Web API Design with Spring Boot Week 3 Coding Assignment

Points possible: 70

Category	Criteria	% of Grade
Functionality	Does the code work?	25
Organization	Is the code clean and organized? Proper use of white space, syntax, and consistency are utilized. Names and comments are concise and clear.	25
Creativity	Student solved the problems presented in the assignment using creativity and out of the box thinking.	25
Completeness	All requirements of the assignment are complete.	25

Instructions: In Eclipse, or an IDE of your choice, write the code that accomplishes the objectives listed below. Ensure that the code compiles and runs as directed. Take screenshots of the code and of the running program (make sure to get screenshots of all required functionality) and paste them in this document where instructed below. Create a new repository on GitHub for this week's assignments and push this document, with your Java project code, to the repository. Add the URL for this week's repository to this document where instructed and submit this document to your instructor when complete.

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).

3) In the DAO implementation class (`DefaultJeepSalesDao`):

   a) Add the class-level annotation: `@Service`.

   b) 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. watch

```
oller       : Model = WRANGLER, Trim = Sport
sService    : The fetchJeeps method was called with arguments: (model= WRANGLER, trim=  Sport)
```

   c)

```
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.HashMap;
7  import java.util.List;
8  import java.util.Map;
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.Component;
14 import org.springframework.stereotype.Service;
15
16 import com.mysql.cj.log.Log;
17 import com.promineotech.jeep.entity.Jeep;
18 import com.promineotech.jeep.entity.JeepModel;
19
20 import lombok.extern.slf4j.Slf4j;
21 @Service
22 @Component
23 @Slf4j
24
25 public class DefaultJeepSalesDao implements JeepSalesDao {
26
27     @Autowired
28     private NamedParameterJdbcTemplate jdbcTemplate;
29
30     @Override
31     public List<Jeep> fetchJeeps(JeepModel model, String trim) {
32         log.debug("DAO: model={}, trim={}", model, trim);
33
34         return null;
35     }
36 }
37
```

   d)

e) In `DefaultJeepSalesDao`, inject an instance variable of type `NamedParameterJdbcTemplate`.

f) 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.

g) 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());`)

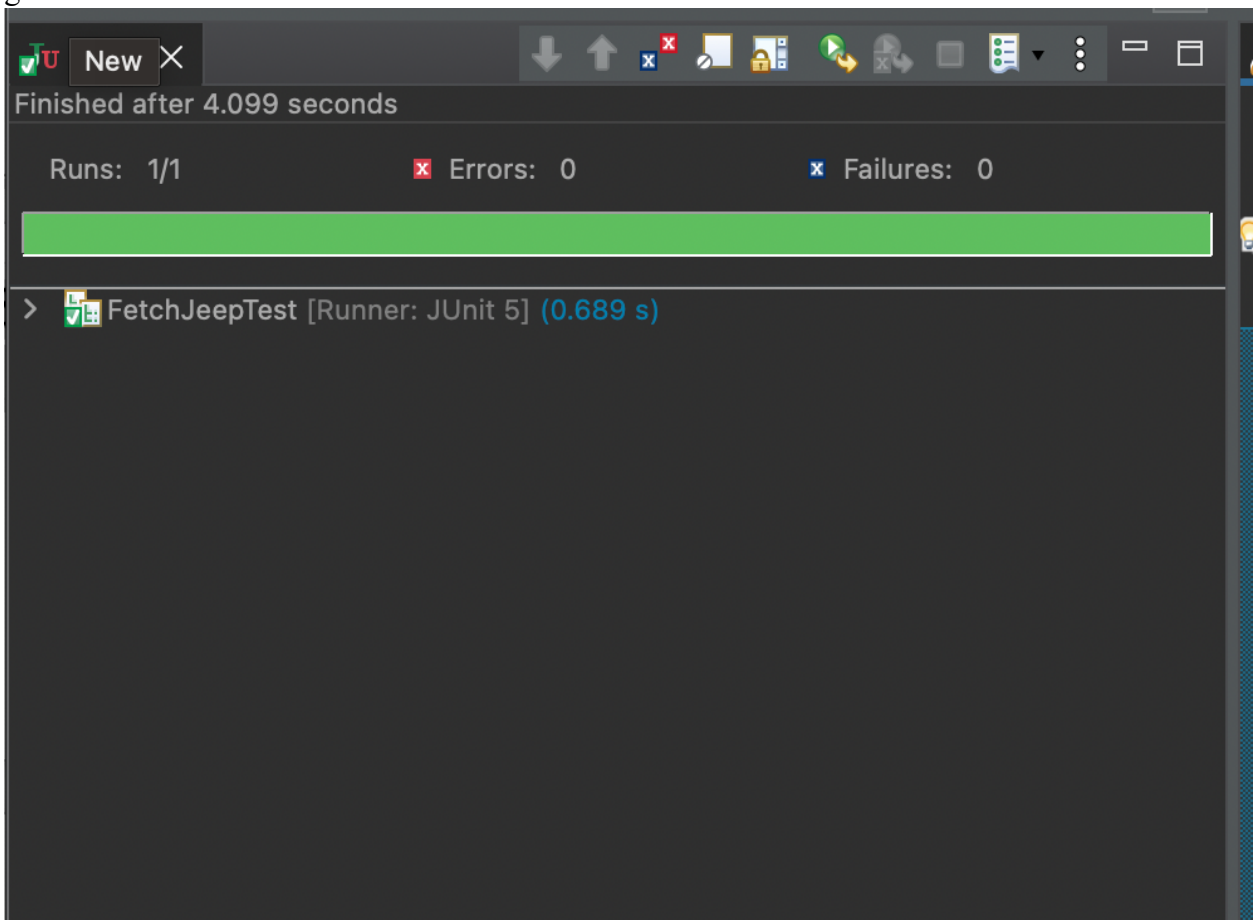
h) Call the `query` method on the `NamedParameterJdbcTemplate` instance variable to return a list of Jeep model objects. Use a RowMapper to map each row of the result set. Remember to convert `modelId` to a `JeepModel`. See the video for details. Produce a

screenshot to show the complete method in the implementation class.

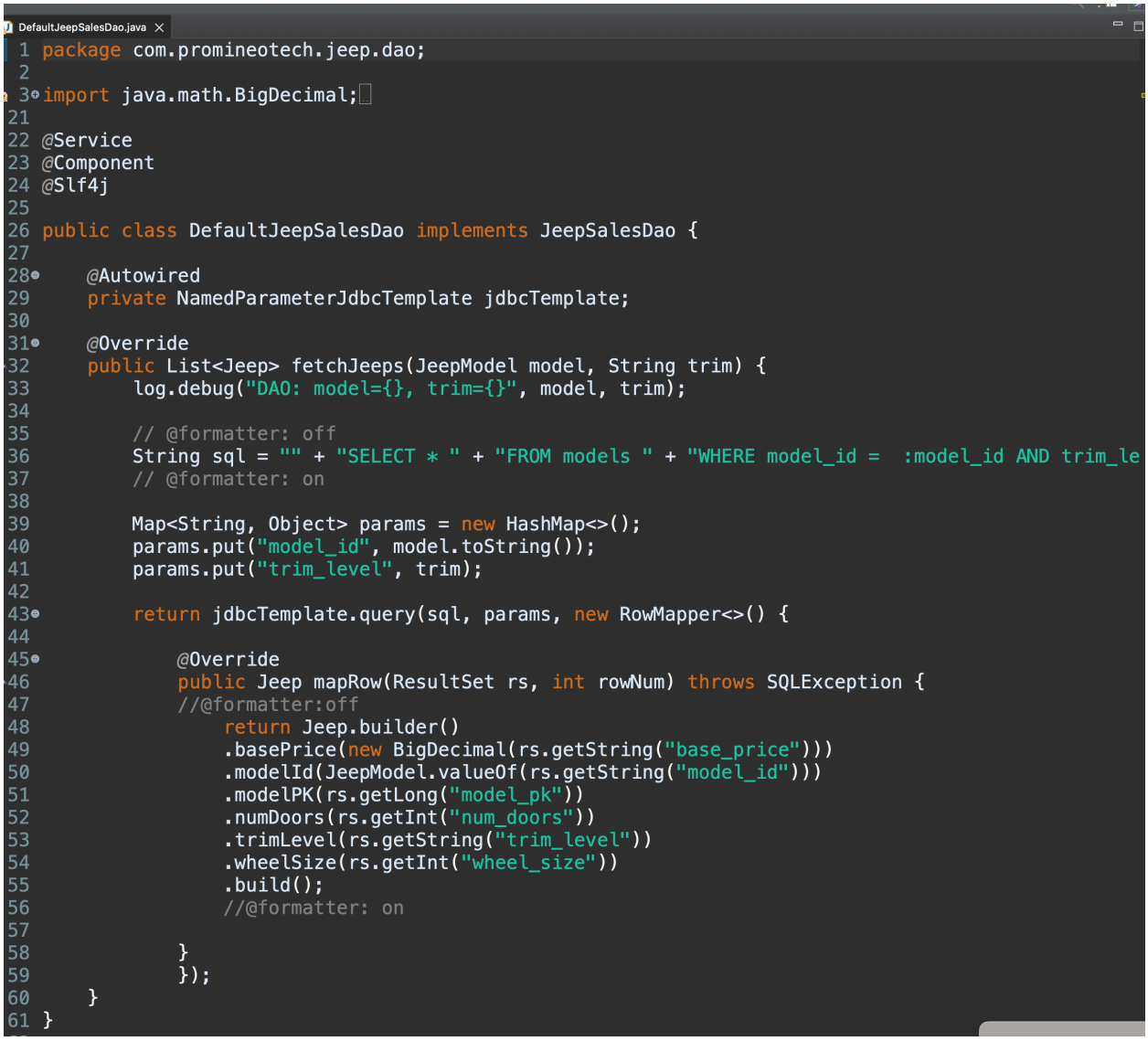
```
import java.util.Map;
Restore
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.jdbc.core.RowMapper;
import org.springframework.jdbc.core.namedparam.NamedParameterJdbcTemplate;
import org.springframework.stereotype.Component;
import org.springframework.stereotype.Service;

import com.mysql.cj.log.Log;
import com.promineotech.jeep.entity.Jeep;
import com.promineotech.jeep.entity.JeepModel;

import lombok.extern.slf4j.Slf4j;

@Service
@Component
@Slf4j

public class DefaultJeepSalesDao implements JeepSalesDao {

    @Autowired
    private NamedParameterJdbcTemplate jdbcTemplate;

    @Override
    public List<Jeep> fetchJeeps(JeepModel model, String trim) {
        log.debug("DAO: model={}, trim={}", model, trim);

        // @formatter: off
        String sql = "" + "SELECT * " + "FROM models " + "WHERE model_id =  :model_id AND trim_level
        // @formatter: on

        Map<String, Object> params = new HashMap<>();
        params.put("model_id", model.toString());
        params.put("trim_level", trim);

        return jdbcTemplate.query(sql, params, new RowMapper<>() {

            @Override
            public Jeep mapRow(ResultSet rs, int rowNum) throws SQLException {
            //@formatter:off
                return Jeep.builder()
                .basePrice(new BigDecimal(rs.getString("base_price")))
                .modelId(JeepModel.valueOf(rs.getString("model_id")))
                .modelPK(rs.getLong("model_pk"))
                .numDoors(rs.getInt("num_doors"))
                .trimLevel(rs.getString("trim_level"))
                .wheelSize(rs.getInt("wheel_size"))
                .build();
                //@formatter: on

            }
            });
    }
}
```

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.



Screenshots of Code:



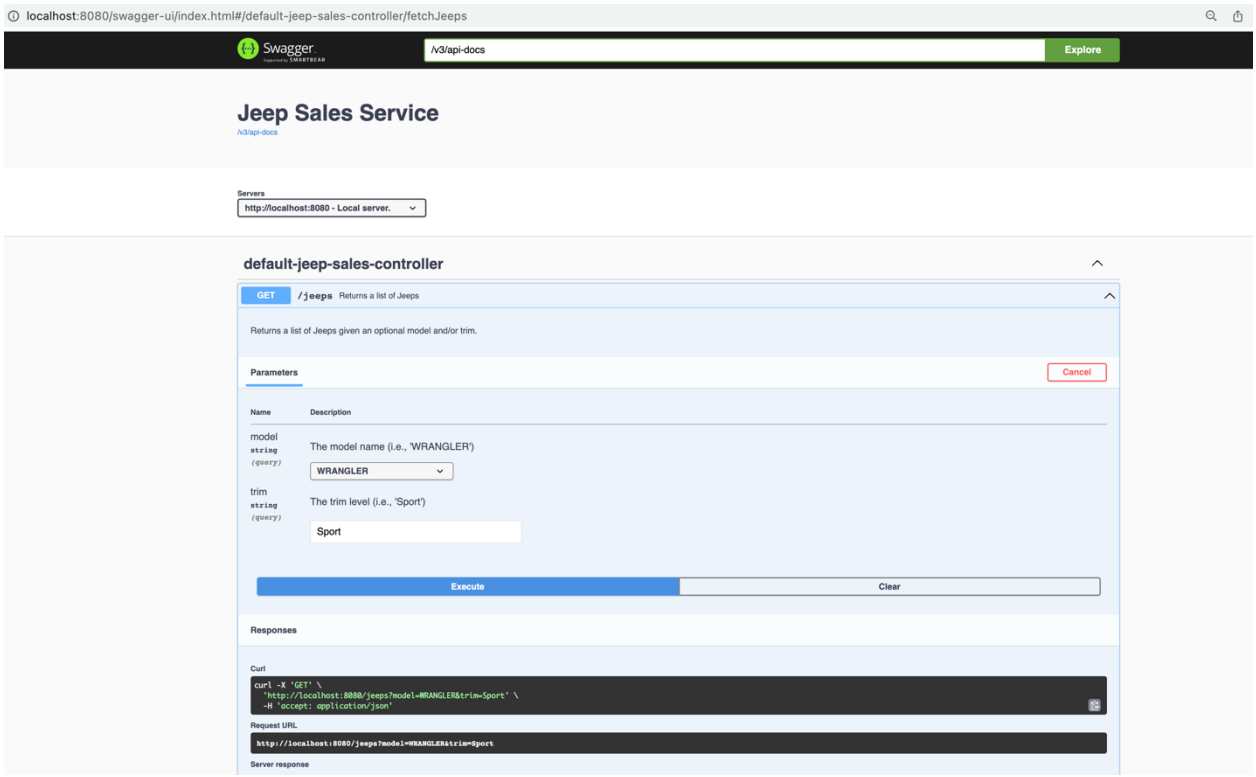
DefaultJeepSalesDao.java

```java
package com.promineotech.jeep.dao;

import java.math.BigDecimal;

@Service
@Component
@Slf4j

public class DefaultJeepSalesDao implements JeepSalesDao {

    @Autowired
    private NamedParameterJdbcTemplate jdbcTemplate;

    @Override
    public List<Jeep> fetchJeeps(JeepModel model, String trim) {
        log.debug("DAO: model={}, trim={}", model, trim);

        // @formatter: off
        String sql = "" + "SELECT * " + "FROM models " + "WHERE model_id =  :model_id AND trim_le
        // @formatter: on

        Map<String, Object> params = new HashMap<>();
        params.put("model_id", model.toString());
        params.put("trim_level", trim);

        return jdbcTemplate.query(sql, params, new RowMapper<>() {

            @Override
            public Jeep mapRow(ResultSet rs, int rowNum) throws SQLException {
            //@formatter:off
                return Jeep.builder()
                .basePrice(new BigDecimal(rs.getString("base_price")))
                .modelId(JeepModel.valueOf(rs.getString("model_id")))
                .modelPK(rs.getLong("model_pk"))
                .numDoors(rs.getInt("num_doors"))
                .trimLevel(rs.getString("trim_level"))
                .wheelSize(rs.getInt("wheel_size"))
                .build();
                //@formatter: on

            }
            });
    }
}
```

DefaultJeepSalesDao.java | JeepSalesDao.java

```java
package com.promineotech.jeep.dao;

import java.util.List;

public interface JeepSalesDao {

    /**
     * @param model
     * @param trim
     * @return
     */
    List<Jeep> fetchJeeps(JeepModel model, String trim);

}

```

Screenshots of Running Application:


localhost:8080/swagger-ui/index.html#/default-jeep-sales-controller/fetchJeeps
Swagger
/v3/api-docs
Explore
Jeep Sales Service
/v3/api-docs
Servers
http://localhost:8080 - Local server.
default-jeep-sales-controller
GET
/jeeps
Returns a list of Jeeps
Returns a list of Jeeps given an optional model and/or trim.
Parameters
Cancel
Name
Description
model
string
(query)
The model name (i.e., 'WRANGLER')
WRANGLER
trim
string
(query)
The trim level (i.e., 'Sport')
Sport
Execute
Clear
Responses
Curl
curl -X 'GET' \
'http://localhost:8080/jeeps?model=WRANGLER&trim=Sport' \
-H 'accept: application/json'
Request URL
http://localhost:8080/jeeps?model=WRANGLER&trim=Sport
Server response

Code	Details

200	**Response body**
	`[ { "modelId": "WRANGLER", "trimLevel": "Sport", "numDoors": 2, "wheelSize": 17, "basePrice": 28475 }, { "modelId": "WRANGLER", "trimLevel": "Sport", "numDoors": 4, "wheelSize": 17, "basePrice": 31975 } ]`
	Download
	Response headers
	`connection: keep-alive`
	`content-type: application/json`
	`date: Wed,05 Oct 2022 22:10:05 GMT`
	`keep-alive: timeout=60`
	`transfer-encoding: chunked`

Responses

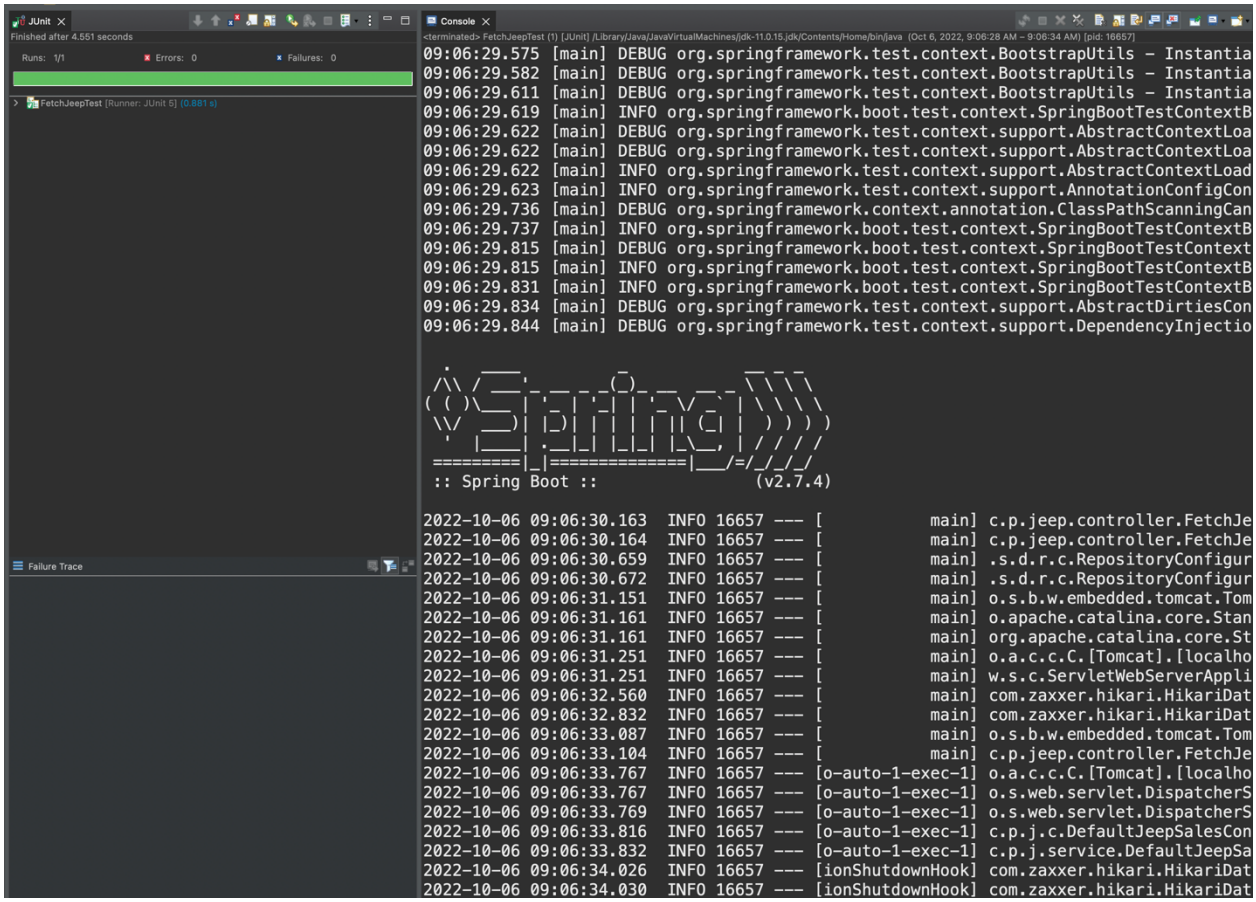
Code	Description	Links
200	A list of Jeeps is returned Media type: application/json Controls Accept header. **Example Value** | Schema `{ "modelId": "GRAND_CHEROKEE", "trimLevel": "string", "numDoors": 0, "wheelSize": 0, "basePrice": 0 }`	*No links*
400	The request parameters are invalid Media type: application/json	*No links*
404	No Jeeps were found with the input criteria Media type: application/json	*No links*
500	An unplanned error occurred Media type: application/json	*No links*

Schemas

Jeep {

modelId	string Enum: > Array [7]
trimLevel	string
numDoors	integer($int32)
wheelSize	integer($int32)
basePrice	number

}



URL to GitHub Repository:

https://github.com/geraldinedepaul17/BackEndClass/tree/main/Week%2015