


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.

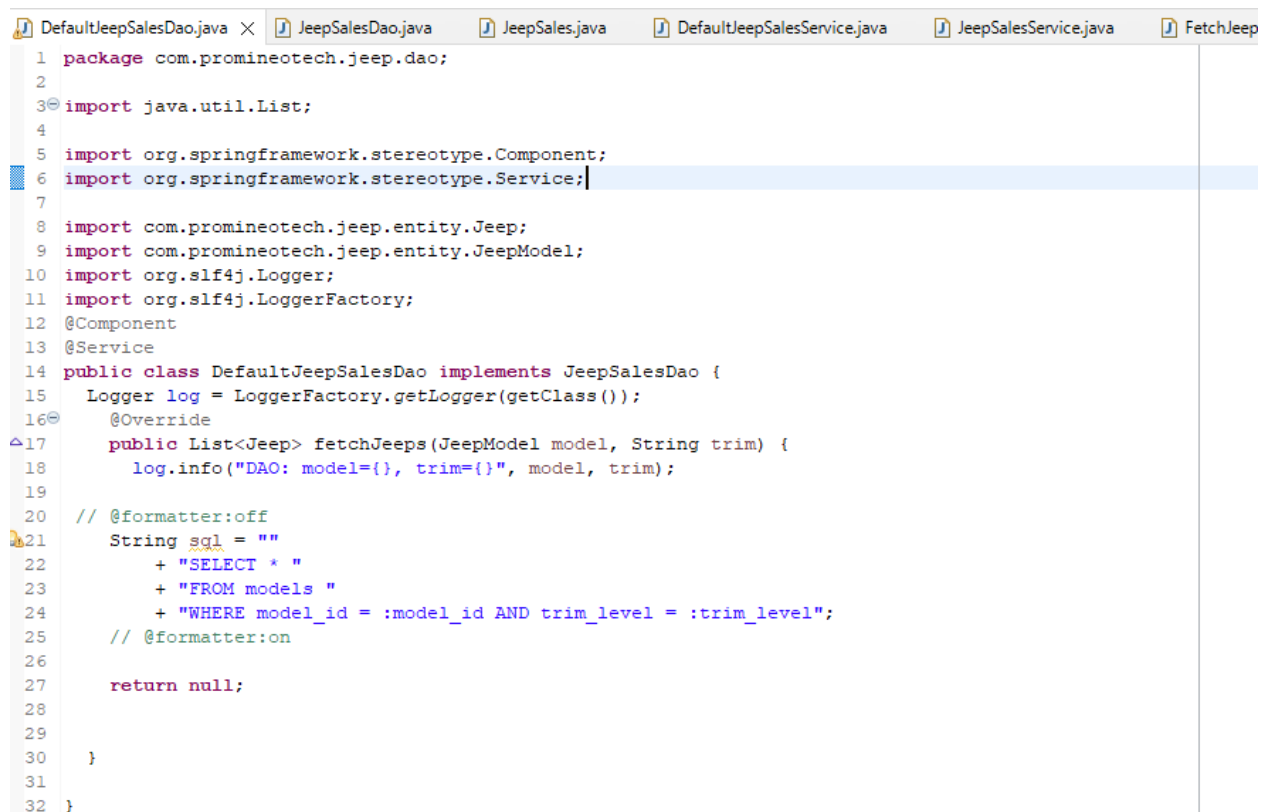
Project Resources: <https://github.com/promineotech/Spring-Boot-Course-Student-Resources>

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);
```

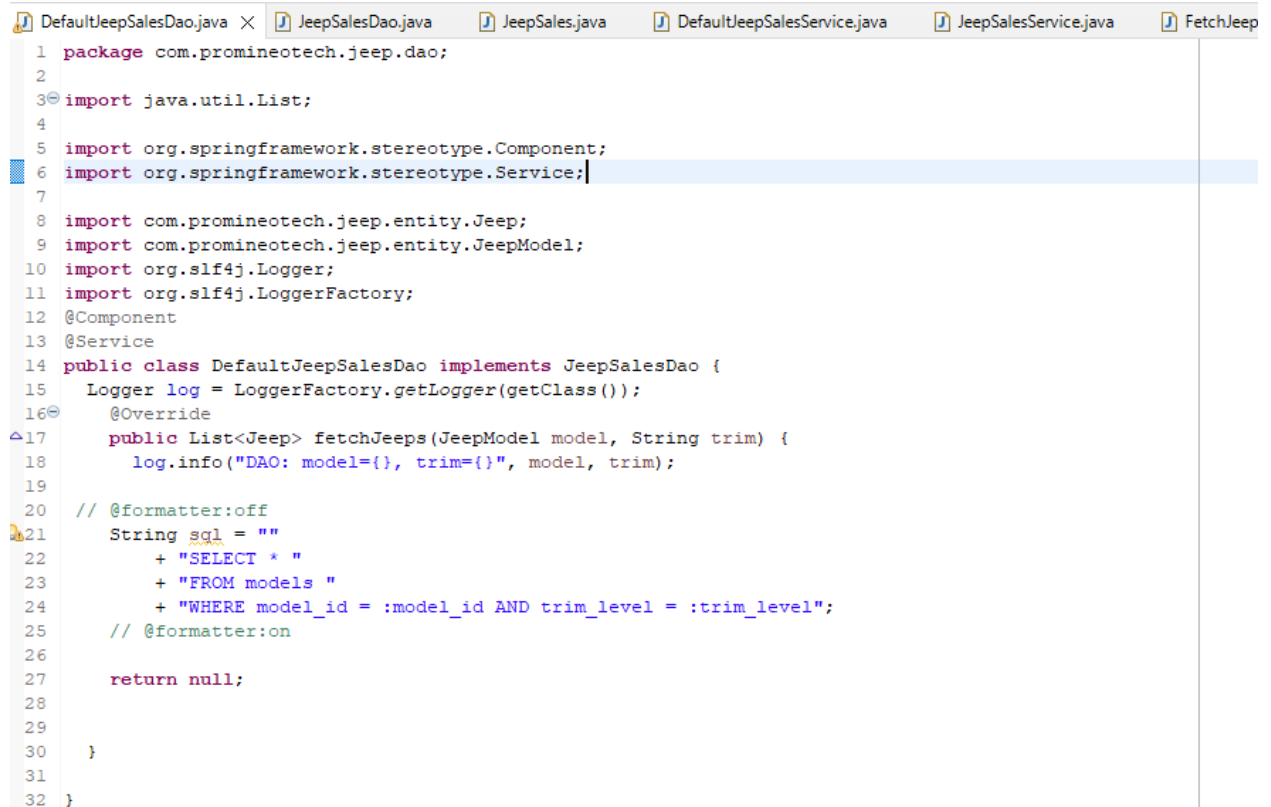
- 2) In the Jeep sales service implementation class, inject the DAO interface as an instance variable. The instance variable should be private and should be named `jeepSalesDao`. Call the DAO method from the service method and store the returned value in a local variable named `jeeps`. Return the value 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
5 import org.springframework.stereotype.Component;
6 import org.springframework.stereotype.Service;
7
8 import com.promineotech.jeep.entity.Jeep;
9 import com.promineotech.jeep.entity.JeepModel;
10 import org.slf4j.Logger;
11 import org.slf4j.LoggerFactory;
12 @Component
13 @Service
14 public class DefaultJeepSalesDao implements JeepSalesDao {
15     Logger log = LoggerFactory.getLogger(getClass());
16     @Override
17     public List<Jeep> fetchJeeps(JeepModel model, String trim) {
18         log.info("DAO: model={}, trim={}", model, trim);
19
20         // @formatter:off
21         String sql = "
22             + "SELECT * "
23             + "FROM models "
24             + "WHERE model_id = :model_id AND trim_level = :trim_level";
25         // @formatter:on
26
27         return null;
28     }
29 }
30
31
32 }
```

- 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.jeepp.dao;
2
3 import java.util.List;
4
5 import org.springframework.stereotype.Component;
6 import org.springframework.stereotype.Service;
7
8 import com.promineotech.jeepp.entity.Jeep;
9 import com.promineotech.jeepp.entity.JeepModel;
10 import org.slf4j.Logger;
11 import org.slf4j.LoggerFactory;
12 @Component
13 @Service
14 public class DefaultJeepSalesDao implements JeepSalesDao {
15     Logger log = LoggerFactory.getLogger(getClass());
16     @Override
17     public List<Jeep> fetchJeeps(JeepModel model, String trim) {
18         log.info("DAO: model={}, trim={}", model, trim);
19
20         // @formatter:off
21         String sql = "
22             + "SELECT * "
23             + "FROM models "
24             + "WHERE model_id = :model_id AND trim_level = :trim_level";
25         // @formatter:on
26
27         return null;
28     }
29 }
30 }
31 }
32 }
```

- 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:

DefaultJeepSalesService.java DefaultJeepSalesDao.java JeepSalesDao.java FetchJeepTest.java X FetchJeepTestSupport.java

```
1 package com.promineotech.jeeptest.controller;
2
3 import static org.assertj.core.api.Assertions.assertThat;
4
5
6 @SpringBootTest(webEnvironment = WebEnvironment.RANDOM_PORT)
7 @ActiveProfiles("test")
8 @Sql(scripts = {
9     "classpath:flyway/migrations/V1.0__Jeep_Schema.sql",
10    "classpath:flyway/migrations/V1.1__Jeep_Data.sql"},
11    config = @SqlConfig(encoding = "utf-8"))
12
13 class FetchJeepTest extends FetchJeepTestSupport {
14
15     @Autowired
16     @Getter
17     private TestRestTemplate restTemplate;
18
19
20     @LocalServerPort
21     private int serverPort;
22
23
24     @Test
25     void testThatJeepsAreReturnedWhenAvalidModelAndTrimAreSupplied() {
26
27         //Given: a valid model, trim and URI
28         JeepModel model = JeepModel.WRANGLER;
29         String trim = "Sport";
30         //String uri =
31         //String.format("http://localhost:%d/jeeps?model=%s&trim=%s", serverPort, model, trim);
32         String uri =
33             String.format("%s?model=%s&trim=%s", getBaseUrl(), model, trim);
34     }
35 }
```

DefaultJeepSalesService.java X DefaultJeepSalesDao.java JeepSalesDao.java FetchJeepTest.java FetchJeepTestSupport.java

```
1 package com.promineotech.jeeptest.service;
2
3 import java.util.Collections;
4
5
6 @Service
7 @Slf4j
8 public class DefaultJeepSalesService implements JeepSalesService {
9
10     @Autowired
11     private JeepSalesDao jeepSalesDao;
12
13
14     @Override
15     public List<Jeep> fetchJeeps(JeepModel model, String trim) {
16         log.info("The fetchJeeps method was called with model={} and trim={}", model, trim);
17
18         List<Jeep> jeeps = jeepSalesDao.fetchJeeps(model, trim);
19
20         Collections.sort(jeeps);
21         return jeepSalesDao.fetchJeeps(model, trim);
22     }
23
24 }
25
26
27
28
29
30
31
```

```

42     new RowMapper<>() {
43
44         @Override
45         public Jeep mapRow(ResultSet rs, int rowNum) throws SQLException {
46             // formatter:off
47             return Jeep.builder()
48                 .basePrice(new BigDecimal(rs.getString("base_price")))
49                 .modelId(JeepModel.valueOf(rs.getString("model_id")))
50                 .modelPK(rs.getLong("model_pk"))
51                 .numDoors(rs.getInt("num_doors"))
52                 .trimLevel(rs.getString("trim_level"))
53                 .wheelSize(rs.getInt("wheel_size"))
54                 .build();
55             // formatter:on
56         }
57     }
58
59
60 }
61
62
63

```

DefaultJeepSalesService.java DefaultJeepSalesDao.java × JeepSalesDao.java FetchJeepTest.java FetchJeepTestSupport.java

```

1  package com.promineotech.jeep.dao;
2
3  import java.math.BigDecimal;
19 @Component
20 @Service
21 @Slf4j
22 public class DefaultJeepSalesDao implements JeepSalesDao {
23
24     @Autowired
25     private NamedParameterJdbcTemplate jdbcTemplate;
26
27     Logger log = LoggerFactory.getLogger(getClass());
28     @Override
29     public List<Jeep> fetchJeeps(JeepModel model, String trim) {
30         log.info("DAO: model={}, trim={}", model, trim);
31
32         // @formatter:off
33         String sql = ""
34             + "SELECT * "
35             + "FROM models "
36             + "WHERE model_id = :model_id AND trim_level = :trim_level";
37         // @formatter:on
38         Map<String, Object> params = new HashMap<>();
39         params.put("model_id", model.toString());
40         params.put("trim_level", trim);
41         return jdbcTemplate.query(sql, params,
42             new RowMapper<>() {
43
44                 @Override
45                 public Jeep mapRow(ResultSet rs, int rowNum) throws SQLException {
46                     // formatter:off
47                     return Jeep.builder()

```

DefaultJeepSalesService.java DefaultJeepSalesDao.java JeepSalesDao.java X FetchJeepTest.java FetchJeepTestSupport.java

```
1 |
2 package com.promineotech.jeep.dao;
3
4 import java.util.List;
5
6
7
8
9 public interface JeepSalesDao {
10     List<Jeep> fetchJeeps(JeepModel model, String trim);
11 }
12
13
14
15
16
```

DefaultJeepSalesService.java DefaultJeepSalesDao.java JeepSalesDao.java FetchJeepTest.java X FetchJeepTestSupport.java

```
51     String uri =
52         String.format("%s?model=%s&trim=%s", getBaseUri(), model, trim);
53     //When: a connection is made to the URI
54
55     ResponseEntity<List<Jeep>> response =
56         getRestTemplate().exchange(uri, HttpMethod.GET, null,
57             new ParameterizedTypeReference<>() {});
58     // HttpMethod.GET, null, new ParameterizedTypeReference<>(){});
59
60     //Then: a success (OK - 200) status code is returned
61     assertThat(response.getStatusCode()).isEqualTo(HttpStatus.OK);
62
63     // And: the actual list returned is the same as the expected list
64
65     List<Jeep> actual = response.getBody();
66     List<Jeep> expected = buildExpected();
67
68     actual.forEach(jeep -> jeep.setModelPK(null));
69
70     assertThat(actual).isEqualTo(expected);
71
72 }
73
74
75
76
77
78
79
80
81
82
```


DefaultJeepSalesService.java DefaultJeepSalesDao.java JeepSalesDao.java FetchJeepTest.java FetchJeepTestSupport.java

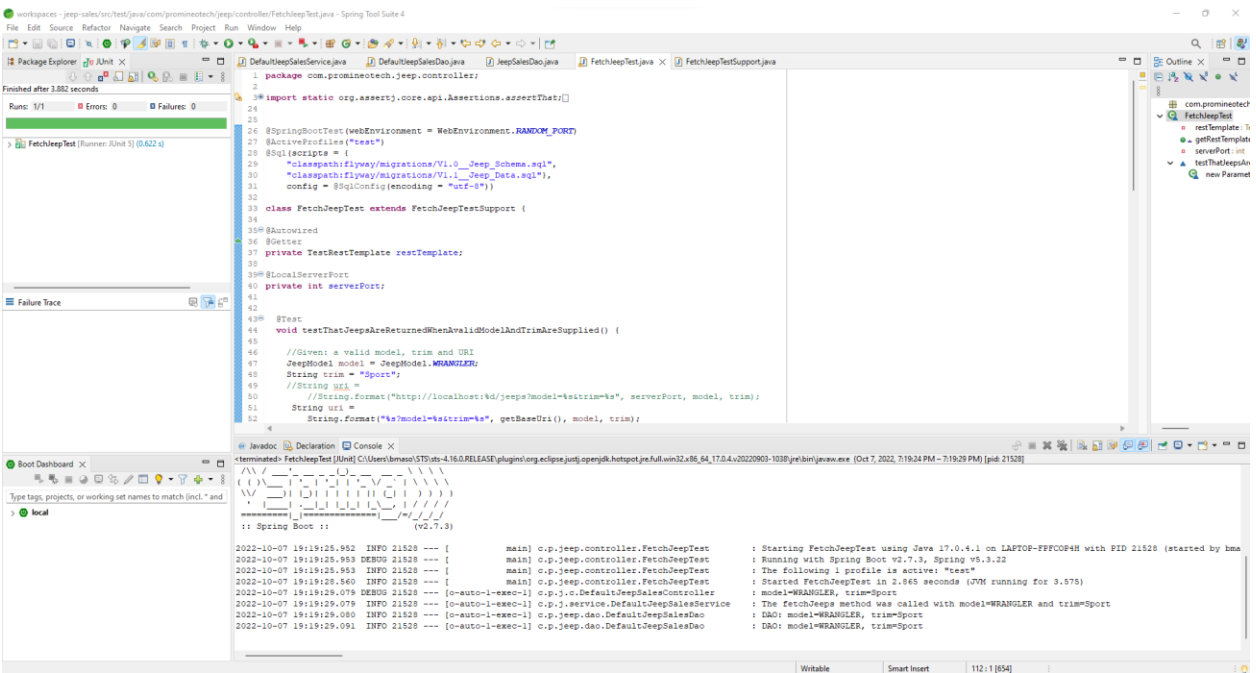
```
1 package com.promineotech.jeep.controller.support;
2
3 import java.math.BigDecimal;
4
5
6
7
8
9
10 public class FetchJeepTestSupport extends BaseTest {
11     protected List<Jeep> buildExpected() {
12         List<Jeep> list = new LinkedList<>();
13         // @formatter:off
14         list.add(Jeep.builder()
15             .modelId(JeepModel.WRANGLER)
16             .trimLevel("Sport")
17             .numDoors(2)
18             .wheelSize(17)
19             .basePrice(new BigDecimal("28475.00"))
20             .build());
21
22         list.add(Jeep.builder()
23             .modelId(JeepModel.WRANGLER)
24             .trimLevel("Sport")
25             .numDoors(4)
26             .wheelSize(17)
27             .basePrice(new BigDecimal("31975.00"))
28             .build());
29         // @formatter:on
30
31         Collections.sort(list);
32         return list;
33     }
34 }
35
```

```
DefaultJeepSalesService.java  DefaultJeepSalesDao.java  JeepSalesDao.java  FetchJeepTest.java  FetchJeepTestSupport.java  Je

1 package com.promineotech.jeep.entity;
2
3 import java.math.BigDecimal;
4
5
6
7
8
9
10
11 @Data
12 @Builder
13 @NoArgsConstructor
14 @AllArgsConstructor
15 public class Jeep implements Comparable<Jeep>{
16
17     private Long modelPK;
18     private JeepModel modelId;
19     private String trimLevel;
20     private int wheelSize;
21     private BigDecimal basePrice;
22     private int numDoors;
23
24     @JsonIgnore
25     public Long getModelPK() {
26         return modelPK;
27     }
28
29     @Override
30     public int compareTo(Jeep that) {
31         // formatter:off
32         return Comparator
33             .comparing(Jeep::getModelId)
34             .thenComparing(Jeep::getTrimLevel)
35             .thenComparing(Jeep::getNumDoors)
36             .compare(this, that);
37         // formatter:on
38     }
39 }
```

@ Javadoc Declaration Console X
<terminated> FetchJeepTest [Unit] C:\Users\hmacos\STS\etc-4.16.0\RELEASE\plugins\org.eclipse.jdt.launcher\hotspot\jre\full\win32\j86_64_17.0.4\20220903-1038\jre\bin\java.exe

Screenshots of Running Application:



URL to GitHub Repository:

<https://github.com/bmason1969/week-15.git>