





Web API Design with Spring Boot Week 15 Coding Assignment
URL to GitHub Repository: https://github.com/Adam-Swim/Week_13_Jeep_sales

URL to Public Link of your Video: <https://youtu.be/ObEYQDB-rW4>

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 functionality into your Video when you see: 
- Create a video showcasing your work:
 - In this video: record and present your project verbally while showing the results of the working project. Don't forget to include the requested functionality, indicated by: 
 - 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 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 you should include something in your video submission, look for the icon:

Note: 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. In your video, show the DAO implementation class and the log line in the IDE's console. 🖥️

```
14
15 import com.promineotech.jeep.entity.Jeep;
16 import com.promineotech.jeep.entity.JeepModel;
17
18 import lombok.extern.slf4j.Slf4j;
19
20
21 @Service
22 @Slf4j
23 public class DefaultJeepSalesDao implements JeepSalesDao {
24
25     @Autowired
26     private NamedParameterJdbcTemplate jdbcTemplate;
27
28     @Override
29     public List<Jeep> fetchJeeps(JeepModel model, String trim) {
30         log.debug("DAO: model={}, trim={}", model, trim);
31     }
32 }
```

- 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.

```
1 package com.promineotech.jeep.dao;
2
3 import java.math.BigDecimal;
4 import java.sql.ResultSet;
5 import java.sql.SQLException;
6 import java.util.List;
7 import java.util.Map;
8 import java.util.HashMap;
9
10 import org.springframework.beans.factory.annotation.Autowired;
11 import org.springframework.jdbc.core.RowMapper;
12 import org.springframework.jdbc.core.namedparam.NamedParameterJdbcTemplate;
13 import org.springframework.stereotype.Service;
14
15 import com.promineotech.jeep.entity.Jeep;
16 import com.promineotech.jeep.entity.JeepModel;
17
18 import lombok.extern.slf4j.Slf4j;
19
20
21 @Service
22 @Slf4j
23 public class DefaultJeepSalesDao implements JeepSalesDao {
24
25     @Autowired
26     private NamedParameterJdbcTemplate jdbcTemplate;
27
28     @Override
29     public List<Jeep> fetchJeeps(JeepModel model, String trim) {
30         log.debug("DAO: model={}, trim={}", model, trim);
31     }
32 }
```

Use a RowMapper to map each row of the result set. Remember to convert modelId to a JeepModel. See the video for details. In your video, show the complete method in the implementation

5 Coding Assignment

class. 🖥️

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. In your video, show the test and the green status bar. 🖥️

```

232 // }
233 //
234 //
235 //
236 //
237 //
238 //
239 protected List<Jeep> buildExpected() {
240     List<Jeep> list = new LinkedList<>();
241
242     // @formatter:off
243     list.add(Jeep.builder()
244         .modelId(JeepModel.WRANGLER)
245         .trimLevel("Sport")
246         .numDoors(2)
247         .wheelSize(17)
248         .basePrice(new BigDecimal("28745.00"))
249         .build());
250
251     list.add(Jeep.builder()
252         .modelId(JeepModel.WRANGLER)
253         .trimLevel("Sport")
254         .numDoors(4)
255         .wheelSize(17)
256         .basePrice(new BigDecimal("31975.00"))
257         .build());
258     // @formatter:on
259
260     // Collections.sort(list);
261     return list;
262 }
263
264 // static Stream<Arguments> parametersForInvalidInput() {
265 //     // @formatter:off

```

Failure Trace

```

org.opentest4j.AssertionFailedError:
expected:
[Jeep(modelPK=null, modelId=WRANGLER, trimLevel=Sport, numDoors=2, wheelSize=17, basePrice=28745.00)]
but was:
[Jeep(modelPK=null, modelId=WRANGLER, trimLevel=Sport, numDoors=4, wheelSize=17, basePrice=31975.00)]

```

14:05:56.417 [main] DEBUG org.springframework.test.context.support.DependencyInjectionTestExecutionListener - Performing de