

Web API Design with Spring Boot Week 15 Coding Assignment


Points possible: 75

URL to GitHub Repository: <https://github.com/AlexWarr/Week-15-Homework>


URL to Public Link of your Video: <https://youtu.be/gjL1Raz5jUI>

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.

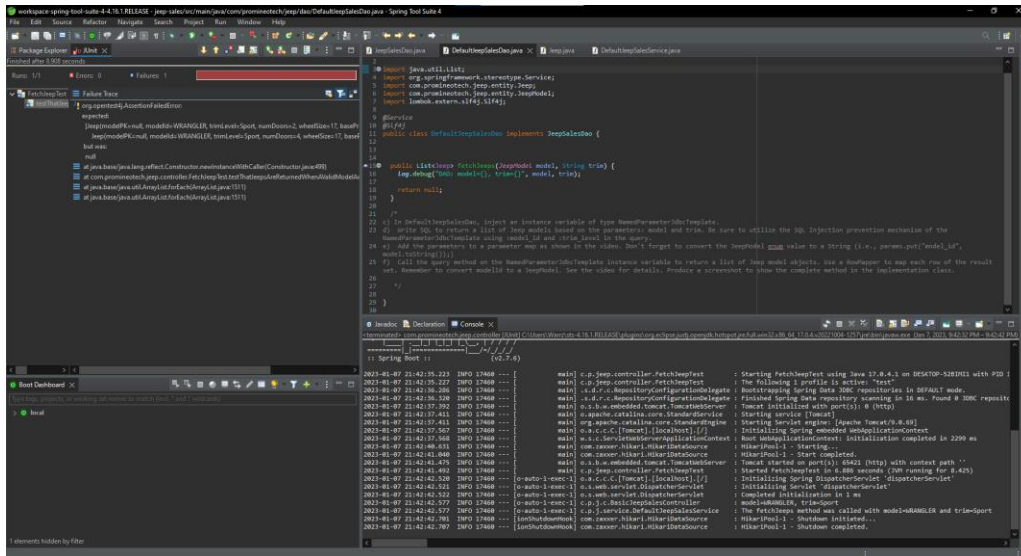
3. Save the Coding Assignment Document as a .pdf and do the following:

- 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

Coding Steps:

- 1) In the application you've been building add a DAO layer:
 - a) Add the package, `com.promineotech.jeeptest.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).
- 3) In the DAO implementation class (`DefaultJeepSalesDao`):
 - a) Add the class-level annotation: `@Service`.
 - b) 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. 

Web API Design with Spring Boot Week 15 Coding Assignment

- c) In DefaultJeepSalesDao, inject an instance variable of type NamedParameterJdbcTemplate.
- d) 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.
- e) 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());)
- f) 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. 🖥️

```
public List<Jeep> fetchJeeps(JeepModel model, String trim) {
    Log.debug("DAO: model={}, trim={}", model, trim);

    //@formatter:off
    String sql = ""
        + "SELECT * "
        + "FROM models "
        + "WHERE model_id = :model_id AND trim_level = :trim_level";
    //@formatter:on

    Map<String, Object> params = new HashMap<String, Object>();
    params.put("model_id", model.toString());
    params.put("trim_level", trim);

    return jdbcTemplate.query(sql, params,
        new RowMapper<Jeep>() {
            @Override
            public Jeep mapRow(ResultSet rs, int rowNum) throws SQLException {
                //@formatter:off
                return Jeep.builder()
                    .basePrice(new BigDecimal(rs.getString("base_price")))
                    .modelId(JeepModel.valueOf(rs.getString("model_id")))
                    .modelPK(rs.getLong("model_pk"))
                    .numDoors(rs.getInt("num_doors"))
                    .trimLevel(rs.getString("trim_level"))
                    .wheelSize(rs.getInt("wheel_size"))
                    .build();
                //@formatter:on
            }
        });
}
```

- 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.

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

