


# 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.

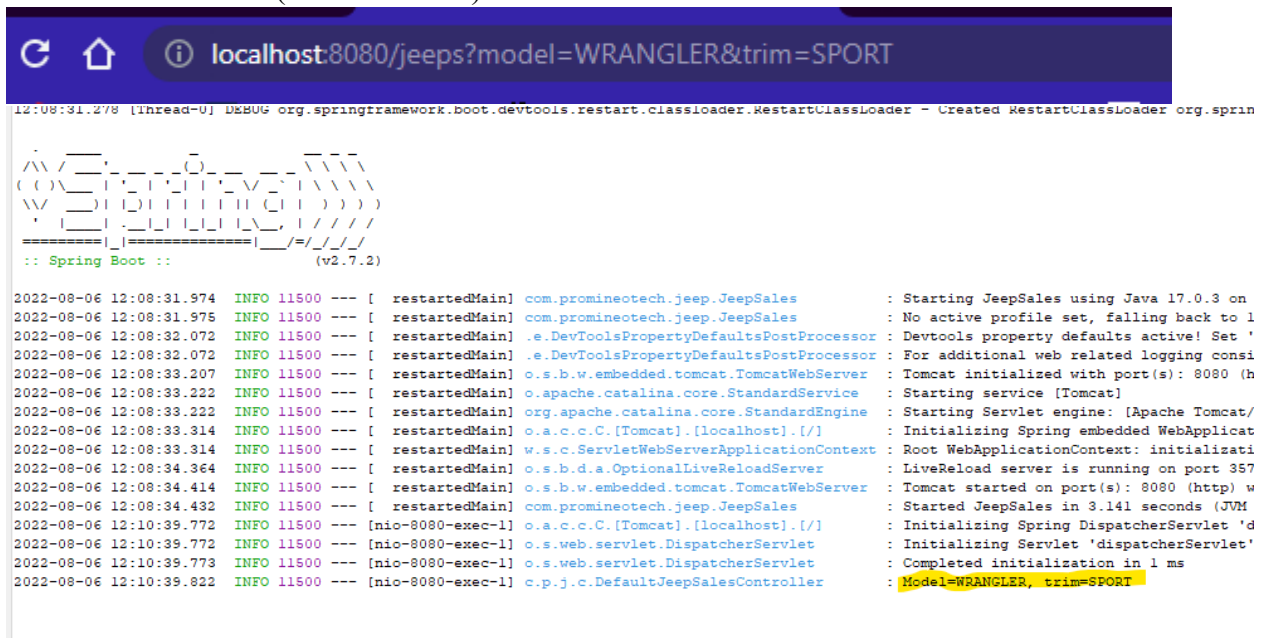
**Here's a friendly tip:** as you watch the videos, code along with the videos. This will help you with the homework. When a screenshot is required, look for the icon:  You will keep adding to this project throughout this part of the course. When it comes time for the final project, use this project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

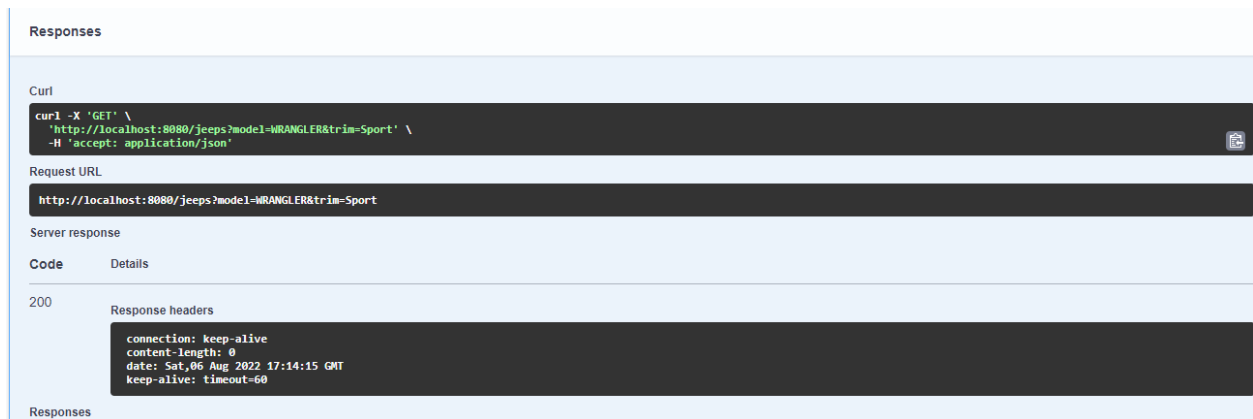


```
12:08:31.278 [Thread-0] DEBUG org.springframework.boot.devtools.restart.classloader.RestartClassLoader - Created RestartClassLoader org.sprin

:: Spring Boot ::
(v2.7.2)

2022-08-06 12:08:31.974 INFO 11500 --- [ restartedMain] com.promineotech.jee JeepSales : Starting JeepSales using Java 17.0.3 on
2022-08-06 12:08:31.975 INFO 11500 --- [ restartedMain] com.promineotech.jee JeepSales : No active profile set, falling back to l
2022-08-06 12:08:32.072 INFO 11500 --- [ restartedMain] .e.DevToolsPropertyDefaultsPostProcessor : Devtools property defaults active! Set '
2022-08-06 12:08:32.072 INFO 11500 --- [ restartedMain] .e.DevToolsPropertyDefaultsPostProcessor : For additional web related logging consi
2022-08-06 12:08:33.207 INFO 11500 --- [ restartedMain] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat initialized with port(s): 8080 (h
2022-08-06 12:08:33.222 INFO 11500 --- [ restartedMain] o.apache.catalina.core.StandardService : Starting service [Tomcat]
2022-08-06 12:08:33.222 INFO 11500 --- [ restartedMain] org.apache.catalina.core.StandardEngine : Starting Servlet engine: [Apache Tomcat/
2022-08-06 12:08:33.314 INFO 11500 --- [ restartedMain] o.a.c.c.C.[Tomcat].[localhost].[/] : Initializing Spring embedded WebApplicat
2022-08-06 12:08:33.314 INFO 11500 --- [ restartedMain] w.s.c.ServletWebServerApplicationContext : Root WebApplicationContext: initializati
2022-08-06 12:08:33.364 INFO 11500 --- [ restartedMain] o.s.b.d.a.OptionalLiveReloadServer : LiveReload server is running on port 357
2022-08-06 12:08:34.414 INFO 11500 --- [ restartedMain] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat started on port(s): 8080 (http) w
2022-08-06 12:08:34.432 INFO 11500 --- [ restartedMain] com.promineotech.jee JeepSales : Started JeepSales in 3.141 seconds (JVM
2022-08-06 12:10:39.772 INFO 11500 --- [nio-8080-exec-1] o.a.c.c.C.[Tomcat].[localhost].[/] : Initializing Spring DispatcherServlet 'd
2022-08-06 12:10:39.773 INFO 11500 --- [nio-8080-exec-1] o.s.web.servlet.DispatcherServlet : Initializing Servlet 'dispatcherServlet'
2022-08-06 12:10:39.822 INFO 11500 --- [nio-8080-exec-1] c.p.j.c.DefaultJeepSalesController : Completed initialization in 1 ms
: Model=WRANGLER, trim=SPORT
```

- 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. 



```
Responses

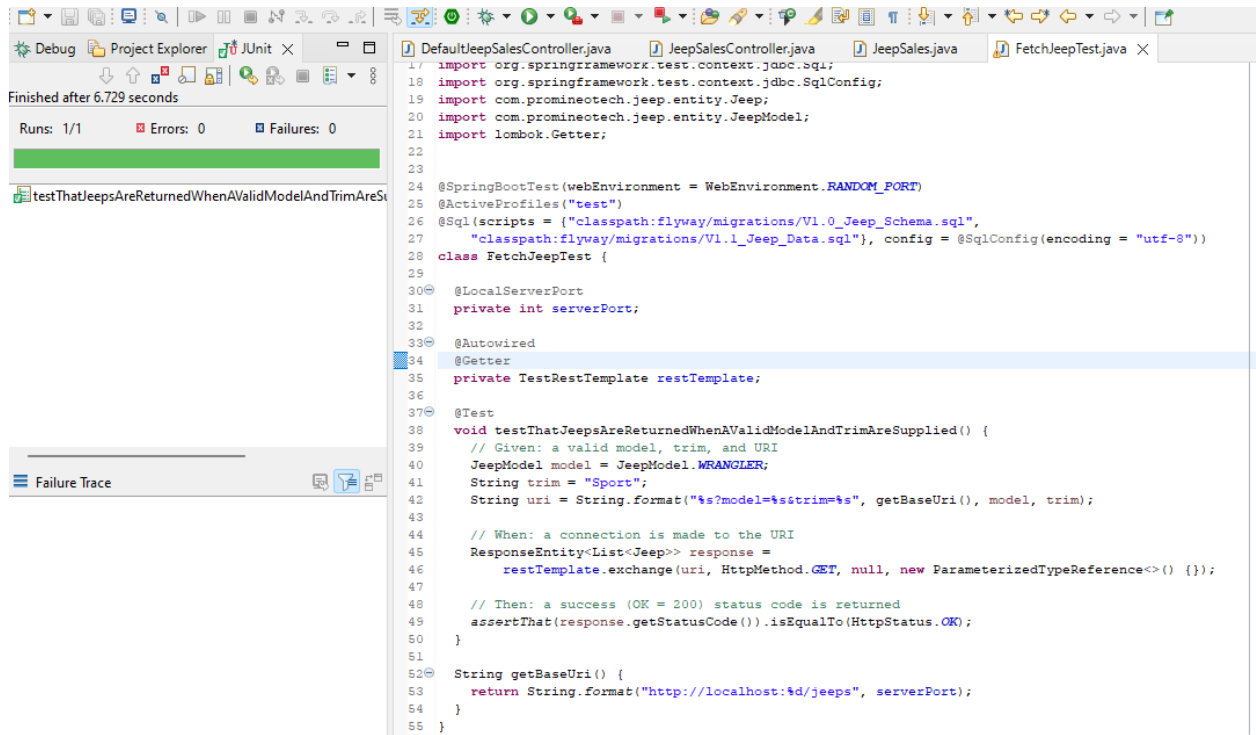
Curl
curl -X 'GET' \
'http://localhost:8080/jeeps?model=WRANGLER&trim=Sport' \
-H 'accept: application/json'

Request URL
http://localhost:8080/jeeps?model=WRANGLER&trim=Sport

Server response
Code Details
200
Response headers
connection: keep-alive
content-length: 0
date: Sat, 06 Aug 2022 17:14:15 GMT
keep-alive: timeout=60

Responses
```

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



- 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:

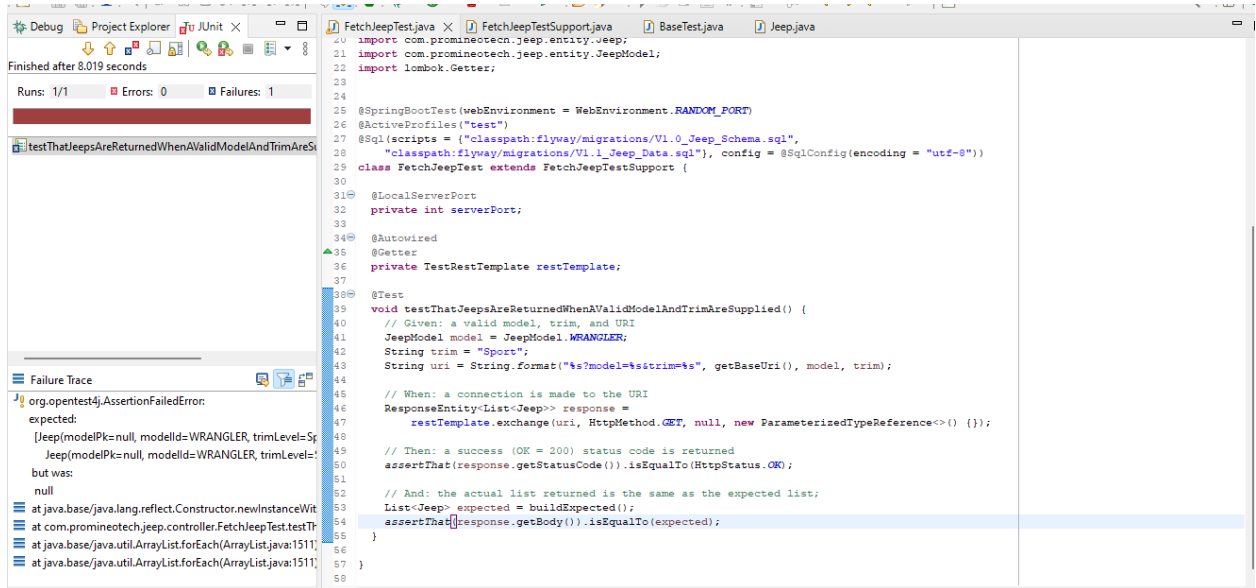
	Row 1	Row 2
Model ID	WRANGLER	WRANGLER
Trim Level	Sport	Sport
Num Doors	2	4
Wheel Size	17	17
Base Price	\$28,475.00	\$31,975.00

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. 🖨

a & b:

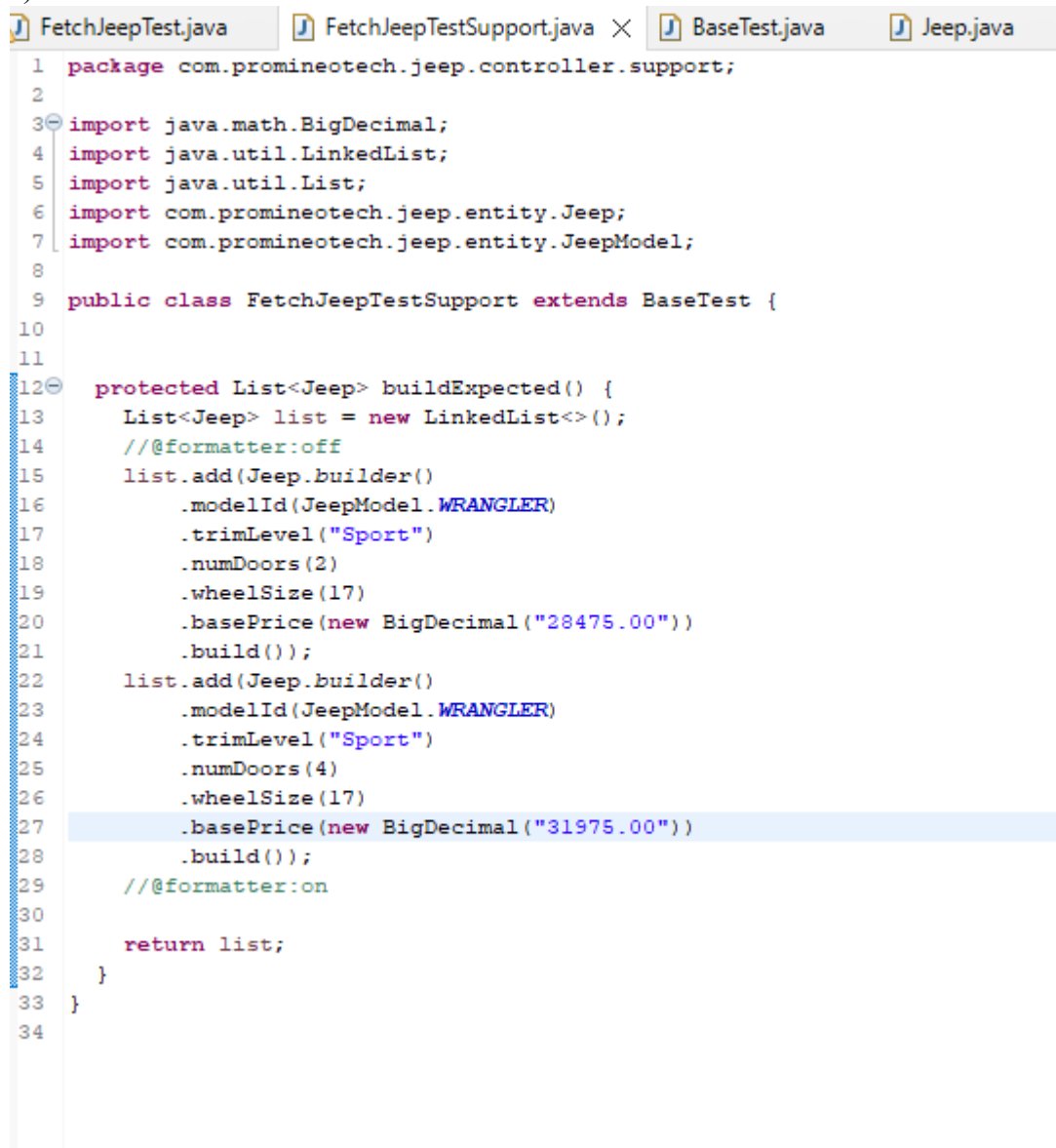


```
FetchJeepTest.java X FetchJeepTestSupport.java BaseTest.java Jeep.java
20 import com.promineotech.jee.entity.jee;
21 import com.promineotech.jee.entity.JeeModel;
22 import lombok.Getter;
23
24
25 @SpringBootTest(webEnvironment = WebEnvironment.RANDOM_PORT)
26 @ActiveProfiles("test")
27 @Sql(scripts = {"classpath:flyway/migrations/V1.0_Jee_Schema.sql",
28               "classpath:flyway/migrations/V1.1_Jee_Data.sql"}, config = @SqlConfig(encoding = "utf-8"))
29 class FetchJeepTest extends FetchJeepTestSupport {
30
31     @LocalServerPort
32     private int serverPort;
33
34     @Autowired
35     @Getter
36     private TestRestTemplate restTemplate;
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 = String.format("%s?model=%s&trim=%s", getBaseUri(), model, trim);
44
45         // When: a connection is made to the URI
46         ResponseEntity<List<Jee>> response =
47             restTemplate.exchange(uri, HttpMethod.GET, null, new ParameterizedTypeReference<>() {});
48
49         // Then: a success (OK = 200) status code is returned
50         assertEquals(response.getStatusCode(), HttpStatus.OK);
51
52         // And: the actual list returned is the same as the expected list:
53         List<Jee> expected = buildExpected();
54         assertEquals(response.getBody(), expected);
55     }
56
57 }
58
```

Failure Trace

```
org.opentest4j.AssertionFailedError:
expected:
[Jeep(modelPk=null, modelId=WRANGLER, trimLevel=Sport,
Jeep(modelPk=null, modelId=WRANGLER, trimLevel=Sport,
but was:
null
at java.base/java.lang.reflect.Constructor.newInstanceWithCaller(Constructor.java:490)
at com.promineotech.jee.controller.FetchJeepTestTest.fetchJeepsWhenValidModelAndTrimAreSupplied(FetchJeepTestTest.java:151)
at java.base/java.util.ArrayList.forEach(ArrayList.java:1511)
at java.base/java.util.ArrayList.forEach(ArrayList.java:1511)
```

c)




```
1 package com.promineotech.jeep.controller.support;
2
3 import java.math.BigDecimal;
4 import java.util.LinkedList;
5 import java.util.List;
6 import com.promineotech.jeep.entity.Jeep;
7 import com.promineotech.jeep.entity.JeepModel;
8
9 public class FetchJeepTestSupport extends BaseTest {
10
11
12 protected List<Jeep> buildExpected() {
13     List<Jeep> list = new LinkedList<>();
14     //@formatter:off
15     list.add(Jeep.builder()
16         .modelId(JeepModel.WRANGLER)
17         .trimLevel("Sport")
18         .numDoors(2)
19         .wheelSize(17)
20         .basePrice(new BigDecimal("28475.00"))
21         .build());
22     list.add(Jeep.builder()
23         .modelId(JeepModel.WRANGLER)
24         .trimLevel("Sport")
25         .numDoors(4)
26         .wheelSize(17)
27         .basePrice(new BigDecimal("31975.00"))
28         .build());
29     //@formatter:on
30
31     return list;
32 }
33 }
34
```

- 7) Add a service layer in your application as shown in the videos:
- Add a package named `com.promineotech.jeep.service`.
  - In the new package, create an interface named `JeepSalesService`.
  - In the same package (service), create a class named `DefaultJeepSalesService` that implements the `JeepSalesService` interface. Add the class-level annotation, `@Service`.
  - 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. 

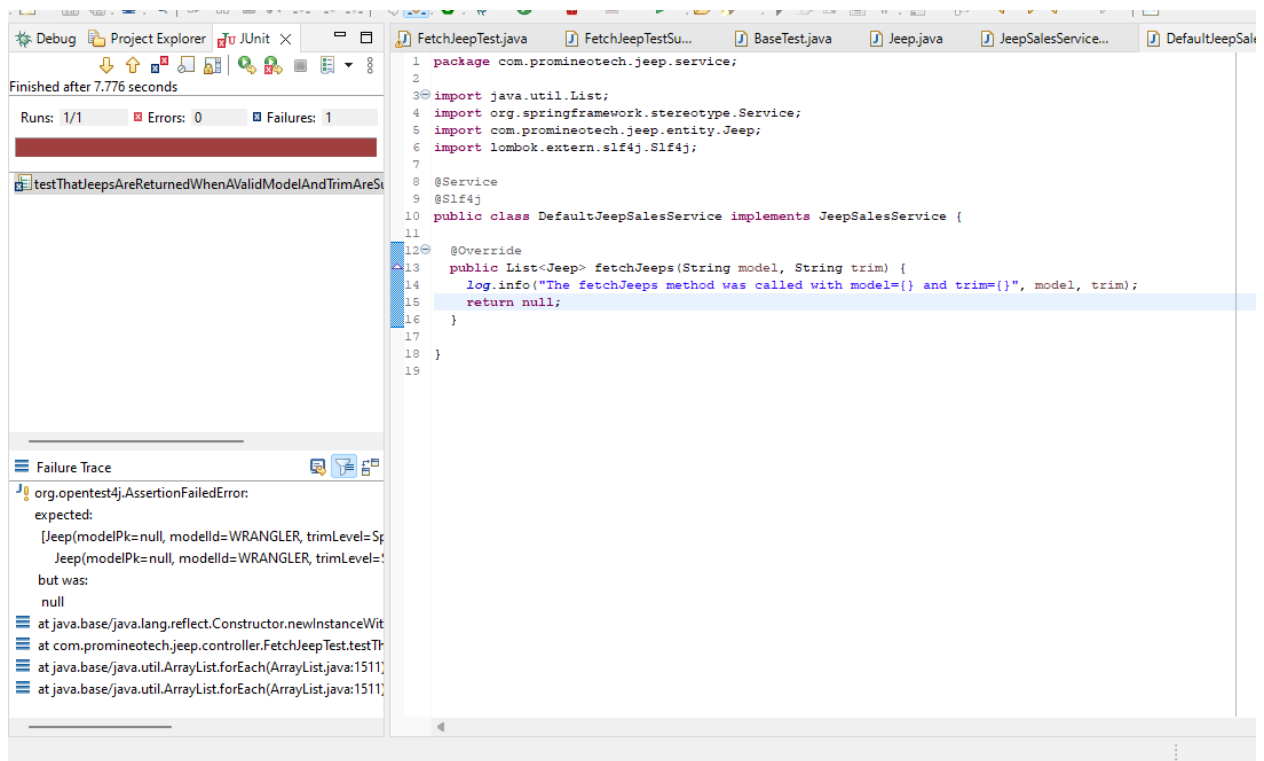
```

terminated- FetchJepTest [Unit] C:\Program Files\Java\jdk-17.0.3\bin\javaw.exe (Aug 6, 2022, 6:36:12 PM - 6:36:21 PM) [pid: 23904]
18:36:13.915 [main] DEBUG org.springframework.test.context.BootstrapUtils - Instantiating CacheAwareContextLoaderDelegate from class [org.springframework.test.context.cache.DefaultCacheAwareContextLoaderDelegate]
18:36:13.945 [main] DEBUG org.springframework.test.context.BootstrapUtils - Instantiating BootstrapContext using constructor [public org.springframework.test.context.support.DefaultBootstrapContext(org.springframework.test.context.CacheAwareContextLoaderDelegate)
18:36:14.066 [main] DEBUG org.springframework.test.context.BootstrapUtils - Instantiating SpringBootContextBootstrapper for test class [com.promineotech.jeeptest.FetchJepTest] from class [com.springframework.boot.test.context.SpringBootContextBootstrapper]
18:36:14.105 [main] INFO org.springframework.boot.test.context.SpringBootTestContextBootstrapper - Neither @ContextConfiguration nor @ContextHierarchy found for test class [com.promineotech.jeeptest.FetchJepTest], using SpringBootContextBootstrapper
18:36:14.120 [main] DEBUG org.springframework.test.context.support.AbstractContextLoader - Did not detect default resource location for test class [com.promineotech.jeeptest.FetchJepTest]: class path resource [com/promineotech/jeeptest/FetchJepTest-context.xml] did not contain resource
18:36:14.122 [main] DEBUG org.springframework.test.context.support.AbstractContextLoader - Did not detect default resource location for test class [com.promineotech.jeeptest.FetchJepTest]: class path resource [com/promineotech/jeeptest/FetchJepTest-context.xml] did not contain resource
18:36:14.122 [main] INFO org.springframework.test.context.support.AbstractContextLoader - Could not detect default resource locations for test class [com.promineotech.jeeptest.FetchJepTest]: no ResourceFoundException found for [ContextLoaderDelegate]
18:36:14.123 [main] INFO org.springframework.test.context.support.AnnotationConfigContextLoaderUtils - Could not detect default configuration classes for test class [com.promineotech.jeeptest.FetchJepTest]: no classes found in package [com.promineotech.jeeptest]
18:36:14.379 [main] DEBUG org.springframework.context.annotation.ClassPathScanningCandidateComponentProvider - Identified candidate component class: file [C:\Users\Jared\OneDrive\Desktop\Project\src\main\java\com\promineotech\jeeptest\FetchJepTest.class]
18:36:14.392 [main] INFO org.springframework.boot.test.context.SpringBootTestContextBootstrapper - Found @SpringBootConfiguration com.promineotech.jeeptest.JeeptestApplication for test class com.promineotech.jeeptest.FetchJepTest
18:36:14.584 [main] DEBUG org.springframework.boot.test.context.SpringBootTestContextBootstrapper - @TestExecutionListeners is not present for class [com.promineotech.jeeptest.FetchJepTest], using [org.springframework.test.context.support.DefaultTestExecutionListener]
18:36:14.576 [main] INFO org.springframework.boot.test.context.SpringBootTestContextBootstrapper - Loaded default TestExecutionListener class names from location [META-INF/spring.factories]
18:36:14.599 [main] DEBUG org.springframework.test.context.support.AbstractContextLoader - Skipping candidate TestExecutionListener [org.springframework.test.context.transaction.TransactionalTestExecutionListener]
18:36:14.600 [main] DEBUG org.springframework.boot.test.context.SpringBootTestContextBootstrapper - Skipping candidate TestExecutionListener [org.springframework.test.context.jdbc.SqlScriptTestExecutionListener]
18:36:14.601 [main] INFO org.springframework.boot.test.context.SpringBootTestContextBootstrapper - Using TestExecutionListeners: [org.springframework.test.context.web.ServletTestExecutionListener, org.springframework.test.context.support.DefaultTestExecutionListener]
18:36:14.612 [main] DEBUG org.springframework.test.context.support.AbstractDirtiesContextTestExecutionListener - Before test class: context [DefaultTestContext@4716eb3b testClass = FetchJepTest, testMethod = FetchJepTest, testClassSourceMethod = FetchJepTest]
18:36:14.637 [main] DEBUG org.springframework.test.context.support.DependencyInjectionTestExecutionListener - Performing dependency injection for test context [[DefaultTestContext@4716eb3b]]

[
  {
    "id": 1,
    "name": "Spring Boot",
    "version": "v2.7.2"
  }
]

2022-08-06 18:36:15.266 INFO 23904 --- [main] c.p.jeeptest.controller.FetchJepTest : Starting FetchJepTest using Java 17.0.3 on LAPTOP-1N0U2V3V with PID 23904 (started by .)
2022-08-06 18:36:15.267 DEBUG 23904 --- [main] c.p.jeeptest.controller.FetchJepTest : Running with Spring Boot v2.7.2, Spring v5.3.22
2022-08-06 18:36:15.269 INFO 23904 --- [main] c.p.jeeptest.controller.FetchJepTest : The following 1 profile is active: "test"
2022-08-06 18:36:20.118 INFO 23904 --- [main] c.p.jeeptest.controller.FetchJepTest : Started FetchJepTest in 4.532 seconds (JVM running for 7.04)
2022-08-06 18:36:21.319 INFO 23904 --- [o-auto-1-exec-1] c.p.j.c.DefaultJeeptestSalesController : Model=WRANGLER, trim=Sport
2022-08-06 18:36:21.323 INFO 23904 --- [o-auto-1-exec-1] c.p.j.s.service.DefaultJeeptestSalesService : The fetchJeeps method was called with model=WRANGLER and trim=Sport

```




- 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.

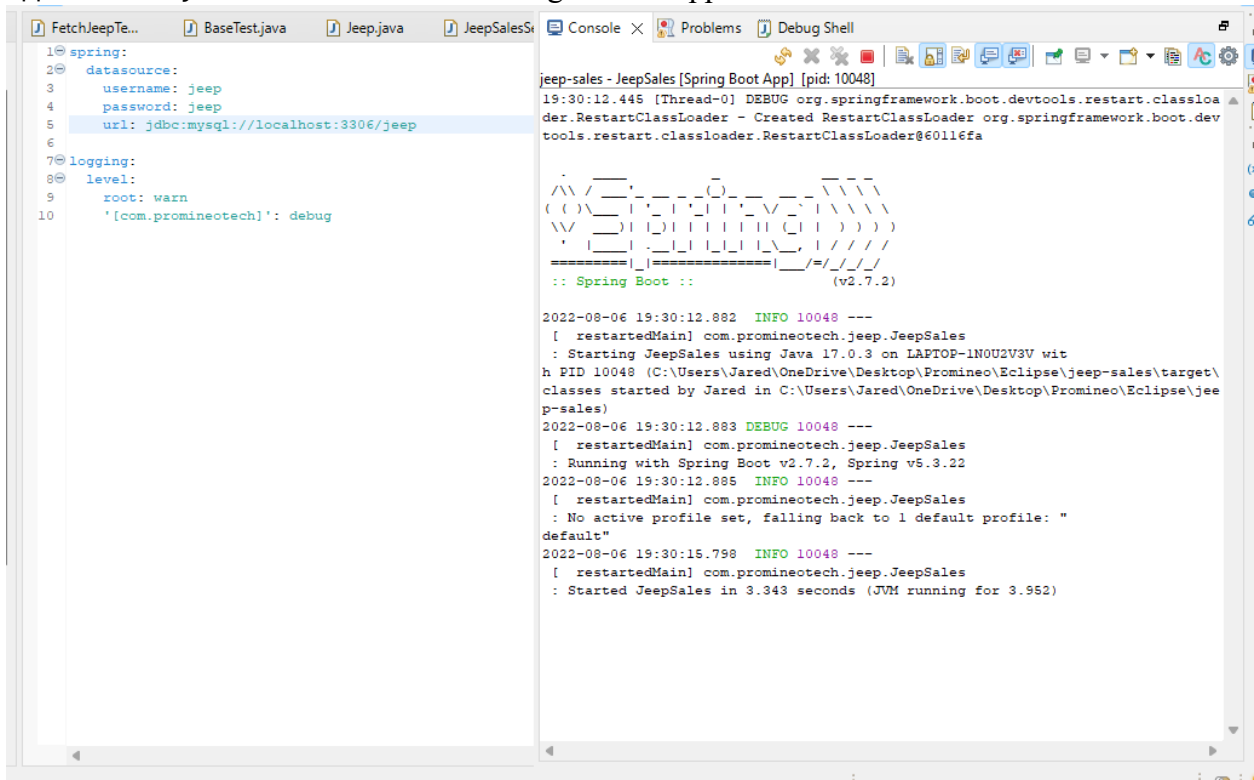
Be careful with the indentation! YAML allows hierarchical configuration but it reads the hierarchy based on the indentation level. The keyword "spring" MUST start in the first column. It should look similar to this when done:

```

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 panels. The left panel displays the `application.yaml` file with the following content:

```
1 spring:
2   datasource:
3     username: jeep
4     password: jeep
5     url: jdbc:mysql://localhost:3306/jeep
6
7 logging:
8   level:
9     root: warn
10    '[com.promineotech]': debug
```

The right panel shows the console output, which includes the Spring Boot logo and the following log messages:

```
jeep-sales - JeepSales [Spring Boot App] [pid: 10048]
19:30:12.445 [Thread-0] DEBUG org.springframework.boot.devtools.restart.classloader.RestartClassLoader - Created RestartClassLoader org.springframework.boot.devtools.restart.classloader.RestartClassLoader@60116fa

:: Spring Boot :: (v2.7.2)


2022-08-06 19:30:12.882 INFO 10048 ---
[ restartedMain] com.promineotech.jeepp.JeeppSales
: Starting JeepSales using Java 17.0.3 on LAPTOP-1N0U2V3V with
h PID 10048 (C:\Users\Jared\OneDrive\Desktop\Promineo\Eclipse\jeep-sales\target\
classes started by Jared in C:\Users\Jared\OneDrive\Desktop\Promineo\Eclipse\jee
p-sales)
2022-08-06 19:30:12.883 DEBUG 10048 ---
[ restartedMain] com.promineotech.jeepp.JeeppSales
: Running with Spring Boot v2.7.2, Spring v5.3.22
2022-08-06 19:30:12.885 INFO 10048 ---
[ restartedMain] com.promineotech.jeepp.JeeppSales
: No active profile set, falling back to 1 default profile: "
default"
2022-08-06 19:30:15.798 INFO 10048 ---
[ restartedMain] com.promineotech.jeepp.JeeppSales
: Started JeepSales in 3.343 seconds (JVM running for 3.952)
```

- 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;MODE=MYSQL
4
5 logging:
6   level:
7     root: warn
8     '[com.promineotech]': debug
```

## Screenshots of Code:

```
DefaultJeepSalesController.java ×
1 package com.promineotech.jeep.controller;
2
3 import java.util.List;
4
5
6 @RestController
7 @Slf4j
8 public class DefaultJeepSalesController implements JeepSalesController {
9
10     @Autowired
11     private JeepSalesService jeepSalesService;
12
13
14     @Override
15     public List<Jeep> fetchJeeps(String model, String trim) {
16         log.info("Model={}, trim={}", model, trim);
17         return jeepSalesService.fetchJeeps(model, trim);
18     }
19 }
20
21
22
23
24
```

DefaultJeepSalesController.java

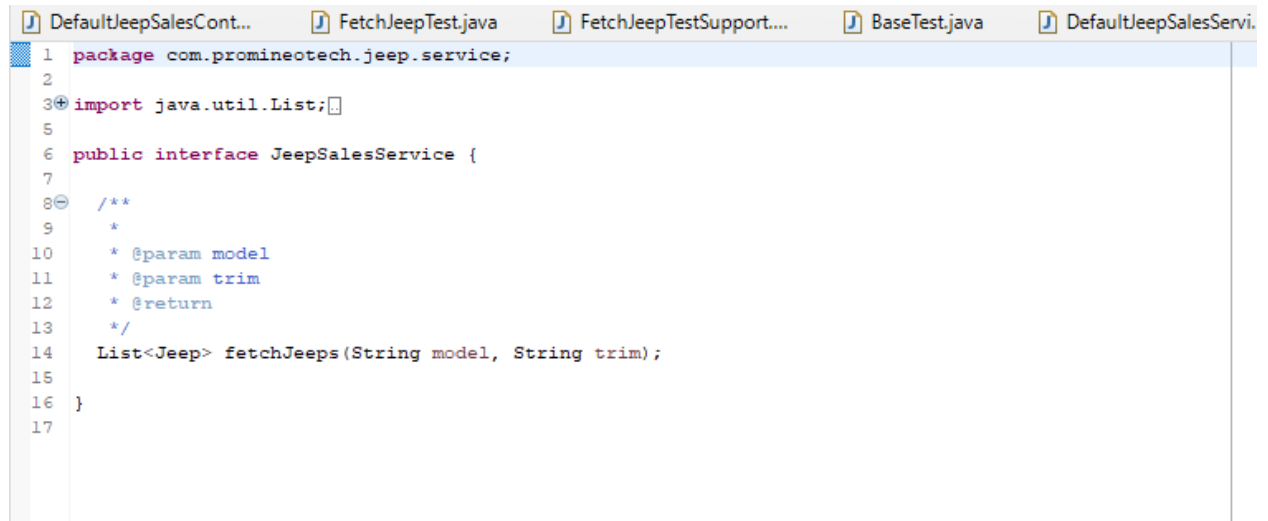
FetchJeepTest.java X

```
1 package com.promineotech.jeep.controller;
2
3 import static org.assertj.core.api.Assertions.assertThat;
4 import java.util.List;
5 import org.junit.jupiter.api.Test;
6 import org.springframework.beans.factory.annotation.Autowired;
7 import org.springframework.boot.test.context.SpringBootTest;
8 import org.springframework.boot.test.context.SpringBootTest.WebEnvironment;
9 import org.springframework.boot.test.web.client.TestRestTemplate;
10 import org.springframework.boot.test.web.server.LocalServerPort;
11 import org.springframework.core.ParameterizedTypeReference;
12 import org.springframework.http.HttpMethod;
13 import org.springframework.http.HttpStatus;
14 import org.springframework.http.ResponseEntity;
15 import org.springframework.test.context.ActiveProfiles;
16 import org.springframework.test.context.jdbc.Sql;
17 import org.springframework.test.context.jdbc.SqlConfig;
18 import com.promineotech.jeep.controller.support.FetchJeepTestSupport;
19 import com.promineotech.jeep.entity.Jeep;
20 import com.promineotech.jeep.entity.JeepModel;
21 import lombok.Getter;
22
23
24 @SpringBootTest(webEnvironment = WebEnvironment.RANDOM_PORT)
25 @ActiveProfiles("test")
26 @Sql(scripts = {"classpath:flyway/migrations/V1.0_Jeep_Schema.sql",
27               "classpath:flyway/migrations/V1.1_Jeep_Data.sql"}, config = @SqlConfig(encoding = "utf-8"))
28 class FetchJeepTest extends FetchJeepTestSupport {
29
30     @LocalServerPort
31     private int serverPort;
32
33     @Autowired
34     @Getter
35     private TestRestTemplate restTemplate;
36
37     @Test
38     void testThatJeepsAreReturnedWhenAValidModelAndTrimAreSupplied() {
39
40         // Given: a valid model, trim, and URI
41         JeepModel model = JeepModel.WRANGLER;
42         String trim = "Sport";
43         String uri = String.format("%s?model=%s&trim=%s", getBaseUri(), model, trim);
44
45         // When: a connection is made to the URI
46         ResponseEntity<List<Jeep>> response =
47             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         // And: the actual list returned is the same as the expected list;
53         List<Jeep> expected = buildExpected();
54         assertThat(response.getBody()).isEqualTo(expected);
55     }
56 }
57
```

```
DefaultJeepSalesController.java FetchJeepTest.java FetchJeepTestSupport.java X
1 package com.promineotech.jeeptest.support;
2
3+ import java.math.BigDecimal;
4
5
6
7
8
9 public class FetchJeepTestSupport extends BaseTest {
10
11
12- protected List<Jeep> buildExpected() {
13     List<Jeep> list = new LinkedList<>();
14     //@formatter:off
15     list.add(Jeep.builder()
16         .modelId(JeepModel.WRANGLER)
17         .trimLevel("Sport")
18         .numDoors(2)
19         .wheelSize(17)
20         .basePrice(new BigDecimal("28475.00"))
21         .build());
22     list.add(Jeep.builder()
23         .modelId(JeepModel.WRANGLER)
24         .trimLevel("Sport")
25         .numDoors(4)
26         .wheelSize(17)
27         .basePrice(new BigDecimal("31975.00"))
28         .build());
29     //@formatter:on
30
31     return list;
32 }
33 }
34
```

```
DefaultJeepSalesController.java FetchJeepTest.java FetchJeepTestSupport.java BaseTest.java X
1 package com.promineotech.jeeptest.support;
2
3+ import org.springframework.beans.factory.annotation.Autowired;
4
5
6
7
8 public class BaseTest {
9- @LocalServerPort
10 private int serverPort;
11
12- @Autowired
13 @Getter
14 private TestRestTemplate restTemplate;
15
16- protected String getBaseUrl() {
17     return String.format("http://localhost:%d/jeeps", serverPort);
18 }
19
20 }
21
```

```
DefaultJeepSalesController.java FetchJeepTest.java FetchJeepTestSupport.java BaseTest.java DefaultJeepSalesService.java X
1 package com.promineotech.jeeptest.service;
2
3+ import java.util.List;
4
5
6
7
8 @Service
9 @Slf4j
10 public class DefaultJeepSalesService implements JeepSalesService {
11
12- @Override
13 public List<Jeep> fetchJeeps(String model, String trim) {
14     log.info("The fetchJeeps method was called with model={} and trim={}", model, trim);
15     return null;
16 }
17
18 }
19
```



The screenshot shows an IDE with five tabs: DefaultJeepSalesCont..., FetchJeepTest.java, FetchJeepTestSupport..., BaseTest.java, and DefaultJeepSalesServi.... The active tab is DefaultJeepSalesCont..., which contains the following Java code:

```
1 package com.promineotech.jeepp.service;
2
3 import java.util.List;
4
5
6 public interface JeepSalesService {
7
8     /**
9      *
10     * @param model
11     * @param trim
12     * @return
13     */
14     List<Jeep> fetchJeeps(String model, String trim);
15
16 }
17
```

**Screenshots of Running Application:**

```
19:39:43.851 [Thread-0] DEBUG org.springframework.boot.devtools.restart.classloader.RestartClassLoader - Created RestartClassLoader org.springframework.boot.devtools.restart.classloader.RestartClassLoader@2b1eb68f
```

```
  _ _ _ _ _  
 / \ / \ _ _ _ _ _ ( _ ) _ _ _ _ _ \ \ \ \ \  
( ( ) \ _ _ _ _ _ | ' | ' | ' | ' \ / _ _ _ _ _ \ \ \ \ \  
 \ \ / \ _ _ _ _ _ | | | | | | | | | | ( | | ) ) ) )  
  ' | _ _ _ _ _ | _ _ _ _ _ | _ _ _ _ _ | / / / / /  
=====|_|=====|_|_/=/_/_/_/_  
:: Spring Boot ::                (v2.7.2)
```

```
2022-08-06 19:39:44.182 INFO 22000 ---
```

```
[ restartedMain] com.promineotech.jeepp.JeeppSales  
: Starting JeeppSales using Java 17.0.3 on LAPTOP-1N0U2V3V with  
h PID 22000 (C:\Users\Jared\OneDrive\Desktop\Promineo\Eclipse\jeepp-sales\target\classes started by Jared in C:\Users\Jared\OneDrive\Desktop\Promineo\Eclipse\jeepp-sales)
```

```
2022-08-06 19:39:44.182 DEBUG 22000 ---
```

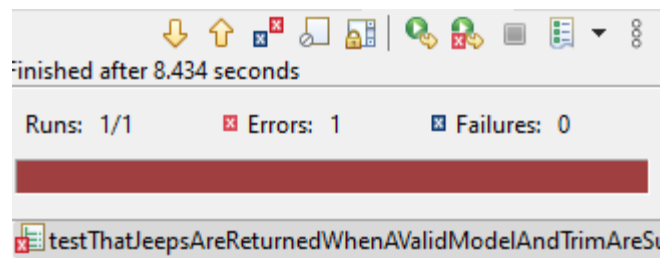
```
[ restartedMain] com.promineotech.jeepp.JeeppSales  
: Running with Spring Boot v2.7.2, Spring v5.3.22
```

```
2022-08-06 19:39:44.183 INFO 22000 ---
```

```
[ restartedMain] com.promineotech.jeepp.JeeppSales  
: No active profile set, falling back to 1 default profile: "default"
```

```
2022-08-06 19:39:46.704 INFO 22000 ---
```

```
[ restartedMain] com.promineotech.jeepp.JeeppSales  
: Started JeeppSales in 2.843 seconds (JVM running for 3.429)
```



**URL to GitHub Repository:**

<https://github.com/JaredBears/JeepSales>