

CMRapp Web API Design w/ Spring Boot Week 15 Coding Assignment


Points possible: 75

URL to GitHub Repository: <https://github.com/CMRapp/jeep-sales>


URL to Public Link of your Video: <http://videos.cmrwebstudio.com/promineo-tech-be-videos/CMR-wk15.mp4>

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.


CMRapp Web API Design w/ 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).
- 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. 


CMRapp Web API Design w/ Spring Boot Week 15 Coding Assignment

The screenshot shows an IDE with the Package Explorer on the left, the Java editor in the center, and the Spring Boot console at the bottom. The Package Explorer shows the project structure for 'jeep-sales'. The Java editor displays the code for 'DefaultJeepSalesDao.java', which implements 'JeepSalesDao'. The code includes an '@Autowired' field for 'NamedParameterJdbcTemplate', an '@Override' method 'fetchJeeps' that constructs an SQL query with parameters for 'model_id' and 'trim_level', and a 'return' statement using 'jdbcTemplate.query' with a 'RowMapper'.

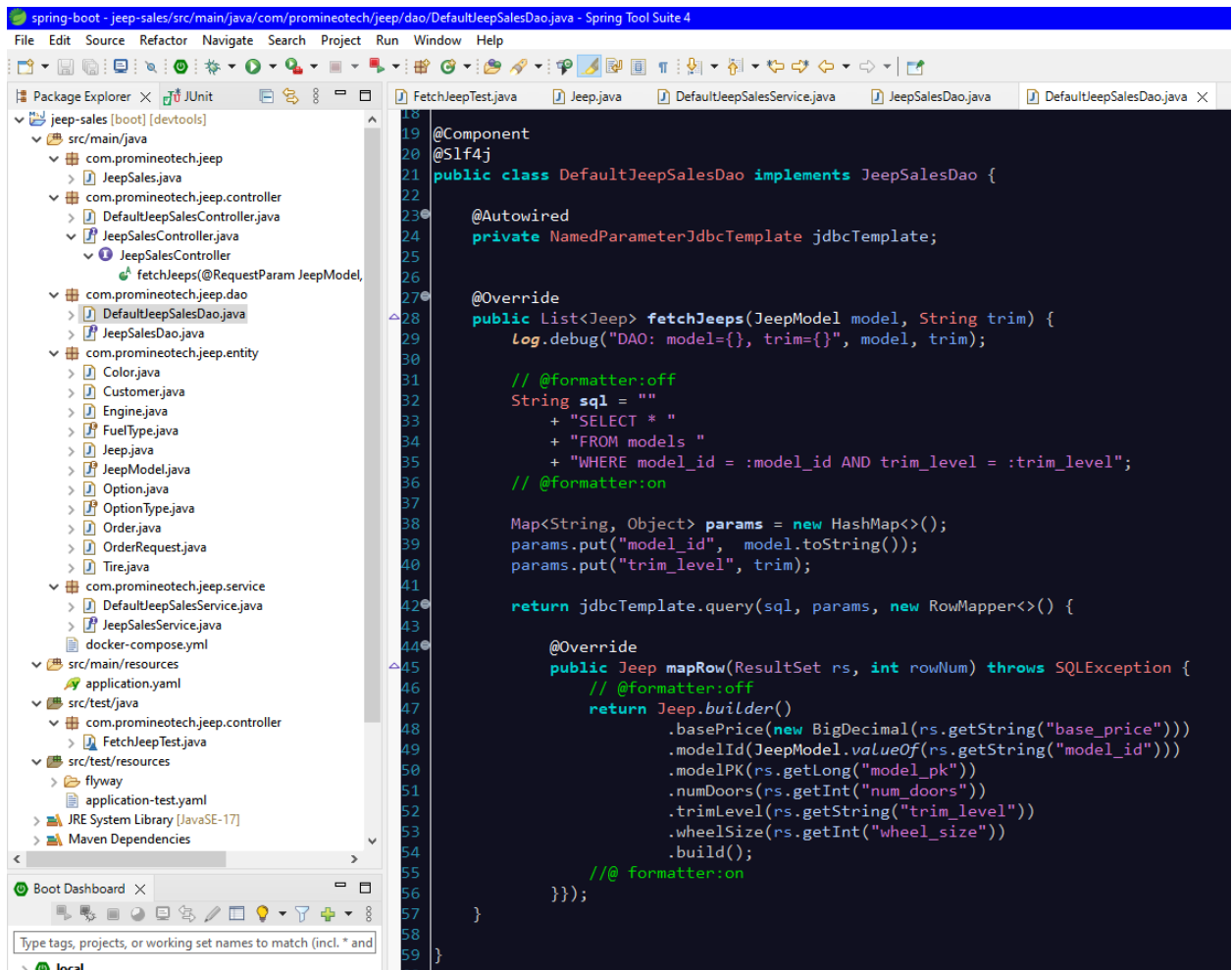
```
1 package com.promineotech.jeep.dao;
2
3 import java.math.BigDecimal;
4
5
6
7
8
9
10
11 @Component
12 @Slf4j
13 public class DefaultJeepSalesDao implements JeepSalesDao {
14
15     @Autowired
16     private NamedParameterJdbcTemplate jdbcTemplate;
17
18
19     @Override
20     public List<Jeep> fetchJeeps(JeepModel model, String trim) {
21         log.debug("DAO: model={}, trim={}", model, trim);
22
23         // @formatter:off
24         String sql = "
25             + "SELECT * "
26             + "FROM models "
27             + "WHERE model_id = :model_id AND trim_level = :trim_level";
28         // @formatter:on
29
30         Map<String, Object> params = new HashMap<>();
31         params.put("model_id", model.toString());
32         params.put("trim_level", trim);
33
34         return jdbcTemplate.query(sql, params, new RowMapper<>() {
35
36         });
37     }
38 }
```


The console output shows the following log messages:

```
2022-12-23 11:57:03.601 INFO 16496 --- [main] c.p.jeep.controller.FetchJeepTest : Starting FetchJeepTest using Java 1
2022-12-23 11:57:03.602 DEBUG 16496 --- [main] c.p.jeep.controller.FetchJeepTest : Running with Spring Boot v2.7.6, Sp
2022-12-23 11:57:03.602 INFO 16496 --- [main] c.p.jeep.controller.FetchJeepTest : The following 1 profile is active:
2022-12-23 11:57:06.058 INFO 16496 --- [main] c.p.jeep.controller.FetchJeepTest : Started FetchJeepTest in 2.749 seco
2022-12-23 11:57:06.858 INFO 16496 --- [o-auto-1-exec-1] c.p.j.c.DefaultJeepSalesController : Retrieving Jeeps with model=WRANGLER
2022-12-23 11:57:06.859 INFO 16496 --- [o-auto-1-exec-1] c.p.j.service.DefaultJeepSalesService : The fetchJeeps method was called wi
2022-12-23 11:57:06.859 DEBUG 16496 --- [o-auto-1-exec-1] c.p.jeep.dao.DefaultJeepSalesDao : DAO: model=WRANGLER, trim=Sport
```

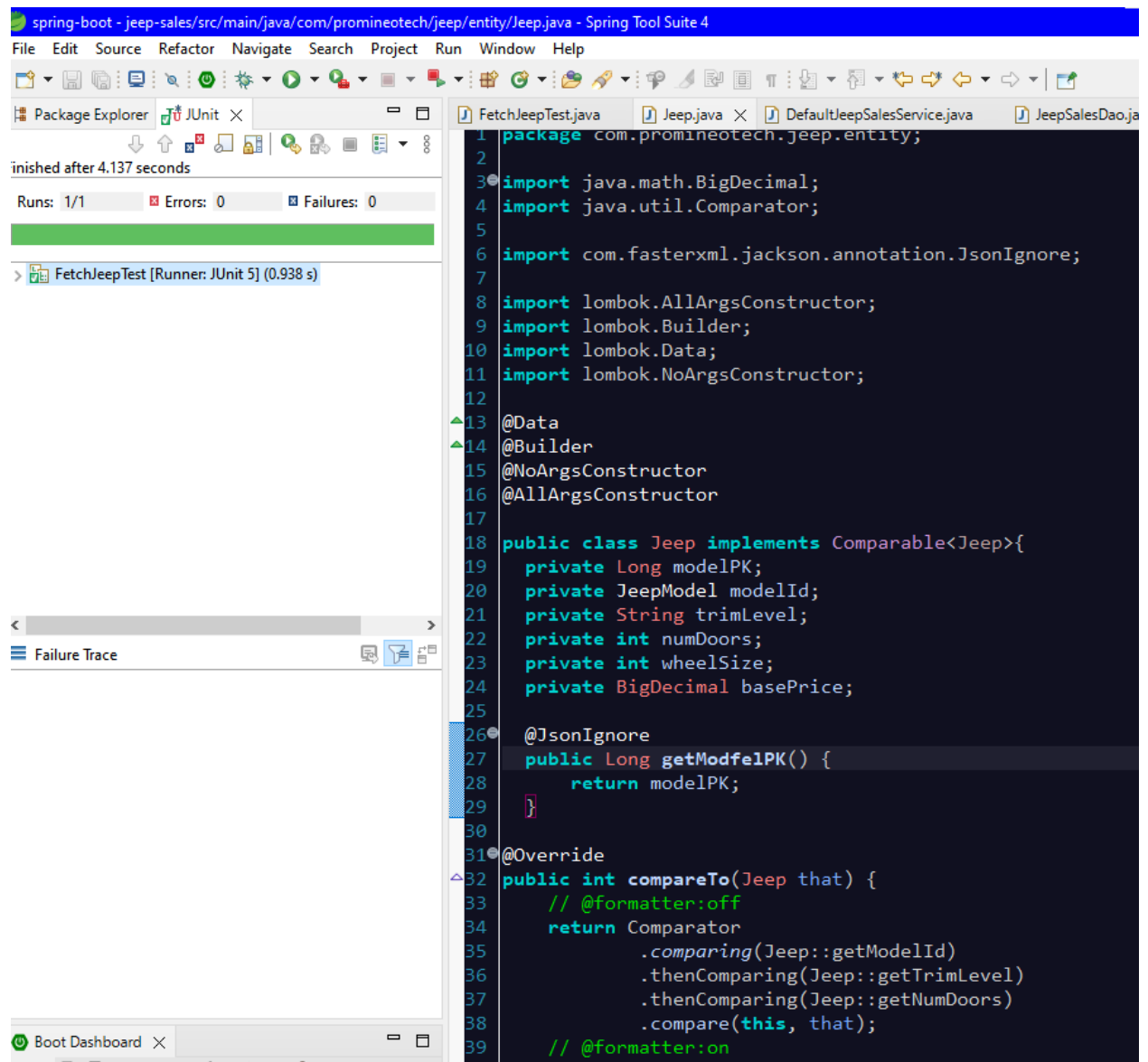
- c) In `DefaultJeepSalesDao`, inject an instance variable of type `NamedParameterJdbcTemplate`.
- d) Write SQL to return a list of Jeep models based on the parameters: model and trim. Be sure to utilize the SQL Injection prevention mechanism of the `NamedParameterJdbcTemplate` using `:model_id` and `: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. 

CMRapp Web API Design w/ Spring Boot Week 15 Coding Assignment



- 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. 

CMRapp Web API Design w/ Spring Boot Week 15 Coding Assignment



The screenshot shows an IDE window titled "spring-boot - jeep-sales/src/main/java/com/promineotech/jeep/entity/Jeep.java - Spring Tool Suite 4". The interface includes a menu bar (File, Edit, Source, Refactor, Navigate, Search, Project, Run, Window, Help), a toolbar, and a Package Explorer on the left. The Package Explorer shows a project structure with a "JUnit" folder containing a "FetchJeepTest" class. Below the Package Explorer, a status bar indicates "finished after 4.137 seconds", "Runs: 1/1", "Errors: 0", and "Failures: 0". A "Failure Trace" section is also visible. The main editor displays the code for "FetchJeepTest.java". The code defines a "Jeep" class that implements the "Comparable<Jeep>" interface. It includes imports for "java.math.BigDecimal", "java.util.Comparator", "com.fasterxml.jackson.annotation.JsonIgnore", "lombok.AllArgsConstructor", "lombok.Builder", "lombok.Data", and "lombok.NoArgsConstructor". The class is annotated with "@Data", "@Builder", "@NoArgsConstructor", and "@AllArgsConstructor". It has private fields for "modelPK", "modelId", "trimLevel", "numDoors", "wheelSize", and "basePrice". The "getModfelPK()" method is annotated with "@JsonIgnore". The "compareTo()" method is annotated with "@Override" and uses a comparator to compare "Jeep" objects based on "modelId", "trimLevel", and "numDoors".

```
1 package com.promineotech.jeep.entity;
2
3 import java.math.BigDecimal;
4 import java.util.Comparator;
5
6 import com.fasterxml.jackson.annotation.JsonIgnore;
7
8 import lombok.AllArgsConstructor;
9 import lombok.Builder;
10 import lombok.Data;
11 import lombok.NoArgsConstructor;
12
13 @Data
14 @Builder
15 @NoArgsConstructor
16 @AllArgsConstructor
17
18 public class Jeep implements Comparable<Jeep>{
19     private Long modelPK;
20     private JeepModel modelId;
21     private String trimLevel;
22     private int numDoors;
23     private int wheelSize;
24     private BigDecimal basePrice;
25
26     @JsonIgnore
27     public Long getModfelPK() {
28         return modelPK;
29     }
30
31     @Override
32     public int compareTo(Jeep that) {
33         // @formatter:off
34         return Comparator
35             .comparing(Jeep::getModelId)
36             .thenComparing(Jeep::getTrimLevel)
37             .thenComparing(Jeep::getNumDoors)
38             .compare(this, that);
39         // @formatter:on
40     }
41 }
```