

Web API Design with Spring Boot Week 15 Coding Assignment


Points possible: 75

URL to GitHub Repository: <https://github.com/admmoore21/JeepSalesAPI>


URL to Public Link of your Video: <https://youtu.be/MRr8ZR09h48>

Instructions :

1. Follow the **Coding Steps** below to complete this assignment.

- In Spring Tool Suite (STS), or an IDE of your choice, write the code that accomplishes the objectives listed below. Ensure that the code compiles and runs as directed.
- Use your existing repo or create a new repository on GitHub for this week's assignment and push your completed code to the repo, including your entire Maven Project Directory (e.g., jeep-sales) and any additional files (e.g. .sql files) that you create. In addition, screenshot your ERD and push the screenshot to your GitHub repo.
- Include the screenshots into this Assignment Document indicated by: 
- Create a video showcasing your work:
 - In this video: record and present your project verbally while showing the results of the working project.
 - Easy way to Create a video: Start a meeting in Zoom, share your screen, open Eclipse with the code and your Console window, start recording & record yourself describing and running the program showing the results.
 - Your video should be a maximum of 5 minutes.
 - Upload your video with a public link.
 - Easy way to Create a Public Video Link: Upload your video recording to YouTube with a public link.


2. In addition, please include the following in your Coding Assignment Document:

- The requested screenshots, indicated by: 
- The URL for this week's GitHub repository.
- The URL of the public link of your video.

3. Save the Coding Assignment Document as a .pdf and do the following:

- Push the .pdf to the GitHub repo for this week.
 - Upload the .pdf to the LMS in your Coding Assignment Submission.
-

Web API Design with Spring Boot Week 15 Coding Assignment

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.

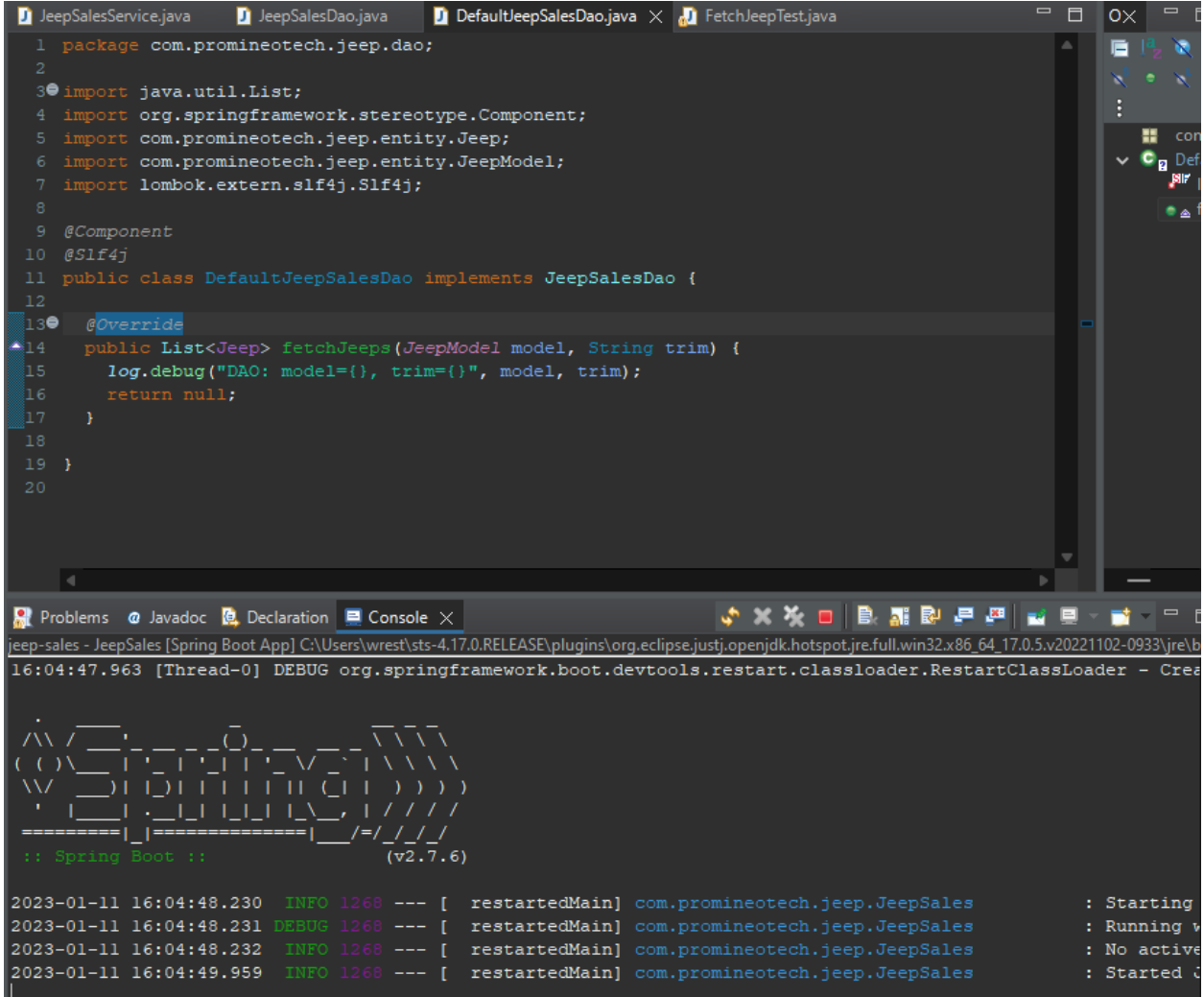
Project Resources: <https://github.com/promineotech/Spring-Boot-Course-Student-Resources>

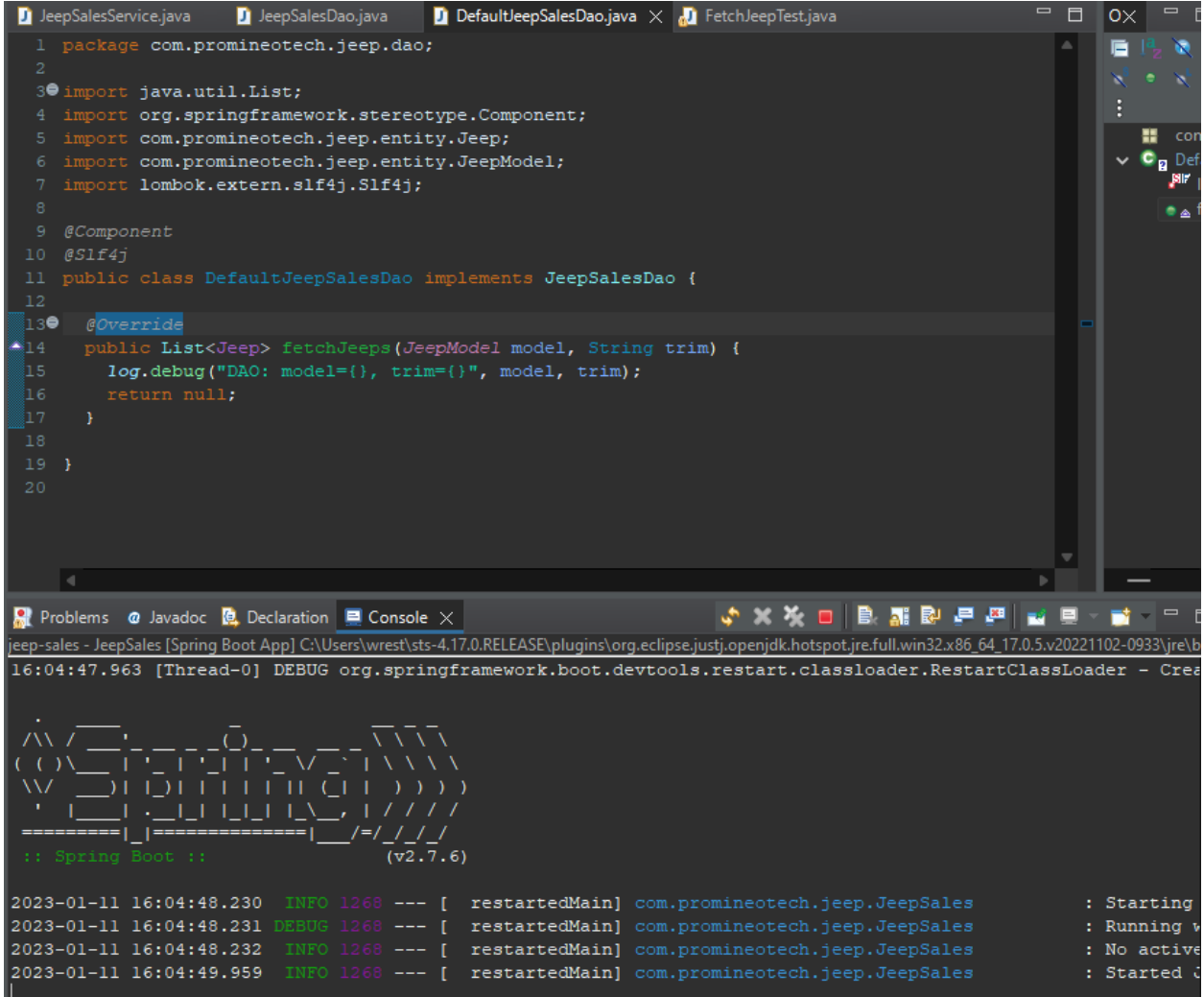
Coding Steps:

- 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).

Web API Design with Spring Boot Week 15 Coding Assignment

- 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. 



```
1 package com.promineotech.jeep.dao;
2
3 import java.util.List;
4 import org.springframework.stereotype.Component;
5 import com.promineotech.jeep.entity.Jeep;
6 import com.promineotech.jeep.entity.JeepModel;
7 import lombok.extern.slf4j.Slf4j;
8
9 @Component
10 @Slf4j
11 public class DefaultJeepSalesDao implements JeepSalesDao {
12
13     @Override
14     public List<Jeep> fetchJeeps(JeepModel model, String trim) {
15         log.debug("DAO: model={}, trim={}", model, trim);
16         return null;
17     }
18
19 }
20
```

Problems Javadoc Declaration Console

jeep-sales - JeepSales [Spring Boot App] C:\Users\wrest\sts-4.17.0.RELEASE\plugins\org.eclipse.justi.openjdk.hotspot.jre.full.win32.x86_64_17.0.5.v20221102-0933\jre\bin\java.exe -Xmx1024m -Xms64m -XX:MaxPermSize=256m -Dspring.datasource.url=jdbc:h2:mem:jeep-sales --spring-boot.run --spring-boot.run.arguments=--spring.datasource.url=jdbc:h2:mem:jeep-sales

16:04:47.963 [Thread-0] DEBUG org.springframework.boot.devtools.restart.classloader.RestartClassLoader - Creating new classloader

Spring Boot (v2.7.6)

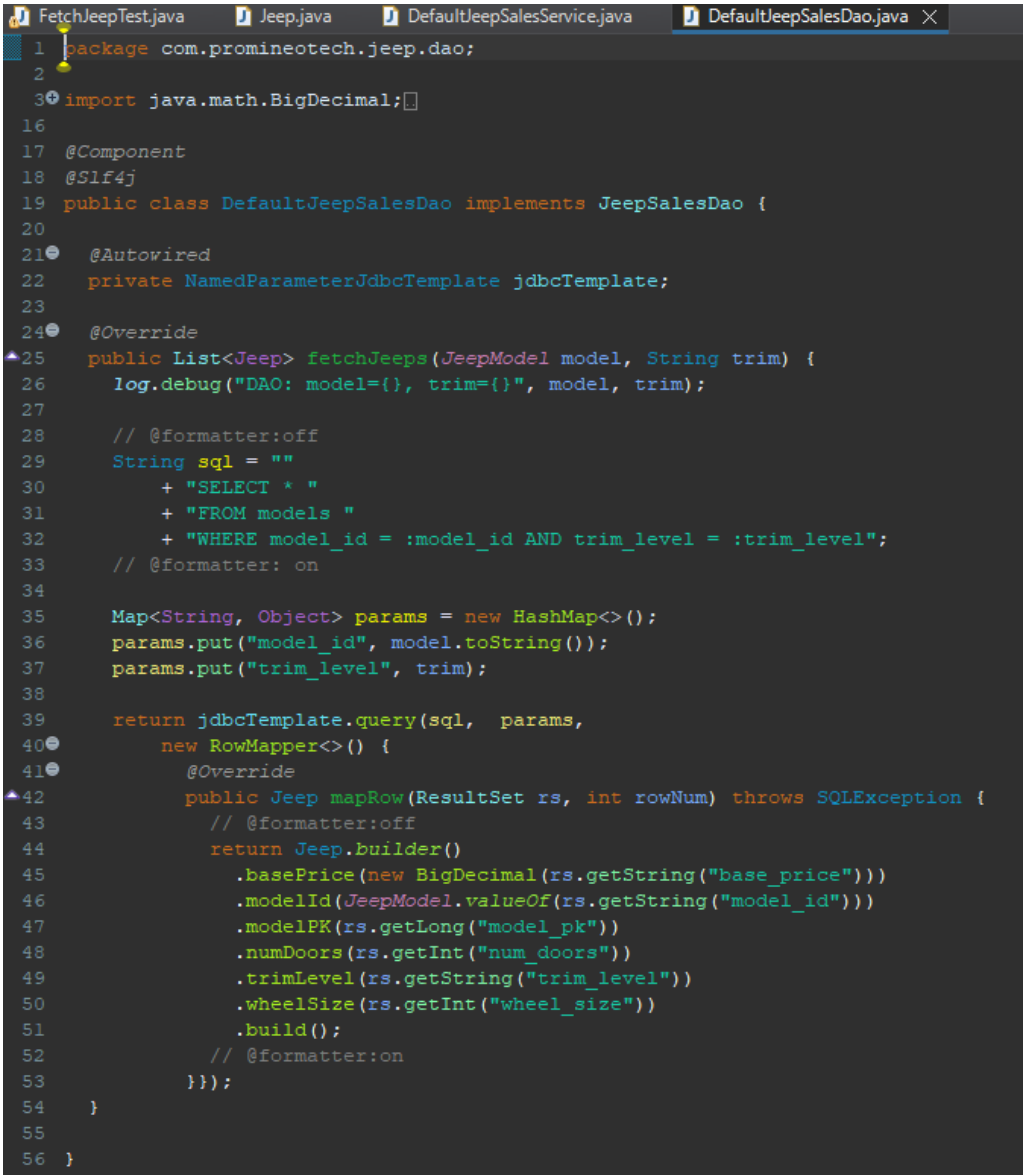
2023-01-11 16:04:48.230 INFO 1268 --- [restartedMain] com.promineotech.jeep.JeepSales : Starting

2023-01-11 16:04:48.231 DEBUG 1268 --- [restartedMain] com.promineotech.jeep.JeepSales : Running v

2023-01-11 16:04:48.232 INFO 1268 --- [restartedMain] com.promineotech.jeep.JeepSales : No active

2023-01-11 16:04:49.959 INFO 1268 --- [restartedMain] com.promineotech.jeep.JeepSales : Started

Web API Design with Spring Boot Week 15 Coding Assignment

- 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. 

```
FetchJeepTest.java  Jeep.java  DefaultJeepSalesService.java  DefaultJeepSalesDao.java X
1 package com.promineotech.jeeo.dao;
2
3 import java.math.BigDecimal;
16
17 @Component
18 @Slf4j
19 public class DefaultJeepSalesDao implements JeepSalesDao {
20
21     @Autowired
22     private NamedParameterJdbcTemplate jdbcTemplate;
23
24     @Override
25     public List<Jeep> fetchJeeps(JeepModel model, String trim) {
26         log.debug("DAO: model={}, trim={}", model, trim);
27
28         // @formatter:off
29         String sql = "
30             + "SELECT * "
31             + "FROM models "
32             + "WHERE model_id = :model_id AND trim_level = :trim_level";
33         // @formatter: on
34
35         Map<String, Object> params = new HashMap<>();
36         params.put("model_id", model.toString());
37         params.put("trim_level", trim);
38
39         return jdbcTemplate.query(sql, params,
40             new RowMapper<>() {
41                 @Override
42                 public Jeep mapRow(ResultSet rs, int rowNum) throws SQLException {
43                     // @formatter:off
44                     return Jeep.builder()
45                         .basePrice(new BigDecimal(rs.getString("base_price")))
46                         .modelId(JeepModel.valueOf(rs.getString("model_id")))
47                         .modelPK(rs.getLong("model_pk"))
48                         .numDoors(rs.getInt("num_doors"))
49                         .trimLevel(rs.getString("trim_level"))
50                         .wheelSize(rs.getInt("wheel_size"))
51                         .build();
52                     // @formatter:on
53                 }
54             });
55     }
56 }
```

Web API Design with Spring Boot Week 15 Coding Assignment

- 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.

The screenshot displays an IDE with the following components:

- Package Explorer:** Shows the project structure with a green status bar indicating a successful test run.
- JUnit Runner:** Shows the test results for 'FetchJeepTest [Runner: JUnit 5] (0.755 s)'. The status bar is green, and the output shows 'Runs: 1/1', 'Errors: 0', and 'Failures: 0'.
- Code Editor:** Displays the source code for 'FetchJeepTest.java'. The code includes imports for Spring Boot test annotations, a test class with a single test method, and a helper method 'buildExpected()' that constructs a list of two Jeep objects for comparison.
- Console:** Shows the output of the test run, including the Spring Boot logo and the version (v2.7.6).

```
1 package com.promineotech.jeepp.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
12 class FetchJeepTest {
13     @Autowired
14     private TestRestTemplate restTemplate;
15     @LocalServerPort
16     private int serverPort;
17
18     @Test
19     void testThatJeepsAreReturnedWhenAValidModelAndTrimAreSupplied() {
20         //Given: A valid model trim and URI
21         JeepModel model = JeepModel.WRANGLER;
22         String trim = "Sport";
23         String uri =
24             String.format("http://localhost:%d/jeeps?model=%s&trim=%s",
25                 serverPort, model, trim);
26
27         //When: A connection is made to the URI
28         ResponseEntity<List<Jeep>> response =
29             restTemplate.exchange(uri, HttpMethod.GET, null,
30                 new ParameterizedTypeReference<>() {});
31
32         //Then: a success (OK - 200) is returned
33         assertThat(response.getStatusCode()).isEqualTo(HttpStatus.OK);
34
35         // And: the actual list returned is the same as the expected list
36         List<Jeep> actual = response.getBody();
37         List<Jeep> expected = buildExpected();
38
39         assertThat(actual).isEqualTo(expected);
40     }
41
42     protected List<Jeep> buildExpected() {
43         List<Jeep> list = new LinkedList<>();
44
45         // @formatter:off
46         list.add(Jeep.builder()
47             .modelId(JeepModel.WRANGLER)
48             .trimLevel("Sport")
49             .numDoors(2)
50             .wheelSize(17)
51             .basePrice(new BigDecimal("28475.00"))
52             .build());
53
54         list.add(Jeep.builder()
55             .modelId(JeepModel.WRANGLER)
56             .trimLevel("Sport")
57             .numDoors(4)
58             .wheelSize(17)
59             .basePrice(new BigDecimal("31975.00"))
60             .build());
61
62         // @formatter:on
63
64         Collections.sort(list);
65         return list;
66     }
67 }
```