

Web API Design with Spring Boot Week 3 Coding Assignment

Points possible: 70

Category	Criteria	% of Grade
Functionality	Does the code work?	25
Organization	Is the code clean and organized? Proper use of white space, syntax, and consistency are utilized. Names and comments are concise and clear.	25
Creativity	Student solved the problems presented in the assignment using creativity and out of the box thinking.	25
Completeness	All requirements of the assignment are complete.	25

Instructions: In Eclipse, or an IDE of your choice, write the code that accomplishes the objectives listed below. Ensure that the code compiles and runs as directed. Take screenshots of the code and of the running program (make sure to get screenshots of all required functionality) and paste them in this document where instructed below. Create a new repository on GitHub for this week's assignments and push this document, with your Java project code, to the repository. Add the URL for this week's repository to this document where instructed and submit this document to your instructor when complete.




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. 
 - 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. 

Screenshots of Code:

Screenshots of Running Application:

Q3.b

```
1 package com.promineotech.jeepp.dao;
2
3 import java.math.BigDecimal;
4
5
6
7
8 @Service
9 @Component
10 @Slf4j
11 public class DefaultJeepSalesDao implements JeepSalesDao {
12
13
14
15
16
17
18
19
20
21
22
23 @Autowired
24 private NamedParameterJdbcTemplate jdbcTemplate;
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 }
```

com.promineotech.jeepp.dao
DefaultJeepSalesDao
log : Logger
jdbcTemplate : NamedParameterJdbcTemplate
fetchJeeps(JeepModel, String) : List<Jeep>

Spring Boot (v2.5.5)

```
2021-10-02 04:38:03.039 INFO 17920 --- [main] c.p.jeepp.controller.FetchJeepTest : Starting FetchJeepTest using Java 16.0.2 on Lilith with
2021-10-02 04:38:03.040 DEBUG 17920 --- [main] c.p.jeepp.controller.FetchJeepTest : Running with Spring Boot v2.5.5, Spring v5.3.10
2021-10-02 04:38:03.040 INFO 17920 --- [main] c.p.jeepp.controller.FetchJeepTest : The following profiles are active: test
2021-10-02 04:38:05.940 INFO 17920 --- [main] c.p.jeepp.controller.FetchJeepTest : Started FetchJeepTest in 3.241 seconds (JVM running for
2021-10-02 04:38:06.603 DEBUG 17920 --- [o-auto-1-exec-1] c.p.j.c.BasicJeepSalesController : model=WRANGLER, trim=Sport
2021-10-02 04:38:06.604 INFO 17920 --- [o-auto-1-exec-1] c.p.j.service.DefaultJeepSalesService : The fetchJeeps method was called with model=WRANGLER and
2021-10-02 04:38:06.605 DEBUG 17920 --- [o-auto-1-exec-1] c.p.jeepp.dao.DefaultJeepSalesDao : DAO: model=WRANGLER, trim=Sport
```

Q3.f

```
JeepSalesDao.java *DefaultJeepSalesDao.java FetchJeepTest.java FetchJeepTestSupport.java Jeep.java
17
18 @Service
19 @Component
20 @Slf4j
21 public class DefaultJeepSalesDao implements JeepSalesDao {
22
23     @Autowired
24     private NamedParameterJdbcTemplate jdbcTemplate;
25
26     @Override
27     public List<Jeep> fetchJeeps(JeepModel model, String trim) {
28         log.debug("DAO: model={}, trim={}", model, trim);
29
30         // @formatter:off
31         String sql = ""
32             + "SELECT * "
33             + "FROM models "
34             + "WHERE model_id = :model_id AND trim_level = :trim_level";
35         // @formatter:on
36
37         Map<String, Object> params = new HashMap<>();
38         params.put("model_id", model.toString());
39         params.put("trim_level", trim);
40
41         return jdbcTemplate.query(sql, params,
42             new RowMapper<>() {
43
44             @Override
45             public Jeep mapRow(ResultSet rs, int rowNum) throws SQLException {
46                 // @formatter:off
47                 return Jeep.builder()
48                     .basePrice(new BigDecimal(rs.getString("base_price")))
49                     .modelId(JeepModel.valueOf(rs.getString("model_id")))
50                     .modelPK(rs.getLong("model_pk"))
51                     .numDoors(rs.getInt("num_doors"))
52                     .trimLevel(rs.getString("trim_level"))
53                     .wheelSize(rs.getInt("wheel_size"))
54                     .build();
55                 // @formatter:on
56             }
57         });
58     }
59 }
```

Q5

The screenshot displays an IDE interface. On the left, the Package Explorer shows the project structure. The JUnit console at the top indicates a successful test run: 'Finished after 4,596 seconds', 'Runs: 1/1', 'Errors: 0', and 'Failures: 0'. Below this, the 'FetchJeepTest' is listed with a duration of '0.811 s'. The Failure Trace panel is empty. At the bottom, the Boot Dashboard shows the 'local' environment. The main editor on the right shows the source code for 'JeepSalesDao.java'. The code defines a 'Jeep' class that implements 'Comparable<Jeep>'. It includes imports for 'BigDecimal', 'Comparator', 'JsonIgnore', 'AllArgsConstructor', 'Builder', 'Data', and 'NoArgsConstructor'. The class has private fields for 'modelPK', 'modelId', 'trimLevel', 'numDoors', 'wheelSize', and 'basePrice'. It includes a 'getModelPK()' method annotated with '@JsonIgnore' and a 'compareTo()' method annotated with '@Override' that uses a 'Comparator' to compare the objects based on 'modelId', 'trimLevel', and 'numDoors'.

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

URL to GitHub Repository:

<https://github.com/allobrandt/Spring-Boot-Wk3>