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:

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

- 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).
- 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

DefaultJeepSalesService.java | JeepSalesDao.java | DefaultJeepSalesDao.java | FetchJe

```
19
20  @Component
21  @Slf4j
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.debug("DAO: model={}, trim={}", model, trim);
30
31          String sql = ""
32                  + "SELECT * "
33                  + "FROM models "
34                  + "WHERE model_id = :model_id AND trim_level = :trim_level";
35
36          Map<String, Object> params = new HashMap<>();
37          params.put("model_id", model.toString());
38          params.put("trim_level", trim);
39
```

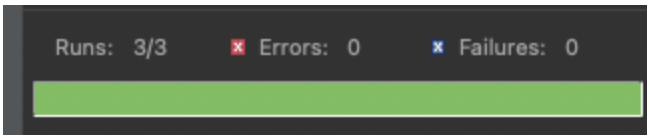
console.

```
c.p.j.service.DefaultJeepSalesService    : The fetchJeeps method was called v
c.p.jeep.dao.DefaultJeepSalesDao         : DAO: model=WRANGLER, trim=Sport
```

- 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.

```
19
20  @Component
21  @Slf4j
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.debug("DAO: model={}, trim={}", model, trim);
30
31          String sql = ""
32                  + "SELECT * "
33                  + "FROM models "
34                  + "WHERE model_id = :model_id AND trim_level = :trim_level";
35
36          Map<String, Object> params = new HashMap<>();
37          params.put("model_id", model.toString());
38          params.put("trim_level", trim);
39
40          return jdbcTemplate.query(sql, params,
41                  new RowMapper<>() {
42
43                      @Override
44                      public Jeep mapRow(ResultSet rs, int rowNum) throws SQLException {
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  }
```

- 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.
- Run the test to produce a green status bar. Produce a screenshot showing the test and the green status bar.



Screenshots of Code:

GlobalErrorHandler.java | FetchJeepTest.java

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

import static org.assertj.core.api.Assertions.assertThat;

@SpringBootTest(webEnvironment = WebEnvironment.RANDOM_PORT)
@ActiveProfiles("test")
@Sql(scripts = { "classpath:flyway/migrations/V1.0__Jeep_Schema.sql",
        "classpath:flyway/migrations/V1.1__Jeep_Data.sql" }, config = @SqlConfig(encoding = "utf-8"))

class FetchJeepTest {

    @Nested
    @SpringBootTest(webEnvironment = WebEnvironment.RANDOM_PORT)
    @ActiveProfiles("test")
    @Sql(scripts = { "classpath:flyway/migrations/V1.0__Jeep_Schema.sql",
            "classpath:flyway/migrations/V1.1__Jeep_Data.sql" }, config = @SqlConfig(encoding = "utf-8"))
    class TestsThatDoNotPolluteTheApplicationContext extends FetchJeepTestSupport {
        @Test
        void testThatJeepsAreReturnedWhenAValidModelAndTrimAreSupplied() {

            JeepModel model = JeepModel.WRANGLER;
            String trim = "Sport";
            String uri = String.format("%s?model=%s&trim=%s", getBaseUri(), model, trim);

            // when: a connection is made to the URI
            ResponseEntity<List<Jeep>> response = getRestTemplate().exchange(uri, HttpMethod.GET, null,
                    new ParameterizedTypeReference<>() {
                    });

            // then: a success (OK - 200) status code is returned
            assertThat(response.getStatusCode()).isEqualTo(HttpStatus.OK);

            // and: the actual list returned is the same as the expected list
            List<Jeep> actual = response.getBody();
            List<Jeep> expected = buildExpected();

            actual.forEach(jeep -> jeep.setModelPK(null));

            assertThat(actual).isEqualTo(expected);
        }

        @Test
        void testThatAnErrorMessageIsReturnedWhenAnInvalidTrimIsSupplied() {

            JeepModel model = JeepModel.WRANGLER;
            String trim = "Invalid Value";
            String uri = String.format("%s?model=%s&trim=%s", getBaseUri(), model, trim);

            // when: a connection is made to the URI
            ResponseEntity<Map<String, Object>> response = getRestTemplate().exchange(uri, HttpMethod.GET, null,
                    new ParameterizedTypeReference<>() {
                    });

            // then: a not found (404) status code is returned
            assertThat(response.getStatusCode()).isEqualTo(HttpStatus.NOT_FOUND);
```

GlobalErrorHandler.java | FetchJeepTest.java

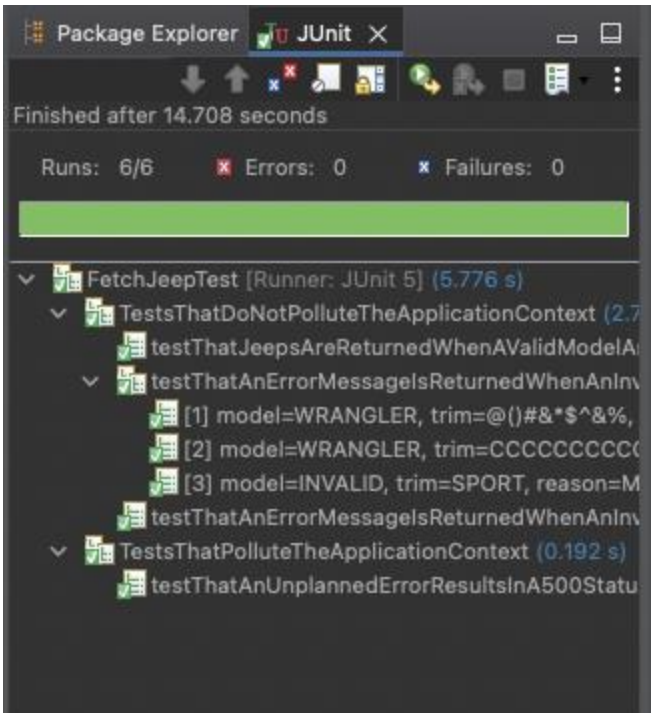
```java
            assertThat(response.getStatusCode()).isEqualTo(HttpStatus.NOT_FOUND);

            // and: an error message is returned
            Map<String, Object> error = response.getBody();

            assertErrorMessageValid(error, HttpStatus.NOT_FOUND);
        }

        @ParameterizedTest
        @MethodSource("com.promineotech.jeep.controller.FetchJeepTest#parametersForInvalidInput")
        void testThatAnErrorMessageIsReturnedWhenAnInvalidValueIsSupplied(String model, String trim, String reason) {

            String uri = String.format("%s?model=%s&trim=%s", getBaseUri(), model, trim);

            // when: a connection is made to the URI
            ResponseEntity<Map<String, Object>> response = getRestTemplate().exchange(uri, HttpMethod.GET, null,
                    new ParameterizedTypeReference<>() {
                    });

            // then: a not found (404) status code is returned
            assertThat(response.getStatusCode()).isEqualTo(HttpStatus.BAD_REQUEST);

            // and: an error message is returned
            Map<String, Object> error = response.getBody();

            assertErrorMessageValid(error, HttpStatus.BAD_REQUEST);
        }
    }

    static Stream<Arguments> parametersForInvalidInput() {
        return Stream.of(arguments("WRANGLER", "@()#&*$^&%", "trim contains non-alpha-numeric characters."),
                arguments("WRANGLER", "C".repeat(Constants.TRIM_MAX_LENGTH + 1), "Trim length too long"),
                arguments("INVALID", "SPORT", "Model is not enum value"));
    }

    @Nested
    @SpringBootTest(webEnvironment = WebEnvironment.RANDOM_PORT)
    @ActiveProfiles("test")
    @Sql(scripts = { "classpath:flyway/migrations/V1.0__Jeep_Schema.sql",
            "classpath:flyway/migrations/V1.1__Jeep_Data.sql" }, config = @SqlConfig(encoding = "utf-8"))
    class TestsThatPolluteTheApplicationContext extends FetchJeepTestSupport {
        @MockBean
        private JeepSalesService jeepSalesService;

        @Test
        void testThatAnUnplannedErrorResultsInA500Status() {

            JeepModel model = JeepModel.WRANGLER;
            String trim = "Invalid";
            String uri =
                    String.format("%s?model=%s&trim=%s", getBaseUri(), model, trim);

            doThrow(new RuntimeException("Ouch!")).when(jeepSalesService)
                .fetchJeeps(model, trim);
```

```
                String.format("%s?model=%s&trim=%s", getBaseUri(), model, trim);

        doThrow(new RuntimeException("Ouch!")).when(jeepSalesService)
            .fetchJeeps(model, trim);

        // when: a connection is made to the URI
        ResponseEntity<Map<String, Object>> response = getRestTemplate().exchange(uri, HttpMethod.GET, null,
                new ParameterizedTypeReference<>() {
                });

        // then: an internal server error is returned
        assertThat(response.getStatusCode()).isEqualTo(HttpStatus.INTERNAL_SERVER_ERROR);

        // and: an error message is returned
        Map<String, Object> error = response.getBody();

        assertErrorMessageValid(error, HttpStatus.INTERNAL_SERVER_ERROR);
    }

  }
}
```

GlobalErrorHandler.java | FetchJeepTest.java

```
@Slf4j
public class GlobalErrorHandler {

    private enum LogStatus {
        STACK_TRACE, MESSAGE_ONLY
    }
    @ExceptionHandler(MethodArgumentTypeMismatchException.class)
    @ResponseStatus(code = HttpStatus.BAD_REQUEST)
    public Map<String, Object> handleMethodArgumentTypeMismatchException(
            MethodArgumentTypeMismatchException e, WebRequest webRequest) {
        return createExceptionMessage(e, HttpStatus.BAD_REQUEST, webRequest,
                LogStatus.MESSAGE_ONLY);
    }

    @ExceptionHandler(ConstraintViolationException.class)
    @ResponseStatus(code = HttpStatus.BAD_REQUEST)
    public Map<String, Object> handleConstraintViolationException(
            ConstraintViolationException e, WebRequest webRequest) {
        return createExceptionMessage(e, HttpStatus.BAD_REQUEST, webRequest, LogStatus.MESSAGE_ONLY);
    }

    @ExceptionHandler(NoSuchElementException.class)
    @ResponseStatus(code = HttpStatus.NOT_FOUND)
    public Map<String, Object> handleNoSuchElementException(
            NoSuchElementException e, WebRequest webRequest) {
        return createExceptionMessage(e, HttpStatus.NOT_FOUND, webRequest, LogStatus.MESSAGE_ONLY);
    }

    @ExceptionHandler(Exception.class)
    @ResponseStatus(code = HttpStatus.INTERNAL_SERVER_ERROR)
    public Map<String, Object> handleException(Exception e, WebRequest webRequest) {
        return createExceptionMessage(e, HttpStatus.INTERNAL_SERVER_ERROR, webRequest, LogStatus.STACK_TRACE);
    }

    private Map<String, Object> createExceptionMessage(Exception e,
            HttpStatus status, WebRequest webRequest, LogStatus logStatus) {
        Map<String, Object> error = new HashMap<>();
        String timestamp =
                ZonedDateTime.now().format(DateTimeFormatter.RFC_1123_DATE_TIME);

        if(webRequest instanceof ServletWebRequest) {
            error.put("uri", ((ServletWebRequest)webRequest).getRequest().getRequestURI());
        }

        error.put("message", e.toString());
        error.put("status code", status.value());
        error.put("timestamp", timestamp);
        error.put("reason", status.getReasonPhrase());

        if(logStatus == LogStatus.MESSAGE_ONLY) {
            log.error("Exception: {}", e.toString());
        } else {
            log.error("Exception:", e);
        }

        return error;
    }
}
```

Screenshots of Running Application:



URL to GitHub Repository:

https://github.com/Asosa809/Week15