


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). 


- 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. 
- 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. 
- 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. 
- 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. 
- 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:

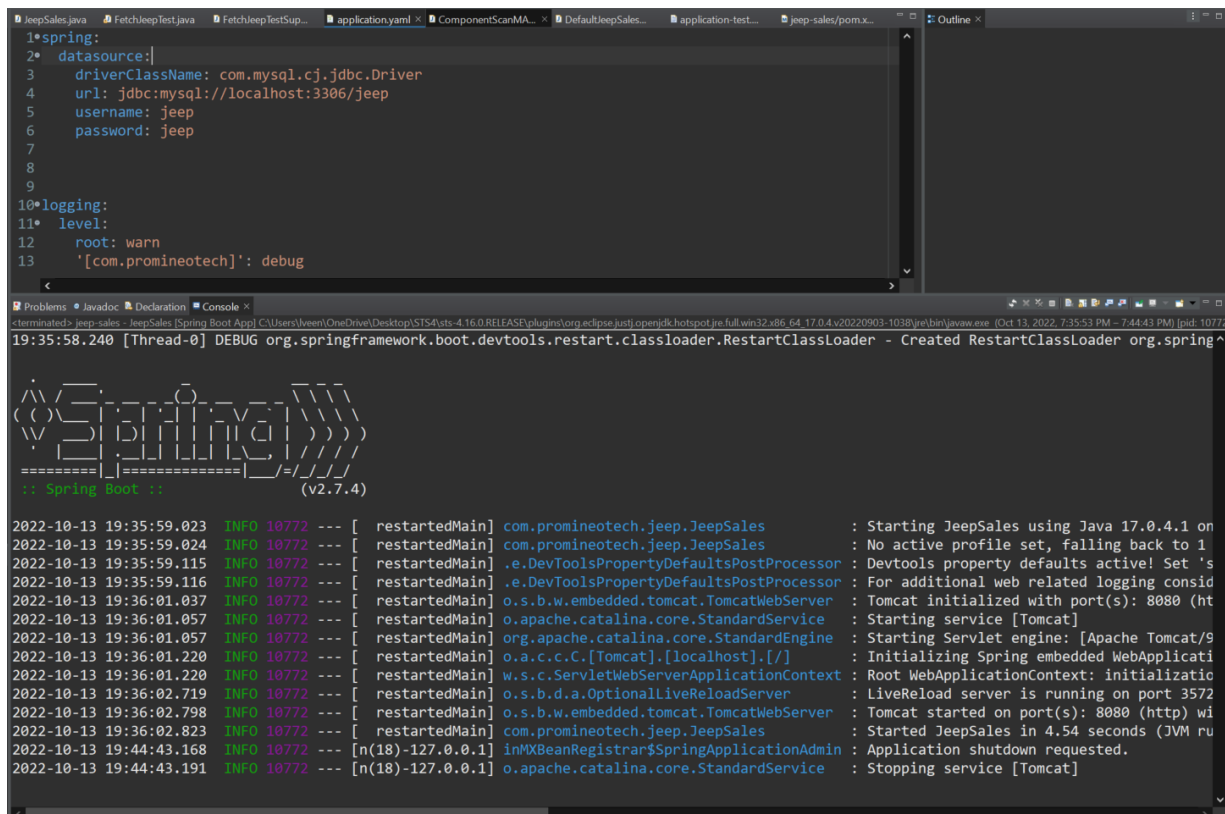
```
spring:
  datasource:
    url: jdbc:h2:mem:jeep
```

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. 

Screenshots of Code:

10)



The screenshot shows an IDE with two main panels. The top panel displays the content of application-test.yaml, and the bottom panel shows a console log.

application-test.yaml:

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

Console Log:

```
19:35:58.240 [Thread-0] DEBUG org.springframework.boot.devtools.restart.classloader.RestartClassLoader - Created RestartClassLoader org.spring...

:: Spring Boot :: (v2.7.4)

2022-10-13 19:35:59.023 INFO 10772 --- [ restartedMain] com.promineotech.jee... : Starting JeepSales using Java 17.0.4.1 on
2022-10-13 19:35:59.024 INFO 10772 --- [ restartedMain] com.promineotech.jee... : No active profile set, falling back to 1
2022-10-13 19:35:59.115 INFO 10772 --- [ restartedMain] .e.DevToolsPropertyDefaultsPostProcessor : Devtools property defaults active! Set 's
2022-10-13 19:35:59.116 INFO 10772 --- [ restartedMain] .e.DevToolsPropertyDefaultsPostProcessor : For additional web related logging consid
2022-10-13 19:36:01.037 INFO 10772 --- [ restartedMain] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat initialized with port(s): 8080 (ht
2022-10-13 19:36:01.057 INFO 10772 --- [ restartedMain] o.apache.catalina.core.StandardService : Starting service [Tomcat]
2022-10-13 19:36:01.057 INFO 10772 --- [ restartedMain] org.apache.catalina.core.StandardEngine : Starting Servlet engine: [Apache Tomcat/9
2022-10-13 19:36:01.220 INFO 10772 --- [ restartedMain] o.a.c.c.C.[Tomcat].[localhost].[/] : Initializing Spring embedded WebApplicati
2022-10-13 19:36:01.220 INFO 10772 --- [ restartedMain] w.s.c.ServletWebServerApplicationContext : Root WebApplicationContext: initializatio
2022-10-13 19:36:02.719 INFO 10772 --- [ restartedMain] o.s.b.d.a.OptionalLiveReloadServer : LiveReload server is running on port 3572
2022-10-13 19:36:02.798 INFO 10772 --- [ restartedMain] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat started on port(s): 8080 (http) wi
2022-10-13 19:36:02.823 INFO 10772 --- [ restartedMain] com.promineotech.jee... : Started JeepSales in 4.54 seconds (JVM ru
2022-10-13 19:44:43.168 INFO 10772 --- [n(18)-127.0.0.1] inMXBeanRegistrar$SpringApplicationAdmin : Application shutdown requested.
2022-10-13 19:44:43.191 INFO 10772 --- [n(18)-127.0.0.1] o.apache.catalina.core.StandardService : Stopping service [Tomcat]
```

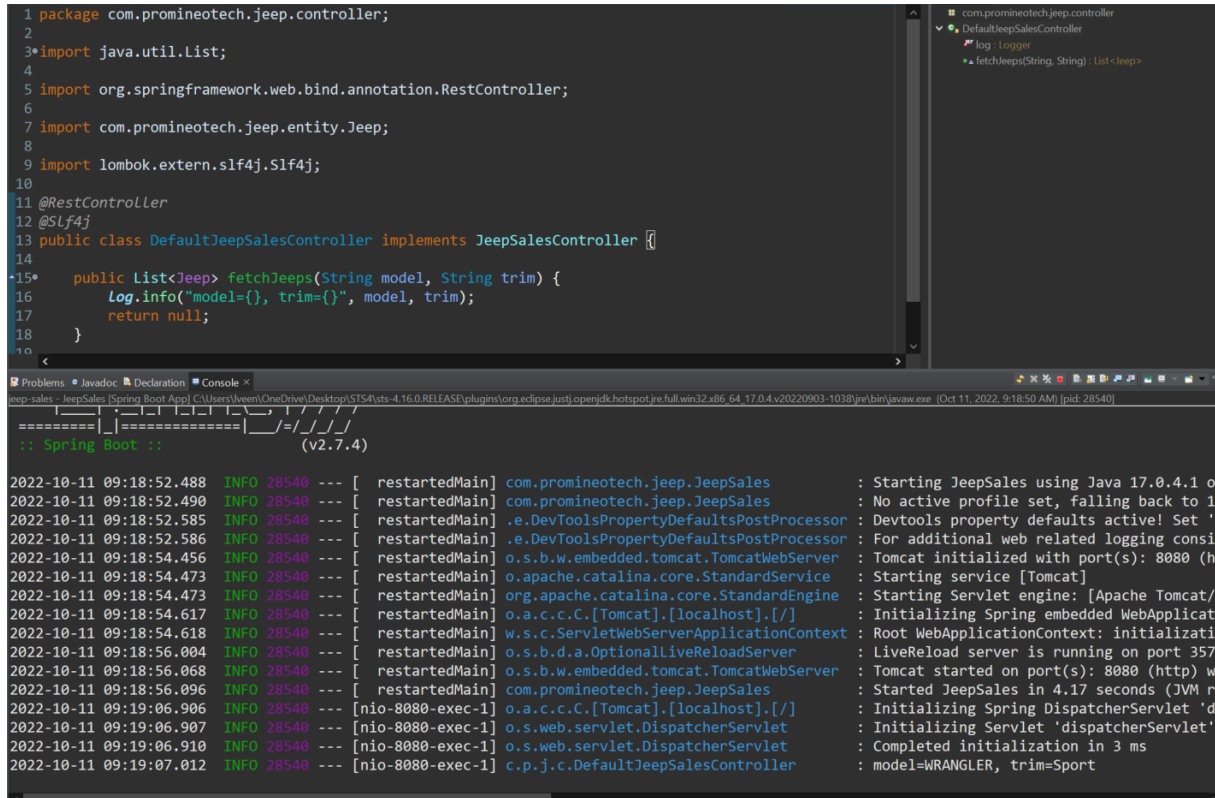
12)

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

Screenshots of Running Application:

2)

<http://localhost:8080/jeeps?model=WRANGLER&trim=Sport>



```
1 package com.promineotech.jeep.controller;
2
3 import java.util.List;
4
5 import org.springframework.web.bind.annotation.RestController;
6
7 import com.promineotech.jeep.entity.Jeep;
8
9 import lombok.extern.slf4j.Slf4j;
10
11 @RestController
12 @Slf4j
13 public class DefaultJeepSalesController implements JeepSalesController {
14
15     public List<Jeep> fetchJeeps(String model, String trim) {
16         log.info("model={}, trim={}", model, trim);
17         return null;
18     }
19 }
```

com.promineotech.jeep.controller
DefaultJeepSalesController
log : Logger
fetchJeeps(String, String) : List<Jeep>

jeep-sales - JeepSales [Spring Boot App] C:\Users\veen\OneDrive\Desktop\STS4\sts-4.16.0.RELEASE\plugins\org.eclipse.justi.openjdk.hotspot.jre.full.win32.x86_64-17.0.4.v20220903-1038\jre\bin\java.exe (Oct 11, 2022, 9:18:50 AM) [pid: 28540]

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

```
2022-10-11 09:18:52.488 INFO 28540 --- [ restartedMain] com.promineotech.jeep.JeepSales : Starting JeepSales using Java 17.0.4.1 o
2022-10-11 09:18:52.490 INFO 28540 --- [ restartedMain] com.promineotech.jeep.JeepSales : No active profile set, falling back to 1
2022-10-11 09:18:52.585 INFO 28540 --- [ restartedMain] .e.DevToolsPropertyDefaultsPostProcessor : Devtools property defaults active! Set '
2022-10-11 09:18:52.586 INFO 28540 --- [ restartedMain] .e.DevToolsPropertyDefaultsPostProcessor : For additional web related logging consi
2022-10-11 09:18:54.456 INFO 28540 --- [ restartedMain] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat initialized with port(s): 8080 (h
2022-10-11 09:18:54.473 INFO 28540 --- [ restartedMain] o.apache.catalina.core.StandardService : Starting service [Tomcat]
2022-10-11 09:18:54.473 INFO 28540 --- [ restartedMain] org.apache.catalina.core.StandardEngine : Starting Servlet engine: [Apache Tomcat/
2022-10-11 09:18:54.617 INFO 28540 --- [ restartedMain] o.a.c.c.C.[Tomcat].[localhost].[/] : Initializing Spring embedded WebApplicat
2022-10-11 09:18:54.618 INFO 28540 --- [ restartedMain] w.s.c.ServletWebServerApplicationContext : Root WebApplicationContext: initializati
2022-10-11 09:18:56.004 INFO 28540 --- [ restartedMain] o.s.b.d.a.OptionalLiveReloadServer : LiveReload server is running on port 357
2022-10-11 09:18:56.068 INFO 28540 --- [ restartedMain] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat started on port(s): 8080 (http) w
2022-10-11 09:18:56.096 INFO 28540 --- [ restartedMain] com.promineotech.jeep.JeepSales : Started JeepSales in 4.17 seconds (JVM r
2022-10-11 09:19:06.906 INFO 28540 --- [nio-8080-exec-1] o.a.c.c.C.[Tomcat].[localhost].[/] : Initializing Spring DispatcherServlet 'd
2022-10-11 09:19:06.907 INFO 28540 --- [nio-8080-exec-1] o.s.web.servlet.DispatcherServlet : Initializing Servlet 'dispatcherServlet'
2022-10-11 09:19:06.910 INFO 28540 --- [nio-8080-exec-1] o.s.web.servlet.DispatcherServlet : Completed initialization in 3 ms
2022-10-11 09:19:07.012 INFO 28540 --- [nio-8080-exec-1] c.p.j.c.DefaultJeepSalesController : model=WRANGLER, trim=Sport
```

3)

Parameters

Cancel

Name	Description
model string <small>(query)</small>	The model name (i.e. 'WRANGLER')
trim string <small>(query)</small>	The trim name (i.e. 'SPORT')

Execute

Clear

Responses

Curl

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

Request URL

```
http://localhost:8080/jeeps?model=CHEROKEE&trim=Sport
```

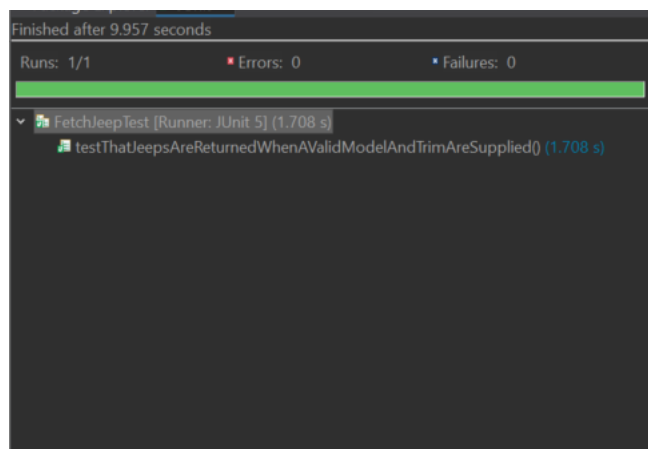
Server response

Code	Details
200	<div>Response headers</div> <div><pre>connection: keep-alive content-length: 0 date: Tue, 11 Oct 2022 19:39:32 GMT keep-alive: timeout=60</pre></div>

Responses

4)

```
2022-10-11 14:38:13.863 INFO 6360 --- [ restartedMain] com.promineotech.jeep.JeepSales : Starting JeepSales using Java 17.0.4.1 on
2022-10-11 14:38:13.877 INFO 6360 --- [ restartedMain] com.promineotech.jeep.JeepSales : No active profile set, falling back to 1 d
2022-10-11 14:38:14.003 INFO 6360 --- [ restartedMain] .e.DevToolsPropertyDefaultsPostProcessor : Devtools property defaults active! Set 'sp
2022-10-11 14:38:14.003 INFO 6360 --- [ restartedMain] .e.DevToolsPropertyDefaultsPostProcessor : For additional web related logging consid
2022-10-11 14:38:17.161 INFO 6360 --- [ restartedMain] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat initialized with port(s): 8080 (ht
2022-10-11 14:38:17.184 INFO 6360 --- [ restartedMain] o.apache.catalina.core.StandardService : Starting service [Tomcat]
2022-10-11 14:38:17.184 INFO 6360 --- [ restartedMain] org.apache.catalina.core.StandardEngine : Starting Servlet engine: [Apache Tomcat/9.
2022-10-11 14:38:17.386 INFO 6360 --- [ restartedMain] o.a.c.c.C.[Tomcat].[localhost].[/] : Initializing Spring embedded WebApplicatio
2022-10-11 14:38:17.386 INFO 6360 --- [ restartedMain] w.s.c.ServletWebServerApplicationContext : Root WebApplicationContext: initializatio
2022-10-11 14:38:19.222 INFO 6360 --- [ restartedMain] o.s.b.d.a.OptionalLiveReloadServer : LiveReload server is running on port 35729
2022-10-11 14:38:19.311 INFO 6360 --- [ restartedMain] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat started on port(s): 8080 (http) wi
2022-10-11 14:38:19.347 INFO 6360 --- [ restartedMain] com.promineotech.jeep.JeepSales : Started JeepSales in 6.243 seconds (JVM ru
2022-10-11 14:38:27.146 INFO 6360 --- [nio-8080-exec-1] o.a.c.c.C.[Tomcat].[localhost].[/] : Initializing Spring DispatcherServlet 'dis
2022-10-11 14:38:27.147 INFO 6360 --- [nio-8080-exec-1] o.s.web.servlet.DispatcherServlet : Initializing Servlet 'dispatcherServlet'
2022-10-11 14:38:27.149 INFO 6360 --- [nio-8080-exec-1] o.s.web.servlet.DispatcherServlet : Completed initialization in 2 ms
2022-10-11 14:38:52.720 INFO 6360 --- [nio-8080-exec-2] o.springdoc.api.AbstractOpenApiResource : Init duration for springdoc-openapi is: 4
2022-10-11 14:39:32.363 INFO 6360 --- [nio-8080-exec-9] c.p.j.c.DefaultJeepSalesController : model=CHEROKEE, trim=Sport
```



6-(A-C)


```

30
31 class FetchJeepTest extends FetchJeepTestSupport {
32
33     @Test
34     void testThatJeepsAreReturnedWhenAValidModelAndTrimAreSupplied() {
35         // Given: a valid model, trim and URE
36
37         JeepModel model = JeepModel.WRANGLER;
38         String trim = "Sport";
39         String uri =
40             String.format("%s?model=%s&trim=%s", getBaseUri(), model, trim);
41
42         // When: a connection is made to the URI
43
44         ResponseEntity<List<Jeep>> response =
45             getRestTemplate().exchange(uri, HttpMethod.GET, null,
46                 new ParameterizedTypeReference<List<Jeep>>() {});
47
48         // Then: a success (OK - 200) status code is returned
49         assertThat(response.getStatusCode()).isEqualTo(HttpStatus.OK);
50
51         // And: the actual list returned is the same as the expected list
52
53         List<Jeep> expected = buildExpected();
54         assertThat(response.getBody()).isEqualTo(expected);
55
56     }
57
58
59 }
60 }
61

```

Package Explorer: JUnit x
 Finished after 11.212 seconds
 Runs: 1/1 Errors: 0 Failures: 1

FetchJeepTest [Runner: JUnit 5] (1.857 s)
 testThatJeepsAreReturnedWhenAValidModelAndTrimAreSupplied() (1.857 s)

Failure Trace

```

org.opentest4j.AssertionFailedError:
expected:
  [Jeep(modelId=WRANGLER, trimLevel=null, numDoors=2, wheelSize=17, basePrice=28
    Jeep(modelId=WRANGLER, trimLevel=null, numDoors=4, wheelSize=17, basePrice=3
  but was:
  null
  at java.base/java.lang.reflect.Constructor.newInstanceWithCaller(Constructor.java:499)
  at com.promineotech.jeeptest.FetchJeepTest.testThatJeepsAreReturnedWhenAValid
  at java.base/java.util.ArrayList.forEach(ArrayList.java:1511)
  at java.base/java.util.ArrayList.forEach(ArrayList.java:1511)
  
```



```

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

```

7-G

The screenshot shows an IDE with three main panels:

- Package Explorer:** Shows a test case `FetchJeepTest` with a failure trace. The failure is an `org.springframework.Assert$AssertionError` where the expected `JeepModelId=WRANGLER` does not match the actual `JeepModelId=WRANGLER` in the test data.
- Code Editor:** Displays the `FetchJeepTestSupport` class and the `DefaultJeepSalesService` class. The `fetchJeeps` method in `DefaultJeepSalesService` is highlighted, showing it returns a list of `Jeep` objects.
- Console:** Shows the output of the test run. It indicates that the test failed due to an `AssertionError` where the expected `JeepModelId=WRANGLER` does not match the actual `JeepModelId=WRANGLER` in the test data.

URL to GitHub Repository:

<https://github.com/LVeenendaal/SpringBoot2>