

Web API Design with Spring Boot Week 15 Coding Assignment

Points possible: 75


URL to GitHub Repository:

https://github.com/DracaQueen/ProminenoTech_SpringbootProject


URL to Public Link of your Video: <https://youtu.be/B5Vx0K1K1Wg>

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):
 - 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. 
 - 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. 
- 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. 

Web API Design with Spring Boot Week 15 Coding Assignment

The screenshot displays an IDE window for a Spring Boot application. The top pane shows the `DefaultJeepSalesDao.java` file, which implements the `JeepSalesDao` interface. The code includes a `@Service` annotation, a `@Component` annotation, and a `@Autowired` dependency on `NamedParameterJdbcTemplate`. The `fetchJeeps` method is implemented to return a list of `Jeep` objects based on the provided `model` and `trim` parameters.

```
1 package com.promineotech.jeeo.dao;
2
3 import java.util.List;
4
5 @Service
6 @Component
7 @Slf4j
8 public class DefaultJeepSalesDao implements JeepSalesDao {
9
10     @Autowired
11     private NamedParameterJdbcTemplate jdbcTemplate;
12
13     @Override
14     public List<Jeep> fetchJeeps(JeepModel model, String trim) {
15         log.debug("DAO: model={}, trim={}", model, trim);
16
17         return null;
18     }
19 }
```

The bottom pane shows the console output, which includes the Spring Boot logo and the following log messages:

```
2023-01-05 11:53:26.387 INFO 13660 --- [ restartedMain] com.promineotech.jeeo.JeeoSales : Starting JeepSales using Java 17.0.5 on LAPTOP-C
2023-01-05 11:53:26.387 DEBUG 13660 --- [ restartedMain] com.promineotech.jeeo.JeeoSales : Running with Spring Boot v2.7.6, Spring v5.3.24
2023-01-05 11:53:26.388 INFO 13660 --- [ restartedMain] com.promineotech.jeeo.JeeoSales : No active profile set, falling back to 1 default
2023-01-05 11:53:29.211 INFO 13660 --- [ restartedMain] com.promineotech.jeeo.JeeoSales : Started JeepSales in 3.241 seconds (JVM running)
```

The bottom pane also shows the `DefaultJeepSalesController.java` file, which implements the `JeepSalesController` interface. The code includes a `@Service` annotation, a `@Component` annotation, and a `@Autowired` dependency on `DefaultJeepSalesDao`. The `fetchJeeps` method is implemented to return a list of `Jeep` objects based on the provided `model` and `trim` parameters.

```
1 package com.promineotech.jeeo.dao;
2
3 import java.math.BigDecimal;
4
5 @Service
6 @Component
7 @Slf4j
8 public class DefaultJeepSalesController implements JeepSalesController {
9
10     @Autowired
11     private DefaultJeepSalesDao jeepSalesDao;
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);
26         params.put("trim_level", trim);
27
28         return jdbcTemplate.query(sql, params, new RowMapper<>() {
29
30             @Override
31             public Jeep mapRow(ResultSet rs, int rowNum) throws SQLException {
32                 // @formatter:off
33                 return Jeep.builder()
34                     .basePrice(new BigDecimal(rs.getString("base_price")))
35                     .modelId(JeepModel.valueOf(rs.getString("model_id")))
36                     .modelPK(rs.getLong("trim_level"))
37                     .numDoors(rs.getInt("num_doors"))
38                     .trimLevel(rs.getString("trim_level"))
39                     .wheelSize(rs.getInt("wheel_size"))
40                     .build();
41                 // @formatter:on
42             }
43         });
44     }
45 }
```

Web API Design with Spring Boot Week 15 Coding Assignment

```
25
26
27 @Override
28 public List<Jeep> fetchJeeps(JeepModel model, String trim) {
29     log.debug("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
38     Map<String, Object> params = new HashMap<>();
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         // $formatter:off
46         public Jeep mapRow(ResultSet rs, int rowNum) throws SQLException {
47             return Jeep.builder()
48                 .basePrice(new BigDecimal(rs.getString("base_price")))
49                 .modelId(JeepModel.valueOf(rs.getString("model_id")))
50         }
51     });
52 }
```

2023-01-06 15:05:26.911 INFO 5608 --- [main] c.p.jeep.controller.FetchJeepTest : Starting FetchJeepTest using Java 17.0.5 on LAPTC
2023-01-06 15:05:26.914 DEBUG 5608 --- [main] c.p.jeep.controller.FetchJeepTest : Running with Spring Boot v2.7.6, Spring v5.3.24
2023-01-06 15:05:26.915 INFO 5608 --- [main] c.p.jeep.controller.FetchJeepTest : The following 1 profile is active: "test"
2023-01-06 15:05:31.084 INFO 5608 --- [main] c.p.jeep.controller.FetchJeepTest : Started FetchJeepTest in 4.609 seconds (JVM runni
2023-01-06 15:05:31.998 DEBUG 5608 --- [o-auto-1-exec-1] c.p.j.c.DefaultJeepSalesController : model=WRANGLER, trim=3port
2023-01-06 15:05:32.000 INFO 5608 --- [o-auto-1-exec-1] c.p.j.service.DefaultJeepSalesService : The fetchJeeps method was called with model=WRANG
2023-01-06 15:05:32.000 DEBUG 5608 --- [o-auto-1-exec-1] c.p.jeep.dao.DefaultJeepSalesDao : DAO: model=WRANGLER, trim=3port