


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`.

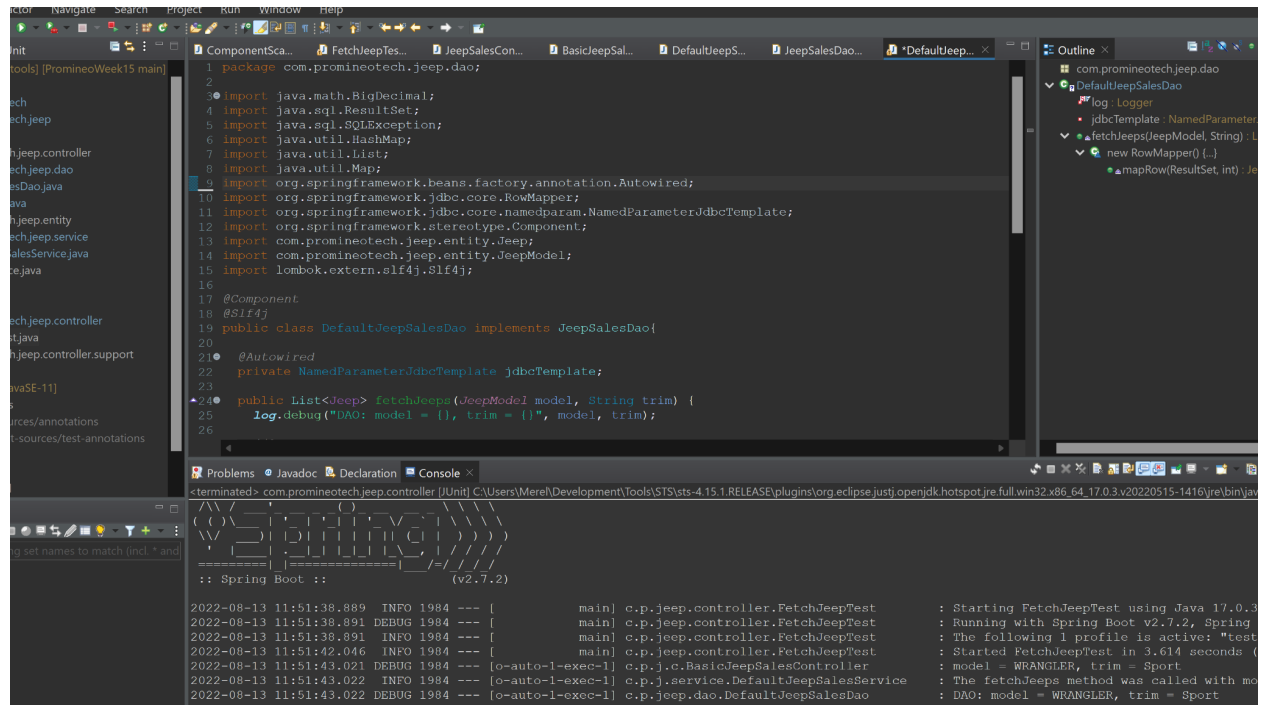
- d) Add a method in the DAO interface and implementation that returns a list of Jeep models (class Jeep) and takes the model and trim parameters. Here is the method signature:

```
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):
- Add the class-level annotation: @Service.
 - 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. 
 - 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. 
- 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:

3 b)

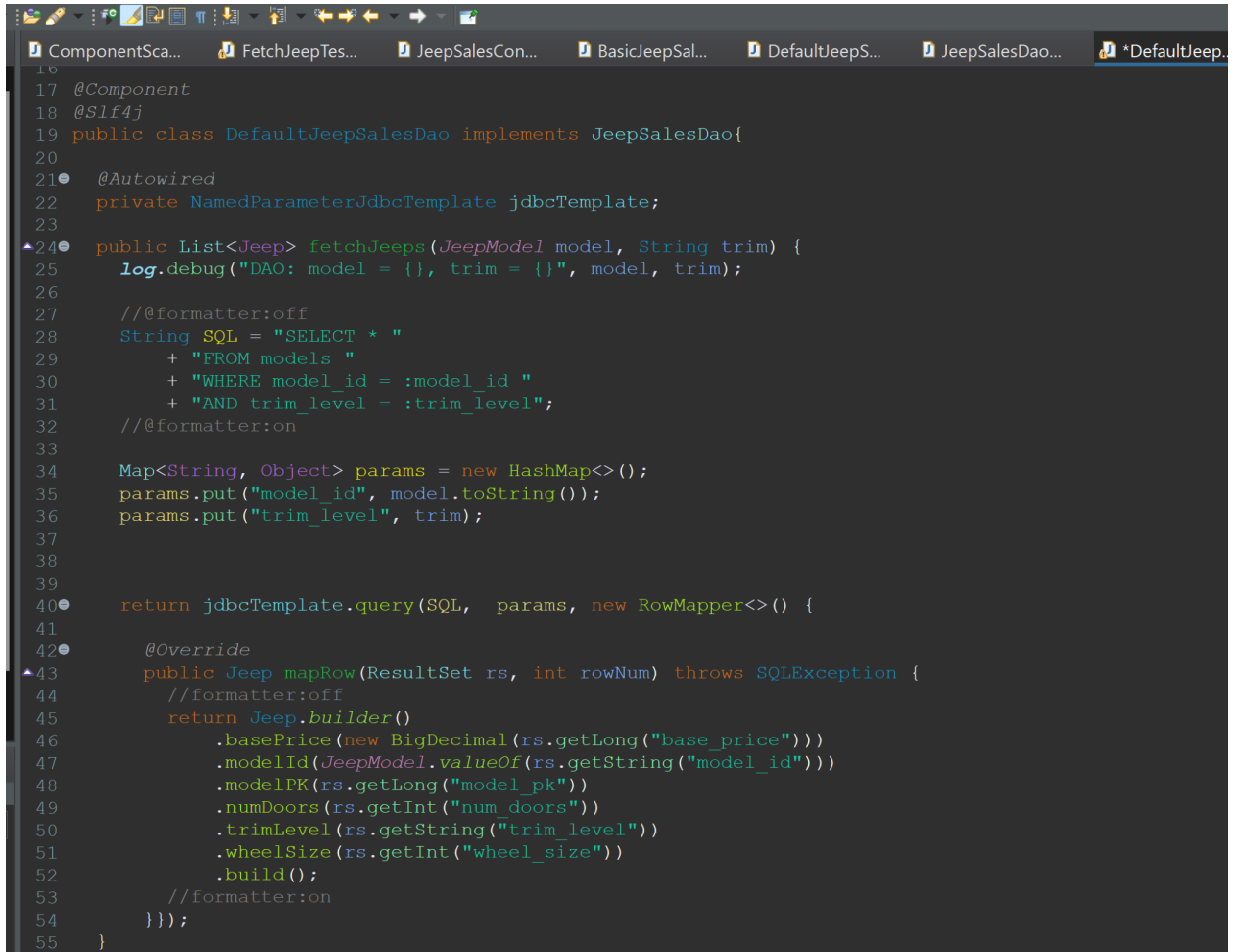


The screenshot displays the Eclipse IDE interface. The main editor shows the `DefaultJeepSalesDao` class, which implements `JeepSalesDao`. The code includes imports for `BigDecimal`, `ResultSet`, `SQLException`, `HashMap`, `List`, `Map`, `SpringFrameworkBeansFactoryAnnotationAutowired`, `SpringFrameworkJdbcCoreRowMapper`, `SpringFrameworkJdbcCoreNamedParamNamedParameterJdbcTemplate`, `SpringFrameworkStereotypeComponent`, `Jeep`, `JeepModel`, and `Slf4j`. The class is annotated with `@Component` and `@Slf4j`. It contains a `fetchJeeps` method that logs the DAO model and trim values.

The console output shows the following log messages:

```
<terminated> com.promineotech.jee...
2022-08-13 11:51:38.889 INFO 1984 --- [main] c.p.jee.controller.FetchJeepTest : Starting FetchJeepTest using Java 17.0.3
2022-08-13 11:51:38.891 DEBUG 1984 --- [main] c.p.jee.controller.FetchJeepTest : Running with Spring Boot v2.7.2, Spring
2022-08-13 11:51:38.891 INFO 1984 --- [main] c.p.jee.controller.FetchJeepTest : The following 1 profile is active: "test"
2022-08-13 11:51:42.046 INFO 1984 --- [main] c.p.jee.controller.FetchJeepTest : Started FetchJeepTest in 3.614 seconds (
2022-08-13 11:51:43.021 DEBUG 1984 --- [o-auto-1-exec-1] c.p.j.c.BasicJeepSalesController : model = WRANGLER, trim = Sport
2022-08-13 11:51:43.022 INFO 1984 --- [o-auto-1-exec-1] c.p.j.service.DefaultJeepSalesService : The fetchJeeps method was called with mo
2022-08-13 11:51:43.022 DEBUG 1984 --- [o-auto-1-exec-1] c.p.jee.dao.DefaultJeepSalesDao : DAO: model = WRANGLER, trim = Sport
```

3 f)



```
16
17 @Component
18 @Slf4j
19 public class DefaultJeepSalesDao implements JeepSalesDao{
20
21     @Autowired
22     private NamedParameterJdbcTemplate jdbcTemplate;
23
24     public List<Jeep> fetchJeeps(JeepModel model, String trim) {
25         log.debug("DAO: model = {}, trim = {}", model, trim);
26
27         //formatter:off
28         String SQL = "SELECT * "
29             + "FROM models "
30             + "WHERE model_id = :model_id "
31             + "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
39
40     return jdbcTemplate.query(SQL, params, new RowMapper<>() {
41
42         @Override
43         public Jeep mapRow(ResultSet rs, int rowNum) throws SQLException {
44             //formatter:off
45             return Jeep.builder()
46                 .basePrice(new BigDecimal(rs.getLong("base_price")))
47                 .modelId(JeepModel.valueOf(rs.getString("model_id")))
48                 .modelPK(rs.getLong("model_pk"))
49                 .numDoors(rs.getInt("num_doors"))
50                 .trimLevel(rs.getString("trim_level"))
51                 .wheelSize(rs.getInt("wheel_size"))
52                 .build();
53             //formatter:on
54         }
55     });
56 }
```

5)

The screenshot shows the Eclipse IDE with the `FetchJeepTest` class open. The class is a JUnit 5 test for the `FetchJeepTest` controller. It uses `SpringBootTest` with `WebEnvironment` set to `RANDOM_PORT`. The test method `testThatJeepsAreReturnedWhenValidModelAndTrimAreSupplied()` sends a GET request to `http://localhost:8080/jeeps?model=WRANGLER&trim=Sport` and expects a 200 status code and a list of jeeps. The console output shows the test starting at 12:11:29.493 and completing at 12:11:30.532. The test is successful, and the console shows the response body as a list of jeeps.

```
1 package com.promineotech.jeeptest.controller;
2
3 import static org.assertj.core.api.Assertions.assertThat;
4
5 @SpringBootTest(webEnvironment = WebEnvironment.RANDOM_PORT)
6 @ActiveProfiles("test")
7 @Sql(scripts = {
8     "classpath:flyway/migrations/V1.0_jeep_schema.sql",
9     "classpath:flyway/migrations/V1.1_jeep_data.sql"
10 }, config = @SqlConfig(encoding = "UTF-8"))
11 class FetchJeepTest extends FetchJeepTestSupport {
12     @Autowired
13     private TestRestTemplate restTemplate;
14
15     @LocalServerPort
16     private int serverPort;
17
18     @Test
19     void testThatJeepsAreReturnedWhenValidModelAndTrimAreSupplied() {
20         //given: a valid model, trim, and URL
21         JeepModel model = JeepModel.WRANGLER;
22         String trim = "Sport";
23         String url = String.format("http://localhost:%d/jeeps?model=%s&trim=%s", serverPort, model, trim);
24
25         //when: a connection is made to the URL
26         ResponseEntity<List<Jeep>> response = restTemplate.exchange(url, HttpMethod.GET, null,
27             new ParameterizedTypeReference<>() {});
28
29         //then: a success (OK - 200) status code is returned
30         assertThat(response.getStatusCode()).isEqualTo(HttpStatus.OK);
31
32         //And: the actual list is the same as the expected list
33         List<Jeep> actual = response.getBody();
34         List<Jeep> expected = buildExpected();
35
36         assertThat(actual).isEqualTo(expected);
37     }
38
39     protected List<Jeep> buildExpected() {
40         List<Jeep> list = new LinkedList<>();
41
42         //formatter off
43         list.add(Jeep.builder()
44             .modelId(JeepModel.WRANGLER)
45             .trimLevel("Sport")
46             .numDoors(1)
47             .wheelSize(17)
48             .basePrice(new BigDecimal("31975.00"))
49             .build());
50         list.add(Jeep.builder()
51             .modelId(JeepModel.WRANGLER)
52             .trimLevel("Sport")
53             .numDoors(2)
54             .wheelSize(17)
55             .basePrice(new BigDecimal("28475.00"))
56             .build());
57         //formatter on
58         Collections.sort(list);
59
60         return list;
61     }
62 }
```

Console Output:

```
<terminated> com.promineotech.jeeptest.controller [JUnit 5] C:\Users\Merel\Development\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
2022-08-13 12:11:29.493 INFO 3976 --- [main] c.p.jeeptest.controller.FetchJeepTest : Started FetchJeepTest in 3.631 seconds (JVM
2022-08-13 12:11:30.530 DEBUG 3976 --- [o-auto-1-exec-1] c.p.j.c.BasicJeepSalesController : model = WRANGLER, trim = Sport
2022-08-13 12:11:30.532 INFO 3976 --- [o-auto-1-exec-1] c.p.j.service.DefaultJeepSalesService : The fetchJeeps method was called with model
2022-08-13 12:11:30.532 DEBUG 3976 --- [o-auto-1-exec-1] c.p.jeeptest.dao.DefaultJeepSalesDao : DAO: model = WRANGLER, trim = Sport
```

The screenshot shows the Eclipse IDE with the `FetchJeepTest` class open. The `buildExpected` method is highlighted, which returns a list of jeeps. The console output shows the test starting at 12:11:29.493 and completing at 12:11:30.532. The test is successful, and the console shows the response body as a list of jeeps.

```
55 List<Jeep> actual = response.getBody();
56 List<Jeep> expected = buildExpected();
57
58 assertThat(actual).isEqualTo(expected);
59
60
61 protected List<Jeep> buildExpected() {
62     List<Jeep> list = new LinkedList<>();
63
64     //formatter off
65     list.add(Jeep.builder()
66         .modelId(JeepModel.WRANGLER)
67         .trimLevel("Sport")
68         .numDoors(1)
69         .wheelSize(17)
70         .basePrice(new BigDecimal("31975.00"))
71         .build());
72     list.add(Jeep.builder()
73         .modelId(JeepModel.WRANGLER)
74         .trimLevel("Sport")
75         .numDoors(2)
76         .wheelSize(17)
77         .basePrice(new BigDecimal("28475.00"))
78         .build());
79     //formatter on
80     Collections.sort(list);
81
82     return list;
83 }
84
85
86
87
88
89
90
```

Console Output:

```
<terminated> com.promineotech.jeeptest.controller [JUnit 5] C:\Users\Merel\Development\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
2022-08-13 12:11:29.493 INFO 3976 --- [main] c.p.jeeptest.controller.FetchJeepTest : Started FetchJeepTest in 3.631 seconds (JVM ru
2022-08-13 12:11:30.530 DEBUG 3976 --- [o-auto-1-exec-1] c.p.j.c.BasicJeepSalesController : model = WRANGLER, trim = Sport
2022-08-13 12:11:30.532 INFO 3976 --- [o-auto-1-exec-1] c.p.j.service.DefaultJeepSalesService : The fetchJeeps method was called with model =
2022-08-13 12:11:30.532 DEBUG 3976 --- [o-auto-1-exec-1] c.p.jeeptest.dao.DefaultJeepSalesDao : DAO: model = WRANGLER, trim = Sport
```

Screenshots of Running Application:

URL to GitHub Repository:

<https://github.com/MerelOhler/PromineoWeek15>