

Web API Design with Spring Boot Week 15 Coding Assignment


Points possible: 75

URL to GitHub Repository: <https://github.com/iroda-n/JeepSales-Springboot>


URL to Public Link of your Video: https://drive.google.com/file/d/1ZYsQE83Y0g-QIzIdAMPhtcXmo6KMIL8N/view?usp=share_link

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.

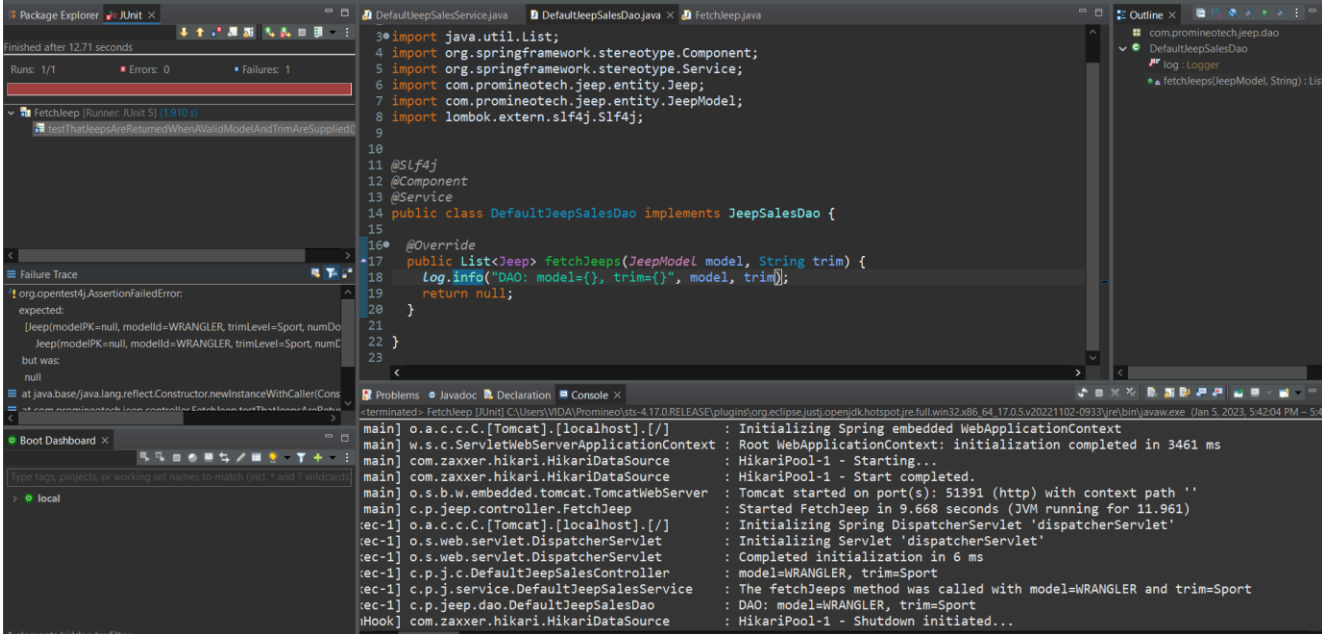
Project Resources: <https://github.com/promineotech/Spring-Boot-Course-Student-Resources>

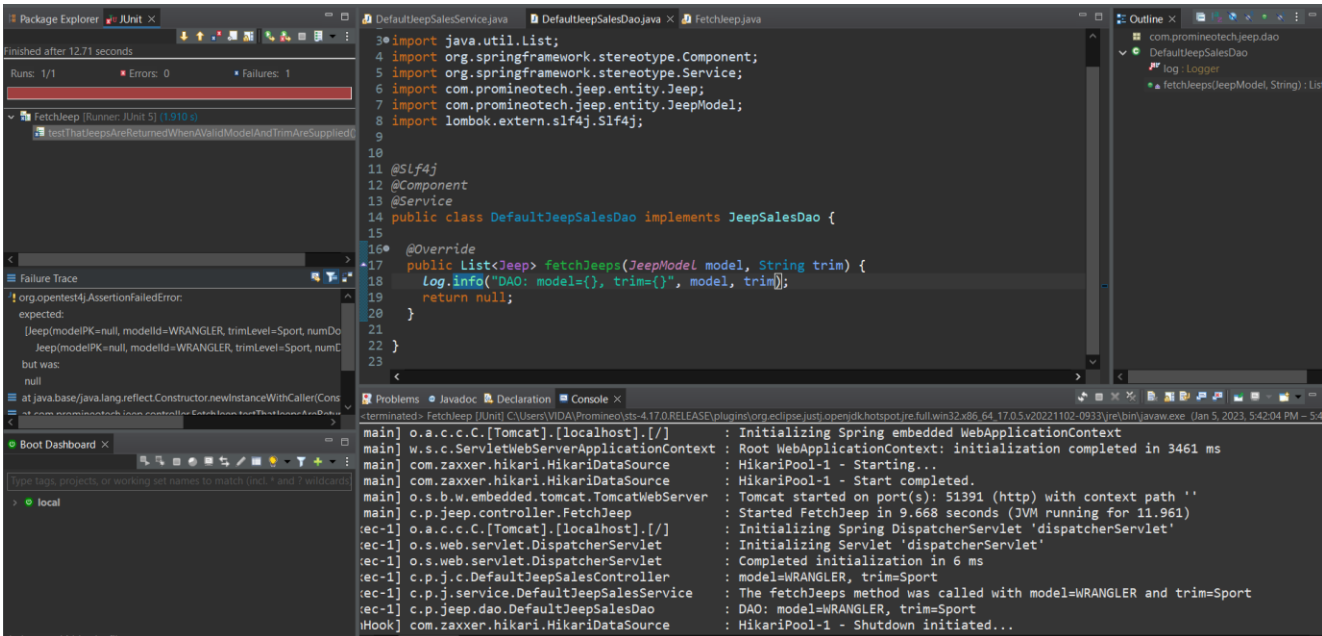
Coding Steps:

- 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 three panels. The top panel displays the `DefaultJeepSalesDao.java` file with the following code:

```
3 import java.util.List;
4 import org.springframework.stereotype.Component;
5 import org.springframework.stereotype.Service;
6 import com.promineotech.jee.entity.Jee;
7 import com.promineotech.jee.entity.JeeModel;
8 import lombok.extern.slf4j.Slf4j;
9
10
11 @Slf4j
12 @Component
13 @Service
14 public class DefaultJeepSalesDao implements JeepSalesDao {
15
16     @Override
17     public List<Jee> fetchJeeps(JeeModel model, String trim) {
18         log.info("DAO: model={}, trim={}", model, trim);
19         return null;
20     }
21 }
22
23
```


The bottom panel shows the console output with the following log line:

```
main] o.a.c.c.C.[Tomcat].[localhost].[/] : Initializing Spring embedded WebApplicationContext
main] w.s.c.ServletWebServerApplicationContext : Root WebApplicationContext: initialization completed in 3461 ms
main] com.zaxxer.hikari.HikariDataSource : HikariPool-1 - Starting...
main] com.zaxxer.hikari.HikariDataSource : HikariPool-1 - Start completed.
main] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat started on port(s): 51391 (http) with context path ''
main] c.p.jee.controller.FetchJeep : Started FetchJeep in 9.668 seconds (JVM running for 11.961)
[ec-1] o.a.c.c.C.[Tomcat].[localhost].[/] : Initializing Spring DispatcherServlet 'dispatcherServlet'
[ec-1] o.s.web.servlet.DispatcherServlet : Initializing Servlet 'dispatcherServlet'
[ec-1] o.s.web.servlet.DispatcherServlet : Completed initialization in 6 ms
[ec-1] c.p.j.c.DefaultJeepSalesController : model=WRANGLER, trim=Sport
[ec-1] c.p.j.service.DefaultJeepSalesService : The fetchJeeps method was called with model=WRANGLER and trim=Sport
[ec-1] c.p.jee.dao.DefaultJeepSalesDao : DAO: model=WRANGLER, trim=Sport
[Hook] com.zaxxer.hikari.HikariDataSource : HikariPool-1 - Shutdown initiated...
```

- 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. 

Web API Design with Spring Boot Week 15 Coding Assignment

```
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.info("DAO: model={}, trim={}", model, trim);
30
31         String sql = "SELECT * "
32             + "FROM models "
33             + "WHERE model_id = :model_id AND trim_level = :trim_level";
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
42             @Override
43             public Jeep mapRow(ResultSet rs, int rowNum) throws SQLException {
44
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             }
54         });
55     }
56 }
```

- 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. 

Web API Design with Spring Boot Week 15 Coding Assignment

```
51
52 //And: the actual list returned is the same as the expected list
53 List<Jeep> expected = buildExpected();
54 assertThat(response.getBody(), equalTo(expected));
55 }
56
57 protected List<Jeep> buildExpected() {
58     List<Jeep> list = new ArrayList<>();
59
60     list.add(Jeep.builder()
61         .modelId(JeepModel.WRANGLER)
62         .trimLevel("Sport")
63         .numDoors(2)
64         .wheelSize(17)
65         .basePrice(new BigDecimal("28475.00"))
66         .build());
67
68     list.add(Jeep.builder()
69         .modelId(JeepModel.WRANGLER)
70         .trimLevel("Sport")
71         .numDoors(4)
72         .wheelSize(17)
73         .basePrice(new BigDecimal("31975.00"))
74         .build());
75 }
```

```
<terminated> FetchJeep [JUnit] C:\Users\VIDA\Promineo\sts-4.17.0.RELEASE\plugins\org.eclipse.justi.openjdk.hotspot.jre.full.win32.x86_64.17.0.5.v20221102-0933\jre\bin\javaw.exe -Jan 5, 2023, 6:39:57 PM - 6:40:
[main] c.p.jeep.controller.FetchJeep : Started FetchJeep in 9.806 seconds (JVM running for 12.807)
[o-auto-1-exec-1] o.a.c.c.C.[Tomcat].[localhost].[/] : Initializing Spring DispatcherServlet 'dispatcherServlet'
[o-auto-1-exec-1] o.s.web.servlet.DispatcherServlet : Initializing Servlet 'dispatcherServlet'
[o-auto-1-exec-1] c.p.j.c.DefaultJeepSalesController : Completed initialization in 5 ms
[o-auto-1-exec-1] c.p.j.service.DefaultJeepSalesService : model=WRANGLER, trim=Sport
[o-auto-1-exec-1] c.p.jeep.dao.DefaultJeepSalesDao : The fetchJeeps method was called with model=WRANGLER and trim=Sport
[ionShutdownHook] com.zaxxer.hikari.HikariDataSource : DAO: model=WRANGLER, trim=Sport
[ionShutdownHook] com.zaxxer.hikari.HikariDataSource : HikariPool-1 - Shutdown initiated...
[ionShutdownHook] com.zaxxer.hikari.HikariDataSource : HikariPool-1 - Shutdown completed.
```