

# Web API Design with Spring Boot Week 15 Coding Assignment

**Points possible: 75**


**URL to GitHub Repository:** <https://github.com/Brierre/SpringBoot-jeep-sales-repo>

**URL to Public Link of your Video:** <https://rumble.com/v20b44i-week-15-video-explanation.html>


---

## 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 15 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>

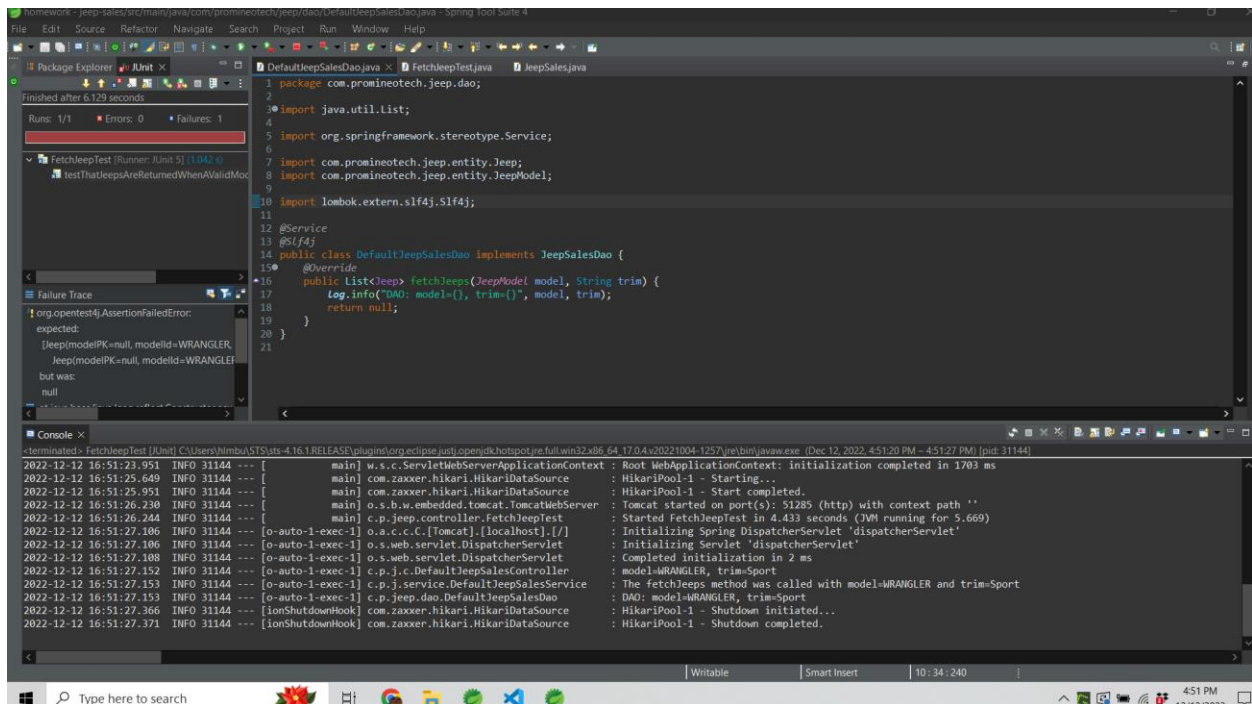
## Coding Steps:

- 1) In the application you've been building add a DAO layer:
  - a) Add the package, `com.promineotech.jeepp.dao`.
  - b) In the new package, create an interface named `JeepSalesDao`.
  - c) In the same package, create a class named `DefaultJeepSalesDao` that implements `JeepSalesDao`.
  - d) Add a method in the DAO interface and implementation that returns a list of Jeep models (class `Jeep`) and takes the model and trim parameters. Here is the method signature:

```
List<Jeep> fetchJeeps(JeepModel model, String trim);
```
- 2) In the Jeep sales service implementation class, inject the DAO interface as an instance variable. The instance variable should be private and should be named `jeepSalesDao`. Call the DAO method from the service method and store the returned value in a local variable named `jeeps`. Return the value in the `jeeps` variable (we will add to this later).

# Web API Design with Spring Boot Week 15 Coding Assignment

- 3) In the DAO implementation class (DefaultJeepSalesDao):
- Add the class-level annotation: `@Service`.
  - Add a log statement in `DefaultJeepSalesDao.fetchJeeps()` that logs the model and trim level. Run the integration test. Produce a screenshot showing the DAO implementation class and the log line in the IDE's console. 🖨️



```
package com.promineotech.jeeptest.dao;

import java.util.List;
import org.springframework.stereotype.Service;
import com.promineotech.jeeptest.entity.Jeeptest;
import com.promineotech.jeeptest.entity.JeeptestModel;
import lombok.extern.slf4j.Slf4j;

@Service
@Slf4j
public class DefaultJeepSalesDao implements JeepSalesDao {

    @Override
    public List<Jeep> fetchJeeps(JeeptestModel model, String trim) {
        log.info("DAO: model={}, trim={}", model, trim);
        return null;
    }
}
```

```
2022-12-12 16:51:23.951 INFO 31144 --- [main] w.s.c.ServletWebServerApplicationContext : Root WebApplicationContext: Initialization completed in 1703 ms
2022-12-12 16:51:25.649 INFO 31144 --- [main] com.zaxxer.hikari.HikariDataSource : HikariPool-1 - Starting...
2022-12-12 16:51:25.951 INFO 31144 --- [main] com.zaxxer.hikari.HikariDataSource : HikariPool-1 - Start completed.
2022-12-12 16:51:26.230 INFO 31144 --- [main] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat started on port(s): 51285 (http) with context path ''
2022-12-12 16:51:26.244 INFO 31144 --- [main] c.p.jeeptest.controller.FetchJeepTest : Started FetchJeepTest in 4.433 seconds (JVM running for 5.669)
2022-12-12 16:51:27.106 INFO 31144 --- [o-auto-1-exec-1] o.a.c.c.C.[Tomcat].[localhost].[/] : Initializing Spring DispatcherServlet 'dispatcherServlet'
2022-12-12 16:51:27.106 INFO 31144 --- [o-auto-1-exec-1] o.s.web.servlet.DispatcherServlet : Initializing Servlet 'dispatcherServlet'
2022-12-12 16:51:27.106 INFO 31144 --- [o-auto-1-exec-1] o.s.web.servlet.DispatcherServlet : Completed initialization in 2 ms
2022-12-12 16:51:27.152 INFO 31144 --- [o-auto-1-exec-1] c.p.jeeptest.controller.FetchJeepTest : model=WRANGLER, trim=Sport
2022-12-12 16:51:27.153 INFO 31144 --- [o-auto-1-exec-1] c.p.jeeptest.service.DefaultJeepSalesService : The fetchJeeps method was called with model=WRANGLER and trim=Sport
2022-12-12 16:51:27.153 INFO 31144 --- [o-auto-1-exec-1] c.p.jeeptest.dao.DefaultJeepSalesDao : DAO: model=WRANGLER, trim=Sport
2022-12-12 16:51:27.366 INFO 31144 --- [ionShutdownHook] com.zaxxer.hikari.HikariDataSource : HikariPool-1 - Shutdown initiated...
2022-12-12 16:51:27.371 INFO 31144 --- [ionShutdownHook] com.zaxxer.hikari.HikariDataSource : HikariPool-1 - Shutdown completed.
```


- In `DefaultJeepSalesDao`, inject an instance variable of type `NamedParameterJdbcTemplate`.
- Write SQL to return a list of Jeep models based on the parameters: model and trim. Be sure to utilize the SQL Injection prevention mechanism of the `NamedParameterJdbcTemplate` using `:model_id` and `:trim_level` in the query.
- Add the parameters to a parameter map as shown in the video. Don't forget to convert the `JeepModel` enum value to a String (i.e., `params.put("model_id", model.toString());`)
- Call the query method on the `NamedParameterJdbcTemplate` instance variable to return a list of Jeep model objects. Use a `RowMapper` to map each row of the result set. Remember to convert `modelId` to a `JeepModel`. See the video for details. Produce a screenshot to show the complete method in the implementation class. 🖨️

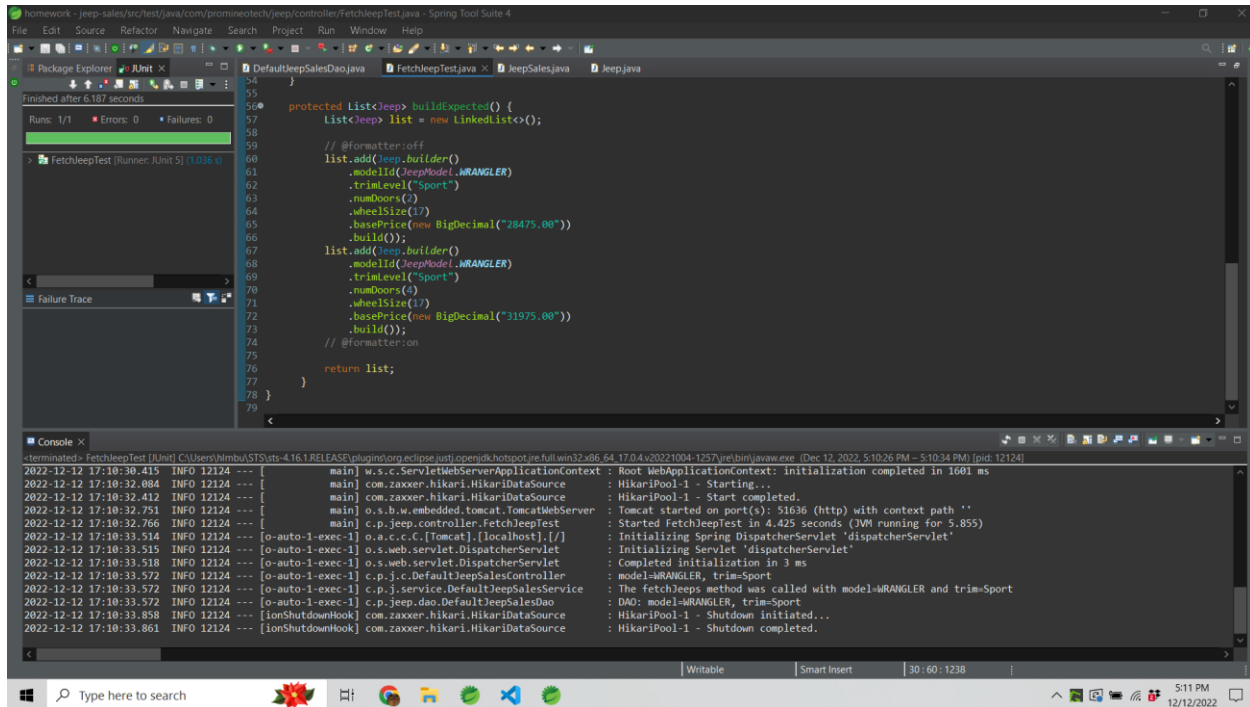
## Web API Design with Spring Boot Week 15 Coding Assignment

```
DefaultJeepSalesDao.java × FetchJeepTest.java JeepSales.java
1 package com.promineotech.jeep.dao;
2
3 import java.math.BigDecimal;
4 import java.sql.ResultSet;
5 import java.sql.SQLException;
6 import java.util.HashMap;
7 import java.util.List;
8 import java.util.Map;
9
10 import org.springframework.beans.factory.annotation.Autowired;
11 import org.springframework.jdbc.core.RowMapper;
12 import org.springframework.jdbc.core.namedparam.NamedParameterJdbcTemplate;
13 import org.springframework.stereotype.Service;
14
15 import com.promineotech.jeep.entity.Jeep;
16 import com.promineotech.jeep.entity.JeepModel;
17
18 import lombok.extern.slf4j.Slf4j;
19
20 @Service
21 @Slf4j
22 public class DefaultJeepSalesDao implements JeepSalesDao {
23
24     @Autowired
25     private NamedParameterJdbcTemplate jdbcTemplate;
26
27     @Override
28     public List<Jeep> fetchJeeps(JeepModel model, String trim) {
29         log.info("DAO: model={}, trim={}", model, trim);
30
31         // @formatter:off
32         String sql = "
33             + "SELECT * "
34             + "FROM models "
35             + "WHERE model_id = :model_id AND trim_level = :trim_level";
36         // @formatter:on
37         Map<String, Object> params = new HashMap<>();
38
39         params.put("model_id", model.toString());
40         params.put("trim_level", trim);
41
42         return jdbcTemplate.query(sql, params, new RowMapper<>() {
43
44             @Override
45             public Jeep mapRow(ResultSet rs, int rowNum) throws SQLException {
46                 // @formatter:off
47                 return Jeep.builder()
48                     .basePrice(new BigDecimal(rs.getString("base_price")))
49                     .modelId(JeepModel.valueOf(rs.getString("model_id")))
50                     .modelPK(rs.getLong("model_pk"))
51                     .numDoors(rs.getInt("num_doors"))
52                     .trimLevel(rs.getString("trim_level"))
53                     .wheelSize(rs.getInt("wheel_size"))
54                     .build();
55                 // @formatter:on
56             }
57         });
58     }
59 }
60
```

- 4) Add a getter in the Jeep class for modelPK. Add the @JsonIgnore annotation to the getter to exclude the modelPK value from the returned object.

# Web API Design with Spring Boot Week 15 Coding Assignment

- 5) Run the test to produce a green status bar. Produce a screenshot showing the test and the green status bar. 



The screenshot shows an IDE window with the following components:

- Package Explorer:** Shows the project structure with 'JUnit' and 'FetchJeepTest' visible.
- JUnit Runner:** Displays 'FetchJeepTest (Runner: JUnit 5) (1.036 s)' with a green status bar and 'Runs: 1/1', 'Errors: 0', 'Failures: 0'.
- Failure Trace:** Empty, indicating no failures.
- Console:** Shows the following log output:

```
<terminated> FetchJeepTest [JUnit] C:\Users\hmbu\STSD\sts-4161\RELEASE\plugins\org.eclipse.justi.openjdk.hotspot.jre.full.win32.x86_64.17.0.4.v20221004-1257\jre\bin\java.exe (Dec 12, 2022, 5:10:26 PM - 5:10:34 PM) [pid: 12124]
2022-12-12 17:10:30.415 INFO 12124 --- [main] w.s.c.ServletWebServerApplicationContext : Root WebApplicationContext: initialization completed in 1601 ms
2022-12-12 17:10:32.084 INFO 12124 --- [main] com.zaxxer.hikari.HikariDataSource : HikariPool-1 - Starting...
2022-12-12 17:10:32.412 INFO 12124 --- [main] com.zaxxer.hikari.HikariDataSource : HikariPool-1 - Start completed.
2022-12-12 17:10:32.751 INFO 12124 --- [main] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat started on port(s): 51636 (http) with context path ''
2022-12-12 17:10:32.766 INFO 12124 --- [main] c.p.j.e.controller.FetchJeepTest : Started FetchJeepTest in 4.425 seconds (JVM running for 5.855)
2022-12-12 17:10:33.514 INFO 12124 --- [o-auto-1-exec-1] o.a.c.c.C.[Tomcat].[localhost].[/] : Initializing Spring DispatcherServlet 'dispatcherServlet'
2022-12-12 17:10:33.518 INFO 12124 --- [o-auto-1-exec-1] o.s.web.servlet.DispatcherServlet : Initializing Servlet 'dispatcherServlet'
2022-12-12 17:10:33.518 INFO 12124 --- [o-auto-1-exec-1] o.s.web.servlet.DispatcherServlet : Completed initialization in 3 ms
2022-12-12 17:10:33.572 INFO 12124 --- [o-auto-1-exec-1] c.p.j.c.DefaultJeepSalesController : model=WRANGLER, trim=Sport
2022-12-12 17:10:33.572 INFO 12124 --- [o-auto-1-exec-1] c.p.j.service.DefaultJeepSalesService : The fetchJeeps method was called with model=WRANGLER and trim=Sport
2022-12-12 17:10:33.572 INFO 12124 --- [o-auto-1-exec-1] c.p.j.e.dao.DefaultJeepSalesDao : DAO: model=WRANGLER, trim=Sport
2022-12-12 17:10:33.858 INFO 12124 --- [ionShutdownHook] com.zaxxer.hikari.HikariDataSource : HikariPool-1 - Shutdown initiated...
2022-12-12 17:10:33.861 INFO 12124 --- [ionShutdownHook] com.zaxxer.hikari.HikariDataSource : HikariPool-1 - Shutdown completed.
```