


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.

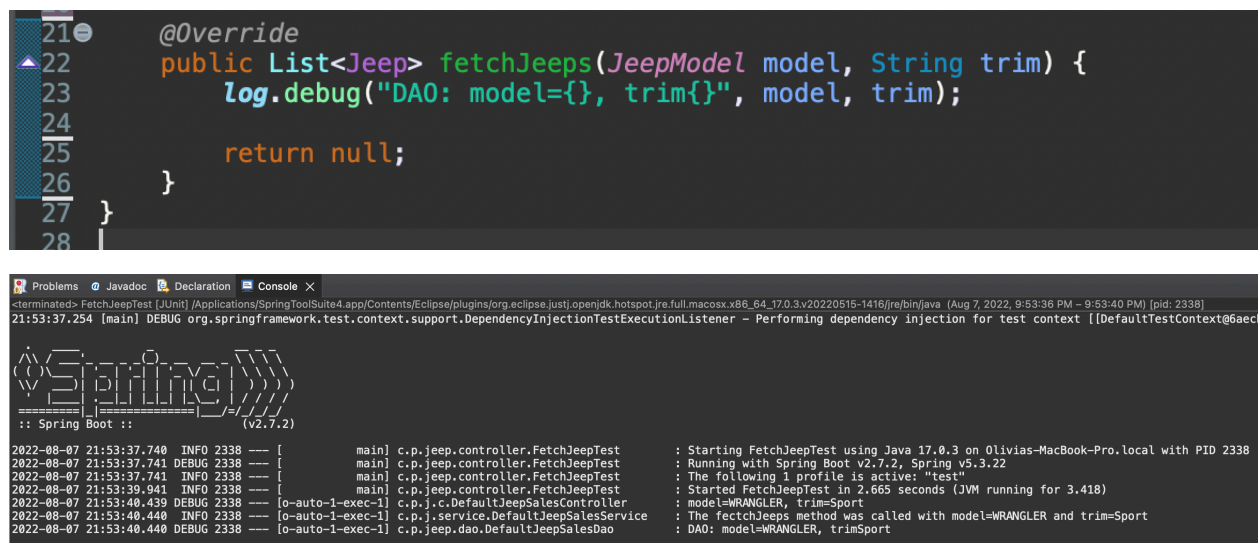
Project Resources: <https://github.com/promineotech/Spring-Boot-Course-Student-Resources>

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.



The screenshot shows an IDE with two panels. The top panel displays a Java class with the following code:

```
21 @Override
22 public List<Jeep> fetchJeeps(JeepModel model, String trim) {
23     log.debug("DAO: model={}, trim={}", model, trim);
24
25     return null;
26 }
27 }
28
```

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


```
2022-08-07 21:53:37.740 INFO 2338 --- [main] c.p.jeep.controller.FetchJeepTest : Starting FetchJeepTest using Java 17.0.3 on Olivias-MacBook-Pro.local with PID 2338
2022-08-07 21:53:37.741 INFO 2338 --- [main] c.p.jeep.controller.FetchJeepTest : Running with Spring Boot v2.7.2, Spring v5.3.22
2022-08-07 21:53:39.941 INFO 2338 --- [main] c.p.jeep.controller.FetchJeepTest : The following 1 profile is active: "test"
2022-08-07 21:53:40.439 DEBUG 2338 --- [o-auto-1-exec-1] c.p.j.c.DefaultJeepSalesController : Started FetchJeepTest in 2.665 seconds (JVM running for 3.418)
2022-08-07 21:53:40.440 INFO 2338 --- [o-auto-1-exec-1] c.p.j.service.DefaultJeepSalesService : model=WRANGLER, trim=Sport
2022-08-07 21:53:40.440 DEBUG 2338 --- [o-auto-1-exec-1] c.p.jeep.dao.DefaultJeepSalesDao : The fetchJeeps method was called with model=WRANGLER and trim=Sport
2022-08-07 21:53:40.440 DEBUG 2338 --- [o-auto-1-exec-1] c.p.jeep.dao.DefaultJeepSalesDao : DAO: model=WRANGLER, trim=Sport
```

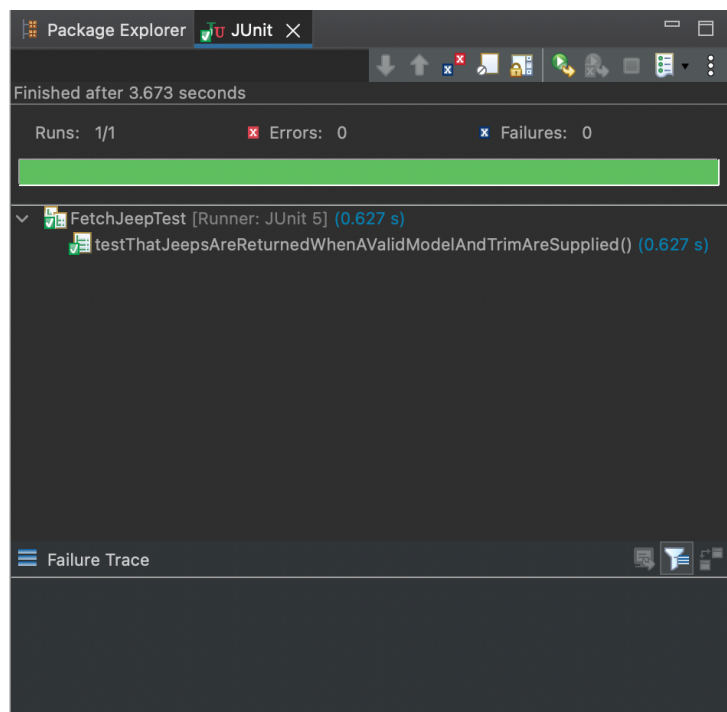
- c) In DefaultJeepSalesDao, inject an instance variable of type NamedParameterJdbcTemplate.
- d) Write SQL to return a list of Jeep models based on the parameters: model and trim. Be sure to utilize the SQL Injection prevention mechanism of the NamedParameterJdbcTemplate using :model_id and :trim_level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

```

21 @Service
22 @Slf4j
23 public class DefaultJeepSalesDao implements JeepSalesDao {
24
25     @Autowired
26     private NamedParameterJdbcTemplate jdbcTemplate;
27
28     @Override
29     public List<Jeep> fetchJeeps(JeepModel model, String trim) {
30         log.debug("DAO: model={}, trim={}", model, trim);
31
32         // @formatter:off
33         String sql = ""
34             + "SELECT * "
35             + "FROM models "
36             + "WHERE model_id = :model_id AND trim_level = :trim_level";
37         // @formatter:on
38
39         Map<String, Object> params = new HashMap<>();
40         params.put("model_id", model.toString());
41         params.put("trim_level", trim);
42
43         return jdbcTemplate.query(sql, params, new RowMapper<>() {
44
45             @Override
46             public Jeep mapRow(ResultSet rs, int rowNum) throws SQLException {
47                 // @formatter:off
48                 return Jeep.builder()
49                     .basePrice(new BigDecimal(rs.getString("base_price")))
50                     .modelId(JeepModel.valueOf(rs.getString("model_id")))
51                     .modelPK(rs.getLong("model_pk"))
52                     .numDoors(rs.getInt("num_doors"))
53                     .trimLevel(rs.getString("trim_level"))
54                     .wheelSize(rs.getInt("wheel_size"))
55                     .build();
56                 // @formatter:on
57             }
58         });
59     }
60 }

```

- 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:

```
DefaultJeepSalesService.java  JeepSalesDao.java X  DefaultJeepSalesDao.java
1 package com.promineotech.jeep.dao;
2
3 import java.util.List;
4
5
6
7
8 public interface JeepSalesDao {
9
10     List<Jeep> fetchJeeps(JeepModel model, String trim);
11
12 }
13
```

```
DefaultJeepSalesService.java  JeepSalesDao.java  DefaultJeepSalesDao.java X  FetchJeepTr
2
3 import java.math.BigDecimal;
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21 @Service
22 @Slf4j
23 public class DefaultJeepSalesDao implements JeepSalesDao {
24
25     @Autowired
26     private NamedParameterJdbcTemplate jdbcTemplate;
27
28     @Override
29     public List<Jeep> fetchJeeps(JeepModel model, String trim) {
30         log.debug("DAO: model={}, trim{}", model, trim);
31
32         // @formatter:off
33         String sql = "
34             + "SELECT * "
35             + "FROM models "
36             + "WHERE model_id = :model_id AND trim_level = :trim_level";
37         // @formatter:on
38
39         Map<String, Object> params = new HashMap<>();
40         params.put("model_id", model.toString());
41         params.put("trim_level", trim);
42
43         return jdbcTemplate.query(sql, params, new RowMapper<>() {
44
45             @Override
46             public Jeep mapRow(ResultSet rs, int rowNum) throws SQLException {
47                 // @formatter:off
48                 return Jeep.builder()
49                     .basePrice(new BigDecimal(rs.getString("base_price")))
50                     .modelId(JeepModel.valueOf(rs.getString("model_id")))
51                     .modelPK(rs.getLong("model_pk"))
52                     .numDoors(rs.getInt("num_doors"))
53                     .trimLevel(rs.getString("trim_level"))
54                     .wheelSize(rs.getInt("wheel_size"))
55                     .build();
56                 // @formatter:on
57             }
58         });
59     }
60 }
```

```
JeepSalesDao.java  DefaultJeepSalesDao.java  Jeep.java X
1 package com.promineotech.jeep.entity;
2
3+ import java.math.BigDecimal;
11
12 @Data
13 @Builder
14 @NoArgsConstructor
15 @AllArgsConstructor
16 public class Jeep {
17     private Long modelPK;
18     private JeepModel modelId;
19     private String trimLevel;
20     private int numDoors;
21     private int wheelSize;
22     private BigDecimal basePrice;
23
24- @JsonIgnore
25     public Long getModelPK() {
26         return modelPK;
27     }
28 }
29
```

Screenshots of Running Application:

```
16:41:14.593 [Thread-8] DEBUG org.springframework.boot.devtools.restart.classloader.RestartClassLoader - Created RestartClassLoader org.springframework.boot.devtools.restart.classloader.RestartClassLoader@3a6bdc4

Spring
=====
:: Spring Boot ::
(v2.7.2)

2022-08-12 16:41:14.845 INFO 1555 --- [ restartedMain] com.promineotech.jeep.JeepSales : Starting JeepSales using Java 17.0.3 on Olivia's-MacBook-Pro.local with PID 1555 (/Users/oliviaguzman/Promineo/SpringToolSuite/jeep-sales/target/cl
2022-08-12 16:41:14.846 DEBUG 1555 --- [ restartedMain] com.promineotech.jeep.JeepSales : Running with Spring Boot v2.7.2, Spring v5.3.22
2022-08-12 16:41:14.846 INFO 1555 --- [ restartedMain] com.promineotech.jeep.JeepSales : No active profile set, falling back to 1 default profile: "default"
2022-08-12 16:41:16.513 INFO 1555 --- [ restartedMain] com.promineotech.jeep.JeepSales : Started JeepSales in 1.913 seconds (JVM running for 2.385)
```

Jeep Sales Service

V3/api-docs

Servers

http://localhost:8080 - Local server. ▾

default-jeep-sales-controller

GET /jeeps Returns a list of Jeeps

Returns a list of Jeeps given an optional model and/or trim

Parameters

Cancel

Name	Description
model string (query)	The model name (i.e., 'WRANGLER')
trim string (query)	The trim level (i.e., 'Sport')

Execute

Clear

Responses

Curl

```
curl -X 'GET' \
  'http://localhost:8080/jeeps?model=WRANGLER&trim=Sport' \
  -H 'accept: application/json'
```

Request URL

http://localhost:8080/jeeps?model=WRANGLER&trim=Sport

Server response

Code Details

200

Response body

```
[
  {
    "modelId": "WRANGLER",
    "trimLevel": "Sport",
    "numDoors": 2,
    "wheelSize": 17,
    "basePrice": 28475
  },
  {
    "modelId": "WRANGLER",
    "trimLevel": "Sport",
    "numDoors": 4,
    "wheelSize": 17,
    "basePrice": 31975
  }
]
```

Response headers

```
connection: keep-alive
content-type: application/json
date: Fri, 12 Aug 2022 21:44:42 GMT
keep-alive: timeout=60
transfer-encoding: Identity
```

Responses

Code Description

Links

200	<p>A list of Jeeps is returned.</p> <p>Media type application/json</p> <p>Controls Accept header.</p> <p>Example Value Schema</p> <pre>{ "modelId": "WRANGLER", "trimLevel": "string", "numDoors": 0, "wheelSize": 0, "basePrice": 0 }</pre>	No links
400	<p>The request parameters are invalid.</p> <p>Media type application/json</p>	No links
404	<p>No Jeeps were found with the input criteria.</p> <p>Media type application/json</p>	No links
500	<p>An unplanned error occurred.</p> <p>Media type application/json</p>	No links

Package ExplorerJUnit X

Finished after 4.364 seconds

Runs: 1/1Errors: 0Failures: 0

FetchJeepTest [Runner: JUnit 5] (0.726 s)

Failure Trace

```
<terminated> FetchJeepTest [JUnit] [Applications/SpringToolSuite4.app/Contents/Eclipse/plugins/org.eclipse.justi.openjdk.hotspot.jre.full.macosx.x86_64.17.0.3.v20220516-1416/jre/bin/java (Aug 12, 2022, 4:39:31 PM - 4:39:37 PM) [pid: 1517]
16:39:33.649 [main] DEBUG org.springframework.test.context.support.DependencyInjectionTestExecutionListener - Performing dependency injection for test context [[DefaultTestContext@6a6cb8d testClass = FetchJeepTest, testInstance = com.promineo.tec

(V)Spring(V))
:: Spring Boot :: (v2.7.2)

2022-08-12 16:39:34.162 INFO 1517 --- [main] c.p.jeeo.controller.FetchJeepTest : Starting FetchJeepTest using Java 17.0.3 on Olivias-MacBook-Pro.local with PID 1517 (started by oliviaguzman in /Users/oliviaguzman/Promineo/Spr
2022-08-12 16:39:34.162 DEBUG 1517 --- [main] c.p.jeeo.controller.FetchJeepTest : Running with Spring Boot v2.7.2, Spring v5.3.22
2022-08-12 16:39:34.162 INFO 1517 --- [main] c.p.jeeo.controller.FetchJeepTest : The following 1 profile is active: "test"
2022-08-12 16:39:36.818 INFO 1517 --- [main] c.p.jeeo.controller.FetchJeepTest : Started FetchJeepTest in 3.144 seconds (JVM running for 4.033)
2022-08-12 16:39:37.438 DEBUG 1517 --- [o-auto-1-exec-1] c.p.j.c.DefaultJeepSalesController : model=WRANGLER, trim=Sport
2022-08-12 16:39:37.431 INFO 1517 --- [o-auto-1-exec-1] c.p.j.service.DefaultJeepSalesService : The fetchJeeps method was called with model=WRANGLER and trim=Sport
2022-08-12 16:39:37.431 DEBUG 1517 --- [o-auto-1-exec-1] c.p.jeeo.dao.DefaultJeepSalesDao : DAO: model=WRANGLER, trim=Sport
```


URL to GitHub Repository:

<https://github.com/OliGuzman/Spring-Boot-Web-API-Design>