


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.Service;
5 import com.promineotech.jeep.entity.Jeep;
6 import com.promineotech.jeep.entity.JeepModel;
7 import lombok.extern.slf4j.Slf4j;
8
9 @Service
10 @Slf4j
11 public class DefaultJeepSalesDao implements JeepSalesDao {
12
13     @Override
14     public List<Jeep> fetchJeeps(JeepModel model, String trim) {
15         log.info("DAO: model = {}, trim = {}", model, trim);
16         return null;
17     }
18 }
19
20
```

```
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. 🖥️

```
24 @Override
25 public List<Jeep> fetchJeeps(JeepModel model, String trim) {
26     Log.info("DAO: model = {}, trim = {}", model, trim);
27
28     String sql = ""
29         + "SELECT * "
30         + "FROM models "
31         + "WHERE model_id = :model_id AND trim_level = :trim_level";
32
33     Map<String, Object> params = new HashMap<>();
34     params.put("model_id", model.toString());
35     params.put("trim_level", trim);
36
37     return jdbcTemplate.query(sql, params, new RowMapper<>(){
38
39         @Override
40         public Jeep mapRow(ResultSet rs, int rowNum) throws SQLException {
41
42             return Jeep.builder()
43                 .basePrice(new BigDecimal(rs.getString("base_price")))
44                 .modelId(JeepModel.valueOf(rs.getString("model_id")))
45                 .modelPK(rs.getLong("model_PK"))
46                 .numDoors(rs.getInt("num_doors"))
47                 .trimLevel(rs.getString("trim_level"))
48                 .wheelSize(rs.getInt("wheel_size"))
49                 .build();
50         }
51     });
52 }
53 }
54 }
```

- 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. 🖥️

Finished after 5.313 seconds

Runs: 1/1	Errors: 0	Failures: 0	<div style="width: 100px; height: 15px; background-color: green;"></div>
-----------	-----------	-------------	--

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

```

31 class FetchJeepTest {
32
33     @Autowired
34     private TestRestTemplate restTemplate;
35
36     @LocalServerPort
37     private int serverPort;
38
39     @Test
40     void testThatJeepsAreReturnedWhenAValidModelAndTrimAreSupplied() {
41
42         JeepModel model = JeepModel.WRANGLER;
43         String trim = "Freedom";
44         String uri = String.format("http://localhost:%d/jeeps?model=%s&trim=%s", serverPort, model, trim);
45
46         ResponseEntity<List<Jeep>> response =
47             restTemplate.exchange(uri, HttpMethod.GET, null, new ParameterizedTypeReference<>() {});
48         assertThat(response.getStatusCode()).isEqualTo(HttpStatus.OK);
49
50         List<Jeep> expected = buildExpected();
51         assertThat(response.getBody()).isEqualTo(expected);
52     }
53
54     List<Jeep> buildExpected(){
55         List<Jeep> list = new LinkedList<>();
56
57         list.add(Jeep.builder()
58             .modelId(JeepModel.WRANGLER)
59             .trimLevel("Freedom")
60             .numDoors(2)
61             .wheelSize(17)
62             .basePrice(new BigDecimal("36110.00"))
63             .build());
64
65         list.add(Jeep.builder()
66             .modelId(JeepModel.WRANGLER)
67             .trimLevel("Freedom")
68             .numDoors(4)
69             .wheelSize(17)
70             .basePrice(new BigDecimal("39365.00"))
71             .build());
72
73         return list;
74     }
75 }
76

```

Screenshots of Code:

```

1 package com.promineotech.jeepp.entity;
2
3 import java.math.BigDecimal;
4 import com.fasterxml.jackson.annotation.JsonIgnore;
5 import lombok.AllArgsConstructor;
6 import lombok.Builder;
7 import lombok.Data;
8 import lombok.NoArgsConstructor;
9
10 @Data
11 @Builder
12 @NoArgsConstructor
13 @AllArgsConstructor
14 public class Jeep {
15     private Long modelPK;
16     private JeepModel modelId;
17     private String trimLevel;
18     private int numDoors;
19     private int wheelSize;
20     private BigDecimal basePrice;
21
22     @JsonIgnore
23     public Long getModelPK() {
24         return modelPK;
25     }
26 }

```

```

1 package com.promineotech.jeepp.controller;
2
3 import java.util.List;
4
5 @RestController
6 @Slf4j
7 public class DefaultJeepSalesController implements JeepSalesController {
8
9     @Autowired
10     private JeepSalesService jeepSalesService;
11
12     @Override
13     public List<Jeep> fetchJeeps(JeepModel model, String trim) {
14         log.info("Model = {}, Trim = {}", model, trim);
15         return jeepSalesService.fetchJeeps(model, trim);
16     }
17 }

```

```

1 package com.promineotech.jeepp.service;
2
3 import java.util.List;
4
5 public interface JeepSalesService {
6
7     List<Jeep> fetchJeeps(JeepModel model, String trim);
8 }

```

```

1 package com.promineotech.jeepp.service;
2
3 import java.util.List;
4 import org.springframework.beans.factory.annotation.Autowired;
5 import org.springframework.stereotype.Service;
6 import com.promineotech.jeepp.dao.JeeppSalesDao;
7 import com.promineotech.jeepp.entity.Jeepp;
8 import com.promineotech.jeepp.entity.JeeppModel;
9 import lombok.extern.slf4j.Slf4j;
10
11 @Service
12 @Slf4j
13 public class DefaultJeeppSalesService implements JeeppSalesService {
14
15     @Autowired
16     private JeeppSalesDao jeeppSalesDao;
17
18
19     public List<Jeepp> fetchJeepps(JeeppModel model, String trim){
20         log.info("The fetchJeepps method was called with arguments: (model = {}, trim = {})", model, trim);
21         List<Jeepp> jeepps = jeeppSalesDao.fetchJeepps(model, trim);
22         return jeepps;
23     }
24 }
25

```

```

1 package com.promineotech.jeepp.dao;
2
3 import java.util.List;
4 import com.promineotech.jeepp.entity.Jeepp;
5 import com.promineotech.jeepp.entity.JeeppModel;
6
7 public interface JeeppSalesDao {
8     List<Jeepp> fetchJeepps(JeeppModel model, String trim);
9 }
10

```

```
1 package com.promineotech.jeeo.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.jeeo.entity.Jeeo;
14 import com.promineotech.jeeo.entity.JeeoModel;
15 import lombok.extern.slf4j.Slf4j;
16
17 @Service
18 @Slf4j
19 public class DefaultJeeoSalesDao implements JeeoSalesDao {
20
21     @Autowired
22     private NamedParameterJdbcTemplate jdbcTemplate;
23
24     @Override
25     public List<Jeeo> fetchJeeos(JeeoModel model, String trim) {
26         log.info("DAO: model = {}, trim = {}", model, trim);
27
28         String sql = ""
29             + "SELECT * "
30             + "FROM models "
31             + "WHERE model_id = :model_id AND trim_level = :trim_level";
32
33     }
```



```

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

```

```

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

```

```

1 spring:
2   datasource:
3     username: jeep
4     password: jeep
5     url: jdbc:mysql://localhost:3306/jeep
6

```

```

1 spring:
2   datasource:
3     url: jdbc:h2:mem:jeep

```

```

1 package com.promineotech.jeep;
2
3 import org.springframework.boot.SpringApplication;
4 import org.springframework.boot.autoconfigure.SpringBootApplication;
5
6 @SpringBootApplication
7 public class JeepSales {
8
9     public static void main(String[] args) {
10
11         SpringApplication.run(JeepSales.class, args);
12     }
13 }
14

```

Screenshots of Running Application:

```
Spring
:: Spring Boot ::      (v2.6.7)

2022-06-02 10:56:52.526 INFO 22972 --- [main] c.p.jee...controller.FetchJee... : Starting FetchJee... using Java 17.0.2 on DESKTOP-V692HUJ with PID 22972 (started by dalto in C:\Users\dalto\Desktop\Developer
2022-06-02 10:56:52.527 INFO 22972 --- [main] c.p.jee...controller.FetchJee... : The following 1 profile is active: "test"
2022-06-02 10:56:53.167 INFO 22972 --- [main] .s.d.r.c.RepositoryConfigurationDelegate : Bootstrapping Spring Data JDBC repositories in DEFAULT mode.
2022-06-02 10:56:53.187 INFO 22972 --- [main] .s.d.r.c.RepositoryConfigurationDelegate : Finished Spring Data repository scanning in 14 ms. Found 0 JDBC repository interfaces.
2022-06-02 10:56:53.859 INFO 22972 --- [main] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat initialized with port(s): 0 (http)
2022-06-02 10:56:53.870 INFO 22972 --- [main] o.apache.catalina.core.StandardService : Starting service [Tomcat]
2022-06-02 10:56:53.870 INFO 22972 --- [main] org.apache.catalina.core.StandardEngine : Starting Servlet engine: [Apache Tomcat/9.0.62]
2022-06-02 10:56:53.998 INFO 22972 --- [main] o.a.c.c.C.[Tomcat].[localhost].[/] : Initializing Spring embedded WebApplicationContext
2022-06-02 10:56:53.999 INFO 22972 --- [main] w.s.c.ServletWebServerApplicationContext : Root WebApplicationContext: initialization completed in 1449 ms
2022-06-02 10:56:55.080 INFO 22972 --- [main] com.zaxxer.hikari.HikariDataSource : HikariPool-1 - Starting...
2022-06-02 10:56:55.813 INFO 22972 --- [main] com.zaxxer.hikari.HikariDataSource : HikariPool-1 - Start completed.
2022-06-02 10:56:56.026 INFO 22972 --- [main] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat started on port(s): 63337 (http) with context path ''
2022-06-02 10:56:56.036 INFO 22972 --- [main] c.p.jee...controller.FetchJee... : Started FetchJee... in 3.87 seconds (JVM running for 4.941)
2022-06-02 10:56:56.708 INFO 22972 --- [o-auto-1-exec-1] o.a.c.c.C.[Tomcat].[localhost].[/] : Initializing Spring DispatcherServlet 'dispatcherServlet'
2022-06-02 10:56:56.708 INFO 22972 --- [o-auto-1-exec-1] o.s.web.servlet.DispatcherServlet : Initializing Servlet 'dispatcherServlet'
2022-06-02 10:56:56.709 INFO 22972 --- [o-auto-1-exec-1] o.s.web.servlet.DispatcherServlet : Completed initialization in 0 ms
2022-06-02 10:56:56.747 INFO 22972 --- [o-auto-1-exec-1] c.p.j.c.DefaultJeeSalesController : Model = WRANGLER, Trim = Freedom
2022-06-02 10:56:56.747 INFO 22972 --- [o-auto-1-exec-1] c.p.j.service.DefaultJeeSalesService : The FetchJee... method was called with arguments: (model = WRANGLER, trim = Freedom)
2022-06-02 10:56:56.747 INFO 22972 --- [o-auto-1-exec-1] c.p.jee...dao.DefaultJeeSalesDao : DAO: model = WRANGLER, trim = Freedom
2022-06-02 10:56:56.965 INFO 22972 --- [ionShutdownHook] com.zaxxer.hikari.HikariDataSource : HikariPool-1 - Shutdown initiated...
2022-06-02 10:56:56.968 INFO 22972 --- [ionShutdownHook] com.zaxxer.hikari.HikariDataSource : HikariPool-1 - Shutdown completed.
```

URL to GitHub Repository:

[DaltonCash/PT-WK15 \(github.com\)](https://github.com/DaltonCash/PT-WK15)