

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.jeep.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 the Eclipse-based Spring Tool Suite 4 IDE. On the left, the Java editor displays the `FetchJeepTest.java` file, which contains JUnit test cases for the `DefaultJeepSalesService` class. The test cases verify that error messages are returned when invalid trim values are supplied. The right side of the interface includes an Outline view showing the project structure and a terminal window at the bottom displaying the execution of the test and its output.

```

workspace-spring-tool-suite-4-4.13.0.RELEASE - jeep-sales/src/test/java/com/prominotech/jeep/controller/FetchJeepTest.java - Spring Tool Suite 4

FetchJeepTest.java
DefaultJeepSalesService.java

seconds
Errors: 0 Failures: 1

[Runner: JUnit 5] [1.137 s]
AssertionFailedError: 
    at org.junit.Assert.fail(Assert.java:80)
    at org.junit.Assert.assertTrue(Assert.java:43)
    at org.junit.Assert.assertNull(Assert.java:192)
    at org.junit.Assert.assertNotNull(Assert.java:748)
    at org.junit.Assert.assertEquals(Assert.java:118)
    at com.prominotech.jeep.controller.FetchJeepTest.testThatAnInvalidTrimIsSupplied(FetchJeepTest.java:54)
    at sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method)
    at sun.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.java:62)
    at sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:43)
    at java.lang.reflect.Method.invoke(Method.java:494)
    at org.junit.runners.model.FrameworkMethod$1.runReflectiveCall(FrameworkMethod.java:50)
    at org.junit.internal.runners.model.ReflectiveCallable.run(ReflectiveCallable.java:12)
    at org.junit.runners.model.FrameworkMethod.invokeExplosively(FrameworkMethod.java:47)
    at org.junit.internal.runners.statements.InvokeMethod.evaluate(InvokeMethod.java:17)
    at org.junit.runners.ParentRunner$3.evaluate(ParentRunner.java:284)
    at org.junit.runners.ParentRunner$1.evaluate(ParentRunner.java:54)
    at org.junit.runners.ParentRunner$2.evaluate(ParentRunner.java:264)
    at org.junit.runners.ParentRunner.run(ParentRunner.java:364)
    at org.eclipse.jdt.internal.junit4.runner.JUnit4TestReference.run(JUnit4TestReference.java:86)
    at org.eclipse.jdt.internal.junit.runner.TestExecution.run(TestExecution.java:38)
    at org.eclipse.jdt.internal.junit.runner.RemoteTestRunner.main(RemoteTestRunner.java:197)

DefaultJeepSalesService.java

scripts = {'classpath:/tyaway/migrations/V1.0__Jeep_Schema.sql',
          'classpath:/tyaway/migrations/V1.1__Jeep_Data.sql'},
config = @SqlConfig(encoding = "utf-8")
class FetchJeepTest extends FetchJeepTestSupport {
    @Test
    void testThatAnErrorMessageIsReturnedWhenAnInvalidTrimIsSupplied() {
        // Given: a valid model, trim and URL
        JeepModel model = JeepModel.WRANGLER;
        String trim = "Invalid Value";
        String uri =
            String.format("%s?model=%s&trim=%s", getBaseUrl(), model, trim);

        // When: a connection is made to the URL
        ResponseEntity<ResponseEntity<List<Jeep>>> response = getRestTemplate().exchange(uri, HttpMethod.GET, null,
            new ParameterizedTypeReference<>() {});

        // Then: a not found (404) status code is returned
        assertEquals(response.getStatusCode(), HttpStatus.NOT_FOUND);
        // And: an error message is returned
    }
}

Console
 FetchJeepTest.testThatAnErrorMessageIsReturnedWhenAnInvalidTrimIsSupplied [JUnit] /Applications/SpringToolSuite4.app/Contents/Eclipse/plugins/org.eclipse.jdt.core/lib/junit.jar
:: Spring Boot ::
(v2.6.2)

2022-01-11 16:55:15.242 INFO 2973 --- [           main] c.p.jeep.controller.FetchJeepTest      : Starting FetchJeepTest using [           main]
2022-01-11 16:55:15.243 DEBUG 2973 --- [           main] c.p.jeep.controller.FetchJeepTest      : Running with Spring Boot v2.6.2
2022-01-11 16:55:15.243 INFO 2973 --- [           main] c.p.jeep.controller.FetchJeepTest      : The following profiles are active: []
2022-01-11 16:55:21.810 INFO 2973 --- [           main] c.p.jeep.controller.FetchJeepTest      : Started FetchJeepTest in 7.476 ms
2022-01-11 16:55:22.811 DEBUG 2973 --- [o-auto-1-exec-1] c.p.j.c.BasicJeepSalesController   : model=WRANGLER, trim=Invalid Value
2022-01-11 16:55:22.845 INFO 2973 --- [o-auto-1-exec-1] c.p.j.service.DefaultJeepSalesService: The fetchJeeps method was called
2022-01-11 16:55:22.845 DEBUG 2973 --- [o-auto-1-exec-1] c.p.j.dao.DefaultJeepSalesDao       : DAO: model=WRANGLER, trim=Invalid Value

```

Caption

The screenshot shows the Spring Tool Suite 4 IDE interface. The central area displays Java code for a DAO class:

```

29     public List<Jeep> fetchJeeps(JeepModel model, String trim) {
30         log.debug("DAO: model={}", model);
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,
43             new RowMapper<Jeep>() {
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     }

```

The JUnit tab shows 0 errors and 0 failures. The Outline tab shows the project structure. The Console tab shows Spring Boot startup logs:

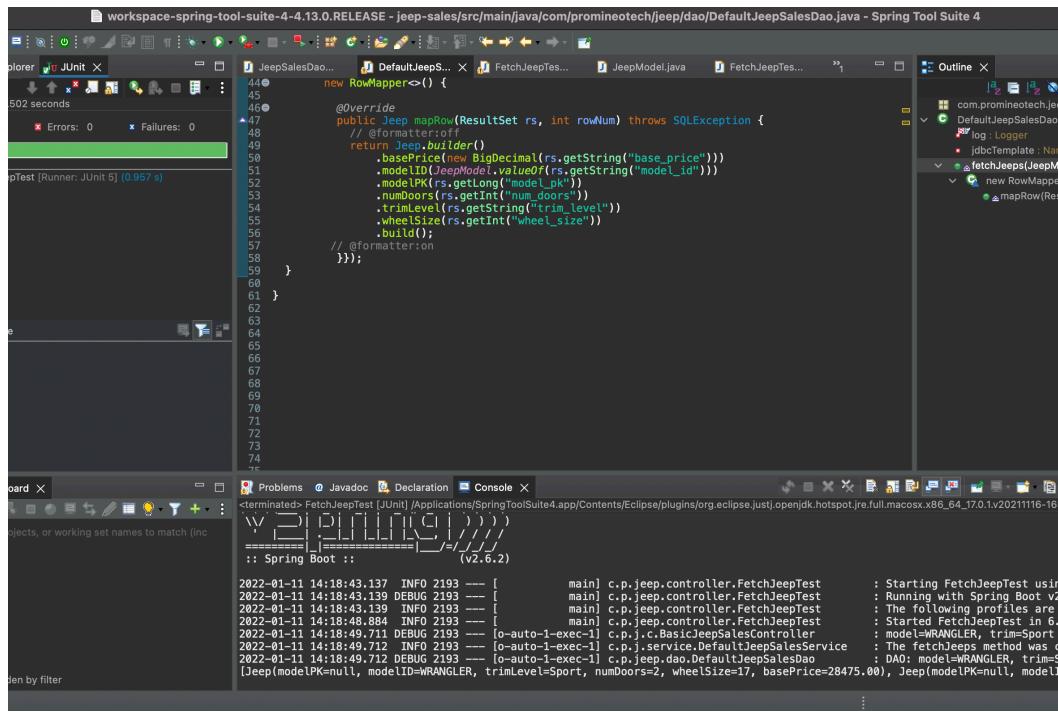
```

2022-01-11 15:34:46.476 INFO 2478 --- [           main] c.p.jeep.controller.FetchJeepTest      : Starting FetchJeepTest us
2022-01-11 15:34:46.477 DEBUG 2478 --- [           main] c.p.jeep.controller.FetchJeepTest      : Running with Spring Boot
2022-01-11 15:34:46.478 INFO 2478 --- [           main] c.p.jeep.controller.FetchJeepTest      : The following profiles are
2022-01-11 15:34:46.478 INFO 2478 --- [           main] c.p.jeep.controller.FetchJeepTest      : Started FetchJeepTest in
2022-01-11 15:34:53.392 DEBUG 2478 --- [o-auto-1-exec-1] c.p.j.c.BasicJeepSalesController   : modelId=RAMONE_FRTIM=SNR
2022-01-11 15:34:53.393 INFO 2478 --- [o-auto-1-exec-1] c.p.j.service
2022-01-11 15:34:53.393 DEBUG 2478 --- [o-auto-1-exec-1] c.p.jeep.dao

```

Caption

- 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.



Caption

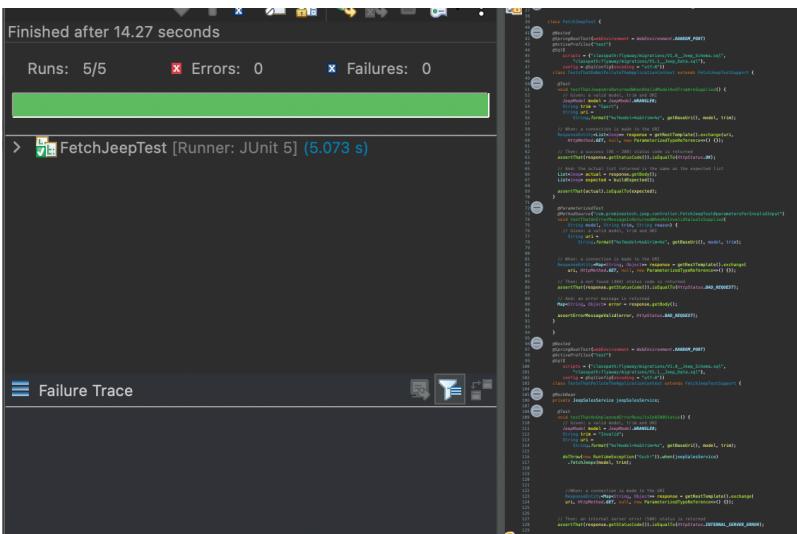
- e) Add the parameters to a parameter map as shown in the video. Don't forget to convert the `JeepModel` enum value to a String (i.e., `params.put("model_id", model.toString());`)
- f) Call the `query` method on the `NamedParameterJdbcTemplate` instance variable to return a list of `Jeep` model objects. 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.

The screenshot shows the IntelliJ IDEA interface with several tabs open. The left sidebar lists project files under 'com.promineotech' and 'rc/main/java'. The right pane displays the Java code for the `DefaultJeepSalesDao` class.

```
1 package com.promineotech.jeep.dao;
2
3 import java.math.BigDecimal;
4
5 @Component
6 @Slf4j
7 public class DefaultJeepSalesDao implements JeepSalesDao {
8
9     @Autowired
10    private NamedParameterJdbcTemplate jdbcTemplate;
11
12    @Override
13    public List<Jeep> fetchJeeps(JeepModel model, String trim) {
14        log.debug("DAO: model={}", model, trim);
15
16        // @formatter:off
17        String sql = """
18            + "SELECT * "
19            + "FROM models "
20            + "WHERE model_id = :model_id AND trim_level = :trim_level";
21        // @formatter:on
22
23        Map<String, Object> params = new HashMap<>();
24        params.put("model_id", model.toString());
25        params.put("trim_level", trim);
26
27        return jdbcTemplate.query(sql, params,
28            new RowMapper<Jeep>() {
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("model_pk"))
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    }
46
47 }
```

F

- 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. 



Caption

Screenshots of Code:

Screenshots of Running Application:

URL to GitHub Repository:

`git@github.com:bgoetz22/SpringBootWeek3.git`