


# Web API Design with Spring Boot Week 2 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.

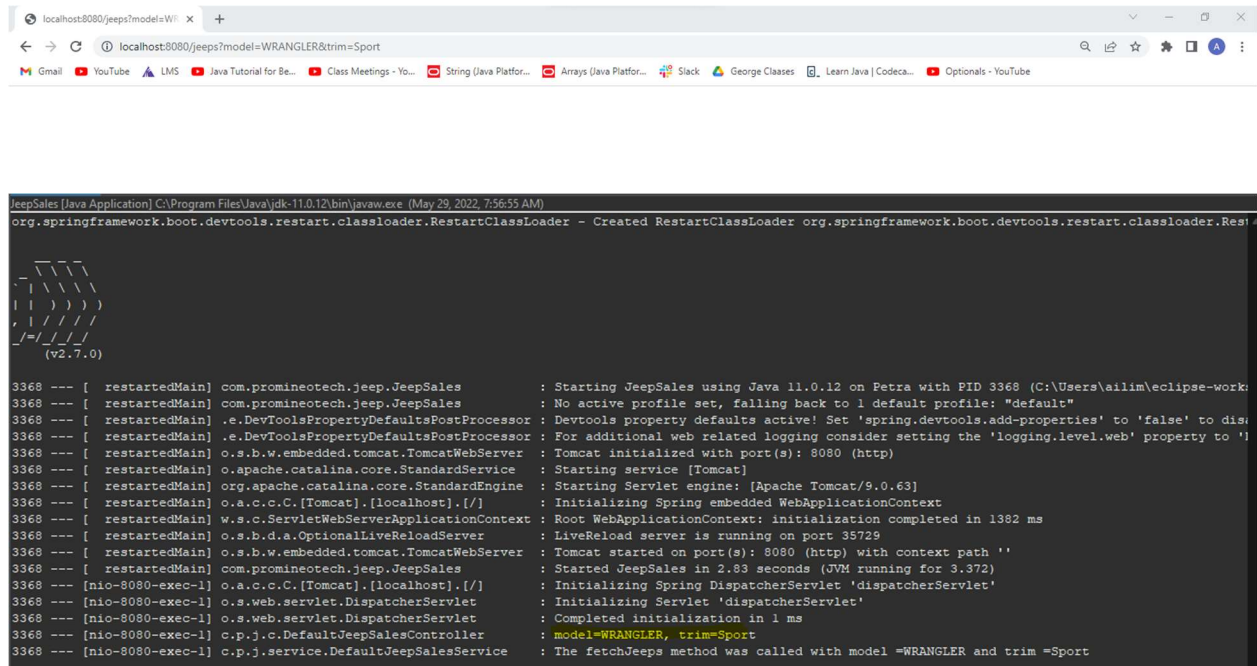
**Here's a friendly tip:** as you watch the videos, code along with the videos. This will help you with the homework. When a screenshot is required, look for the icon:  You will keep adding to this project throughout this part of the course. When it comes time for the final project, use this project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

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

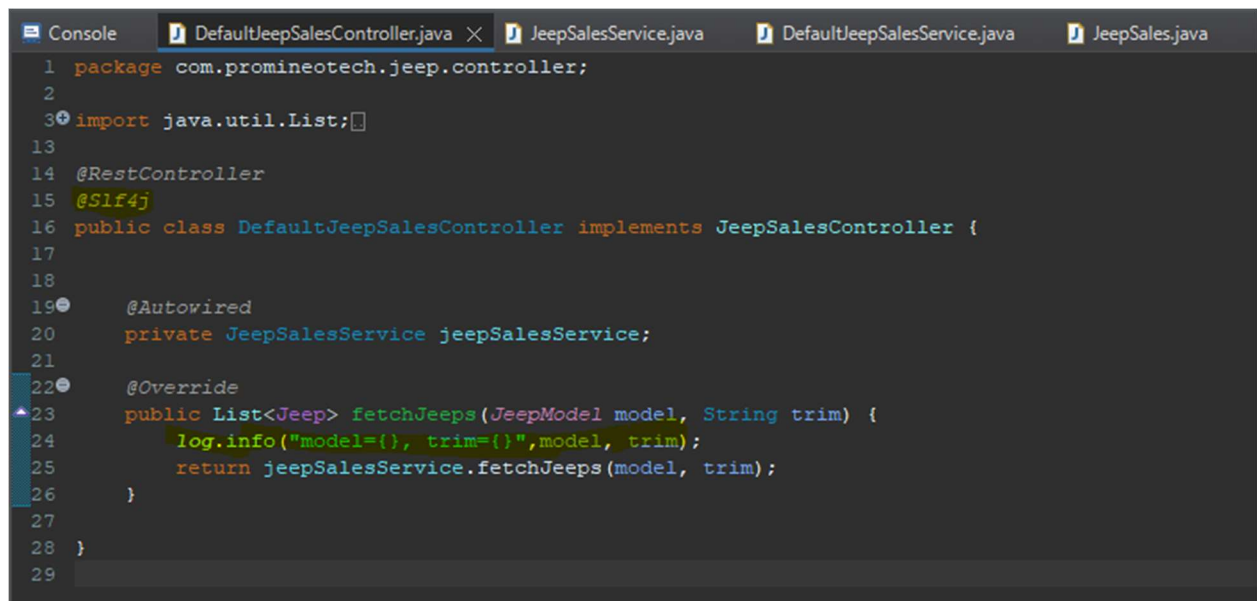
## Coding Steps:

- 1) In the project you started last week, use Lombok to add an info-level logging statement in the controller implementation method that logs the parameters that were input to the method. Remember to add the `@Slf4j` annotation to the class.
- 2) Start the application (not an integration test). Use a browser to navigate to the application passing the parameters required for your selected operation. (A browser, used in this manner, sends an HTTP GET request to the server.) Produce a screenshot showing the browser

navigation bar and the log statement that is in the IDE console showing that the controller method was reached (as in the video). 



The screenshot shows a web browser at `localhost:8080/jeeps?model=WRANGLER&trim=Sport`. The browser's developer tools are open, showing a REST client interface with a GET request to `/jeeps?model=WRANGLER&trim=Sport`. The response is a JSON array of vehicle objects. Below the browser, the IDE console shows the application's startup logs, including the message: `The fetchJeeps method was called with model =WRANGLER and trim =Sport`.



The screenshot shows the source code of `DefaultJeepSalesController.java` in an IDE. The code defines a `RestController` with a `fetchJeeps` method that uses `JeepSalesService` to fetch vehicles. The log statement `log.info("model={}, trim={}", model, trim);` is highlighted, corresponding to the log message seen in the IDE console above.

- 3) With the application still running, use the browser to navigate to the OpenAPI documentation. Use the OpenAPI documentation to send a GET request to the server with a valid model and trim level. (You can get the model and trim from the provided data.sql

file.) Produce a screenshot showing the curl command, the request URL, and the response headers.

## 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 optimal model and/or trim

Parameters

Name	Description
<b>model</b> * required string (query)	Model Name (i.e Wrangler)
<b>trim</b> * required string (query)	Model Name (i.e Sport)

Execute Clear

### Responses

Curl

```
curl -X 'GET' \
  'http://localhost:8080/jeeps?model=GRAND_CHEROKEE&trim=Sport' \
  -H 'accept: application/json'
```

Request URL

http://localhost:8080/jeeps?model=GRAND\_CHEROKEE&trim=Sport

Server response

Code	Details
200	<p>Response headers</p> <pre>connection: keep-alive content-length: 0 date: Sun, 29 May 2022 14:06:15 GMT keep-alive: timeout=60</pre>

Responses

Code	Description	Links
200	SUCCESS	No links

Media type

application/json

Controls Accept header:

Example Value | Schema

```
{}
```

Console

```
JeepSales [Java Application] C:\Program Files\Java\jdk-11.0.12\bin\javaw.exe (May 29, 2022, 7:56:55 AM)
78:06:15.158 INFO 3368 --- [nio-8080-exec-8] c.p.j.c.DefaultJeepSalesController : model=GRAND_CHEROKEE, trim=Sport
78:06:15.158 INFO 3368 --- [nio-8080-exec-8] c.p.j.service.DefaultJeepSalesService : The fetchJeeps method was called with model =GRAND_CHEROKEE and trim =Sport
```

- 4) Run the integration test and show that the test status is green. Produce a screenshot of the test class and the status bar.

```

2022-05-29 08:09:17.547 INFO 7952 --- [main] c.p.jeepp.controller.FetchJeepTest : Starting FetchJeepTest using Java 11.0.12 on Petra with PID 7952
2022-05-29 08:09:18.581 INFO 7952 --- [main] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat initialized with port(s): 0 (http)
2022-05-29 08:09:18.590 INFO 7952 --- [main] org.apache.catalina.core.StandardService : Starting service [Tomcat]
2022-05-29 08:09:18.590 INFO 7952 --- [main] org.apache.catalina.core.StandardEngine : Starting Servlet engine: [Apache Tomcat/9.0.63]
2022-05-29 08:09:18.617 INFO 7952 --- [main] o.a.c.c.C.[Tomcat].[localhost].[/] : Initializing Spring embedded WebApplicationContext
2022-05-29 08:09:18.617 INFO 7952 --- [main] w.s.c.ServletWebServerApplicationContext : Root WebApplicationContext: initialization completed in 1242 ms
2022-05-29 08:09:19.941 INFO 7952 --- [main] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat started on port(s): 50176 (http) with context path ''
2022-05-29 08:09:19.953 INFO 7952 --- [main] c.p.jeepp.controller.FetchJeepTest : Started FetchJeepTest in 2.721 seconds (JVM running for 4.9)
2022-05-29 08:09:20.380 INFO 7952 --- [o-auto-1-exec-1] o.a.c.c.C.[Tomcat].[localhost].[/] : Initializing Servlet 'dispatcherServlet'
2022-05-29 08:09:20.381 INFO 7952 --- [o-auto-1-exec-1] o.s.web.servlet.DispatcherServlet : Completed initialization in 1 ms
2022-05-29 08:09:20.418 INFO 7952 --- [o-auto-1-exec-1] c.p.j.c.DefaultJeepSalesController : model=WRANGLER, trim=Sport
2022-05-29 08:09:20.421 INFO 7952 --- [o-auto-1-exec-1] c.p.j.service.DefaultJeepSalesService : The fetchJeeps method was called with model =WRANGLER and trim =

```

```


1 package com.promineotech.jeepp.controller;
2
3 import org.junit.jupiter.api.Test;
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 })
12 class FetchJeepTest {
13
14     @Autowired
15     private TestRestTemplate restTemplate;
16
17     @LocalServerPort
18     private int serverPort;
19
20     @Test
21     void testThatJeepsAreReturnedWhenAValidModelAndTrimAreSupplied() {
22         JeepModel model = JeepModel.WRANGLER;
23         String trim = "Sport";
24         String uri = String.format("http://localhost:%d/jeeps?model=%s&trim=%s", serverPort, model, trim);
25         System.out.println(uri);
26         ResponseEntity<List<Jeep>> response = restTemplate.exchange(uri, HttpMethod.GET, null, new ParameterizedTypeReference<>() {});
27         assertEquals(response.getStatusCode(), HttpStatus.OK);
28     }
29 }

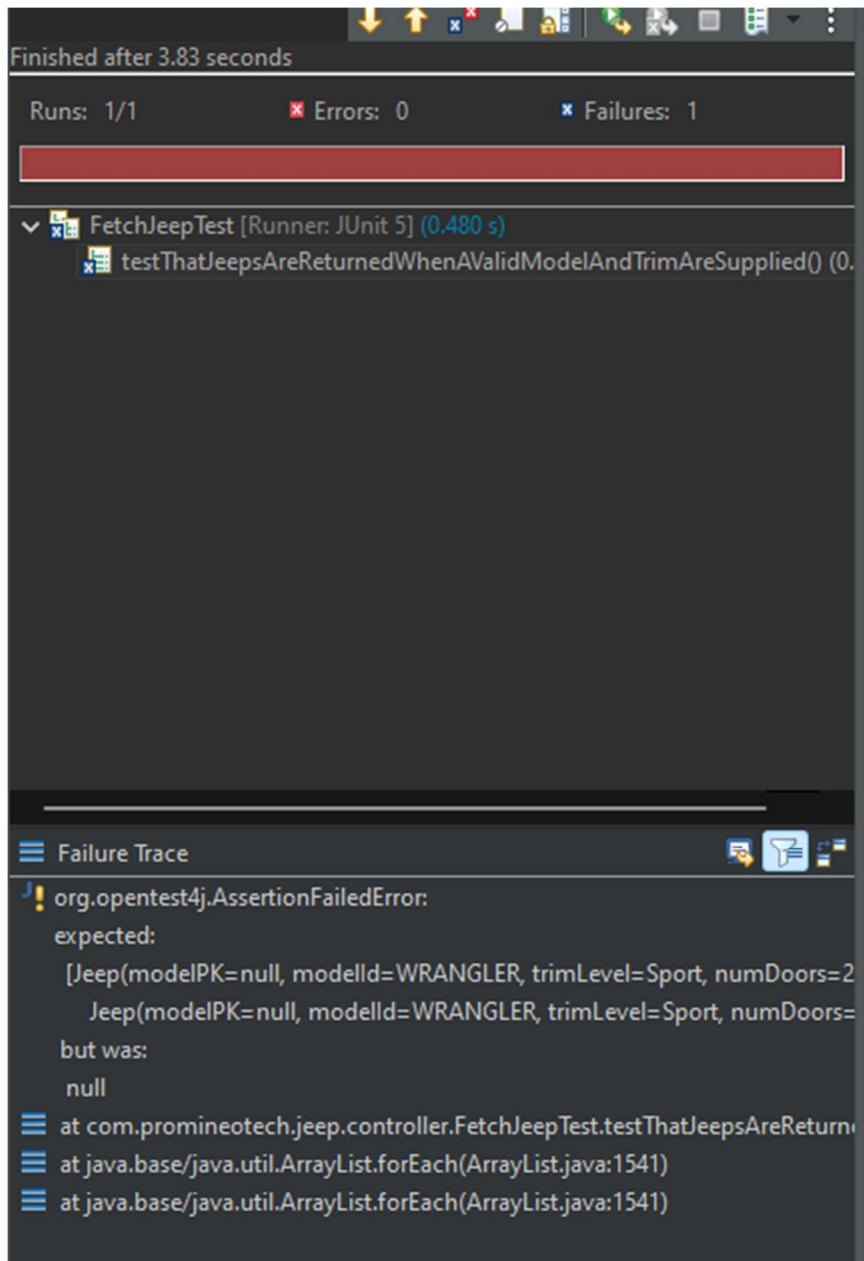
```

- 5) Add a method to the test to return a list of expected Jeep (model) objects based on the model and trim level you selected. You can get the expected list of Jeeps from the file `src/test/resources/flyway/migrations/V1.1__Jeep_Data.sql`. So, for example, using the model Wrangler and trim level "Sport", the query should return two rows:

	Row 1	Row 2
Model ID	WRANGLER	WRANGLER
Trim Level	Sport	Sport
Num Doors	2	4
Wheel Size	17	17
Base Price	\$28,475.00	\$31,975.00

The method should be named `buildExpected()`, and it should return a `List of Jeep`. The video put this method into a support superclass but you can include it in the main test class if you want.

- 6) Write an AssertJ assertion in the test to assert that the actual list of jeeps returned by the server is the same as the expected list. Run the test. Produce a screenshot showing...
  - a) The test with the assertion.
  - b) The JUnit status bar (should be red).
  - c) The method returning the expected list of Jeeps. 





```

1 package com.promineotech.jeep.controller;
2
3 import org.junit.jupiter.api.Disabled;
4
5
6
7 @SpringBootTest(webEnvironment = WebEnvironment.RANDOM_PORT)
8 @ActiveProfiles("test")
9 @Sql(scripts = {
10     "classpath:flyway/migrations/V1.0_Jeep_Schema.sql",
11     "classpath:flyway/migrations/V1.1_Jeep_Data.sql"},
12     config = @SqlConfig(encoding = "utf-8"))
13 class FetchJeepTest {
14
15     @Autowired
16     private TestRestTemplate restTemplate;
17
18     @LocalServerPort
19     private int serverPort;
20
21
22     @Test
23     void testThatJeepsAreReturnedWhenAValidModelAndTrimAreSupplied() {
24         JeepModel model = JeepModel.WRANGLER;
25         String trim = "Sport";
26         String uri = String.format("http://localhost:%d/jeeps?model=%s&trim=%s", serverPort, model, trim);
27         System.out.println(uri);
28         ResponseEntity<List<Jeep>> response = restTemplate.exchange(uri, HttpMethod.GET, null, new ParameterizedTypeReference<>() {});
29         assertEquals(response.getStatusCode(), HttpStatus.OK);
30
31         //Add a method to the test to return a list of expected Jeep (model) objects
32         List<Jeep> expected = buildExpected();
33         System.out.println(expected);
34         assertEquals(response.getBody(), expected);
35     }
36
37     protected List<Jeep> buildExpected() {
38         List<Jeep> list = new LinkedList<>();
39
40         // @formatter:off
41         list.add(Jeep.builder()
42             .modelId(JeepModel.WRANGLER)
43             .trimLevel("Sport")
44             .numDoors(2)
45             .wheelSize(17)
46             .basePrice(new BigDecimal("28475.00"))
47             .build());
48
49         list.add(Jeep.builder()
50             .modelId(JeepModel.WRANGLER)
51             .trimLevel("Sport")
52             .numDoors(4)
53             .wheelSize(17)
54             .basePrice(new BigDecimal("31975.00"))
55             .build());
56         // @formatter:on
57
58         return list;
59     }
60 }

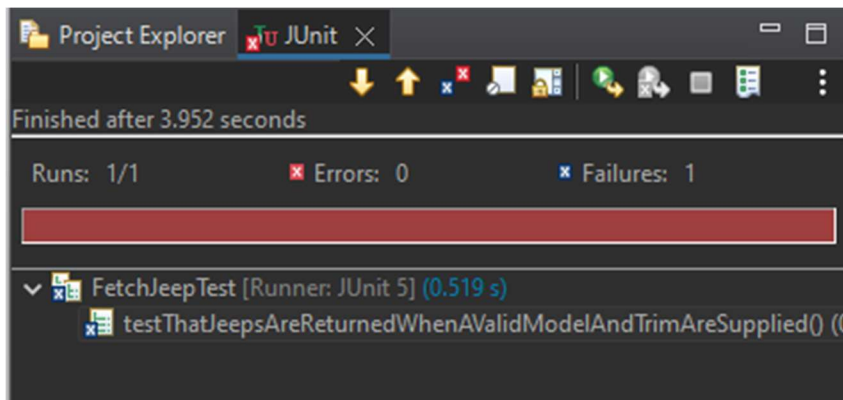
```

7) Add a service layer in your application as shown in the videos:

- a) Add a package named com.promineotech.jeep.service.
- b) In the new package, create an interface named JeepSalesService.

- c) In the same package (service), create a class named DefaultJeepSalesService that implements the JeepSalesService interface. Add the class-level annotation, @Service.
- d) Inject the service interface into DefaultJeepSalesController using the @Autowired annotation. The instance variable should be private, and the variable should be named jeepSalesService.
- e) Define the fetchJeeps method in the interface. Implement the method in the service class. Call the method from the controller (make sure the controller returns the list of Jeeps returned by the service method). The method signature looks like this:  

```
List<Jeep> fetchJeeps(JeepModel model, String trim);
```
- f) Add a Lombok info-level log statement in the service implementation showing that the service was called. Print the parameters passed to the method. Let the method return null for now.
- g) Run the test again. Produce a screenshot showing the service class implementation, the log line in the console, and the red status bar. 🖥️



```
Console X FetchJeepTest.java
<terminated> FetchJeepTest [Unit] C:\Program Files\Java\jdk-11.0.12\bin\javaw.exe (May 29, 2022 8:24:02 AM - 8:24:07 AM)

08:24:04.502 [main] INFO org.springframework.test.context.support.AbstractContextLoader - Could not detect default resource locations for test class [com.promineotech.jeep.controller.FetchJeepTest]: no ResourceBundle found.
08:24:04.502 [main] INFO org.springframework.test.context.support.AnnotationConfigContextLoaderUtils - Could not detect default configuration classes for test class [com.promineotech.jeep.controller.FetchJeepTest]: no configuration classes found in package [com.promineotech.jeep.controller].
08:24:04.035 [main] DEBUG org.springframework.context.annotation.ClassPathScanningCandidateComponentProvider - Identified candidate component class: file [C:\Users\allim\src\jeep\src\main\java\com\promineotech\jeep\JeepSales.class]
08:24:04.037 [main] INFO org.springframework.boot.test.context.SpringBootTestContextBootstrapper - Found @SpringBootConfiguration com.promineotech.jeep.JeepSales for test class [com.promineotech.jeep.controller.FetchJeepTest]
08:24:04.152 [main] DEBUG org.springframework.boot.test.context.SpringBootTestContextBootstrapper - @TestExecutionListeners is not present for class [com.promineotech.jeep.controller.FetchJeepTest]: using the default
08:24:04.153 [main] INFO org.springframework.boot.test.context.SpringBootTestContextBootstrapper - Loaded default TestExecutionListener class names from location [META-INF/test-execution-listeners.properties]: org.springframework.boot.test.autoconfigure.TestExecutionListeners
08:24:04.170 [main] DEBUG org.springframework.boot.test.context.SpringBootTestContextBootstrapper - Skipping candidate TestExecutionListener [org.springframework.test.context.junit4.SpringJUnit4ClassRunner]
08:24:04.171 [main] INFO org.springframework.boot.test.context.SpringBootTestContextBootstrapper - Using TestExecutionListeners: [org.springframework.test.context.web.ServletTestExecutionListener, org.springframework.test.context.support.DefaultTestExecutionListener]
08:24:04.175 [main] DEBUG org.springframework.test.context.support.AbstractDirtiesContextTestExecutionListener - Before test class: context [DefaultTestContext@278bb07e testClass=com.promineotech.jeep.controller.FetchJeepTest]
08:24:04.193 [main] DEBUG org.springframework.test.context.support.DependencyInjectionTestExecutionListener - Performing dependency injection for test context [[DefaultTestContext@278bb07e testClass=com.promineotech.jeep.controller.FetchJeepTest]]

:: Spring Boot ::
(v2.7.0)

2022-05-29 08:24:04.548 INFO 1120 --- [main] c.p.jeep.controller.FetchJeepTest : Starting FetchJeepTest using Java 11.0.12 on Petra with PID 1120 (started by allim)
2022-05-29 08:24:04.549 INFO 1120 --- [main] c.p.jeep.controller.FetchJeepTest : The following 1 profile is active: "test"
2022-05-29 08:24:05.593 INFO 1120 --- [main] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat initialized with port(s): 0 (http)
2022-05-29 08:24:05.602 INFO 1120 --- [main] o.apache.catalina.core.StandardService : Starting service [Tomcat]
2022-05-29 08:24:05.602 INFO 1120 --- [main] org.apache.catalina.core.StandardEngine : Starting Servlet engine: [Apache Tomcat/9.0.63]
2022-05-29 08:24:05.847 INFO 1120 --- [main] o.a.c.c.C.[Tomcat].[localhost].[/] : Initializing Spring embedded WebApplicationContext
2022-05-29 08:24:05.847 INFO 1120 --- [main] w.s.c.ServletWebServerApplicationContext : Root WebApplicationContext: initialization completed in 1270 ms
2022-05-29 08:24:07.063 INFO 1120 --- [main] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat started on port(s): 50241 (http) with context path ''
2022-05-29 08:24:07.077 INFO 1120 --- [main] c.p.jeep.controller.FetchJeepTest : Started FetchJeepTest in 2.851 seconds (JVM running for 3.976)
http://localhost:50241/jeeps?model=WRANGLER&trim=Sport2022-05-29 08:24:07.514 INFO 1120 --- [o-auto-1-exec-1] o.a.c.c.C.[Tomcat].[localhost].[/] : Initializing Spring
2022-05-29 08:24:07.515 INFO 1120 --- [o-auto-1-exec-1] o.s.web.servlet.DispatcherServlet : Initializing Servlet 'dispatcherServlet'
2022-05-29 08:24:07.516 INFO 1120 --- [o-auto-1-exec-1] o.s.web.servlet.DispatcherServlet : Completed initialization in 1 ms
2022-05-29 08:24:07.551 INFO 1120 --- [o-auto-1-exec-1] c.p.j.c.DefaultJeepSalesController : model=WRANGLER, trim=Sport
2022-05-29 08:24:07.554 INFO 1120 --- [o-auto-1-exec-1] c.p.j.service.DefaultJeepSalesService : The fetchJeeps method was called with model =WRANGLER and trim =Sport
[Jeep(modelPK=null, modelId=WRANGLER, trimLevel=Sport, numDoors=2, wheelSize=17, basePrice=28475.00), Jeep(modelPK=null, modelId=WRANGLER, trimLevel=Sport, numDoors=4, wheelSize=17, basePrice=32475.00)]
```

```
Console X FetchJeepTest.java X
1 package com.promineotech.jeep.controller;
2
30 import org.junit.jupiter.api.Disabled;
25
26
27 @SpringBootTest(webEnvironment = WebEnvironment.RANDOM_PORT)
28 @ActiveProfiles("test")
29 @Sql(scripts = {
30     "classpath:flyway/migrations/V1.0__Jeep_Schema.sql",
31     "classpath:flyway/migrations/V1.1__Jeep_Data.sql"},
32     config = @SqlConfig(encoding = "utf-8"))
33 class FetchJeepTest {
34
35     @Autowired
36     private TestRestTemplate restTemplate;
37
38     @LocalServerPort
39     private int serverPort;
40
41
42     @Test
43     void testThatJeepsAreReturnedWhenAValidModelAndTrimAreSupplied() {
44         JeepModel model = JeepModel.WRANGLER;
45         String trim = "Sport";
46         String uri = String.format("http://localhost:%d/jeeps?model=%s&trim=%s", serverPort, model, trim);
47         System.out.println(uri);
48         ResponseEntity<List<Jeep>> response = restTemplate.exchange(uri, HttpMethod.GET, null, new ParameterizedTypeReference<>() {});
49         assertEquals(response.getStatusCode(), HttpStatus.OK);
50
51         //Add a method to the test to return a list of expected Jeep (model) objects
52         List<Jeep> expected = buildExpected();
53         System.out.println(expected);
54         assertEquals(response.getBody(), expected);
55     }
56
57 }
```



```

58 protected List<Jeep> buildExpected() {
59     List<Jeep> list = new LinkedList<>();
60
61     // @formatter:off
62     list.add(Jeep.builder()
63         .modelId(JeepModel.WRANGLER)
64         .trimLevel("Sport")
65         .numDoors(2)
66         .wheelSize(17)
67         .basePrice(new BigDecimal("28475.00"))
68         .build());
69
70     list.add(Jeep.builder()
71         .modelId(JeepModel.WRANGLER)
72         .trimLevel("Sport")
73         .numDoors(4)
74         .wheelSize(17)
75         .basePrice(new BigDecimal("31975.00"))
76         .build());
77     // @formatter:on
78
79
80
81
82     return list;
83 }
84 }
85

```


- 8) Add the database dependencies described in the video to the POM file (MySQL driver and Spring Boot Starter JDBC). To find them, navigate to <https://mvnrepository.com/>. Search for mysql-connector-j and spring-boot-starter-jdbc. In the POM file you don't need version numbers for either dependency because the version is included in the Spring Boot Starter Parent.
- 9) Create application.yaml in src/main/resources. Add the spring.datasource.url, spring.datasource.username, and spring.datasource.password properties to application.yaml. The url should be the same as shown in the video (jdbc:mysql://localhost:3306/jeep). The password and username should match your setup. If you created the database under your root user, the username is "root", and the password is the root user password. If you created a "jeep" user or other user, use the correct username and password.

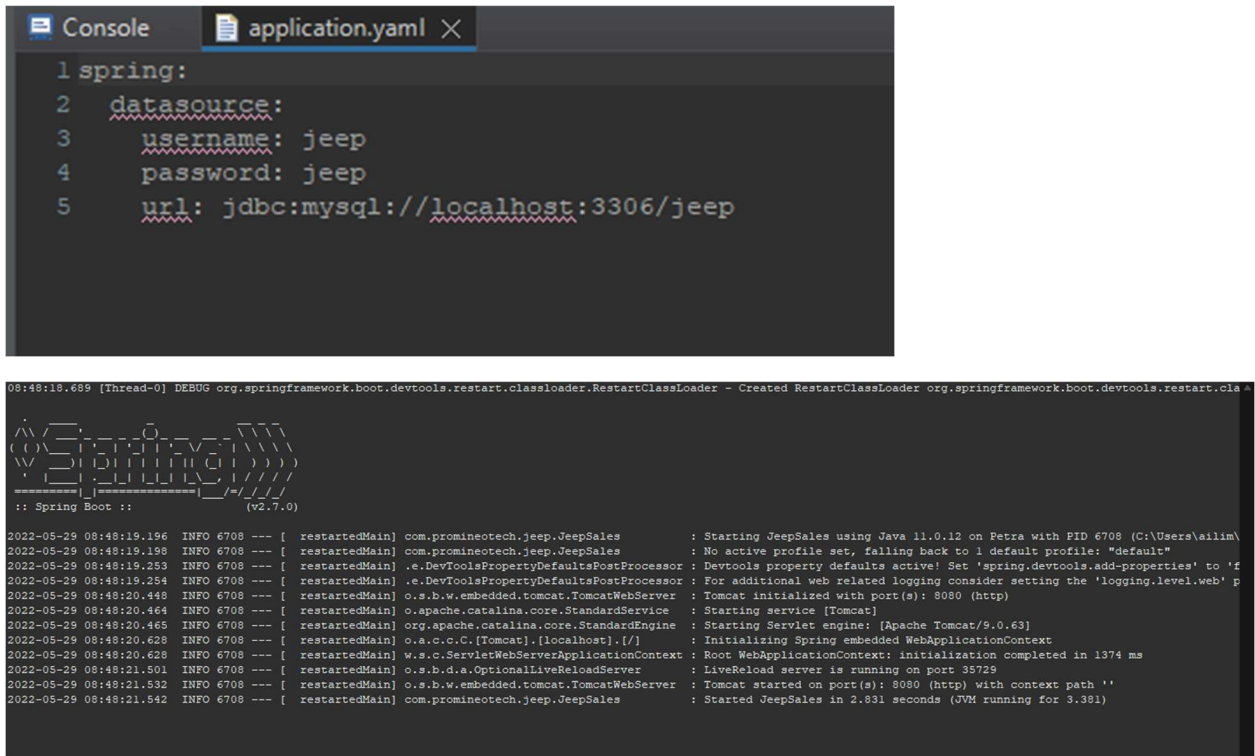
Be careful with the indentation! YAML allows hierarchical configuration but it reads the hierarchy based on the indentation level. The keyword "spring" MUST start in the first column. It should look similar to this when done:

```

spring:
  datasource:
    username: username
    password: password
    url: jdbc:mysql://localhost:3306/jeep

```

- 10) Start the application (the real application, not the test). Produce a screenshot that shows application.yaml and the console showing that the application has started with no errors. 



The screenshot shows an IDE with two tabs: 'Console' and 'application.yaml'. The 'application.yaml' tab is active, displaying the following configuration:

```
1 spring:
2   datasource:
3     username: jeep
4     password: jeep
5     url: jdbc:mysql://localhost:3306/jeep
```

The 'Console' tab shows the following output:

```
08:48:18.689 [Thread-0] DEBUG org.springframework.boot.devtools.restart.classloader.RestartClassLoader - Created RestartClassLoader org.springframework.boot.devtools.restart.clas
:: Spring Boot :: (v2.7.0)


2022-05-29 08:48:19.196 INFO 6708 --- [ restartedMain] com.promineotech.jeepp.JeeppSales : Starting JeeppSales using Java 11.0.12 on Petra with PID 6708 (C:\Users\ailim\
2022-05-29 08:48:19.198 INFO 6708 --- [ restartedMain] com.promineotech.jeepp.JeeppSales : No active profile set, falling back to 1 default profile: "default"
2022-05-29 08:48:19.253 INFO 6708 --- [ restartedMain] .e.DevToolsPropertyDefaultsPostProcessor : DevTools property defaults active! Set 'spring.devtools.add-properties' to 'f
2022-05-29 08:48:19.254 INFO 6708 --- [ restartedMain] .e.DevToolsPropertyDefaultsPostProcessor : For additional web related logging consider setting the 'logging.level.web' p
2022-05-29 08:48:20.448 INFO 6708 --- [ restartedMain] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat initialized with port(s): 8080 (http)
2022-05-29 08:48:20.464 INFO 6708 --- [ restartedMain] o.apache.catalina.core.StandardService : Starting service [Tomcat]
2022-05-29 08:48:20.465 INFO 6708 --- [ restartedMain] org.apache.catalina.core.StandardEngine : Starting Servlet engine: [Apache Tomcat/9.0.63]
2022-05-29 08:48:20.628 INFO 6708 --- [ restartedMain] o.a.c.c.C.[Tomcat].[localhost].[/] : Initializing Spring embedded WebApplicationContext
2022-05-29 08:48:20.628 INFO 6708 --- [ restartedMain] w.s.c.ServletWebServerApplicationContext : Root WebApplicationContext: initialization completed in 1374 ms
2022-05-29 08:48:21.501 INFO 6708 --- [ restartedMain] o.s.b.d.a.OptionalLiveReloadServer : LiveReload server is running on port 35729
2022-05-29 08:48:21.532 INFO 6708 --- [ restartedMain] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat started on port(s): 8080 (http) with context path ''
2022-05-29 08:48:21.542 INFO 6708 --- [ restartedMain] com.promineotech.jeepp.JeeppSales : Started JeeppSales in 2.831 seconds (JVM running for 3.381)
```

- 11) Add the H2 database as dependency. Search for the dependency in the Maven repository like you did above. Search for "h2" and pick the latest version. Again, you don't need the version number, but the scope should be set to "test".

- 12) Create application-test.yaml in src/test/resources. Add the setting spring.datasource.url that points to the H2 database. It should look like this:

```
spring:
  datasource:
    url: jdbc:h2:mem:jeep
```

You do not need to set the username and password because the in-memory H2 database does not require them.

Produce a screenshot showing application-test.yaml. 

```
1 spring:
2   datasource:
3     url: jdbc:h2:mem:jeep
4
5 logging:
6   level:
7     root: warn
8   '[com.promineotech]' : debug
```

```
Console application.yaml application-test.yaml jeep-sales/pom.xml X
1 <project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="ht
2   <modelVersion>4.0.0</modelVersion>
3   <parent>
4     <groupId>org.springframework.boot</groupId>
5     <artifactId>spring-boot-starter-parent</artifactId>
6     <version>2.7.0</version>
7     <relativePath/> <!-- lookup parent from repository -->
```

Finished after 4.482 seconds

Runs: 1/1

Errors: 1

Failures: 0

FetchJeepTest [Runner: JUnit 5] (0.789 s)

testThatJeepsAreReturnedWhenAValidModelAndTrimAreSupplied() (0.789 s)

#### Failure Trace

```
org.springframework.jdbc.datasource.init.ScriptStatementFailedException: Failed to execute
customer_id varchar(40) NOT NULL, first_name varchar(45) NOT NULL, last_name varchar(
tomers ( customer_pk int [*]unsigned NOT NULL AUTO_INCREMENT, customer_id varchar(
, AS, DEFAULT, GENERATED, ON, NOT, NULL, AUTO_INCREMENT, DEFAULT, NULL_TO_DEFAL
CREATE TABLE customers ( customer_pk int unsigned NOT NULL AUTO_INCREMENT, custo
at org.springframework.jdbc.datasource.init.ScriptUtils.executeSqlScript(ScriptUtils.java:28
at org.springframework.jdbc.datasource.init.ResourceDatabasePopulator.populate(Resour
at org.springframework.jdbc.datasource.init.DatabasePopulatorUtils.execute(DatabasePop
at org.springframework.jdbc.datasource.init.ResourceDatabasePopulator.execute(Resourc
at org.springframework.test.context.jdbc.SqlScriptsTestExecutionListener.lambda$execute
at org.springframework.transaction.support.TransactionOperations.lambda$executeWitho
at org.springframework.transaction.support.TransactionTemplate.execute(TransactionTem
at org.springframework.transaction.support.TransactionOperations.executeWithoutResult(
at org.springframework.test.context.jdbc.SqlScriptsTestExecutionListener.executeSqlScript
at org.springframework.test.context.jdbc.SqlScriptsTestExecutionListener.lambda$execute
```

```

09:02:01.616 [main] DEBUG org.springframework.test.context.BootstrapUtils - Instantiating CacheAwareContextLoaderDelegate from class [org.springframework.test.context.cache.DefaultCacheAwareContextLoaderDelegate]
09:02:01.625 [main] DEBUG org.springframework.test.context.BootstrapUtils - Instantiating BootstrapContext using constructor [public org.springframework.test.context.support.DefaultBootstrapContext(java.lang.Class, org.springframework.test.context.CacheAwareContextLoaderDelegate)]
09:02:01.653 [main] DEBUG org.springframework.test.context.BootstrapUtils - Instantiating TestContextBootstrapper for test class [com.promineotech.jeeptest.FetchJeepTest]
09:02:01.661 [main] INFO org.springframework.boot.test.context.SpringBootTestContextBootstrapper - Neither @ContextConfiguration nor @ContextHierarchy found for test class [com.promineotech.jeeptest.FetchJeepTest]
09:02:01.664 [main] DEBUG org.springframework.test.context.support.AbstractContextLoader - Did not detect default resource location for test class [com.promineotech.jeeptest.FetchJeepTest]: resourceLocation [null]
09:02:01.665 [main] INFO org.springframework.test.context.support.AbstractContextLoader - Could not detect default resource locations for test class [com.promineotech.jeeptest.FetchJeepTest]: resourceLocations [null]
09:02:01.665 [main] INFO org.springframework.test.context.support.AnnotationConfigContextLoaderUtils - Could not detect default configuration classes for test class [com.promineotech.jeeptest.FetchJeepTest]: configurationClasses [null]
09:02:01.796 [main] DEBUG org.springframework.test.context.annotation.ClassPathScanningCandidateComponentProvider - Identified candidate component classes: file [C:\Users\ailimuta\workspace\jeep-test\src\test\java\com\promineotech\jeeptest\]
09:02:01.799 [main] INFO org.springframework.boot.test.context.SpringBootTestContextBootstrapper - Found @SpringBootConfiguration com.promineotech.jeeptest.JeeptestApplication for test class [com.promineotech.jeeptest.FetchJeepTest]
09:02:01.886 [main] DEBUG org.springframework.boot.test.context.SpringBootTestContextBootstrapper - @TestExecutionListeners is not present for class [com.promineotech.jeeptest.FetchJeepTest]
09:02:01.886 [main] INFO org.springframework.boot.test.context.SpringBootTestContextBootstrapper - Loaded default TestExecutionListener class names from location [META-INF/spring.org.springframework.test.context.TestExecutionListeners]
09:02:01.903 [main] INFO org.springframework.boot.test.context.SpringBootTestContextBootstrapper - Using TestExecutionListeners: [org.springframework.test.context.web.ServletTestExecutionListener, org.springframework.test.context.support.DependencyInjectionTestExecutionListener, org.springframework.test.context.support.DefaultTestExecutionListener, org.springframework.test.context.support.MethodInvocationTestExecutionListener]
09:02:01.919 [main] DEBUG org.springframework.test.context.support.DependencyInjectionTestExecutionListener - Before test class: context [DefaultTestContext@8b7838a9 testClass = com.promineotech.jeeptest.FetchJeepTest, testMethod = testDb(), testException = null, testContextClassLoader = org.springframework.test.context.support.DefaultTestContextClassLoader]
09:02:01.941 [main] DEBUG org.springframework.core.env.StandardEnvironment - Activating profiles [test]
09:02:01.945 [main] DEBUG org.springframework.test.context.support.TestPropertySourceUtils - Adding inlined properties to environment: (spring.jmx.enabled=false, org.springframework.test.context.cache=org.springframework.test.context.cache.DefaultCacheAwareContextLoaderDelegate)

:: Spring Boot ::
(v2.4.3)

2022-05-29 09:02:02.269 INFO 11124 --- [main] c.p.jeeptest.controller.FetchJeepTest : Starting FetchJeepTest using Java 11.0.12 on Petra with PID 11124 (started by user)
2022-05-29 09:02:02.269 DEBUG 11124 --- [main] c.p.jeeptest.controller.FetchJeepTest : Running with Spring Boot v2.4.3, Spring v5.3.4
2022-05-29 09:02:02.270 INFO 11124 --- [main] c.p.jeeptest.controller.FetchJeepTest : The following profiles are active: test
2022-05-29 09:02:05.006 INFO 11124 --- [main] c.p.jeeptest.controller.FetchJeepTest : Started FetchJeepTest in 3.058 seconds (JVM running for 4.164)

```

Note version was changed to from `<version>2.7.0</version>` to `<version>2.4.3</version>` the same in the video and it passed

```

5 <artifactId>spring-boot-starter-parent</artifactId>
6 <version>2.4.3</version>
7 <relativePath/> <!-- lookup parent from repository -->

```

```

Finished after 4.19 seconds

Runs: 2/2 (1 skipped)  Errors: 0  Failures: 0

FetchJeepTest [Runner: JUnit 5] (0.600 s)

```

```

40 @Test
41 void testDb() {
42 }
43
44 @Disabled
45 @Test
46 void testThatJeepsAreReturnedWhenAValidModelAndTrimAreSupplied() {
47     JeepModel model = JeepModel.WRANGLER;
48     String trim = "Sport";
49     String url = String.format("http://localhost:%d/jeeps?model=%s&trim=%s", serverPort, model, trim);
50     System.out.println(url);
51     ResponseEntity<List<Jeep>> response = restTemplate.exchange(url, HttpMethod.GET, null, new ParameterizedTypeReference<>() {});
52     assertThat(response.getStatusCode(), equalTo(HttpStatus.OK));
53
54     //Add a method to the test to return a list of expected Jeep (model) objects
55     List<Jeep> expected = buildExpected();
56     System.out.println(expected);
57     assertThat(response.getBody(), equalTo(expected));
58 }

```

As instructed in the video, create first a new test db method and disable the method, `testThatJeepsAreReturnedWhenAValidModelAndTrimAreSupplied()`, then run.

URL to GitHub Repository:

<https://github.com/ailimutan/webapi-week14-assignment>