET, null, new ParameterizedTypeReference<List<Jeep>>());
47         //ResponseEntity<Jeep> response = getRestTemplate().getForEntity(uri, Jeep.class);
48
49         // Then: a success (OK - 200) status code is returned
50         assertThat(response.getStatusCode()).isEqualTo(HttpStatus.OK);
51     }

```

```

FetchJeepTest.java JeepModel.java x Jeep.java Order.java JeepSales.java JeepSalesContro
1 package com.promineotech.jeep.entity;
2
3 public enum JeepModel {
4     WRANGLER, GRAND_CHEROKEE, CHEROKEE, COMPASS, RENEGADE, GLADIATOR, WRANGLER_4XE;
5 }

```

```

FetchJeepTest.java JeepModel.java Jeep.java x Order
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

```

```

FetchJeepTest.java  JeepModel.java  Jeep.java  JeepSales.java ×
1 package com.promineotech.jeep;
2
3 import org.springframework.boot.SpringApplication;
4
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  JeepSales.java  JeepSalesController.java ×  DefaultJe
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(responseCode = "200",
30                 description = "A list of Jeeps is returned",
31                 content = @Content(
32                     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 = {

```

```

47     parameters = {
48         @Parameter(
49             name = "model",
50             allowEmptyValue = false,
51             required = false,
52             description = "The model name (i.e., 'WRANGLER')"),
53         @Parameter(
54             name = "trim",
55             allowEmptyValue = false,
56             required = false,
57             description = "The trim level (i.e., 'Sport')"),
58     }
59 )
60
61 @GetMapping
62 @ResponseStatus(code = HttpStatus.OK)
63 List<Jeep> fetchJeeps(@RequestParam JeepModel model,
64                     @RequestParam String trim);
65 // @formatter:on
66 }
67

```

```

FetchJeepTest.java  JeepModel.java  Jeep.java  JeepSales.java  JeepSalesController.java  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         // TODO Auto-generated method stub
14         return null;
15     }
16
17 }

```

**Screenshots of Running Application:**

```
Console x Problems Debug Shell
<terminated> jeep-sales - JeepSales [Spring Boot App] C:\Users\nspad\Downloads\sts-4.16.0.RELEASE\plugins\org.eclipse.justi.openjdk.hotspot.jre.full.win32.x86_64.17.0.4.v20220903
12:41:39.950 [Thread-0] DEBUG org.springframework.boot.devtools.restart.classloader.RestartClassLoader - Created RestartClassLoade

Spring
=====
:: Spring Boot :: (v2.7.4)

2022-09-25 12:41:40.494 INFO 11524 --- [ restartedMain] com.promineotech.jeepp.JeeppSales : Starting JeepSales using Java
2022-09-25 12:41:40.496 INFO 11524 --- [ restartedMain] com.promineotech.jeepp.JeeppSales : No active profile set, fallin
2022-09-25 12:41:40.589 INFO 11524 --- [ restartedMain] .e.DevToolsPropertyDefaultsPostProcessor : Devtools property defaults ac
2022-09-25 12:41:40.589 INFO 11524 --- [ restartedMain] .e.DevToolsPropertyDefaultsPostProcessor : For additional web related lo
2022-09-25 12:41:42.146 INFO 11524 --- [ restartedMain] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat initialized with port(
2022-09-25 12:41:42.160 INFO 11524 --- [ restartedMain] o.apache.catalina.core.StandardService : Starting service [Tomcat]
2022-09-25 12:41:42.161 INFO 11524 --- [ restartedMain] org.apache.catalina.core.StandardEngine : Starting Servlet engine: [Apa
2022-09-25 12:41:42.280 INFO 11524 --- [ restartedMain] o.a.c.c.C.[Tomcat].[localhost].[/] : Initializing Spring embedded
2022-09-25 12:41:42.280 INFO 11524 --- [ restartedMain] w.s.c.ServletWebServerApplicationContext : Root WebApplicationContext: i
2022-09-25 12:41:43.471 INFO 11524 --- [ restartedMain] o.s.b.d.a.OptionalLiveReloadServer : LiveReload server is running
2022-09-25 12:41:43.525 INFO 11524 --- [ restartedMain] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat started on port(s): 80
2022-09-25 12:41:43.541 INFO 11524 --- [ restartedMain] com.promineotech.jeepp.JeeppSales : Started JeepSales in 3.577 se
2022-09-25 12:45:02.172 INFO 11524 --- [nio-8080-exec-1] o.a.c.c.C.[Tomcat].[localhost].[/] : Initializing Spring Dispatche
2022-09-25 12:45:02.173 INFO 11524 --- [nio-8080-exec-1] o.s.web.servlet.DispatcherServlet : Initializing Servlet 'dispatc
2022-09-25 12:45:02.175 INFO 11524 --- [nio-8080-exec-1] o.s.web.servlet.DispatcherServlet : Completed initialization in 2
2022-09-25 12:45:02.988 INFO 11524 --- [io-8080-exec-10] o.springdoc.api.AbstractOpenApiResource : Init duration for springdoc-o
2022-09-25 12:46:20.117 INFO 11524 --- [on(8)-127.0.0.1] inMXBeanRegistrar$SpringApplicationAdmin : Application shutdown requeste
2022-09-25 12:46:20.179 INFO 11524 --- [on(8)-127.0.0.1] o.apache.catalina.core.StandardService : Stopping service [Tomcat]
2022-09-25 12:46:20.182 INFO 11524 --- [on(8)-127.0.0.1] o.a.c.c.C.[Tomcat].[localhost].[/] : Destroying Spring FrameworkSe
```

URL to GitHub Repository:

<https://github.com/tyoung3614/jeep-sales>