


# 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.

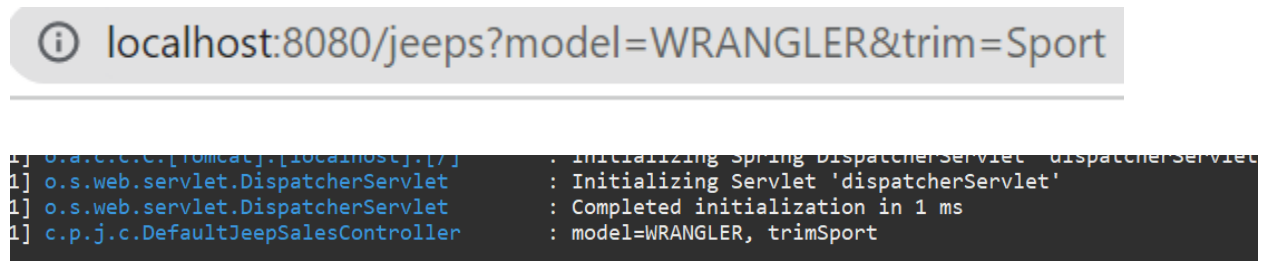
## Project Resources:

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

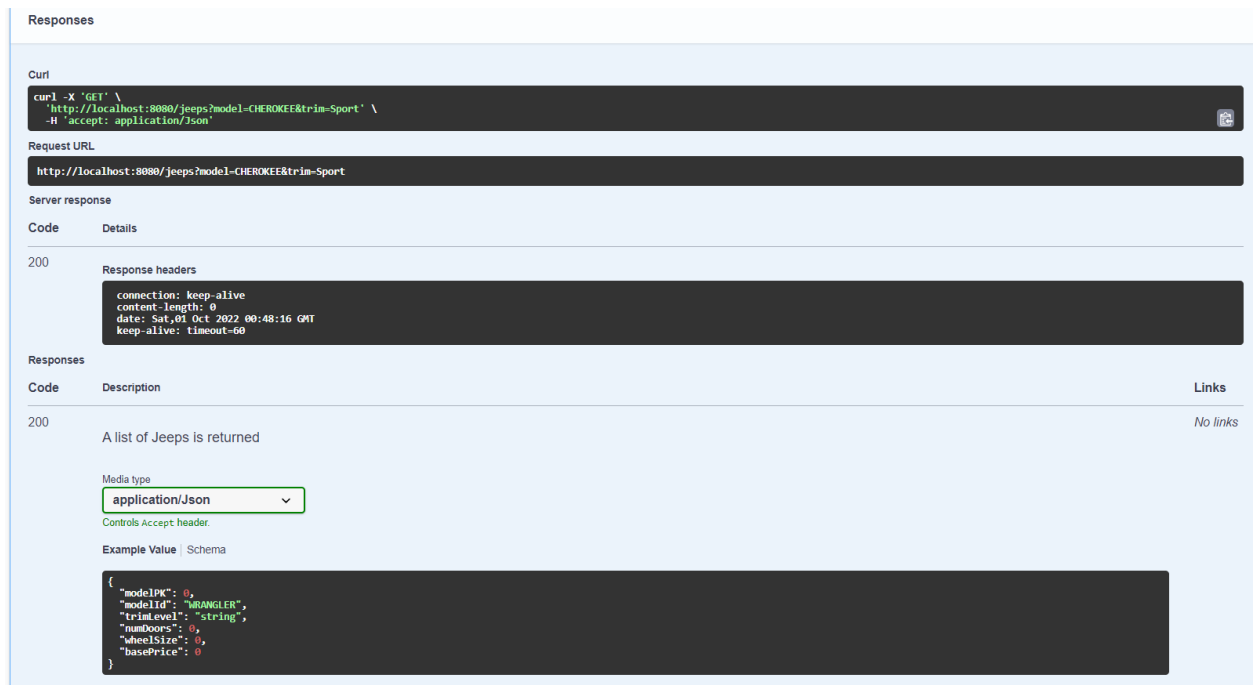
## Coding Steps:

- 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. 🖥️

```

1 package com.promineotech.jeep.controller;
2
30 import static org.assertj.core.api.Assertions.assertThat;
31
32 @SpringBootTest(webEnvironment = WebEnvironment.RANDOM_PORT)
33 @ActiveProfiles("test")
34 @Sql(scripts = {
35     "classpath:flyway/migrations/V1.0__Jeep_Schema.sql",
36     "classpath:flyway/migrations/V1.1__Jeep_Data.sql"}, |
37     config = @SqlConfig(encoding = "utf-8"))
38 class FetchJeepTest {
39
40     @Autowired
41     @Getter
42     private TestRestTemplate restTemplate;
43
44     @LocalServerPort
45     private int serverPort;
46
47     // ...
48 }

```

```


2022-09-30 20:53:02.737 INFO 12148 --- [main] c.p.jeep.controller.FetchJeepTest : Startin
2022-09-30 20:53:02.740 INFO 12148 --- [main] c.p.jeep.controller.FetchJeepTest : The fo
2022-09-30 20:53:04.741 INFO 12148 --- [main] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat
2022-09-30 20:53:04.759 INFO 12148 --- [main] o.apache.catalina.core.StandardService : Startin
2022-09-30 20:53:04.759 INFO 12148 --- [main] org.apache.catalina.core.StandardEngine : Startin
2022-09-30 20:53:04.993 INFO 12148 --- [main] o.a.c.c.C.[Tomcat].[localhost].[/] : Initial
2022-09-30 20:53:04.993 INFO 12148 --- [main] w.s.c.ServletWebServerApplicationContext : Root W
2022-09-30 20:53:07.392 INFO 12148 --- [main] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat
2022-09-30 20:53:07.412 INFO 12148 --- [main] c.p.jeep.controller.FetchJeepTest : Starte
2022-09-30 20:53:08.395 INFO 12148 --- [o-auto-1-exec-1] o.a.c.c.C.[Tomcat].[localhost].[/] : Initial
2022-09-30 20:53:08.395 INFO 12148 --- [o-auto-1-exec-1] o.s.web.servlet.DispatcherServlet : Initial
2022-09-30 20:53:08.398 INFO 12148 --- [o-auto-1-exec-1] o.s.web.servlet.DispatcherServlet : Comple
2022-09-30 20:53:08.458 INFO 12148 --- [o-auto-1-exec-1] c.p.j.c.DefaultJeepSalesController : model=

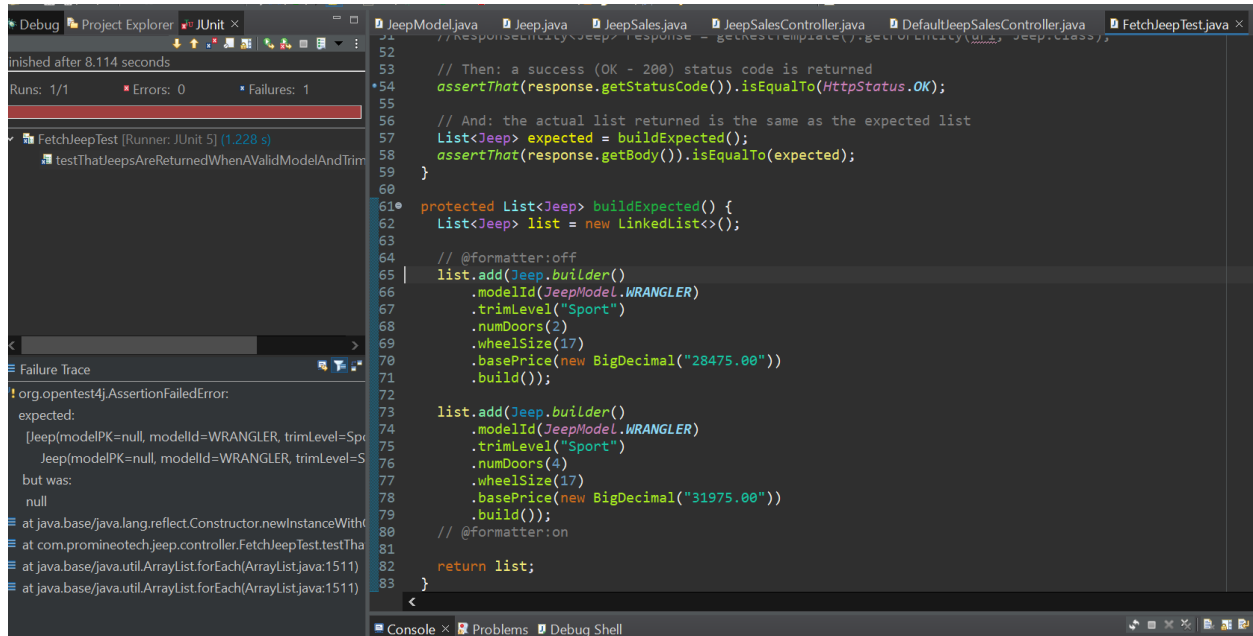
```

- 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
<b>Model ID</b>	WRANGLER	WRANGLER
<b>Trim Level</b>	Sport	Sport
<b>Num Doors</b>	2	4
<b>Wheel Size</b>	17	17
<b>Base Price</b>	\$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...
- The test with the assertion.
  - The JUnit status bar (should be red).
  - 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. 🖥️

The screenshot shows an IDE with a Java test failure and a console log. The test failure is in the 'Failure Trace' pane, showing an 'AssertionFailedError' with the message 'expected: [Jeep(modelPK=null, modelId=WRANGLER, trimLevel=Sport)] but was: null'. The console log shows the test execution details, including the start time, JVM running time, and the log message 'The fetchJeeps method was called with model=WRANGLER and trim=Sport'.

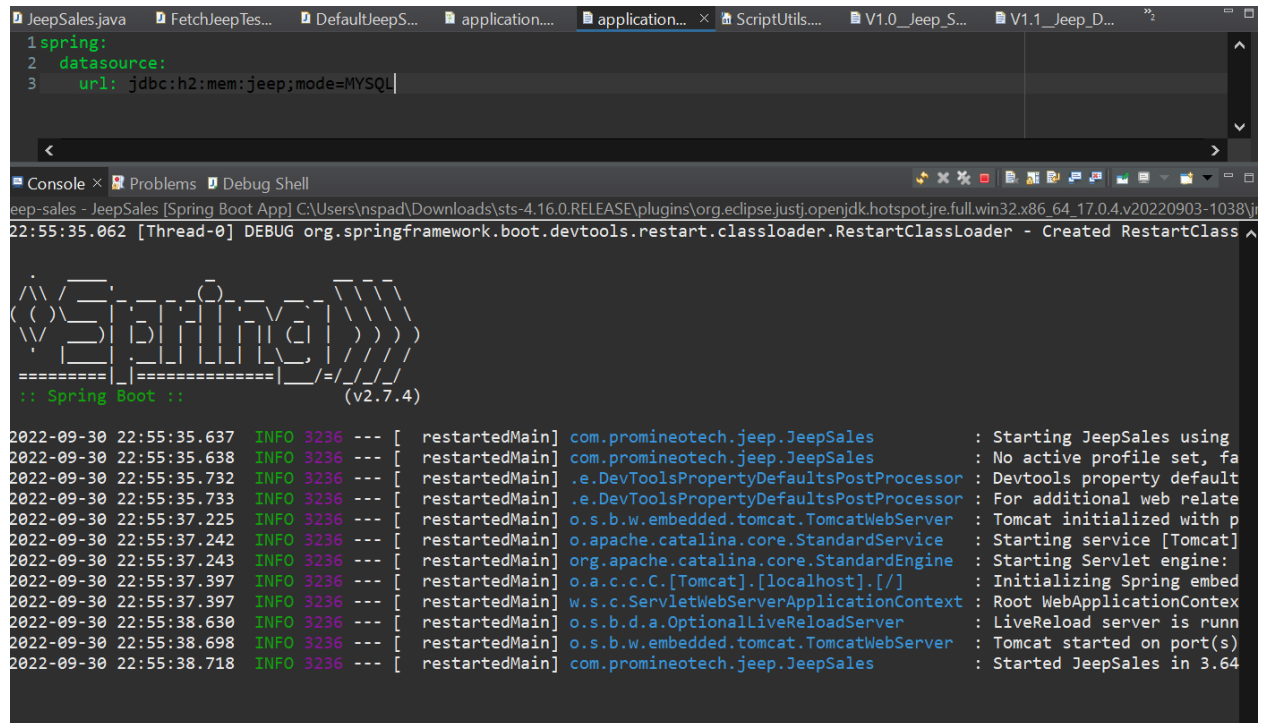
- 8) Add the database dependencies described in the video to the POM file (MySQL driver and Spring Boot Starter JDBC). To find them, navigate to <https://mvnrepository.com/>. Search for `mysql-connector-j` and `spring-boot-starter-jdbc`. In the POM file you don't need version numbers for either dependency 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.





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

Console × Problems Debug Shell

jeep-sales - JeepSales [Spring Boot App] C:\Users\nspad\Downloads\sts-4.16.0.RELEASE\plugins\org.eclipse.justi.openjdk.hotspot.jre.full.win32.x86\_64\_17.0.4.v20220903-1038\jre\bin\java.exe

22:55:35.062 [Thread-0] DEBUG org.springframework.boot.devtools.restart.classloader.RestartClassLoader - Created RestartClass

Spring Boot (v2.7.4)

2022-09-30 22:55:35.637 INFO 3236 --- [ restartedMain] com.promineotech.jeep.JeepSales : Starting JeepSales using

2022-09-30 22:55:35.638 INFO 3236 --- [ restartedMain] com.promineotech.jeep.JeepSales : No active profile set, fa

2022-09-30 22:55:35.732 INFO 3236 --- [ restartedMain] .e.DevToolsPropertyDefaultsPostProcessor : Devtools property default

2022-09-30 22:55:35.733 INFO 3236 --- [ restartedMain] .e.DevToolsPropertyDefaultsPostProcessor : For additional web relate

2022-09-30 22:55:37.225 INFO 3236 --- [ restartedMain] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat initialized with p

2022-09-30 22:55:37.242 INFO 3236 --- [ restartedMain] o.apache.catalina.core.StandardService : Starting service [Tomcat]

2022-09-30 22:55:37.243 INFO 3236 --- [ restartedMain] org.apache.catalina.core.StandardEngine : Starting Servlet engine:

2022-09-30 22:55:37.397 INFO 3236 --- [ restartedMain] o.a.c.c.C.[Tomcat].[localhost].[/] : Initializing Spring embed

2022-09-30 22:55:37.397 INFO 3236 --- [ restartedMain] w.s.c.ServletWebServerApplicationContext : Root WebApplicationContex

2022-09-30 22:55:38.630 INFO 3236 --- [ restartedMain] o.s.b.d.a.OptionalLiveReloadServer : LiveReload server is runn


2022-09-30 22:55:38.698 INFO 3236 --- [ restartedMain] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat started on port(s)

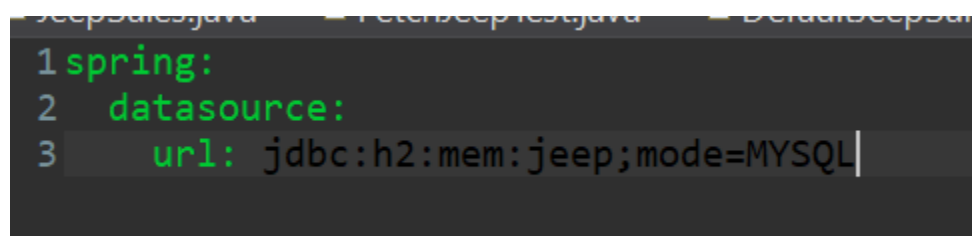
2022-09-30 22:55:38.718 INFO 3236 --- [ restartedMain] com.promineotech.jeep.JeepSales : Started JeepSales in 3.64

- 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
```

**Screenshots of Code:**

```
JeepSales.java  FetchJeepTest.... ×  DefaultJeepSa...  JeepSalesServ...  DefaultJeepSa...  ap
52
53 // Then: a success (OK - 200) status code is returned
54 • 54  assertThat(response.getStatusCode()).isEqualTo(HttpStatus.OK);
55
56 // And: the actual list returned is the same as the expected list
57 56  List<Jeep> expected = buildExpected();
58 57  assertThat(response.getBody()).isEqualTo(expected);
59 58  }
60
61 61  protected List<Jeep> buildExpected() {
62 62  List<Jeep> list = new LinkedList<>();
63
64 63  // @formatter:off
65 64  list.add(Jeep.builder()
66 65  .modelId(JeepModel.WRANGLER)
67 66  .trimLevel("Sport")
68 67  .numDoors(2)
69 68  .wheelSize(17)
70 69  .basePrice(new BigDecimal("28475.00"))
71 70  .build());
72
73 71  list.add(Jeep.builder()
74 72  .modelId(JeepModel.WRANGLER)
75 73  .trimLevel("Sport")
76 74  .numDoors(4)
77 75  .wheelSize(17)
78 76  .basePrice(new BigDecimal("31975.00"))
79 77  .build());
80 78  // @formatter:on
81
82 79  return list;
83 80  }
84
```

```

JeepSales.java  FetchJeepTest....  DefaultJeepSa... ×  JeepSalesServ...  DefaultJeepSa...  ap
1 package com.promineotech.jeep.controller;
2
3 import java.util.List;
4 import org.springframework.beans.factory.annotation.Autowired;
5 import org.springframework.web.bind.annotation.RestController;
6 import com.promineotech.jeep.entity.Jeep;
7 import com.promineotech.jeep.entity.JeepModel;
8 import com.promineotech.jeep.service.JeepSalesService;
9 import lombok.extern.slf4j.Slf4j;
10
11 @RestController
12 @Slf4j
13 public class DefaultJeepSalesController implements JeepSalesController {
14
15     @Autowired
16     private JeepSalesService jeepSalesService;
17
18     @Override
19     public List<Jeep> fetchJeeps(JeepModel model, String trim) {
20         log.info("model={}, trim{}", model, trim);
21         return jeepSalesService.fetchJeeps(model, trim);
22     }
23
24 }
25

```

```

JeepSales.java  FetchJeepTest....  DefaultJeepSa...  JeepSalesServ... ×  Def
1 package com.promineotech.jeep.service;
2
3 import java.util.List;
4 import com.promineotech.jeep.entity.Jeep;
5 import com.promineotech.jeep.entity.JeepModel;
6
7 public interface JeepSalesService {
8
9     List<Jeep> fetchJeeps(JeepModel model, String trim);
10
11 }
12

```



```
JeepSales.java  FetchJeepTest...  DefaultJeepSa...  JeepSalesServ...  DefaultJeepSa... x  a
1 package com.promineotech.jeep.service;
2
3 import java.util.List;
4 import org.springframework.stereotype.Service;
5 import com.promineotech.jeep.entity.Jeep;
6 import com.promineotech.jeep.entity.JeepModel;
7 import lombok.extern.slf4j.Slf4j;
8
9 @Service
10 @Slf4j
11 public class DefaultJeepSalesService implements JeepSalesService {
12
13     @Override
14     public List<Jeep> fetchJeeps(JeepModel model, String trim) {
15         log.info("The fetchJeeps method was called with model={} and trim={}",
16             model, trim);
17         return null;
18     }
19
20 }
21
```

```
JeepSales.java  FetchJeepTest...  DefaultJeepSa...  JeepSalesServ...  DefaultJeepSa...  application.y... x  ap
1 spring:
2     datasource:
3         username: jeep
4         password: jeep
5         url: jdbc:mysql://localhost:3306/jeep
```

```
JeepSales.java  FetchJeepTest...  DefaultJeepSa...  JeepSalesServ...  DefaultJeepSa...  application.y...  application-t... x  S
1 spring:
2     datasource:
3         url: jdbc:h2:mem:jeep;mode=MYSQL
```

## Screenshots of Running Application:

```
Console x Problems Debug Shell
jeep-sales - JeepSales [Spring Boot App] C:\Users\inspad\Downloads\sts-4.16.0.RELEASE\plugins\org.eclipse.justi.openjdk.hotspot.jre.full.win32.x86_64.17.0.4.v20220903-1038\jre\bin\javaw.exe (Sep 30, 2022, 10:55
22:55:35.062 [Thread-0] DEBUG org.springframework.boot.devtools.restart.classloader.RestartClassLoader - Created RestartClassLoader org.springframework.

<>S<>D<>I<>N<>G<>
=====J=====I=====
:: Spring Boot ::
(v2.7.4)

2022-09-30 22:55:35.637 INFO 3236 --- [ restartedMain] com.promineotech.jeep.JeepSales : Starting JeepSales using Java 17.0.4.1 on DESKTOP-VT
2022-09-30 22:55:35.638 INFO 3236 --- [ restartedMain] com.promineotech.jeep.JeepSales : No active profile set, falling back to 1 default pro
2022-09-30 22:55:35.732 INFO 3236 --- [ restartedMain] .e.DevToolsPropertyDefaultsPostProcessor : Devtools property defaults active! Set 'spring.devto
2022-09-30 22:55:35.733 INFO 3236 --- [ restartedMain] .e.DevToolsPropertyDefaultsPostProcessor : For additional web related logging consider setting
2022-09-30 22:55:37.225 INFO 3236 --- [ restartedMain] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat initialized with port(s): 8080 (http)
2022-09-30 22:55:37.242 INFO 3236 --- [ restartedMain] o.apache.catalina.core.StandardService : Starting service [Tomcat]
2022-09-30 22:55:37.243 INFO 3236 --- [ restartedMain] org.apache.catalina.core.StandardEngine : Starting Servlet engine: [Apache Tomcat/9.0.65]
2022-09-30 22:55:37.397 INFO 3236 --- [ restartedMain] o.a.c.c.C.[Tomcat].[localhost].[/] : Initializing Spring embedded WebApplicationContext
2022-09-30 22:55:37.397 INFO 3236 --- [ restartedMain] w.s.c.ServletWebServerApplicationContext : Root WebApplicationContext: initialization completed
2022-09-30 22:55:38.630 INFO 3236 --- [ restartedMain] o.s.b.d.a.OptionalLiveReloadServer : LiveReload server is running on port 35729
2022-09-30 22:55:38.698 INFO 3236 --- [ restartedMain] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat started on port(s): 8080 (http) with context
2022-09-30 22:55:38.718 INFO 3236 --- [ restartedMain] com.promineotech.jeep.JeepSales : Started JeepSales in 3.642 seconds (JVM running for
```

**URL to GitHub Repository:**

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