

# Web API Design with Spring Boot Week 15 Coding Assignment

Points possible: 75


URL to GitHub Repository: <https://github.com/Bwcash/Web-API-Spring-Boot.git>

URL to Public Link of your Video: <https://youtu.be/WxnUhRMVWG0>


---

## 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>

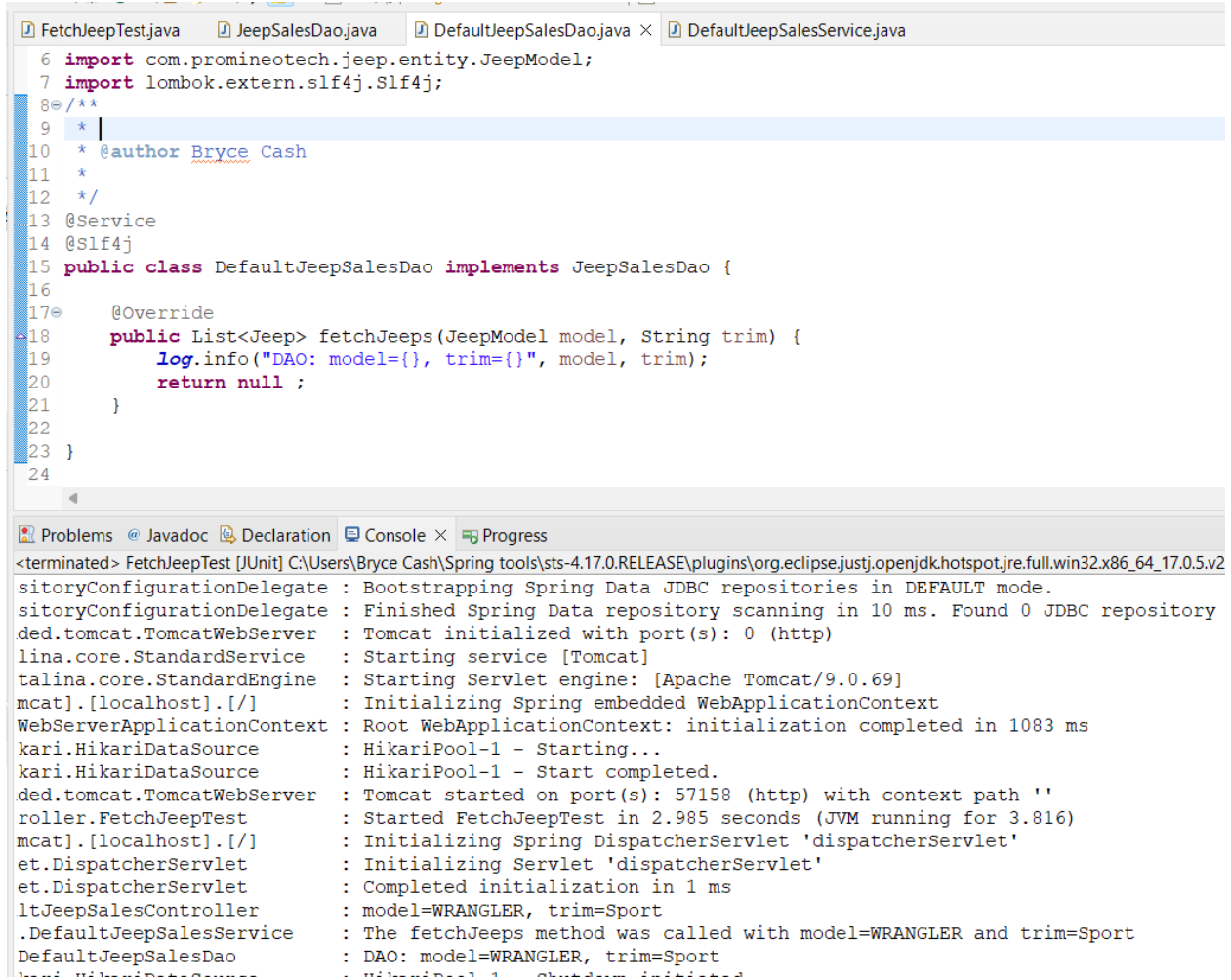
## Coding Steps:

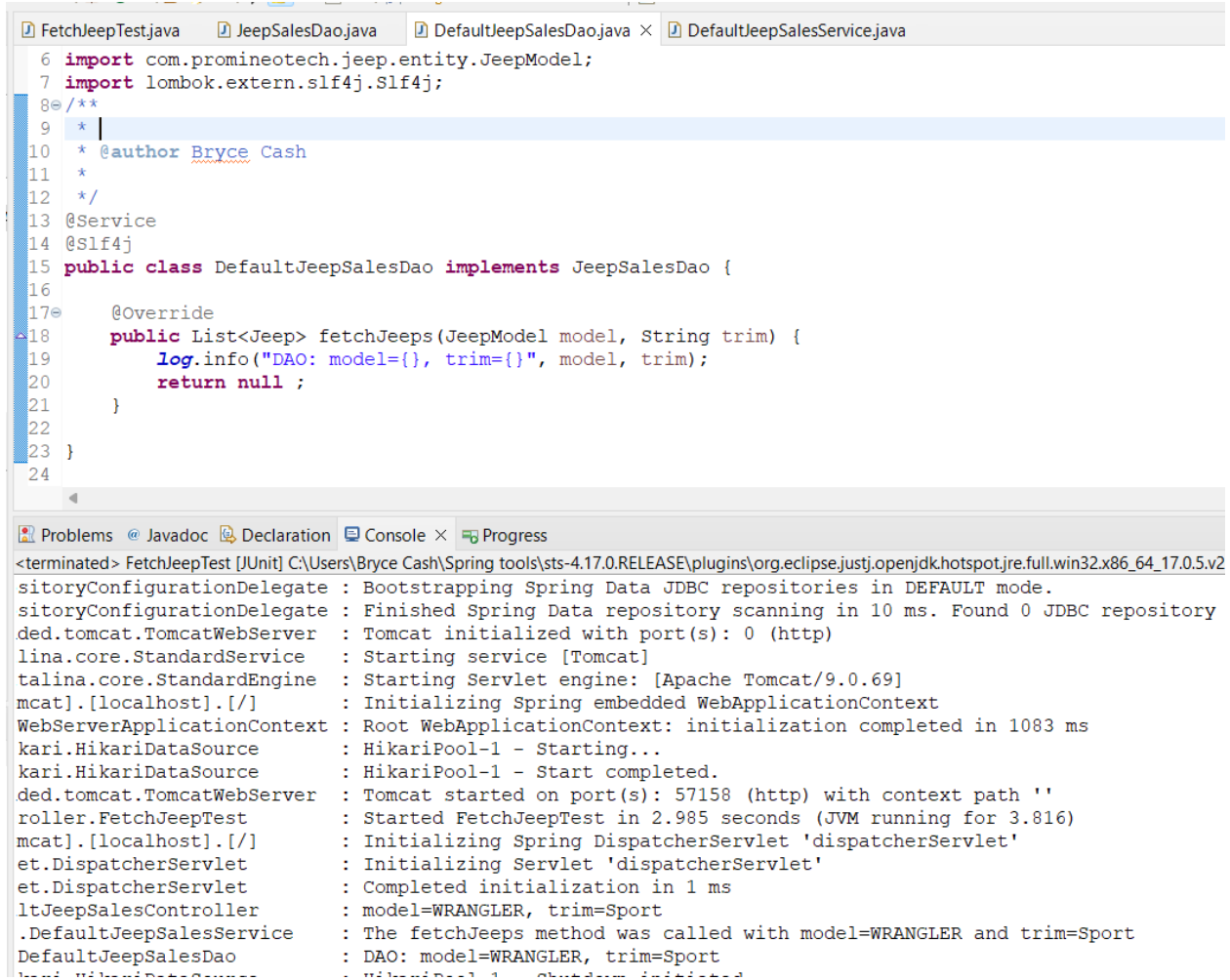
- 1) In the application you've been building add a DAO layer:
  - a) Add the package, `com.promineotech.jeep.dao`.
  - b) In the new package, create an interface named `JeepSalesDao`.
  - c) In the same package, create a class named `DefaultJeepSalesDao` that implements `JeepSalesDao`.
  - d) Add a method 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. 




```
6 import com.promineotech.jee.entity.JeeModel;
7 import lombok.extern.slf4j.Slf4j;
8 /**
9  *
10  * @author Bryce Cash
11  *
12  */
13 @Service
14 @Slf4j
15 public class DefaultJeepSalesDao implements JeepSalesDao {
16
17     @Override
18     public List<Jee> fetchJeeps(JeeModel model, String trim) {
19         log.info("DAO: model={}, trim={}", model, trim);
20         return null ;
21     }
22 }
23
24
```

Problems @ Javadoc Declaration Console × Progress


```
<terminated> FetchJeepTest [JUnit] C:\Users\Bryce Cash\Spring tools\sts-4.17.0.RELEASE\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_17.0.5.v2
sitoryConfigurationDelegate : Bootstrapping Spring Data JDBC repositories in DEFAULT mode.
sitoryConfigurationDelegate : Finished Spring Data repository scanning in 10 ms. Found 0 JDBC repository
ded.tomcat.TomcatWebServer : Tomcat initialized with port(s): 0 (http)
lina.core.StandardService : Starting service [Tomcat]
alina.core.StandardEngine : Starting Servlet engine: [Apache Tomcat/9.0.69]
mcat].[/localhost].[/] : Initializing Spring embedded WebApplicationContext
WebServerApplicationContext : Root WebApplicationContext: initialization completed in 1083 ms
kari.HikariDataSource : HikariPool-1 - Starting...
kari.HikariDataSource : HikariPool-1 - Start completed.
ded.tomcat.TomcatWebServer : Tomcat started on port(s): 57158 (http) with context path ''
oller.FetchJeepTest : Started FetchJeepTest in 2.985 seconds (JVM running for 3.816)
mcat].[/localhost].[/] : Initializing Spring DispatcherServlet 'dispatcherServlet'
et.DispatcherServlet : Initializing Servlet 'dispatcherServlet'
et.DispatcherServlet : Completed initialization in 1 ms
ltJeepSalesController : model=WRANGLER, trim=Sport
.DefaultJeepSalesService : The fetchJeeps method was called with model=WRANGLER and trim=Sport
DefaultJeepSalesDao : DAO: model=WRANGLER, trim=Sport
end: 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());)

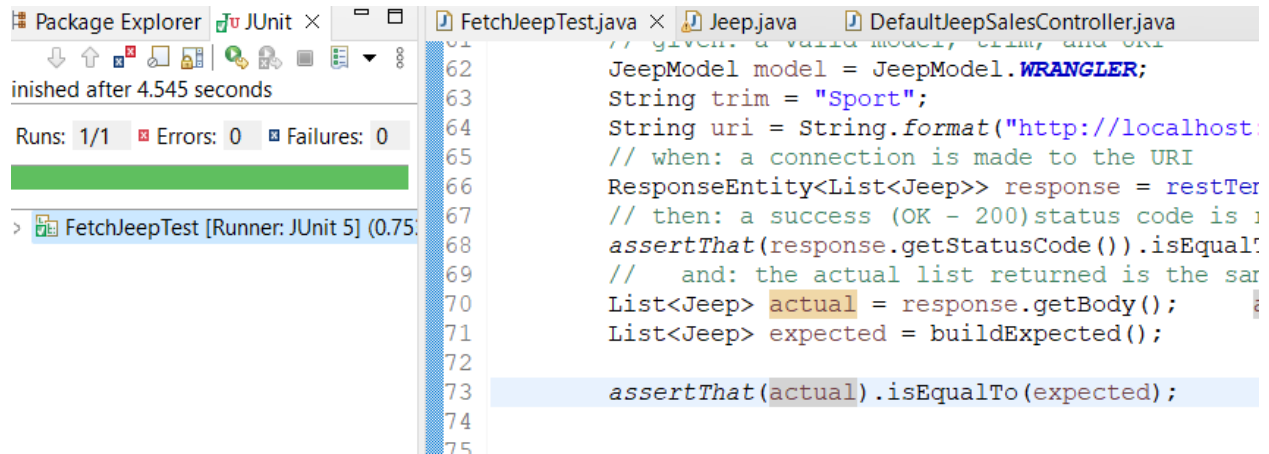
## Web API Design with Spring Boot Week 15 Coding Assignment

- 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 x JeepSalesDao.java DefaultJeepSalesDao.java x DefaultJeepSalesService.java
17 /**
18 jeep-sales/src/test/java/com/promineotech/jeep/controller/FetchJeepTest.java
19 * @author Bryce Cash
20 *
21 */
22 @Service
23 @Slf4j
24 public class DefaultJeepSalesDao implements JeepSalesDao {
25     @Autowired
26     private NamedParameterJdbcTemplate jdbcTemplate;
27
28     @Override
29     public List<Jeep> fetchJeeps(JeepModel model, String trim) {
30         log.info("DAO: model={}, trim={}", model, trim);
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         Map<String, Object> params = new HashMap<>();
38         params.put("model_id", model.toString());
39         params.put("trim_level", trim);
40
41         return jdbcTemplate.query(sql, params, new RowMapper<>() {
42
43             @Override
44             public Jeep mapRow(ResultSet rs, int rowNum) throws SQLException {
45                 //
46                 return Jeep.builder()
47                     .basePrice(new BigDecimal(rs.getString("base_price")))
48                     .modelId(JeepModel.valueOf(rs.getString("model_id")))
49                     .modelPk(rs.getLong("model_pk"))
50                     .numDoors(rs.getInt("num_doors"))
51                     .trimLevel(rs.getString("trim_level"))
52                     .wheelSize(rs.getInt("wheel_size"))
53                     .build();
54                 //
55                 //formatter:on
56             }
57         });
58     }
59 }
```

- 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



The screenshot shows an IDE with three tabs: `FetchJeepTest.java`, `Jeep.java`, and `DefaultJeepSalesController.java`. The `FetchJeepTest.java` tab is active, displaying a test method. The left sidebar shows the Package Explorer and a JUnit runner status. The test method in `FetchJeepTest.java` is as follows:

```
61 // given: a valid model, trim, and URI
62 JeepModel model = JeepModel.WRANGLER;
63 String trim = "Sport";
64 String uri = String.format("http://localhost:8080/api/jeep/%s/%s", model, trim);
65 // when: a connection is made to the URI
66 ResponseEntity<List<Jeep>> response = restTemplate.getForEntity(uri, List.class);
67 // then: a success (OK - 200) status code is returned
68 assertThat(response.getStatusCode()).isEqualTo(HttpStatus.OK);
69 // and: the actual list returned is the same as the expected list
70 List<Jeep> actual = response.getBody();
71 List<Jeep> expected = buildExpected();
72
73 assertThat(actual).isEqualTo(expected);
74
75
```

The JUnit runner status on the left indicates that the test `FetchJeepTest` [Runner: JUnit 5] (0.75s) passed successfully. The status bar shows "Runs: 1/1", "Errors: 0", and "Failures: 0". The test was completed after 4.545 seconds.