

Web API Design with Spring Boot Week 14 Coding Assignment


Points possible: 75

URL to GitHub Repository: <https://github.com/Anghel-Valdehueza/Jeep-Sales.git>


URL to Public Link of your Video: <https://youtu.be/r7vmoFWs-1M>

Instructions :

1. Follow the **Coding Steps** below to complete this assignment.

- In Spring Tool Suite (STS), or an IDE of your choice, write the code that accomplishes the objectives listed below. Ensure that the code compiles and runs as directed.
- Use your existing repo or create a new repository on GitHub for this week's assignment and push your completed code to the repo, including your entire Maven Project Directory (e.g., jeep-sales) and any additional files (e.g. .sql files) that you create. In addition, screenshot your ERD and push the screenshot to your GitHub repo.
- Include the screenshots into this Assignment Document indicated by: 
- Create a video showcasing your work:
 - In this video: record and present your project verbally while showing the results of the working project.
 - Easy way to Create a video: Start a meeting in Zoom, share your screen, open Eclipse with the code and your Console window, start recording & record yourself describing and running the program showing the results.
 - Your video should be a maximum of 5 minutes.
 - Upload your video with a public link.
 - Easy way to Create a Public Video Link: Upload your video recording to YouTube with a public link.


2. In addition, please include the following in your Coding Assignment Document:

- The requested screenshots, indicated by: 
- The URL for this week's GitHub repository.
- The URL of the public link of your video.

3. Save the Coding Assignment Document as a .pdf and do the following:


- Push the .pdf to the GitHub repo for this week.
 - Upload the .pdf to the LMS in your Coding Assignment Submission.
-

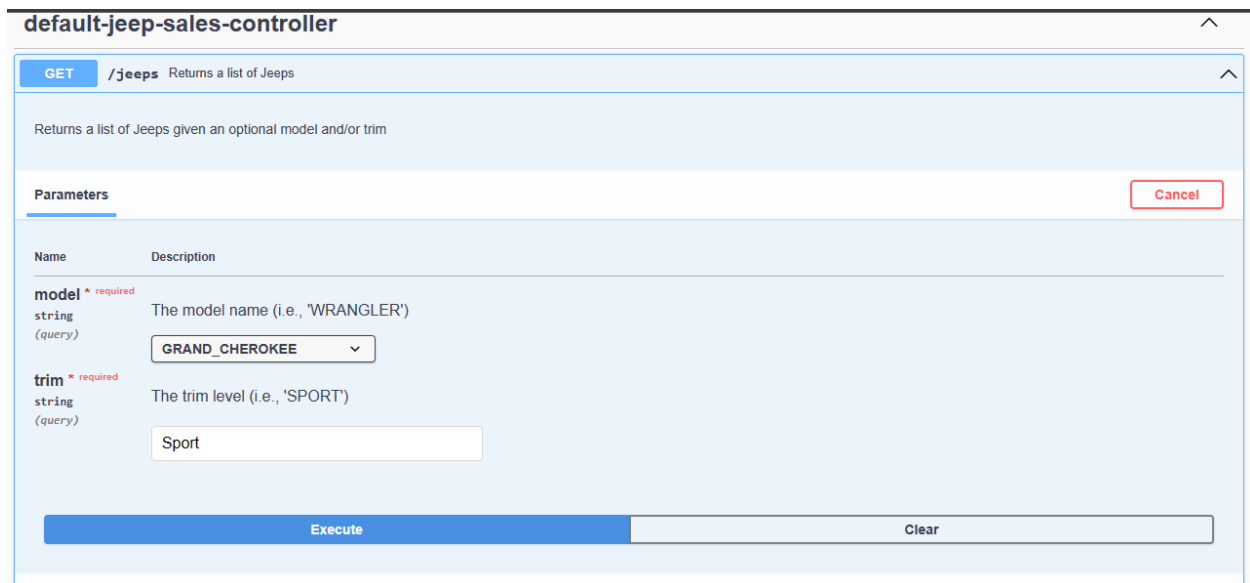
Web API Design with Spring Boot Week 14 Coding Assignment

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). 



default-jeep-sales-controller

GET /jeeps Returns a list of Jeeps

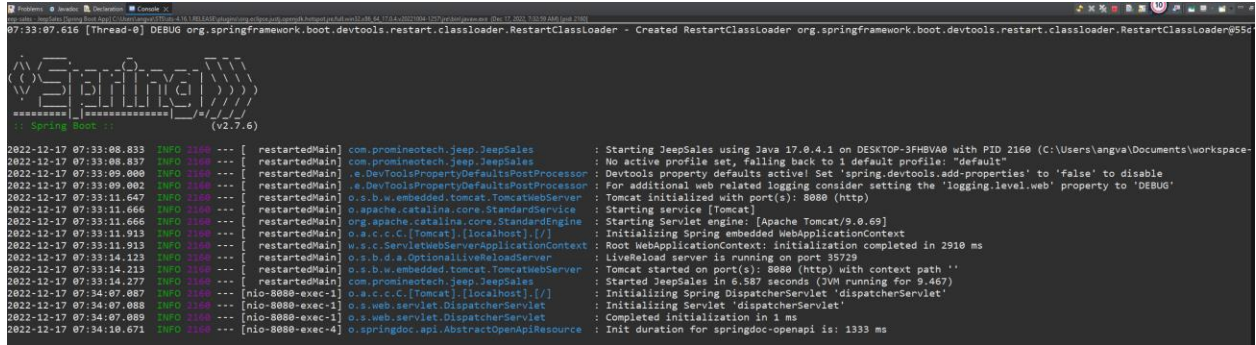
Returns a list of Jeeps given an optional model and/or trim

Parameters

| Name | Description |
|---------------------------------------|-----------------------------------|
| model * required string (query) | The model name (i.e., 'WRANGLER') |
| trim * required string (query) | The trim level (i.e., 'SPORT') |

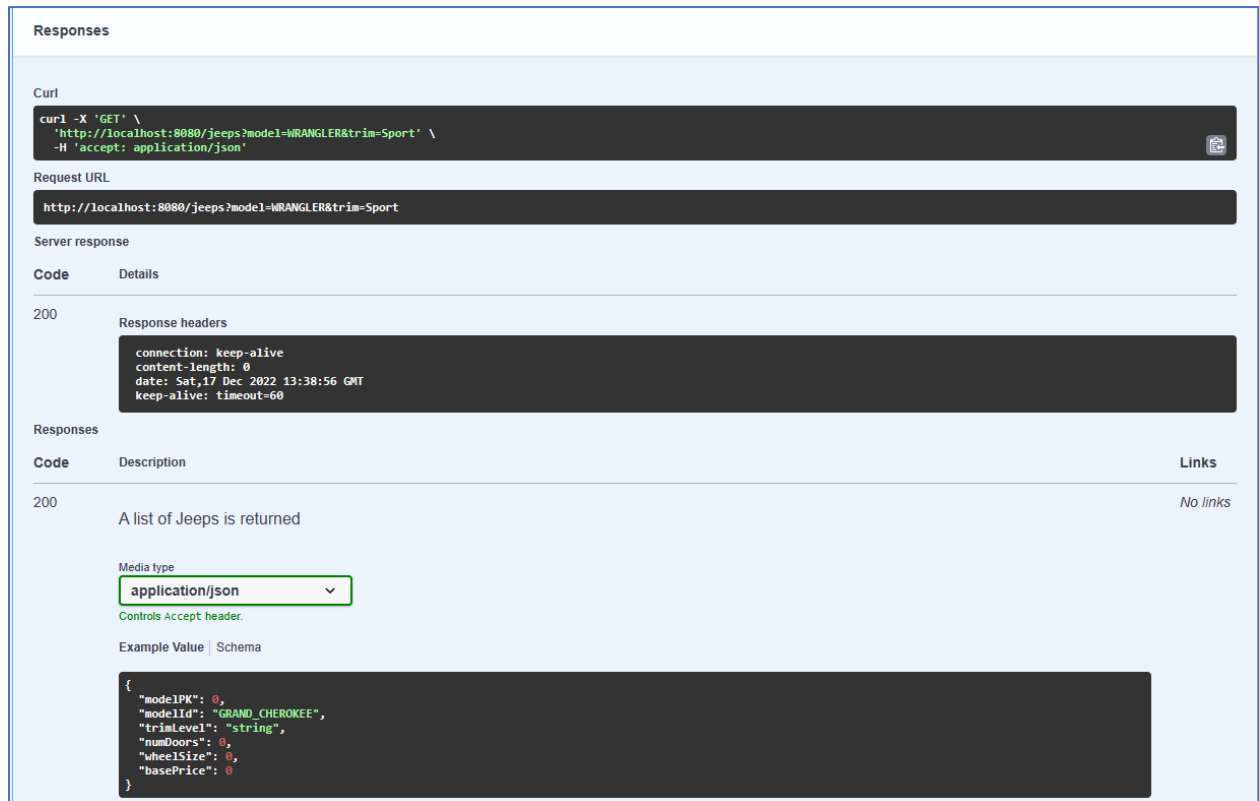
Execute Clear

Web API Design with Spring Boot Week 14 Coding Assignment



```
07:33:07.616 [Thread-0] DEBUG org.springframework.boot.devtools.restart.classloader.RestartClassLoader - Created RestartClassLoader org.springframework.boot.devtools.restart.classloader.RestartClassLoader@55d
[Spring Boot] (v2.7.6)
2022-12-17 07:33:08.833 INFO 1160 --- [ restartedMain] com.promineotech.jeepp.JeeppSales : Starting JeeppSales using Java 17.0.4.1 on DESKTOP-3FHBVA0 with PID 2160 (C:\Users\langva\Documents\workspace-
2022-12-17 07:33:09.000 INFO 1160 --- [ restartedMain] e.DevToolsPropertyDefaultsPostProcessor : DevTools property defaults active! Set 'spring.devtools.add-properties' to 'false' to disable
2022-12-17 07:33:09.002 INFO 1160 --- [ restartedMain] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat initialized with port(s): 8080 (http)
2022-12-17 07:33:11.666 INFO 1160 --- [ restartedMain] org.apache.catalina.core.StandardEngine : Starting service [Tomcat]
2022-12-17 07:33:11.913 INFO 1160 --- [ restartedMain] o.a.c.c.C.[Tomcat].[localhost].[/] : Initializing Spring embedded WebApplicationContext
2022-12-17 07:33:11.913 INFO 1160 --- [ restartedMain] w.s.c.ServletWebServerApplicationContext : Root WebApplicationContext: initialization completed in 2910 ms
2022-12-17 07:33:14.113 INFO 1160 --- [ restartedMain] o.s.b.d.a.OptionalLiveReloadServer : LiveReload server is running on port 35729
2022-12-17 07:33:14.213 INFO 1160 --- [ restartedMain] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat started on port(s): 8080 (http) with context path ''
2022-12-17 07:33:14.277 INFO 1160 --- [ restartedMain] com.promineotech.jeepp.JeeppSales : Started JeeppSales in 6.587 seconds (JVM running for 9.467)
2022-12-17 07:34:07.087 INFO 1160 --- [nio-8080-exec-1] o.a.c.c.C.[Tomcat].[localhost].[/] : Initializing Spring DispatcherServlet 'dispatcherServlet'
2022-12-17 07:34:07.088 INFO 1160 --- [nio-8080-exec-1] o.s.web.servlet.DispatcherServlet : Initializing Servlet 'dispatcherServlet'
2022-12-17 07:34:07.089 INFO 1160 --- [nio-8080-exec-1] o.s.web.servlet.DispatcherServlet : Completed initialization in 1 ms
2022-12-17 07:34:10.671 INFO 1160 --- [nio-8080-exec-4] o.springdoc.api.AbstractOpenApiResource : Init duration for springdoc-openapi is: 1333 ms
```

- 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, 17 Dec 2022 13:38:56 GMT
Keep-alive: timeout=60
```

Responses

| Code | Description | Links |
|------|-----------------------------|----------|
| 200 | A list of Jeeps is returned | No links |

Media type

application/json

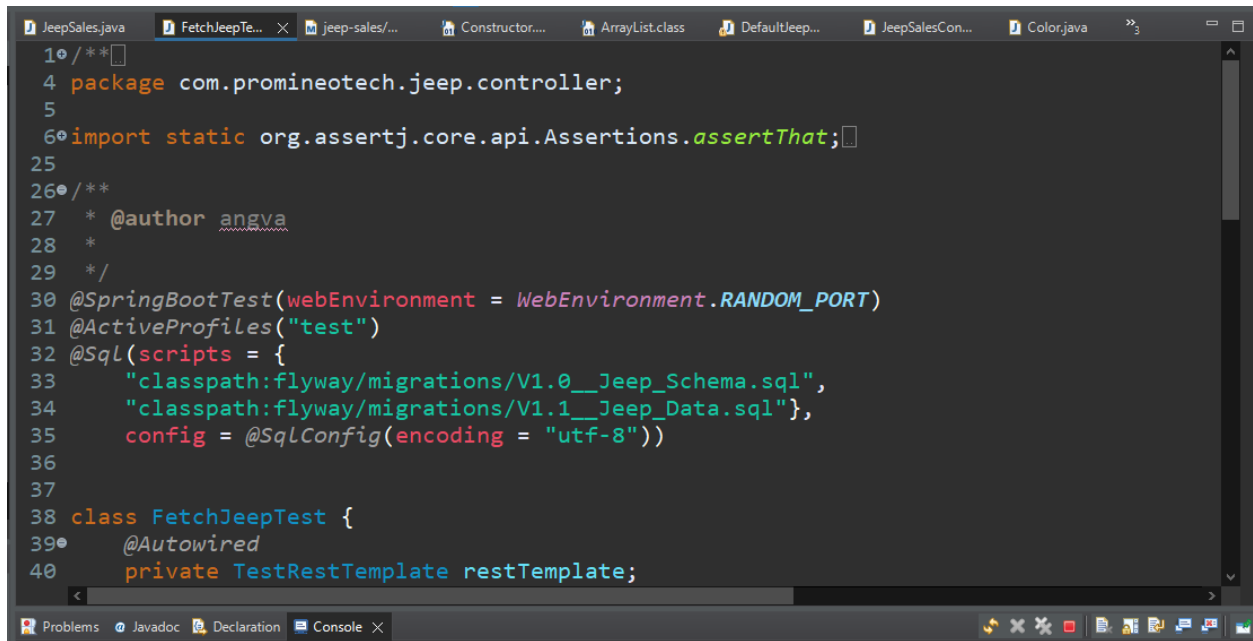
Controls Accept header.

Example Value | Schema

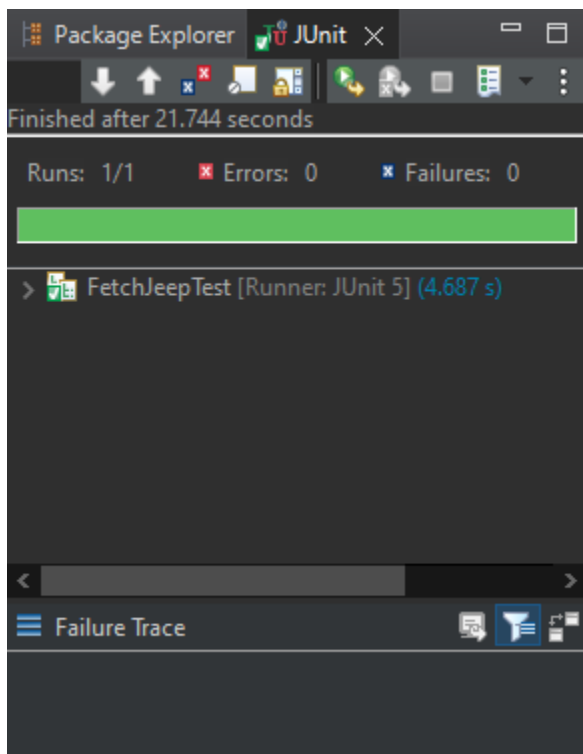
```
{
  "modelId": 0,
  "modelId": "GRAND_CHEROKEE",
  "trimLevel": "string",
  "numDoors": 0,
  "wheelSize": 0,
  "basePrice": 0
}
```

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

Web API Design with Spring Boot Week 14 Coding Assignment



```
1 // **
4 package com.promineotech.jeepp.controller;
5
6 import static org.assertj.core.api.Assertions.assertThat;
25
26 /**
27  * @author angva
28  *
29  */
30 @SpringBootTest(webEnvironment = WebEnvironment.RANDOM_PORT)
31 @ActiveProfiles("test")
32 @Sql(scripts = {
33     "classpath:flyway/migrations/V1.0__Jeep_Schema.sql",
34     "classpath:flyway/migrations/V1.1__Jeep_Data.sql"},
35     config = @SqlConfig(encoding = "utf-8"))
36
37
38 class FetchJeepTest {
39     @Autowired
40     private TestRestTemplate restTemplate;
```




- 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

Web API Design with Spring Boot Week 14 Coding Assignment

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 |
| | | |

- 6) 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.
- 7) 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. 

```
18 public class DefaultJeepSalesService implements JeepSalesService {
19     @Override
20     public List<Jeep> fetchJeeps(JeepModel model, String trim) {
21         Log.info("Received a Request in Jeep Sales service with model {}, trim {}", model, trim);
22         return null;
23     }
24 }
25 }
26 }
```

- 8) 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`.

Web API Design with Spring Boot Week 14 Coding Assignment

- 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.

```
10 /**
4 package com.promineotech.jeep.service;
5
6 import java.util.List;
7 import org.springframework.stereotype.Service;
8 import com.promineotech.jeep.entity.Jeep;
9 import com.promineotech.jeep.entity.JeepModel;
10 import lombok.extern.slf4j.Slf4j;
11
12 /**
13  * @author angva
14  *
15  */
16 @Service
17 @Slf4j
18 public class DefaultJeepSalesService implements JeepSalesService {
19     @Override
20     public List<Jeep> fetchJeeps(JeepModel model, String trim) {
21         log.info("Received a Request in Jeep Sales service with model {}, trim {}", model, trim);
22         return null;
23     }
24 }
25
26
```

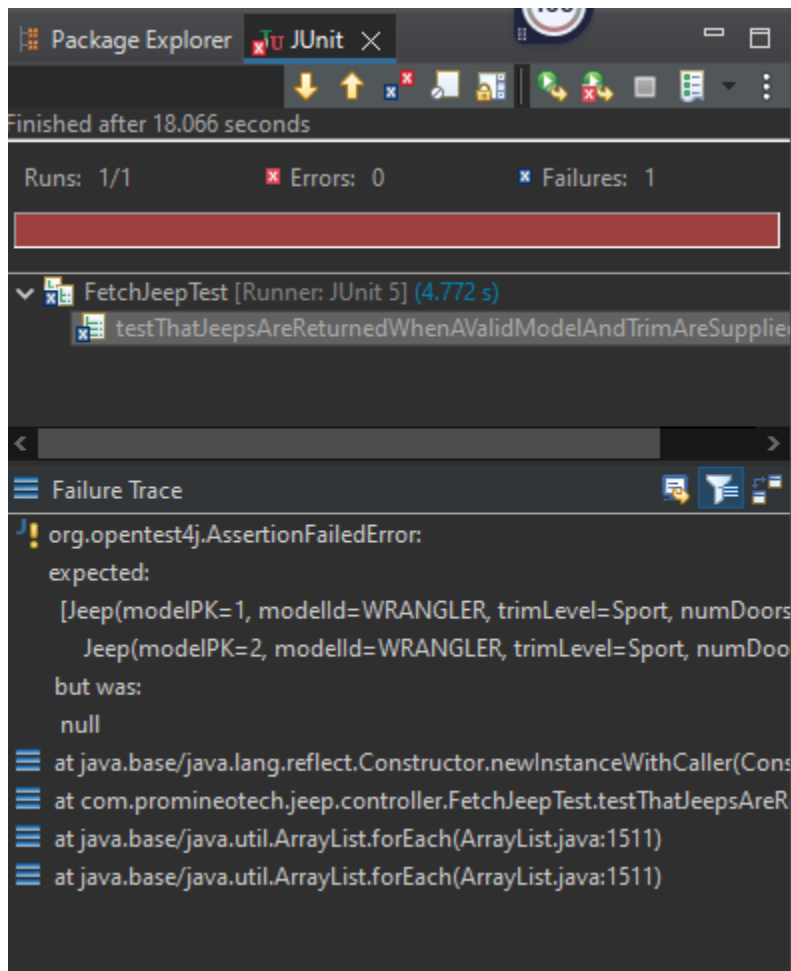
```
06:37:07.414 [main] DEBUG org.springframework.test.context.BootstrapUtils - Instantiating CacheAwareContextLoaderDelegate from class [org.springframework.test.context.cache.DefaultCacheAwareContextLoaderDeleg
06:37:07.460 [main] DEBUG org.springframework.test.context.BootstrapUtils - Instantiating BootstrapContext using constructor [public org.springframework.test.context.support.DefaultBootstrapContext(java.lang.
06:37:07.621 [main] DEBUG org.springframework.test.context.BootstrapUtils - Instantiating TestContextBootstrapper for test class [com.promineotech.jeep.controller.FetchJeepTest] from class [org.springframework
06:37:07.659 [main] INFO org.springframework.boot.test.context.SpringBootTestContextBootstrapper - Neither @ContextConfiguration nor @ContextHierarchy found for test class [com.promineotech.jeep.controller.Fe
06:37:07.675 [main] DEBUG org.springframework.test.context.support.AbstractContextLoader - Did not detect default resource location for test class [com.promineotech.jeep.controller.FetchJeepTest]: class path
06:37:07.679 [main] DEBUG org.springframework.test.context.support.AbstractContextLoader - Did not detect default resource location for test class [com.promineotech.jeep.controller.FetchJeepTest]: class path
06:37:07.680 [main] INFO org.springframework.test.context.support.AbstractContextLoader - Could not detect default resource locations for test class [com.promineotech.jeep.controller.FetchJeepTest]: no resour
06:37:07.685 [main] INFO org.springframework.test.context.support.AnnotationConfigContextLoaderUtils - Could not detect default configuration classes for test class [com.promineotech.jeep.controller.FetchJeep
06:37:08.387 [main] DEBUG org.springframework.test.context.support.AnnotationConfigContextLoaderUtils - Identified candidate component class: file [C:\Users\angva\Documents\workspace-spring-tool-suite-5\
06:37:08.394 [main] INFO org.springframework.boot.test.context.SpringBootTestContextBootstrapper - Found @SpringBootConfiguration com.promineotech.jeep.JeepSales for test class com.promineotech.jeep.controller
06:37:08.903 [main] DEBUG org.springframework.boot.test.context.SpringBootTestContextBootstrapper - @TestExecutionListeners is not present for class [com.promineotech.jeep.controller.FetchJeepTest]: using def
06:37:08.909 [main] INFO org.springframework.boot.test.context.SpringBootTestContextBootstrapper - Loaded default TestExecutionListener class names from location [META-INF/spring-factories]: [org.springframework
06:37:08.980 [main] INFO org.springframework.boot.test.context.SpringBootTestContextBootstrapper - Using TestExecutionListeners: [org.springframework.test.context.web.ServletTestExecutionListener@67d4508d, or
06:37:09.007 [main] DEBUG org.springframework.test.context.support.AbstractJUnit4TestExecutionListener - Before test class: context [[DefaultTestContext@6731787b testClass = FetchJeepTest, testInstance
06:37:09.064 [main] DEBUG org.springframework.test.context.support.DependencyInjectionTestExecutionListener - Performing dependency injection for test context [[DefaultTestContext@6731787b testClass = FetchJe

Spring
=====
:: Spring Boot ::
(v2.7.6)

2022-12-29 06:37:11.028 INFO 6086 --- [main] c.p.jeep.controller.FetchJeepTest : Starting FetchJeepTest using Java 17.0.4.1 on DESKTOP-3FHBVA0 with PID 6086 (started by angva in C:\Users\an
2022-12-29 06:37:11.039 INFO 6086 --- [main] c.p.jeep.controller.FetchJeepTest : The following 1 profile is active: "test"
2022-12-29 06:37:15.956 INFO 6086 --- [main] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat initialized with port(s): 80 (http)
2022-12-29 06:37:15.990 INFO 6086 --- [main] o.apache.catalina.core.StandardService : Starting service [Tomcat]
2022-12-29 06:37:15.992 INFO 6086 --- [main] org.apache.catalina.core.StandardEngine : Starting Servlet engine: [Apache Tomcat/9.0.69]
2022-12-29 06:37:16.398 INFO 6086 --- [main] o.a.c.c.C.[Tomcat].[localhost].[/] : Initializing Spring embedded WebApplicationContext
2022-12-29 06:37:16.398 INFO 6086 --- [main] w.s.c.ServletWebServerApplicationContext : Root WebApplicationContext: initialization completed in 5200 ms
2022-12-29 06:37:24.647 INFO 6086 --- [main] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat started on port(s): 80624 (http) with context path ''
2022-12-29 06:37:24.751 INFO 6086 --- [main] c.p.jeep.controller.FetchJeepTest : Started FetchJeepTest in 15.519 seconds (JVM running for 22.654)
2022-12-29 06:37:25.190 INFO 6086 --- [main] com.zaxxer.hikari.HikariDataSource : HikariPool-1 - Starting...
2022-12-29 06:37:26.852 INFO 6086 --- [main] com.zaxxer.hikari.HikariDataSource : HikariPool-1 - Start completed.
```

Web API Design with Spring Boot Week 14 Coding Assignment

```
2022-12-29 06:37:26.852 INFO 6036 --- [main] com.zaxxer.hikari.HikariDataSource : HikariPool-1 - Start completed.
2022-12-29 06:37:29.786 INFO 6036 --- [o-auto-1-exec-1] o.a.c.c.C.[Tomcat].[localhost].[/] : Initializing Spring DispatcherServlet 'dispatcherServlet'
2022-12-29 06:37:29.786 INFO 6036 --- [o-auto-1-exec-1] o.s.web.servlet.DispatcherServlet : Initializing Servlet 'dispatcherServlet'
2022-12-29 06:37:29.723 INFO 6036 --- [o-auto-1-exec-1] o.s.web.servlet.DispatcherServlet : Completed initialization in 17 ms
2022-12-29 06:37:30.427 INFO 6036 --- [ionShutdownHook] com.zaxxer.hikari.HikariDataSource : HikariPool-1 - Shutdown initiated...
2022-12-29 06:37:30.443 INFO 6036 --- [ionShutdownHook] com.zaxxer.hikari.HikariDataSource : HikariPool-1 - Shutdown completed.
```



- 9) 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.
- 10) 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.

Web API Design with Spring Boot Week 14 Coding Assignment

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:

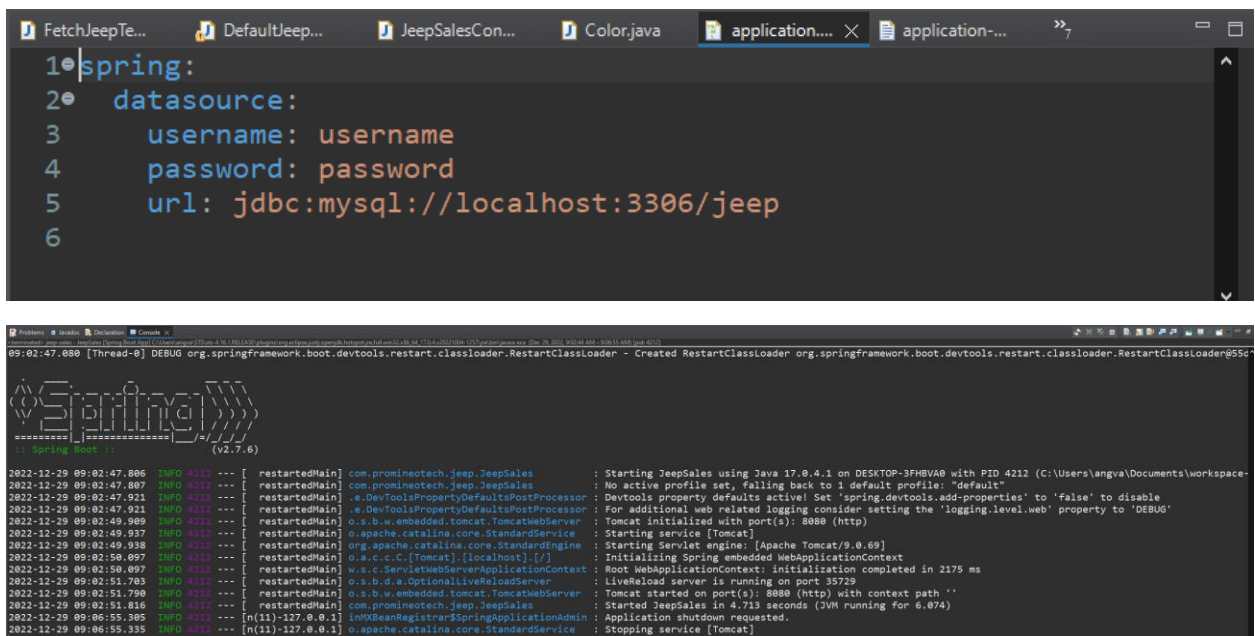
datasource:

```
username: username
```

```
password: password
```

```
url: jdbc:mysql://localhost:3306/jeep
```

- 11) 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. 



- 12) 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".
- 13) 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:

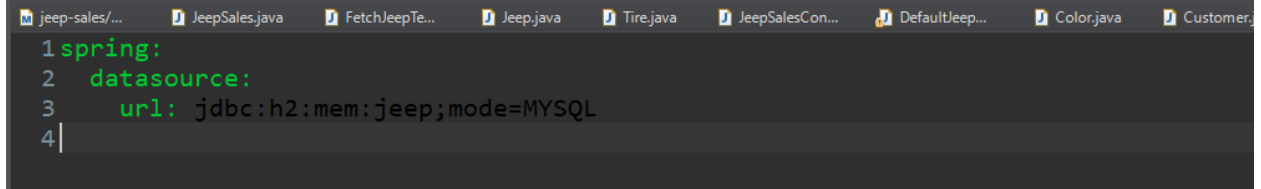
datasource:

```
url: jdbc:h2:mem:jeep;mode=MYSQL
```

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

Web API Design with Spring Boot Week 14 Coding Assignment

Produce a screenshot showing application-test.yaml. 🖥️

A screenshot of an IDE window showing a file named application-test.yaml. The file contains the following YAML configuration:

```
1 spring:
2   datasource:
3     url: jdbc:h2:mem:jeep;mode=MYSQL
4 |
```

The IDE's tab bar at the top shows several open files: jeep-sales/..., JeepSales.java, FetchJeepTe..., Jeep.java, Tire.java, JeepSalesCon..., DefaultJeep..., Color.java, and Customer....