

Web API Design with Spring Boot Week 15 Coding Assignment


Points possible: 75

URL to GitHub Repository: https://github.com/CamiGrace/Springboot_JeepSales


URL to Public Link of your Video: <https://www.youtube.com/watch?v=PKsl9g1wfy4>

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

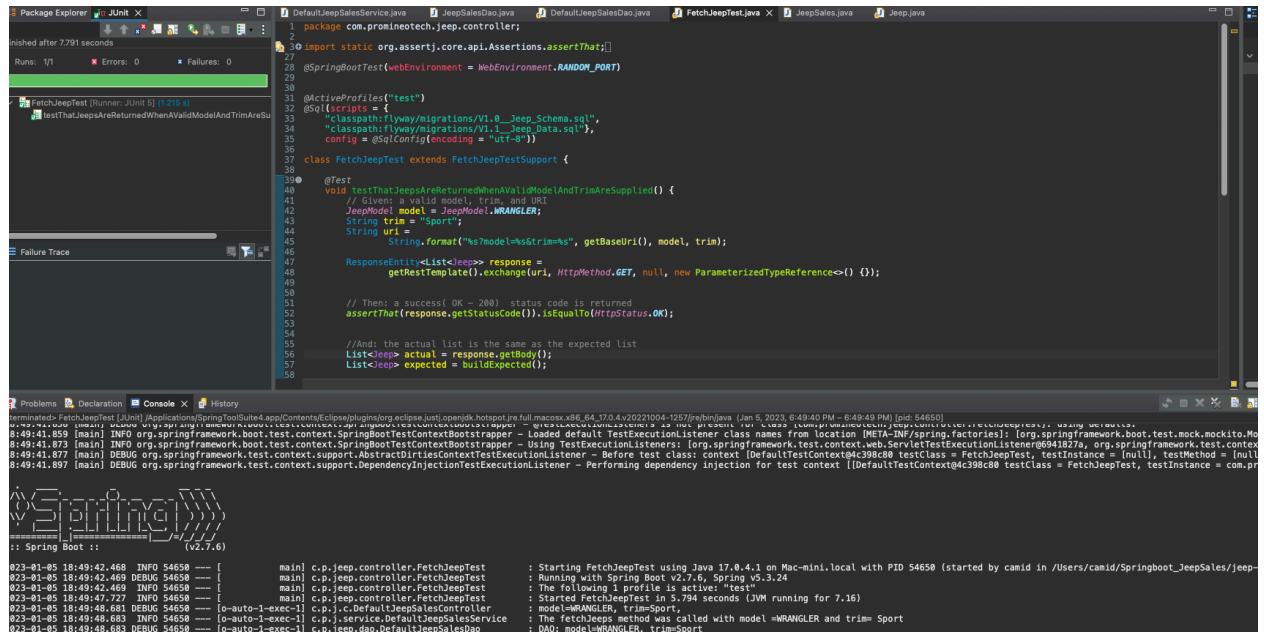
Remember to convert `modelId` to a `JeepModel`. See the video for details. Produce a screenshot to show the complete method in the implementation class. 🖥️

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

- 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 the following components:

- Package Explorer:** Shows the project structure with a green status bar for the `FetchJeepTest` class.
- JUnit Runner:** Shows the test results for `FetchJeepTest` with a green status bar and a message: "FetchJeepTest [Runner: JUnit 5] (1.0/6)".
- Failure Trace:** Shows the test results for `FetchJeepTest` with a green status bar.
- Code Editor:** Displays the `FetchJeepTest` class with the following code:

```
1 package com.promineotech.jeep.controller;
2
3 import static org.assertj.core.api.Assertions.assertThat;
4
5 @SpringBootTest(webEnvironment = WebEnvironment.RANDOM_PORT)
6
7 @ActiveProfiles("test")
8 @Sql(scripts = {
9     "classpath:/migrations/V1.0_jeep_schema.sql",
10    "classpath:/migrations/V1.1_jeep_data.sql",
11    config = @SqlConfig(encoding = "UTF-8")
12 })
13 class FetchJeepTest extends FetchJeepTestSupport {
14
15     @Test
16     void testThatJeepsAreReturnedWhenValidModelAndTrimAreSupplied() {
17         // Given: a valid model, trim, and URI
18         JeepModel model = JeepModel.WRANGLER;
19         String trim = "Sport";
20         String uri =
21             String.format("%s?model=%s&trim=%s", getBaseUrl(), model, trim);
22
23         Response<List<Jeep>> response =
24             getRestTemplate().exchange(uri, HttpMethod.GET, null, new ParameterizedTypeReference<>() {});
25
26         // Then: a success (OK - 200) status code is returned
27         assertThat(response.getStatusCode()).isEqualTo(HttpStatus.OK);
28
29         // And: the actual list is the same as the expected list
30         List<Jeep> actual = response.getBody();
31         List<Jeep> expected = buildExpected();
32     }
33 }
```
- Console:** Shows the output of the test run, including the Spring Boot logo and the following log messages:

```
023-01-05 18:49:42.466 INFO 54650 --- [main] c.p.jeep.controller.FetchJeepTest : Starting FetchJeepTest using Java 17.0.4.1 on Mac-mini.local with PID 54650 (started by camid in /Users/camid/Springboot_JeepSales/jeep-023-01-05)
023-01-05 18:49:42.469 DEBUG 54650 --- [main] c.p.jeep.controller.FetchJeepTest : Running with Spring Boot v2.7.6, Spring v5.3.24
023-01-05 18:49:42.469 INFO 54650 --- [main] c.p.jeep.controller.FetchJeepTest : The following 1 profile is active: "test"
023-01-05 18:49:47.727 INFO 54650 --- [main] c.p.jeep.controller.FetchJeepTest : Started FetchJeepTest in 5.794 seconds (JVM running for 7.16)
023-01-05 18:49:48.683 DEBUG 54650 --- [io-auto-1-exec-1] c.p.j.service.DefaultJeepSalesService : model=WRANGLER, trim=Sport
023-01-05 18:49:48.683 INFO 54650 --- [io-auto-1-exec-1] c.p.j.service.DefaultJeepSalesService : The fetchJeeps method was called with model=WRANGLER and trim=Sport
```