


# 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.

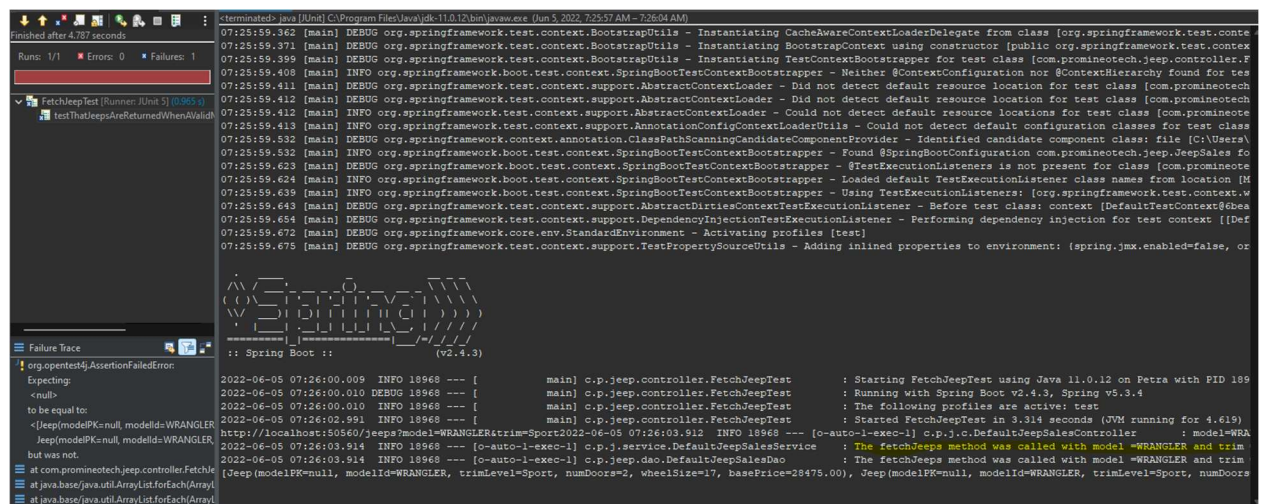
**Project Resources:** <https://github.com/promineotech/Spring-Boot-Course-Student-Resources>

## 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:

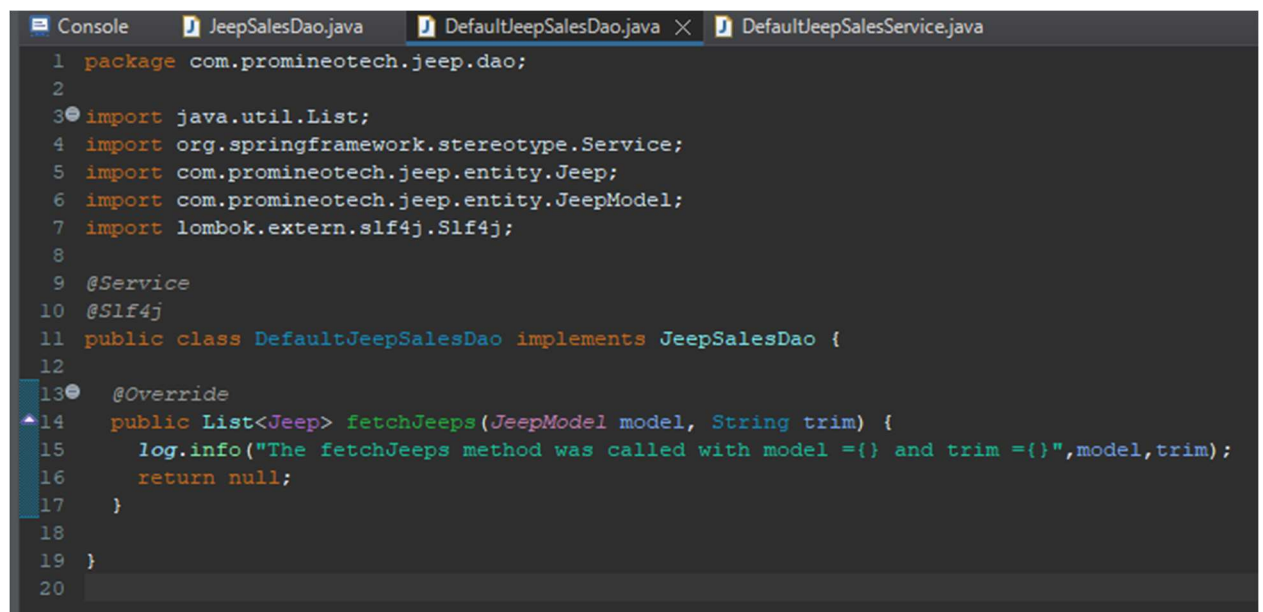
```
List<Jeep> fetchJeeps(JeepModel model, String trim);
```

- 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.



```
Finished after 4.787 seconds
Runs: 1/1 | Errors: 0 | Failures: 1
FetchJeepTest (Runner JUnit 5) [0.985 s]
testThatJeepsAreReturnedWhenValid

2022-06-05 07:26:00.009 INFO 18968 --- [main] c.p.jeep.controller.FetchJeepTest : Starting FetchJeepTest using Java 11.0.12 on Petra with PID 189
2022-06-05 07:26:00.010 DEBUG 18968 --- [main] c.p.jeep.controller.FetchJeepTest : Running with Spring Boot v2.4.3, Spring v5.3.4
2022-06-05 07:26:00.010 INFO 18968 --- [main] c.p.jeep.controller.FetchJeepTest : The following profiles are active: test
2022-06-05 07:26:02.991 INFO 18968 --- [main] c.p.jeep.controller.FetchJeepTest : Started FetchJeepTest in 3.314 seconds (JVM running for 4.619)
2022-06-05 07:26:03.910 INFO 18968 --- [o-auto-1-exec-1] c.p.jeep.service.DefaultJeepSalesService : The fetchJeeps method was called with model={\"modelId\":\"WRANGLER\", \"trimLevel\":\"Sport\", \"numDoors\":2, \"wheelSize\":17, \"basePrice\":28475.00} and trim={\"\"}
2022-06-05 07:26:03.914 INFO 18968 --- [o-auto-1-exec-1] c.p.jeep.dao.DefaultJeepSalesDao : The fetchJeeps method was called with model={\"modelId\":\"WRANGLER\", \"trimLevel\":\"Sport\", \"numDoors\":2, \"wheelSize\":17, \"basePrice\":28475.00} and trim={\"\"}
2022-06-05 07:26:03.914 INFO 18968 --- [o-auto-1-exec-1] c.p.jeep.dao.DefaultJeepSalesDao : The fetchJeeps method was called with model={\"modelId\":\"WRANGLER\", \"trimLevel\":\"Sport\", \"numDoors\":2, \"wheelSize\":17, \"basePrice\":28475.00} and trim={\"\"}
```




```
package com.promineotech.jeep.dao;

import java.util.List;
import org.springframework.stereotype.Service;
import com.promineotech.jeep.entity.Jeep;
import com.promineotech.jeep.entity.JeepModel;
import lombok.extern.slf4j.Slf4j;

@Service
@Slf4j
public class DefaultJeepSalesDao implements JeepSalesDao {


    @Override
    public List<Jeep> fetchJeeps(JeepModel model, String trim) {
        log.info("The fetchJeeps method was called with model={\"modelId\":\"WRANGLER\", \"trimLevel\":\"Sport\", \"numDoors\":2, \"wheelSize\":17, \"basePrice\":28475.00} and trim={\"\"}");
        return null;
    }
}
```

- 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. 

```

1 package com.promineotech.jee.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.Service;
13 import com.promineotech.jee.entity.Jee;
14 import com.promineotech.jee.entity.JeeModel;
15 import lombok.extern.slf4j.Slf4j;
16
17 @Service
18 @Slf4j
19 public class DefaultJeeSalesDao implements JeeSalesDao {
20
21     @Autowired
22     NamedParameterJdbcTemplate jdbcTemplate;
23
24     @Override
25     public List<Jee> fetchJeeps(JeeModel model, String trim) {
26         log.info("The fetchJeeps method was called with model ={} and trim ={}",model,trim);
27         //@formatter: off
28         String sql = ""
29             + "SELECT * "
30             + "FROM models "
31             + "WHERE model_id = :model_id AND trim_level = :trim_level";
32         //@formatter: on
33
34         Map<String,Object> params = new HashMap<>();
35         params.put("model_id", model.toString());
36         params.put("trim_level", trim);
37
38         return jdbcTemplate.query(sql, params,
39             new RowMapper<>() {
40                 @Override
41                 public Jee mapRow(ResultSet rs, int rowNum) throws SQLException {
42                     //@formatter: off
43                     return Jee.builder()
44                         .basePrice(new BigDecimal(rs.getString("base_price")))
45                         .modelId(JeeModel.valueOf(rs.getString("model_id")))
46                         .modelPK(rs.getLong("model_pk"))
47                         .numDoors(rs.getInt("num_doors"))
48                         .trimLevel(rs.getString("trim_level"))
49                         .wheelSize(rs.getInt("wheel_size"))
50                         .build();
51                     //@formatter: on
52                 }
53             });
54     }
55 }
56

```

- 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. 

```
1 package com.promineotech.jeep.controller;
2
3 import org.junit.jupiter.api.Test;
4 import org.springframework.beans.factory.annotation.Autowired;
5 import org.springframework.boot.test.context.SpringBootTest;
6 import org.springframework.boot.test.context.SpringBootTest.WebEnvironment;
7 import org.springframework.boot.test.web.client.TestRestTemplate;
8
9 import org.springframework.core.ParameterizedTypeReference;
10 import org.springframework.test.context.ActiveProfiles;
11 import org.springframework.test.context.jdbc.Sql;
12 import org.springframework.test.context.jdbc.SqlConfig;
13 import org.springframework.boot.web.server.*;
14 import com.promineotech.jeep.entity.Jeep;
15 import com.promineotech.jeep.entity.JeepModel;
16 import org.springframework.http.HttpMethod;
17 import org.springframework.http.HttpStatus;
18 import org.springframework.http.ResponseEntity;
19
20 import java.math.BigDecimal;
21 import java.util.LinkedList;
22 import java.util.List;
23 import static org.assertj.core.api.Assertions.assertThat;
24
25
26 @SpringBootTest(webEnvironment = WebEnvironment.RANDOM_PORT)
27 @ActiveProfiles("test")
28 @Sql(scripts = {
29     "classpath:flyway/migrations/V1.0__Jeep_Schema.sql",
30     "classpath:flyway/migrations/V1.1__Jeep_Data.sql"},
31     config = @SqlConfig(encoding = "utf-8"))
32 class FetchJeepTest {
33
34     @Autowired
35     private TestRestTemplate restTemplate;
36
```



```

37  @LocalServerPort
38  private int serverPort;
39
40
41  @Test
42  void testThatJeepsAreReturnedWhenAValidModelAndTrimAreSupplied() {
43      JeepModel model = JeepModel.WRANGLER;
44      String trim = "Sport";
45      String uri = String.format("http://localhost:%d/jeeps?model=%s&trim=%s", serverPort, model, trim);
46      System.out.print(uri);
47      ResponseEntity<List<Jeep>> response = restTemplate.exchange(uri, HttpMethod.GET, null, new ParameterizedTypeReference<>() {});
48      assertThat(response.getStatusCode()).isEqualTo(HttpStatus.OK);
49
50      //Add a method to the test to return a list of expected Jeep (model) objects
51      List<Jeep> expected = buildExpected();
52      System.out.println(expected);
53      assertThat(response.getBody()).isEqualTo(expected);
54  }
55
56  protected List<Jeep> buildExpected() {
57      List<Jeep> list = new LinkedList<>();
58
59      // @formatter:off
60      list.add(Jeep.builder()
61          .modelId(JeepModel.WRANGLER)
62          .trimLevel("Sport")
63          .numDoors(2)
64          .wheelSize(17)
65          .basePrice(new BigDecimal("28475.00"))
66          .build());
67  }

```

```

68
69      list.add(Jeep.builder()
70          .modelId(JeepModel.WRANGLER)
71          .trimLevel("Sport")
72          .numDoors(4)
73          .wheelSize(17)
74          .basePrice(new BigDecimal("31975.00"))
75          .build());
76      // @formatter:on
77
78
79
80
81      return list;
82  }
83  }

```

The screenshot shows an IDE with the following components:

- Project Explorer:** Shows a project named "JUnit" with a file "FetchJeepTest.java" selected.
- Console:** Displays logs from the Spring Boot application. The logs show the application starting successfully on port 8080. The logs also show the test results for the "testThatJeepsAreReturnedWhenAValidModelAndTrimAreSupplied" test, which passed.
- Failure Trace:** Shows the details of the test failure, including the expected and actual values for the "basePrice" field.

The console output includes the following information:

- Spring Boot version: 2.4.3
- Application name: FetchJeepTest
- Application version: 1.0.0
- Application started on port 8080.
- Test results for "testThatJeepsAreReturnedWhenAValidModelAndTrimAreSupplied":
  - Expected: [Jeep(modelId=WRANGLER, trimLevel=Sport, numDoors=2, wheelSize=17, basePrice=28475.00)]
  - Actual: [Jeep(modelId=WRANGLER, trimLevel=Sport, numDoors=2, wheelSize=17, basePrice=28475.00)]
  - Test passed.

**URL to GitHub Repository:**

**<https://github.com/ailimutan/alect-springboot-week15-assignment>**