

Web API Design with Spring Boot Week 15 Coding Assignment


Points possible: 75

URL to GitHub Repository: <https://github.com/adrianedirisinghe1989/Week-13-15-SpringBoot>


URL to Public Link of your Video: https://www.youtube.com/watch?v=q_aGHQliVZw

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>

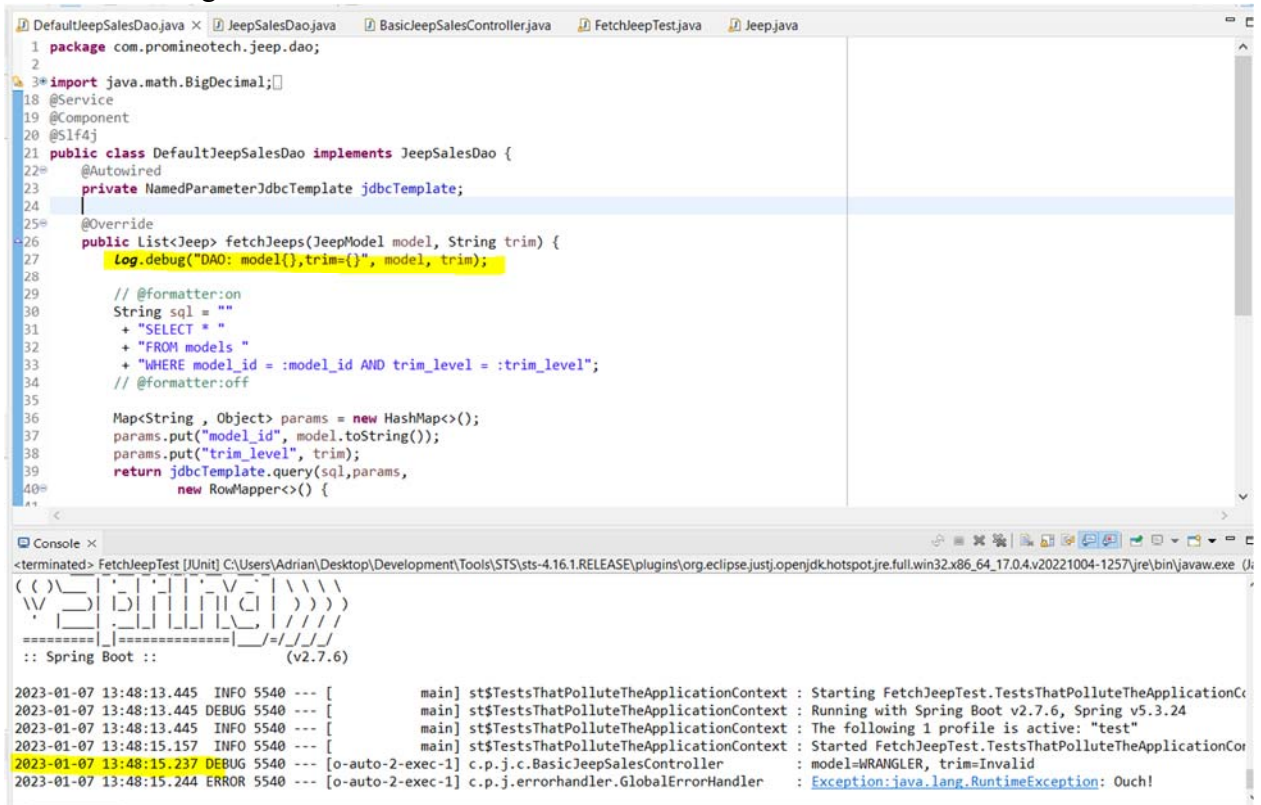
Coding Steps:

- 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.

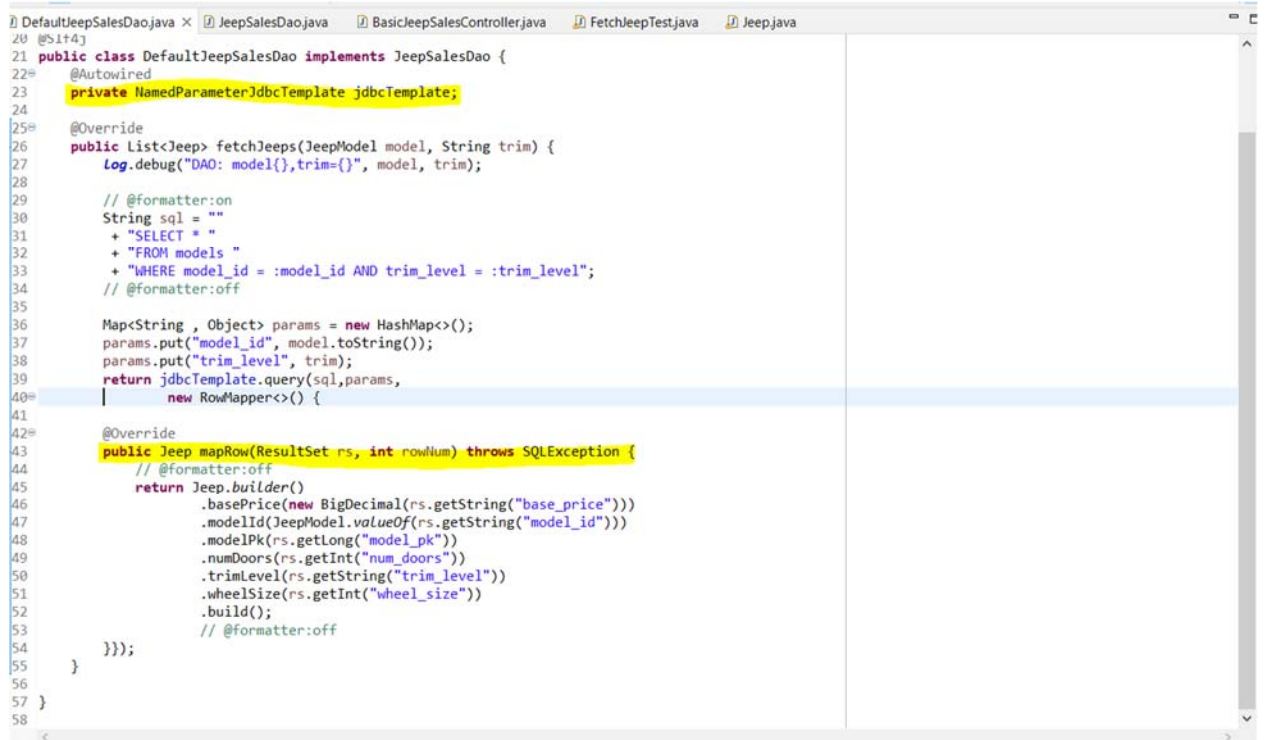


The screenshot shows an IDE with the `DefaultJeepSalesDao.java` file open. The code includes the `@Service` annotation and a `fetchJeeps` method that uses `Log.debug` to log the model and trim level. The console output shows the application running with Spring Boot v2.7.6 and the following log line:

```
2023-01-07 13:48:15.237 DEBUG 5540 --- [o-auto-2-exec-1] c.p.j.c.BasicJeepSalesController : model=WRANGLER, trim=Invalid
```

- 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

Web API Design with Spring Boot Week 15 Coding Assignment

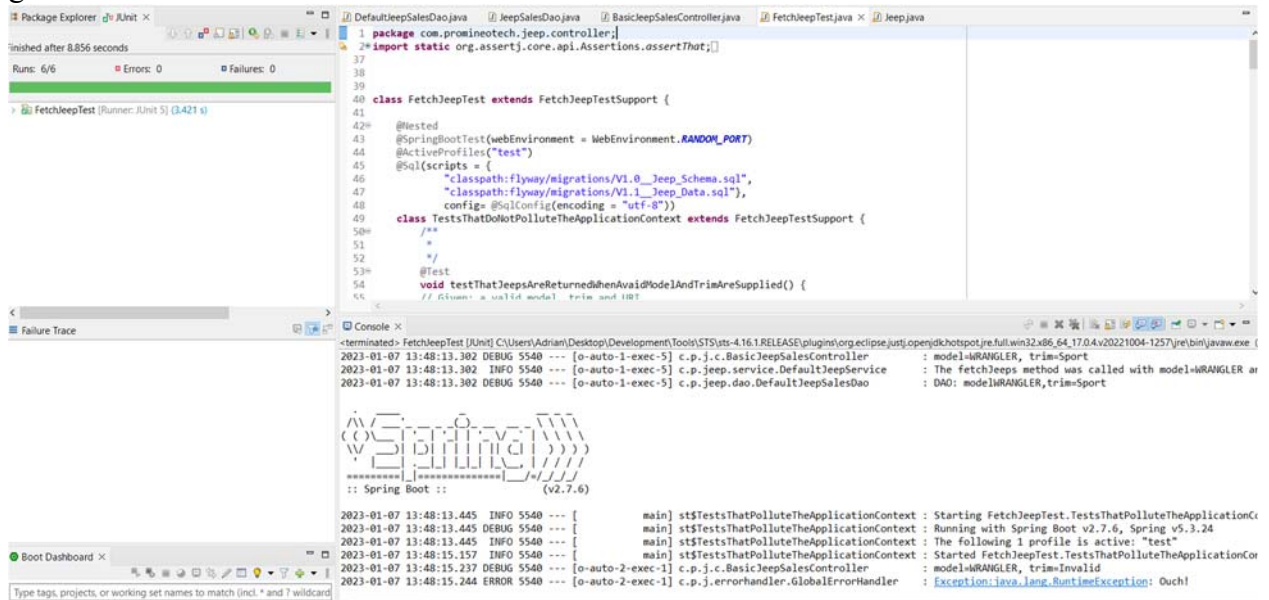
screenshot to show the complete method in the implementation class. 

```
DefaultJeepSalesDao.java x JeepSalesDao.java BasicJeepSalesController.java FetchJeepTest.java Jeep.java
20 @S114j
21 public class DefaultJeepSalesDao implements JeepSalesDao {
22     @Autowired
23     private NamedParameterJdbcTemplate jdbcTemplate;
24
25     @Override
26     public List<Jeep> fetchJeeps(JeepModel model, String trim) {
27         Log.debug("DAO: model={},trim={}", model, trim);
28
29         // @formatter:on
30         String sql = "
31             + "SELECT * "
32             + "FROM models "
33             + "WHERE model_id = :model_id AND trim_level = :trim_level";
34         // @formatter:off
35
36         Map<String, Object> params = new HashMap<>();
37         params.put("model_id", model.toString());
38         params.put("trim_level", trim);
39         return jdbcTemplate.query(sql, params,
40             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.getString("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:off
54             }
55         });
56     }
57 }
58 }
```

- 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.

Web API Design with Spring Boot Week 15 Coding Assignment

- 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 interface with the following components:

- Package Explorer:** Shows the project structure with files like `DefaultJeepSalesDao.java`, `JeepSalesDao.java`, `BasicJeepSalesController.java`, `FetchJeepTest.java`, and `Jeep.java`.
- Test Runner:** Shows the test `FetchJeepTest` (Runner: JUnit 5) with a green status bar and a message "Finished after 8.856 seconds".
- Source Editor:** Contains the code for `FetchJeepTest`, which extends `FetchJeepTestSupport`. It includes annotations for `@SpringBootTest`, `@ActiveProfiles("test")`, and `@Sql` scripts. The test method `testThatJeepsAreReturnedWhenValidModelAndTrimAreSupplied()` is shown.
- Console:** Displays the output of the test run, including the Spring Boot logo and the following log messages:

```
2023-01-07 13:48:13.302 DEBUG 5540 --- [o-auto-1-exec-5] c.p.j.c.BasicJeepSalesController : model=WRANGLER, trim=Sport
2023-01-07 13:48:13.302 INFO 5540 --- [o-auto-1-exec-5] c.p.jee.service.DefaultJeepService : The fetchJeeps method was called with model=WRANGLER and
2023-01-07 13:48:13.302 DEBUG 5540 --- [o-auto-1-exec-5] c.p.jee.dao.DefaultJeepSalesDao : DAO: model=WRANGLER, trim=Sport

:: Spring Boot ::
(v2.7.6)

2023-01-07 13:48:13.445 INFO 5540 --- [main] st$TestsThatPolluteTheApplicationContext : Starting FetchJeepTest.TestsThatPolluteTheApplicationC
2023-01-07 13:48:13.445 DEBUG 5540 --- [main] st$TestsThatPolluteTheApplicationContext : Running with Spring Boot v2.7.6, Spring v5.3.24
2023-01-07 13:48:13.445 INFO 5540 --- [main] st$TestsThatPolluteTheApplicationContext : The following 1 profile is active: "test"
2023-01-07 13:48:15.157 INFO 5540 --- [main] st$TestsThatPolluteTheApplicationContext : Started FetchJeepTest.TestsThatPolluteTheApplicationC
2023-01-07 13:48:15.237 DEBUG 5540 --- [o-auto-2-exec-1] c.p.j.c.BasicJeepSalesController : model=WRANGLER, trim=Invalid
2023-01-07 13:48:15.244 ERROR 5540 --- [o-auto-2-exec-1] c.p.j.errorhandler.GlobalExceptionHandler : Exception: java.lang.RuntimeException: Ouch!
```