


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>

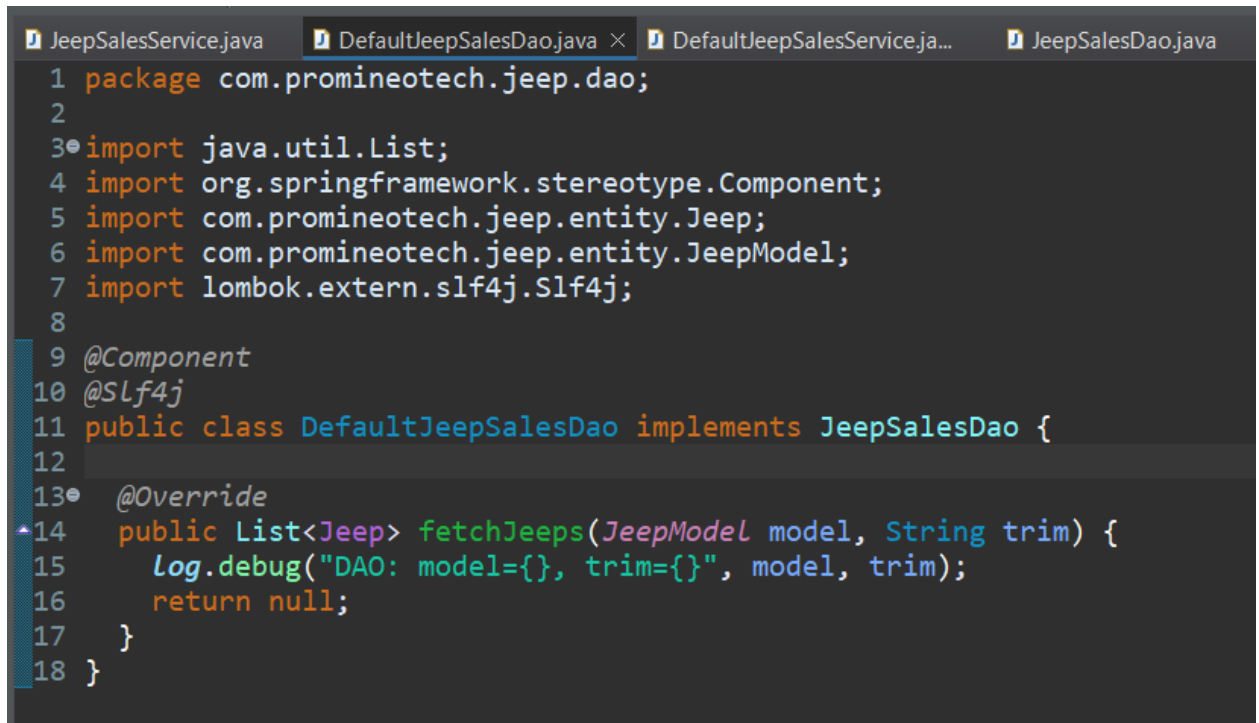
Coding Steps:

- 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. 



```
1 package com.promineotech.jeep.dao;
2
3 import java.util.List;
4 import org.springframework.stereotype.Component;
5 import com.promineotech.jeep.entity.Jeep;
6 import com.promineotech.jeep.entity.JeepModel;
7 import lombok.extern.slf4j.Slf4j;
8
9 @Component
10 @Slf4j
11 public class DefaultJeepSalesDao implements JeepSalesDao {
12
13     @Override
14     public List<Jeep> fetchJeeps(JeepModel model, String trim) {
15         log.debug("DAO: model={}, trim={}", model, trim);
16         return null;
17     }
18 }
```

The screenshot shows the Eclipse IDE with the following components:

- JUnit Runner:** Shows a test failure for `FetchJeepTest`. The failure is an `org.opentest4j.AssertionFailedError` with the message: "expected: [Jeep(modelPK=null, modelId=WRANGLER, trimLevel=Sport, numDoc=1), Jeep(modelPK=null, modelId=WRANGLER, trimLevel=Sport, numDoc=2)] but was: null".
- Source Editor:** Displays the `DefaultJeepSalesDao` class. The `fetchJeeps` method is currently implemented to return `null`.


```

1 package com.promineotech.jeeo.dao;
2
3 import java.util.List;
4 import org.springframework.stereotype.Component;
5 import com.promineotech.jeeo.entity.Jeeo;
6 import com.promineotech.jeeo.entity.JeeoModel;
7 import lombok.extern.slf4j.Slf4j;
8
9 @Component
10 @Slf4j
11 public class DefaultJeepSalesDao implements JeepSalesDao {
12
13     @Override
14     public List<Jeep> fetchJeeps(JeepModel model, String trim) {
15         log.debug("DAO: model={}, trim={}", model, trim);
16         return null;
17     }
18 }

```
- Outline:** Shows the project structure with `com.promineotech.jeeo.dao` and `DefaultJeepSalesDao`.
- Console:** Shows the output of the test run, including the failure message and the stack trace.


The screenshot shows the Eclipse IDE console with the following output:

```

<terminated> FetchJeepTest [JUnit] C:\Users\j\cur\Downloads\tools\STS\sts-4.15.1.RELEASE\plugins\org.eclipse.justi.openjdk.hotspot.jre.full.win32.x86_64.17.0.3.v20220515-1416\jre\bin\java.exe
(v2.7.2)

35 INFO 5624 --- [main] c.p.jeeo.controller.FetchJeepTest : Starting FetchJeepTest using Java 17.0.3 on DESKTOP-VIUARQV with PID 5624
41 INFO 5624 --- [main] c.p.jeeo.controller.FetchJeepTest : Running with Spring Boot v2.7.2, Spring v5.3.22
43 INFO 5624 --- [main] c.p.jeeo.controller.FetchJeepTest : The following 1 profile is active: "test"
56 INFO 5624 --- [main] c.p.jeeo.controller.FetchJeepTest : Started FetchJeepTest in 7.234 seconds (JVM running for 9.285)
62 DEBUG 5624 --- [o-auto-1-exec-1] c.p.j.c.DefaultJeepSalesController : model = WRANGLER, trim = Sport
65 INFO 5624 --- [o-auto-1-exec-1] c.p.j.service.DefaultJeepSalesService : the fetchJeeps method was called with model=WRANGLER and trim=Sport
66 DEBUG 5624 --- [o-auto-1-exec-1] c.p.jeeo.dao.DefaultJeepSalesDao : DAO: model=WRANGLER, trim=Sport

```


- In `DefaultJeepSalesDao`, inject an instance variable of type `NamedParameterJdbcTemplate`.
- 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.
- 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());`).
- 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. 

```
JeepSalesService.java  DefaultJeepSalesDao.java x DefaultJeepSalesService.java  JeepSalesDao.java  Jeep.java
1 package com.promineotech.jeep.dao;
2
3 import java.math.BigDecimal;
4 import java.sql.ResultSet;
5 import java.sql.SQLException;
6 import java.util.HashMap;
7 import java.util.List;
8 import java.util.Map;
9 import org.springframework.beans.factory.annotation.Autowired;
10 import org.springframework.jdbc.core.RowMapper;
11 import org.springframework.jdbc.core.namedparam.NamedParameterJdbcTemplate;
12 import org.springframework.stereotype.Component;
13 import com.promineotech.jeep.entity.Jeep;
14 import com.promineotech.jeep.entity.JeepModel;
15 import lombok.extern.slf4j.Slf4j;
16
17 @Component
18 @Slf4j
19 public class DefaultJeepSalesDao implements JeepSalesDao {
20     @Autowired
21     private NamedParameterJdbcTemplate jdbcTemplate;
22
23
24     @Override
25     public List<Jeep> fetchJeeps(JeepModel model, String trim) {
26         log.debug("DAO: model={}, trim={}", model, trim);
27
28         // @formatter:off
29         String sql = ""
30             + "SELECT * "
31             + "FROM models "
32             + "WHERE model_id = :model_id AND trim_level = :trim_level";
33         // @formatter:on
```

```

34     Map<String, Object> params = new HashMap<>();
35     params.put("model_id", model.toString());
36     params.put("trim_level", trim);
37
38
39
40     return jdbcTemplate.query(sql, params,
41         new RowMapper<>() {
42             @Override
43             public Jeep mapRow(ResultSet rs, int rowNum)
44                 throws SQLException{
45                 //Formatter: off
46                 return Jeep.builder()
47                     .basePrice(new BigDecimal(rs.getString("base_price")))
48                     .modelId(JeepModel.valueOf(rs.getString("model_id")))
49                     .modelPK(rs.getLong("model_pk"))
50                     .numDoors(rs.getInt("num_doors"))
51                     .trimLevel(rs.getString("trim_level"))
52                     .wheelSize(rs.getInt("wheel_size"))
53                     .build();
54                 // @formatter:on
55             }
56         });
57 }
58

```

- 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. 

DefaultJeepSalesDao.java

FetchJeepTest.java ×

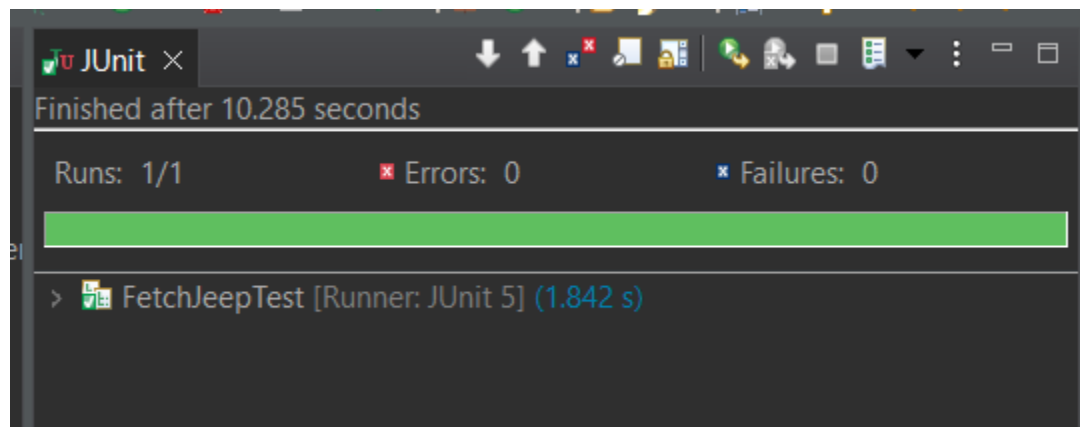
```
1 package com.promineotech.jeep.controller;
2 import static org.assertj.core.api.Assertions.assertThat;
3 import java.math.BigDecimal;
4 import java.util.LinkedList;
5 import java.util.List;
6 import org.junit.jupiter.api.Test;
7 import org.springframework.beans.factory.annotation.Autowired;
8 import org.springframework.boot.test.context.SpringBootTest;
9 import org.springframework.boot.test.context.SpringBootTest.WebEnvironment;
10 import org.springframework.boot.test.web.client.TestRestTemplate;
11 import org.springframework.boot.test.web.server.LocalServerPort;
12 import org.springframework.core.ParameterizedTypeReference;
13 import org.springframework.http.HttpMethod;
14 import org.springframework.http.HttpStatus;
15 import org.springframework.http.ResponseEntity;
16 import org.springframework.test.context.ActiveProfiles;
17 import org.springframework.test.context.jdbc.Sql;
18 import org.springframework.test.context.jdbc.SqlConfig;
19 import com.promineotech.jeep.entity.Jeep;
20 import com.promineotech.jeep.entity.JeepModel;
21 import lombok.Getter;
22
23 @SpringBootTest(webEnvironment = WebEnvironment.RANDOM_PORT)
24 @ActiveProfiles("test")
25 @Sql(
26     scripts = {"classpath:flyway/migrations/v1.0__Jeep_Schema.sql",
27         "classpath:flyway/migrations/v1.1__Jeep_Data.sql" },
28     config = @SqlConfig(encoding = "utf-8"))
29 class FetchJeepTest {
30     @Autowired
31     @Getter
32     private TestRestTemplate restTemplate;
33
34     @LocalServerPort
35     private int serverPort;
36
```

DefaultJeepSalesDao.java

FetchJeepTest.java ×

```
37● @Test
38 void testThatJeepsAreReturnedWhenAValidModelAndTrimAreSupplied() {
39     // Given: a valid model, trim and URI
40     JeepModel model = JeepModel.WRANGLER;
41     String trim = "Sport";
42     String uri = String.format("http://localhost:%d/jeeps?model=%s&trim=%s",
43         serverPort, model, trim);
44
45     // When: a connection is made to the URI
46
47
48     // Then: a success (OK - 200) status code is returned
49     ResponseEntity<List<Jeep>> response =
50         getRestTemplate().exchange(uri, HttpMethod.GET, null,
51             new ParameterizedTypeReference<>() {});
52
53     assertThat(response.getStatusCode()).isEqualTo(HttpStatus.OK);
54     // And the actual list returned is the same as the expected list
55     List<Jeep> expected = buildExpected();
56     assertThat(response.getBody()).isEqualTo(expected);
57 }
58
59● protected List<Jeep> buildExpected() {
60     List<Jeep> list = new LinkedList<>();
61     // @formatter:off
62     list.add(Jeep.builder()
63         .modelId(JeepModel.WRANGLER)
64         .trimLevel("Sport")
65         .numDoors(2)
66         .wheelSize(17)
67         .basePrice(new BigDecimal("28475.00"))
68         .build());
69 }
```

```
69
70     list.add(Jeep.builder()
71         .modelId(JeepModel.WRANGLER)
72         .trimLevel("Sport")
73         .numDoors(4)
74         .wheelSize(17)
75         .basePrice(new BigDecimal("31975.00"))
76         .build());
77
78
79     Jeep j1 = new Jeep();
80     //formatter:on
81
82     return list;
83 }
84 }
```



Screenshots of Code:

Screenshots of Running Application:

URL to GitHub Repository:

<https://github.com/juliecurran3/Week-3-Spring-Boot-In-Progress.git>