

Web API Design with Spring Boot Week 15 Coding Assignment


Points possible: 75

URL to GitHub Repository: <https://github.com/AvannR/SpringBoot-Jeep-Sales-Week-13-to-16->


URL to Public Link of your Video: <https://youtu.be/T6jb62fERKA>

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.

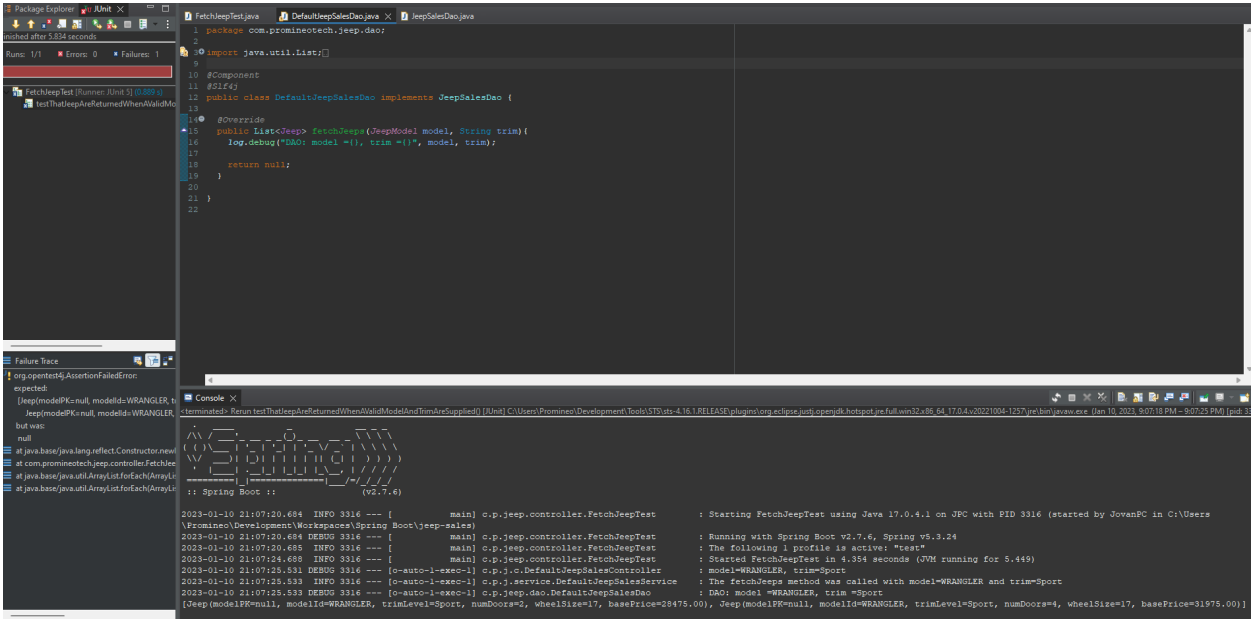
Project Resources: <https://github.com/promineotech/Spring-Boot-Course-Student-Resources>

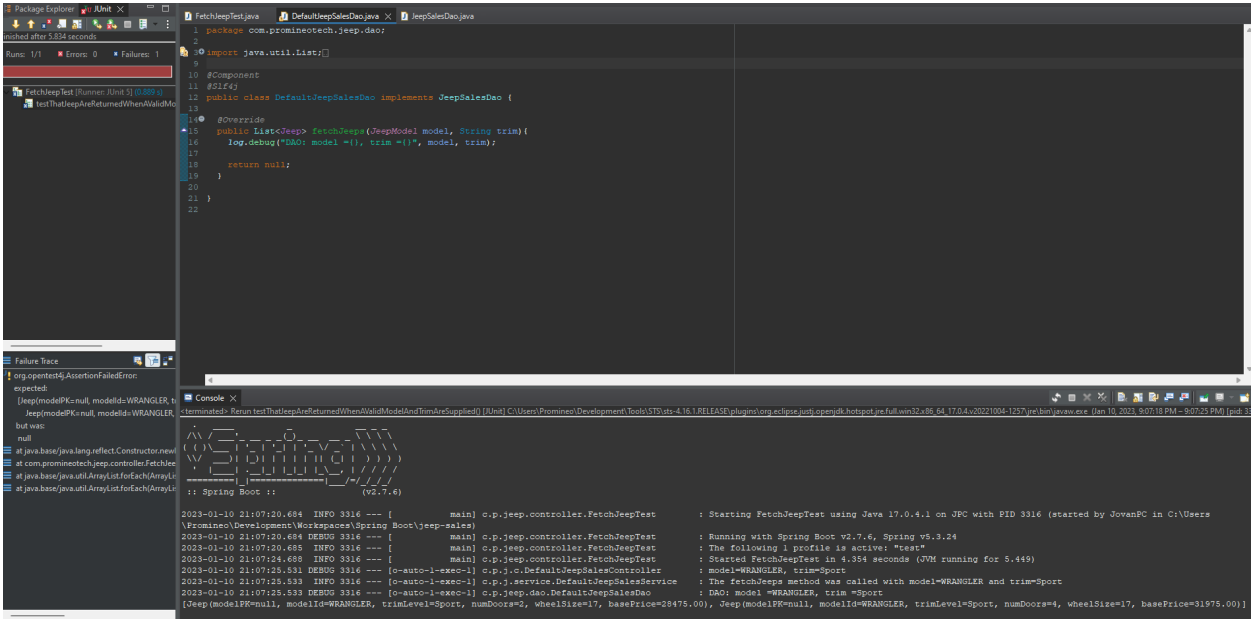
Coding Steps:

- 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).

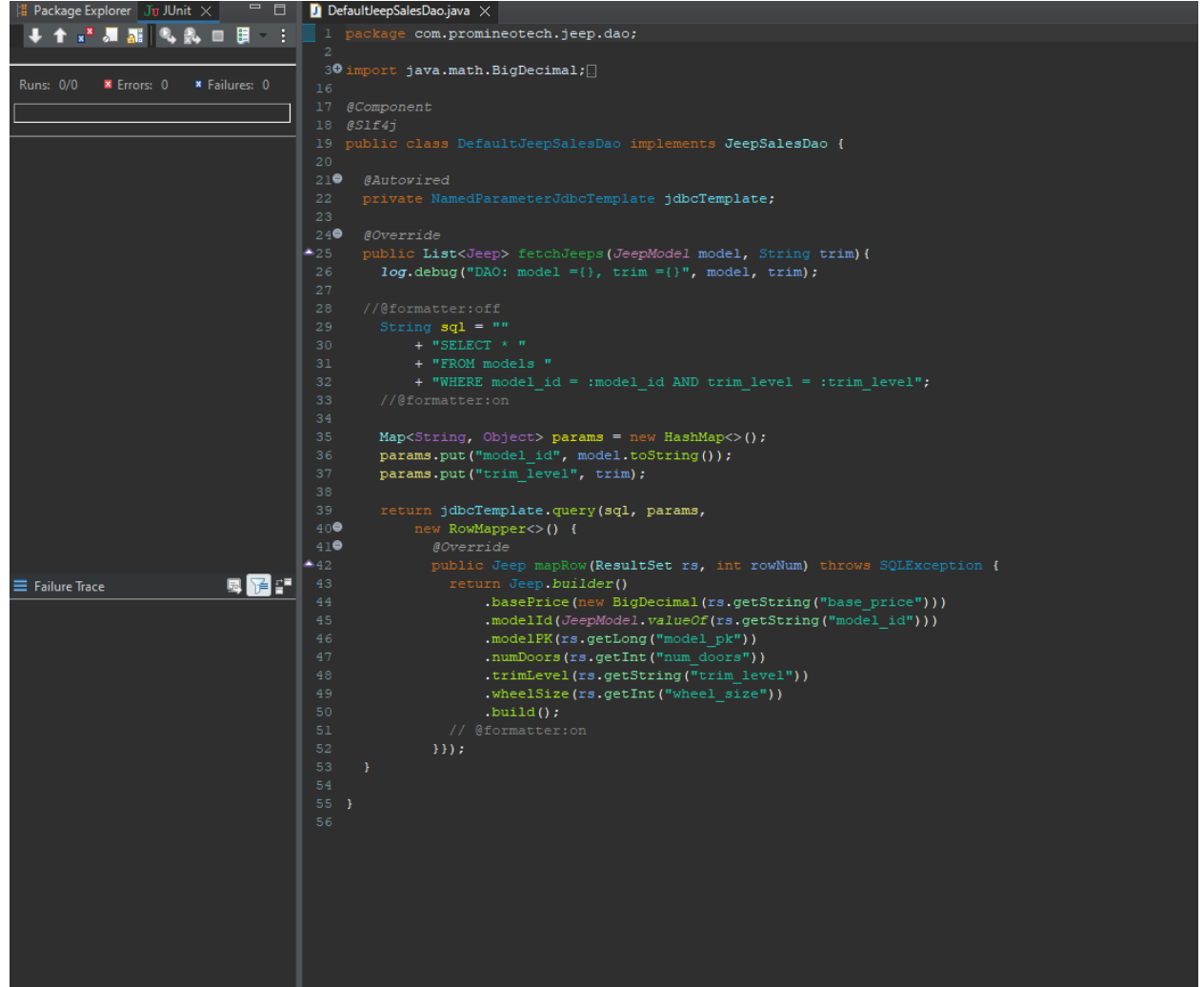
Web API Design with Spring Boot Week 15 Coding Assignment

- 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. 



- 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

Web API Design with Spring Boot Week 15 Coding Assignment

screenshot to show the complete method in the implementation class. 

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

- 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 displays an IDE with three panels. The top panel shows the source code of `FetchJeepTest.java`. The middle panel shows the test execution results, indicating a green status bar. The bottom panel shows the console output, which includes the Spring Boot startup logs and the test execution details.

```
30 "classpath:/migrations/V1_1_jeep_data.sql",
31 config = @SqlConfig(encoding = "utf-8"))
32 // @formatter:off
33 class FetchJeepTest {
34
35     // @formatter:off
36     @Test
37     void testThatJeepsAreReturnedWhenRequested() {
38         JeepModel model = JeepModel.WRANGLER;
39         String trim = "Sport";
40         String url =
41             "http://localhost:8080/jeeps?model=" + model + "&trim=" + trim;
42         Response response = restTemplate.exchange(
43             url, HttpMethod.GET, null, new ParameterizedTypeReference<>() {});
44         assertEquals(response.getStatusCode(), HttpStatus.OK);
45         List<Jeep> actual = response.getBody();
46         List<Jeep> expected = buildExpected();
47         assertEquals(actual, expected);
48     }
49     private List<Jeep> buildExpected() {
50         List<Jeep> list = new LinkedList<>();
51         // @formatter:off
52         list.add(Jeep.builder()
53             .model(JeepModel.WRANGLER)
54             .trimLevel("Sport")
55             .numDoors(2)
56             .wheelSize(17)
57             .basePrice(new BigDecimal("28475.00"))
58             .build());
59         list.add(Jeep.builder()
60             .model(JeepModel.WRANGLER)
61             .trimLevel("Sport")
62             .numDoors(2)
63             .wheelSize(17)
64             .basePrice(new BigDecimal("31975.00"))
65             .build());
66         // @formatter:off
67         Collections.sort(list);
68         return list;
69     }
70     @Autowired
71     private TestRestTemplate restTemplate;
72     @LocalServerPort
73     private int serverPort;
74 }
75
```

Failure Trace

```
2023-01-10 22:42:24.159 INFO 19424 --- [main] c.p.jee.controller.FetchJeepTest : Starting FetchJeepTest using Java 17.0.4.1 on JPC with PID 19424 (started by JovanPC in C:\Users\JovanPC\Development\Workspaces\Spring Boot\jeep-sales)
2023-01-10 22:42:24.160 DEBUG 19424 --- [main] c.p.jee.controller.FetchJeepTest : Running with Spring Boot v2.7.4, Spring v5.3.24
2023-01-10 22:42:24.161 INFO 19424 --- [main] c.p.jee.controller.FetchJeepTest : The following 1 profile is active: "test"
2023-01-10 22:42:25.104 INFO 19424 --- [o-auto-1-exec-1] c.p.jee.controller.FetchJeepTest : Started FetchJeepTest in 4.616 seconds (JVM running for 5.473)
2023-01-10 22:42:25.104 DEBUG 19424 --- [o-auto-1-exec-1] c.p.jee.dao.DefaultJeepSalesController : model=WRANGLER, trim=Sport
2023-01-10 22:42:25.107 INFO 19424 --- [o-auto-1-exec-1] c.p.jee.service.DefaultJeepSalesService : The fetchJeeps method was called with model=WRANGLER and trim=Sport
2023-01-10 22:42:25.107 DEBUG 19424 --- [o-auto-1-exec-1] c.p.jee.dao.DefaultJeepSalesDao : DAO: model=WRANGLER, trim=Sport
```