


# 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:

- In the application you've been building add a DAO layer:
  - Add the package, com.promineotech.jeepp.dao.
  - In the new package, create an interface named JeepSalesDao.
  - In the same package, create a class named DefaultJeepSalesDao that implements JeepSalesDao.

- 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);
```
- 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).
- 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. 
- 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.
- Run the test to produce a green status bar. Produce a screenshot showing the test and the green status bar. 

### Screenshots of Code:

```
DefaultJeepSalesDao.java x
30 import java.math.BigDecimal;
19
20 @Service
21 @Slf4j
22 public class DefaultJeepSalesDao implements JeepSalesDao {
23
24     @Autowired
25     private NamedParameterJdbcTemplate jdbcTemplate;
26
27     @Override
28     public List<Jeep> fetchJeeps(JeepModel model, String trim) {
29         Log.debug("DAO: model={}, trim={}", model, trim);
30
31         // @formatter:off
32         String sql = "
33             + "SELECT * "
34             + "FROM models "
35             + "WHERE model_id = :model_id AND trim_level = :trim_level";
36         // @formatter:on
37
38         Map<String, Object> params = new HashMap<>();
39         params.put("model_id", model.toString());
40         params.put("trim_level", trim);
41
42         return jdbcTemplate.query(sql, params, 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
```

**Screenshots of Running Application:**

```
DefaultJeepSaleService.java  JeepSalesDao.java  DefaultJeepSalesDao.java  FetchJeepTest.java
1 package com.promineotech.jeep.dao;
2
30 import java.util.List;
11
12 @Service
13 @Slf4j
14 public class DefaultJeepSalesDao implements JeepSalesDao {
15
16     @Override
17     public List<Jeep> fetchJeeps(JeepModel model, String trim) {
18         log.debug("DAO: model={}, trim={}", model, trim);
19         return null;
20     }
21
22 }
23

Problems  Javadoc  Declaration  Console x
<terminated> FetchJeepTest [JUnit] C:\Users\adilh\CodingTools\STS\sts-4.16.0.RELEASE\plugins\org.eclipse.justi.openjdk.hotspot.jre.full.win32.x86_64_17.0.4.v20220903-1038\jre\bin\javaw.exe (Oct 4, 2022)
10:56:03.359 [main] DEBUG org.springframework.test.context.support.AbstractDirContextTestExecutionListener - Before test class: context [DefaultTestCon
10:56:03.390 [main] DEBUG org.springframework.test.context.support.DependencyInjectionTestExecutionListener - Performing dependency injection for test cont

=====
:: Spring Boot ::
(v2.7.3)

2022-10-04 10:56:04.072 INFO 14516 --- [main] c.p.jee.controller.FetchJeepTest : Starting FetchJeepTest using Java 17.0.4.1 on DESKTOP-
2022-10-04 10:56:04.074 DEBUG 14516 --- [main] c.p.jee.controller.FetchJeepTest : Running with Spring Boot v2.7.3, Spring v5.3.22
2022-10-04 10:56:04.075 INFO 14516 --- [main] c.p.jee.controller.FetchJeepTest : The following 1 profile is active: "test"
2022-10-04 10:56:10.007 INFO 14516 --- [main] c.p.jee.controller.FetchJeepTest : Started FetchJeepTest in 6.559 seconds (JVM running fo
2022-10-04 10:56:11.226 DEBUG 14516 --- [o-auto-1-exec-1] c.p.j.c.DefaultJeepSalesController : model=WRANGLER, trim=Sport
2022-10-04 10:56:11.229 INFO 14516 --- [o-auto-1-exec-1] c.p.jee.service.DefaultJeepSaleService : The fetchJeeps method was called with model=WRANGLER a
2022-10-04 10:56:11.232 DEBUG 14516 --- [o-auto-1-exec-1] c.p.jee.dao.DefaultJeepSalesDao : DAO: model=WRANGLER, trim=Sport
```

```
sts-workspace - jeep-sales/src/test/java/com/promineotech/jeep/controller/FetchJeepTest.java - Spring Tool Suite 4
File Edit Source Refactor Navigate Search Project Run Window Help
Package Explorer  JUnit x
Finished after 6.745 seconds
Runs: 1/1  Errors: 0  Failures: 0
FetchJeepTest [Runner: JUnit 5] (1.426 s)
Failure Trace
Boot Dashboard x
Type tags, projects, or working set names to match find, * and ? wild
> local
1 elements hidden by filter

64
65 // Then a success (OK - 200) status code is returned.
66 assertEquals(response.getStatusCode(), HttpStatus.OK);
67
68 // And: the actual List returned is the same as the expected List
69 List<Jeep> actual = response.getBody();
70 List<Jeep> expected = buildExpected();
71
72 assertEquals(actual, expected);
73
74
75 protected List<Jeep> buildExpected() {
76     List<Jeep> list = new LinkedList<Jeep>();
77
78     // @formatter:off
79     list.add(Jeep.builder()
80         .modelId(JeepModel.WRANGLER)
81         .trimLevel("Sport")
82         .numDoors(2)
83         .wheelSize(17)
84         .basePrice(new BigDecimal("28475.00"))
85         .build());
86
87     list.add(Jeep.builder()
88         .modelId(JeepModel.WRANGLER)
89         .build());
90 }

Problems  Javadoc  Declaration  Console x
<terminated> FetchJeepTest [JUnit] C:\Users\adilh\CodingTools\STS\sts-4.16.0.RELEASE\plugins\org.eclipse.justi.openjdk.hotspot.jre.full.win32.x86_64_17.0.4.v20220903-1038\jre\bin\javaw.exe (Oct 4, 2022)
=====
:: Spring Boot ::
(v2.7.3)

2022-10-04 11:15:12.302 INFO 14484 --- [main] c.p.jee.controller.FetchJeepTest : Starting FetchJeepTest using Java 17.0.4.1 on DESKTOP-
2022-10-04 11:15:12.306 DEBUG 14484 --- [main] c.p.jee.controller.FetchJeepTest : Running with Spring Boot v2.7.3, Spring v5.3.22
2022-10-04 11:15:12.388 INFO 14484 --- [main] c.p.jee.controller.FetchJeepTest : The following 1 profile is active: "test"
2022-10-04 11:15:18.151 INFO 14484 --- [main] c.p.jee.controller.FetchJeepTest : Started FetchJeepTest in 6.405 seconds (JVM running
2022-10-04 11:15:19.323 DEBUG 14484 --- [o-auto-1-exec-1] c.p.j.c.DefaultJeepSalesController : model=WRANGLER, trim=Sport
2022-10-04 11:15:19.325 INFO 14484 --- [o-auto-1-exec-1] c.p.jee.service.DefaultJeepSaleService : The fetchJeeps method was called with model=WRANGLER
2022-10-04 11:15:19.325 DEBUG 14484 --- [o-auto-1-exec-1] c.p.jee.dao.DefaultJeepSalesDao : DAO: model=WRANGLER, trim=Sport
```

URL to GitHub Repository:

[www.github.com/jfalzini99/jeep-sales](https://www.github.com/jfalzini99/jeep-sales)