


# 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>

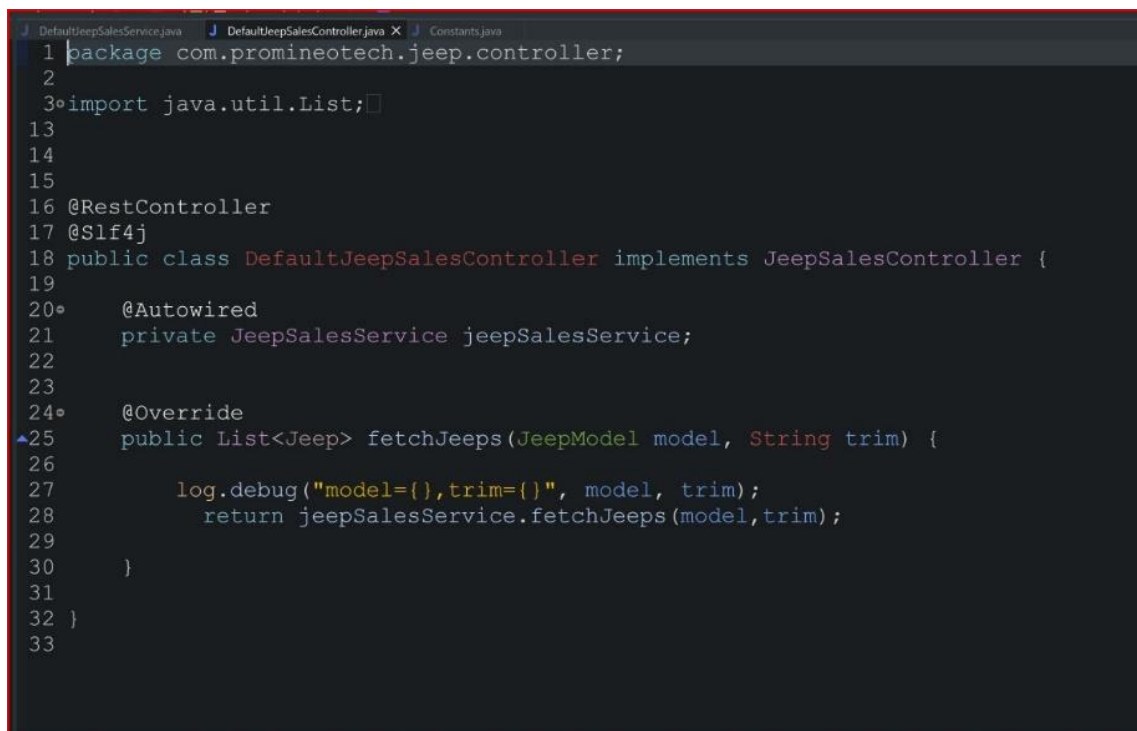
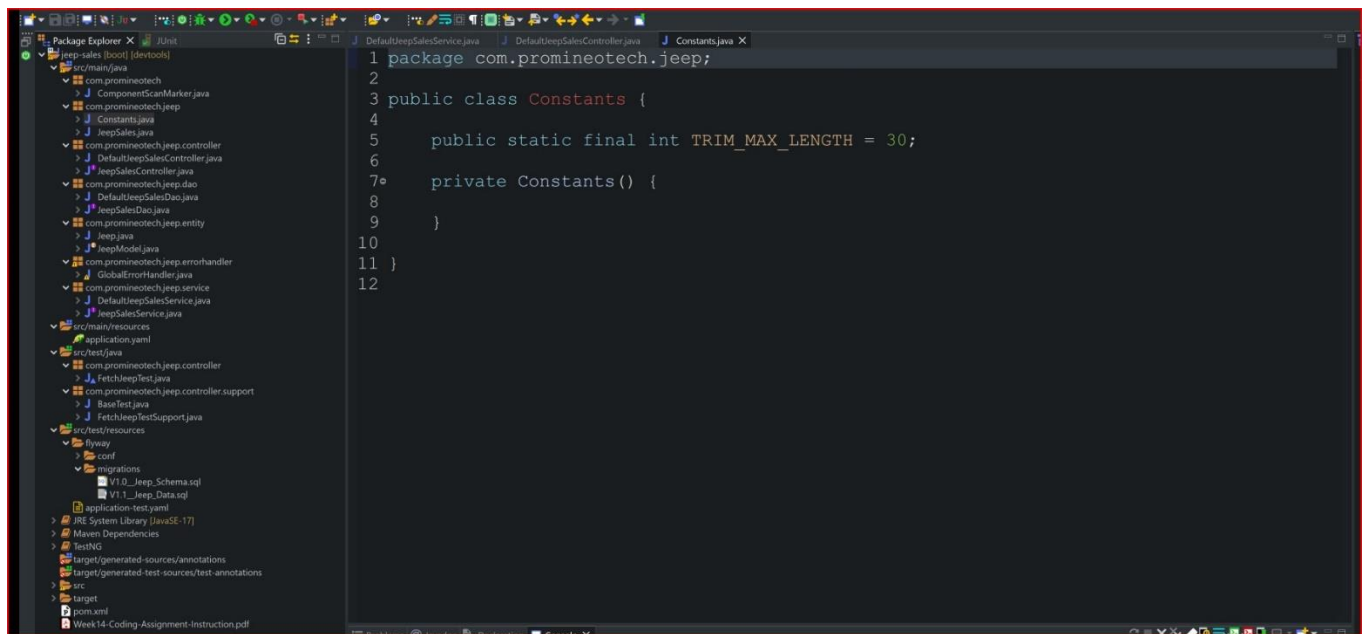
## Coding Steps:

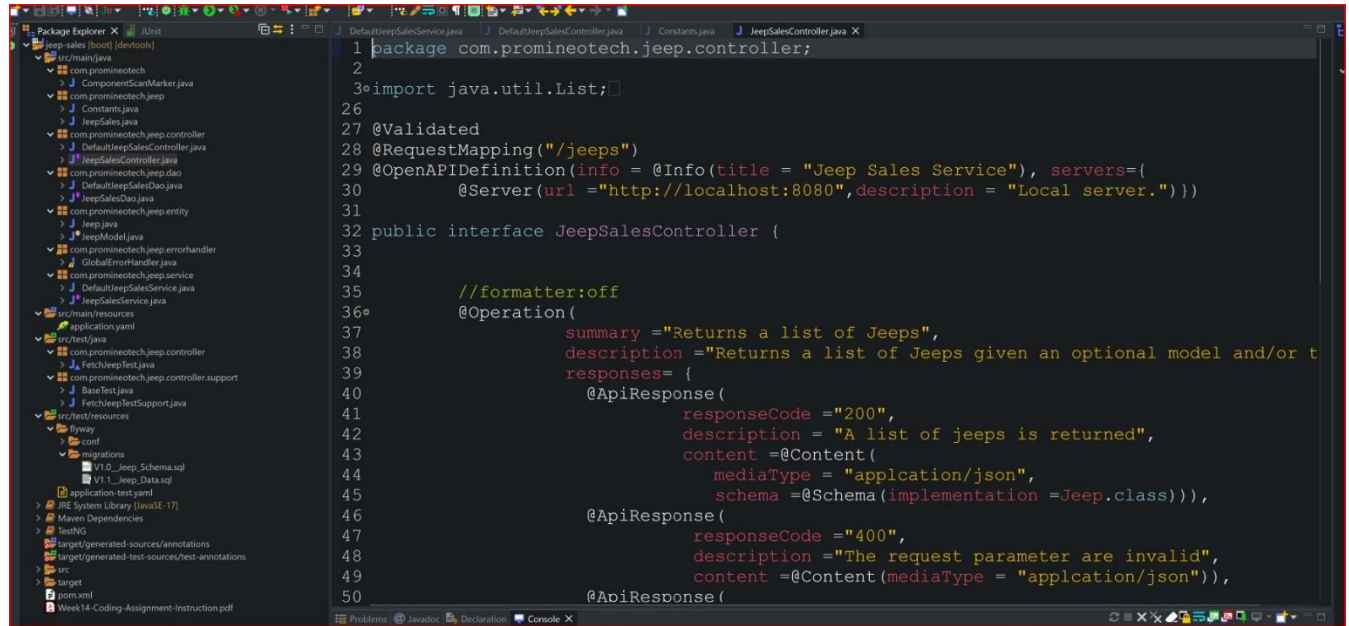
- 1) In the application you've been building add a DAO layer:
  - a) Add the package, `com.promineotech.jeepp.dao`.
  - b) In the new package, create an interface named `JeepSalesDao`.
  - c) In the same package, create a class named `DefaultJeepSalesDao` that implements `JeepSalesDao`.
  - d) Add a 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).
- 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. 
  - c) In `DefaultJeepSalesDao`, inject an instance variable of type `NamedParameterJdbcTemplate`.
  - d) 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.
  - e) 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());`)
  - 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. 
- 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:



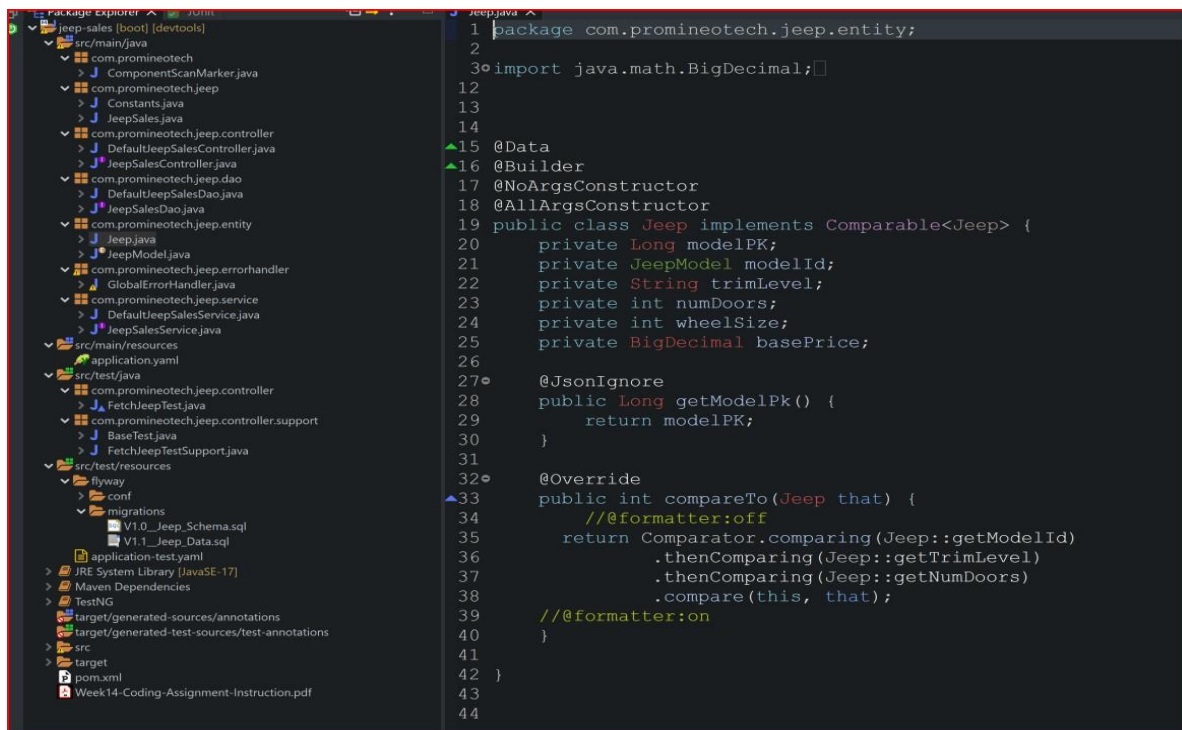


```

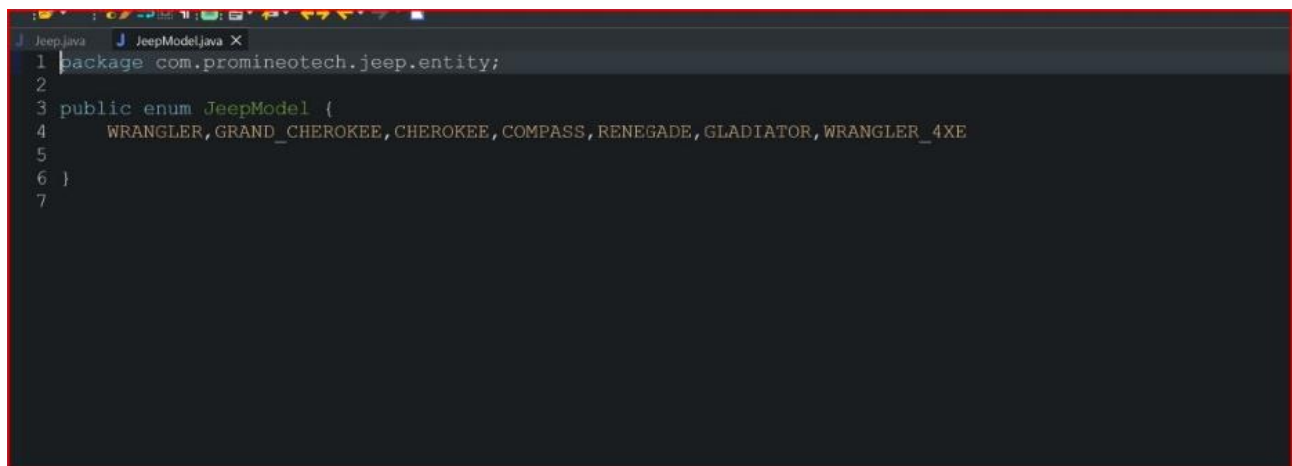
48         description = "The request parameter are invalid",
49         content = @Content(mediaType = "application/json")),
50     @ApiResponse(
51         responseCode = "404",
52         description = "No jeep were found with the input criteria",
53         content = @Content(mediaType = "application/json")),
54     @ApiResponse(
55         responseCode = "500",
56         description = "An Unplanned error occured",
57         content = @Content(mediaType = "application/json")),
58     ),
59     parameters = {
60         @Parameter(
61             name = "model",
62             allowEmptyValue = false,
63             required = false,
64             description = "The model name(i.e., 'WRANGLER')",
65         ),
66         @Parameter(
67             name = "trim",
68             allowEmptyValue = false,
69             required = false,
70             description = "The trim level(i.e., 'Sport')",
71         )
72     },
73     @GetMapping
74     @ResponseStatus(code = HttpStatus.OK)
75     List<Jeep> fetchJeeps(
76         @RequestParam(required = false)
77         JeepModel model,
78
79         @Length(max=Constants.TRIM_MAX_LENGTH)
80         @Pattern(regexp = "[\\w\\s]*")
81         @RequestParam(required = false)
82         String trim);
83 // @formatter:on
84 }

```



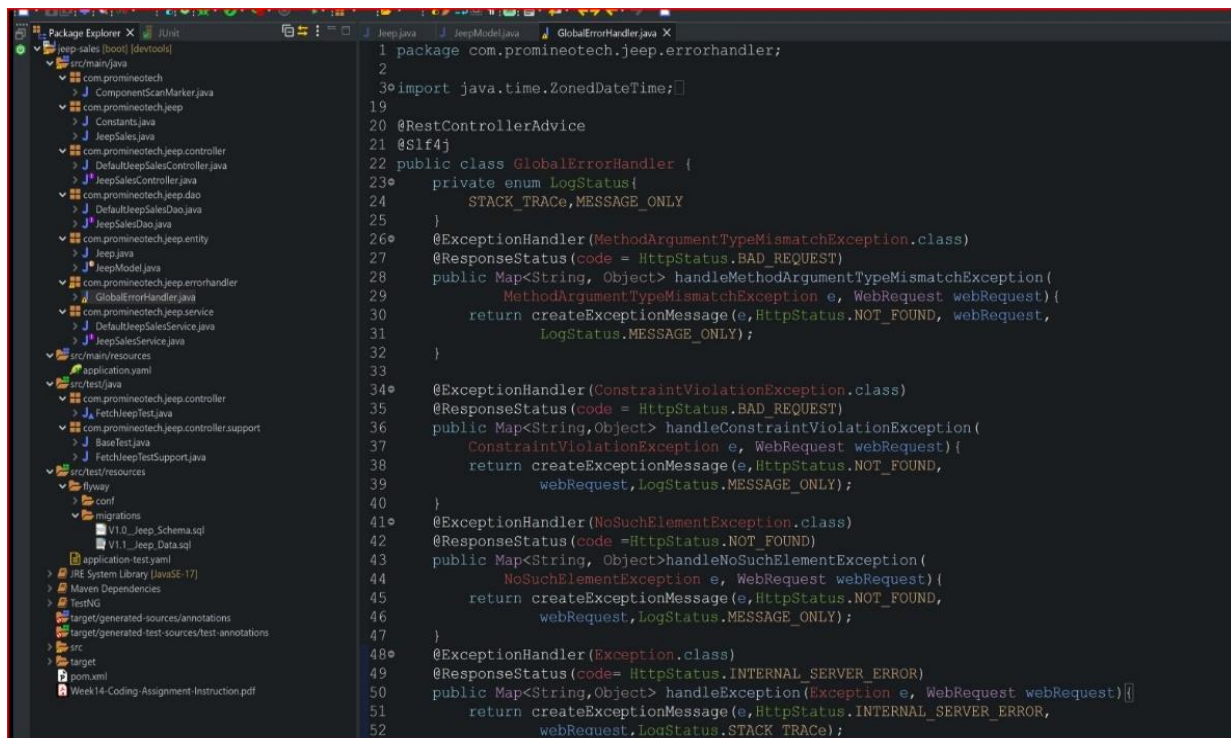


```
1 package com.promineotech.jeep.entity;
2
3 import java.math.BigDecimal;
4
5
6
7
8
9
10
11
12
13
14
15 @Data
16 @Builder
17 @NoArgsConstructor
18 @AllArgsConstructor
19 public class Jeep implements Comparable<Jeep> {
20     private Long modelPK;
21     private JeepModel modelId;
22     private String trimLevel;
23     private int numDoors;
24     private int wheelSize;
25     private BigDecimal basePrice;
26
27     @JsonIgnore
28     public Long getModelPk() {
29         return modelPK;
30     }
31
32     @Override
33     public int compareTo(Jeep that) {
34         // @formatter:off
35         return Comparator.comparing(Jeep::getModelId)
36             .thenComparing(Jeep::getTrimLevel)
37             .thenComparing(Jeep::getNumDoors)
38             .compare(this, that);
39         // @formatter:on
40     }
41
42 }
43
44
```



```
1 package com.promineotech.jeep.entity;
2
3 public enum JeepModel {
4     WRANGLER, GRAND_CHEROKEE, CHEROKEE, COMPASS, RENEGADE, GLADIATOR, WRANGLER_4XE
5
6 }
7
```





```
1 package com.promineotech.jee.errorhandler;
2
3 import java.time.ZonedDateTime;
4
5 @RestControllerAdvice
6 @Slf4j
7 public class GlobalExceptionHandler {
8     private enum LogStatus {
9         STACK_TRACE, MESSAGE_ONLY
10    }
11
12    @ExceptionHandler(MethodArgumentTypeMismatchException.class)
13    @ResponseStatus(code = HttpStatus.BAD_REQUEST)
14    public Map<String, Object> handleMethodArgumentTypeMismatchException(
15        MethodArgumentTypeMismatchException e, WebRequest webRequest) {
16        return createExceptionMessage(e, HttpStatus.NOT_FOUND, webRequest,
17            LogStatus.MESSAGE_ONLY);
18    }
19
20    @ExceptionHandler(ConstraintViolationException.class)
21    @ResponseStatus(code = HttpStatus.BAD_REQUEST)
22    public Map<String, Object> handleConstraintViolationException(
23        ConstraintViolationException e, WebRequest webRequest) {
24        return createExceptionMessage(e, HttpStatus.NOT_FOUND,
25            webRequest, LogStatus.MESSAGE_ONLY);
26    }
27
28    @ExceptionHandler(NoSuchElementException.class)
29    @ResponseStatus(code = HttpStatus.NOT_FOUND)
30    public Map<String, Object> handleNoSuchElementException(
31        NoSuchElementException e, WebRequest webRequest) {
32        return createExceptionMessage(e, HttpStatus.NOT_FOUND,
33            webRequest, LogStatus.MESSAGE_ONLY);
34    }
35
36    @ExceptionHandler(Exception.class)
37    @ResponseStatus(code = HttpStatus.INTERNAL_SERVER_ERROR)
38    public Map<String, Object> handleException(Exception e, WebRequest webRequest) {
39        return createExceptionMessage(e, HttpStatus.INTERNAL_SERVER_ERROR,
40            webRequest, LogStatus.STACK_TRACE);
41    }
42
43    private Map<String, Object> createExceptionMessage(Exception e,
44        HttpStatus status, WebRequest webRequest, LogStatus logStatus) {
45        Map<String, Object> error = new HashMap<>();
46        String timestamp =
47            ZonedDateTime.now().format(DateTimeFormatter.RFC_1123_DATE_TIME);
48
49        if (webRequest instanceof ServletWebRequest) {
50            error.put("uri",
51                ((ServletWebRequest) webRequest).getRequest().getRequestURI());
52        }
53        error.put("message", e.toString());
54        error.put("status code", status.value());
55        error.put("timestamp", timestamp);
56        error.put("reason", status.getReasonPhrase());
57
58        if (logStatus == logStatus.MESSAGE_ONLY) {
59            log.error("Exception: {}", e.toString());
60        } else {
61            log.error("Exception:", e);
62        }
63        return error;
64    }
65 }
```

```
52         webRequest, logStatus.STACK_TRACE);
53     }
54     private Map<String, Object> createExceptionMessage(Exception e,
55         HttpStatus status, WebRequest webRequest, LogStatus logStatus) {
56         Map<String, Object> error = new HashMap<>();
57         String timestamp =
58             ZonedDateTime.now().format(DateTimeFormatter.RFC_1123_DATE_TIME);
59
60         if (webRequest instanceof ServletWebRequest) {
61             error.put("uri",
62                 ((ServletWebRequest) webRequest).getRequest().getRequestURI());
63         }
64         error.put("message", e.toString());
65         error.put("status code", status.value());
66         error.put("timestamp", timestamp);
67         error.put("reason", status.getReasonPhrase());
68
69         if (logStatus == logStatus.MESSAGE_ONLY) {
70             log.error("Exception: {}", e.toString());
71         } else {
72             log.error("Exception:", e);
73         }
74         return error;
75     }
76 }
77 }
```



```

1 package com.promineotech.jeepp.service;
2
3 import java.util.Collections;
4
5
6
7
8
9
10
11
12
13
14
15
16
17 @Service
18 @Slf4j
19 public class DefaultJeepSalesService implements JeepSalesService{
20     @Autowired
21     private JeepSalesDao jeepSalesDao;
22
23     @Transactional(readOnly = true)
24     @Override
25     public List<Jeep> fetchJeeps(JeepModel model, String trim) {
26         log.info("The fetchJeeps method was called with model={} and trim={}",
27             model,trim);
28
29         List<Jeep> jeeps =jeepSalesDao.fetchJeeps(model,trim);
30
31         if(jeeps.isEmpty()) {
32             String msg = String.format("no jeeps found with model =%s and trim =%s", model,trim);
33             throw new NoSuchElementException(msg);
34         }
35         Collections.sort(jeeps);
36         return jeeps;
37     }
38 }

```

```

1 package com.promineotech.jeepp.service;
2
3 import java.util.List;
4
5
6
7
8 public interface JeepSalesService {
9
10     List<Jeep> fetchJeeps(JeepModel model, String trim);
11
12 }
13

```



```

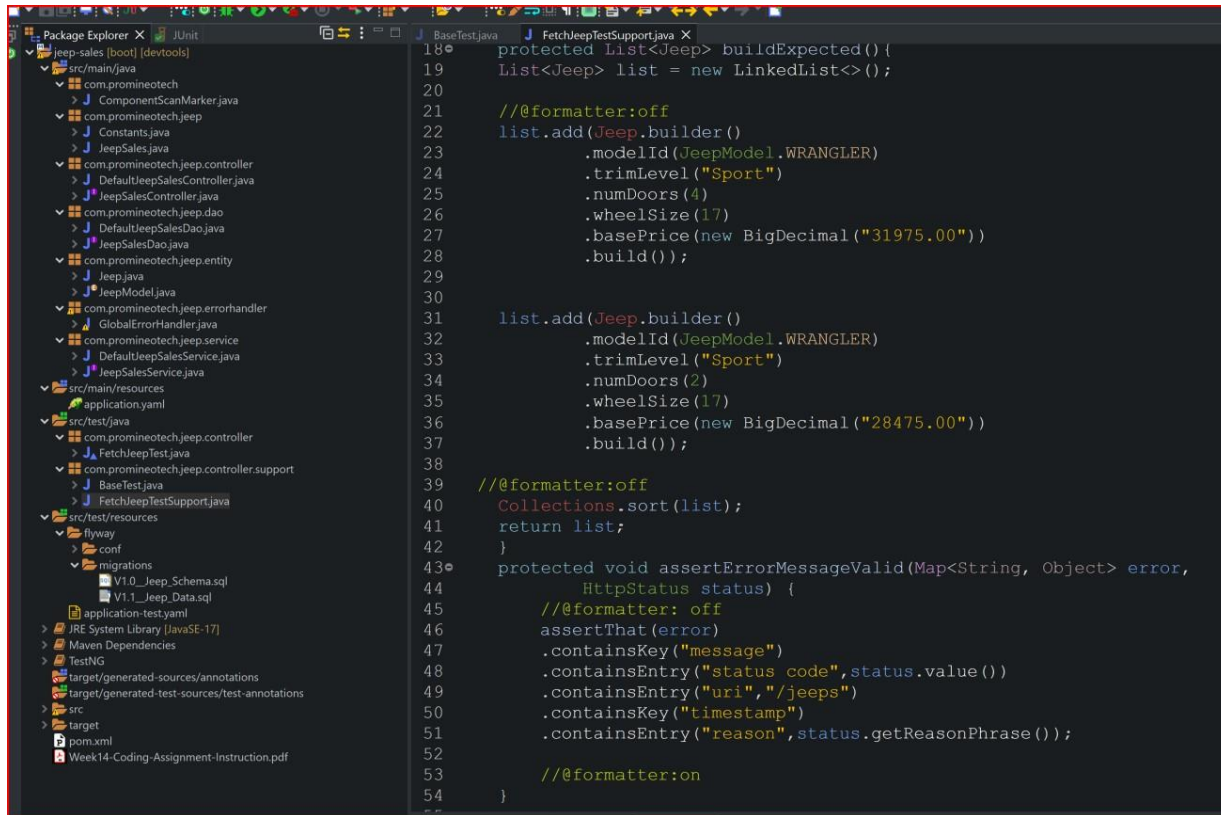
J FetchJeepTest.java X
98      //And: an error message is returned
99
100     Map<String,Object> error = response.getBody();
101     assertErrorMessageValid(error,HttpStatus.NOT_FOUND);
102 }
103 }
104
105 static Stream<Arguments>parametersForInvalidInput() {
106     //formatter: off
107     return Stream.of(
108         arguments("WRANGLER", "@#$$^&&%", "Trim contains non-alpha-numeric chars "),
109         arguments("WRANGLER", "C".repeat(Constants.TRIM_MAX_LENGTH +1), "Trim length too long"),
110         arguments("INVALID", "Sport", "Model is not enum value")
111     );
112 }
113
114 @Nested
115 @SpringBootTest(webEnvironment = WebEnvironment.RANDOM_PORT)
116 @ActiveProfiles("test")
117 @Sql(
118     scripts = {"classpath:flyway/migrations/V1.0__Jeep_Schema.sql",
119               "classpath:flyway/migrations/V1.1__Jeep_Data.sql"},
120     config = @SqlConfig(encoding = "utf-8"))
121 class TestThatPolluteTheApplicationContext extends FetchJeepTestSupport{
122     @MockBean
123     private JeepSalesService jeepSalesService;
124
125     @Test
126     void testThatUnPlannedErrorResultsInA500Status() {
127         //Given: a valid model, trim and URI
128         JeepModel model = JeepModel.WRANGLER;
129         String trim = "Invalid";
130         String uri = String.format("%s?model=%s&trim=%s",getBaseUri(),model,trim);
131
132         doThrow(new RuntimeException("Ouch!")).when(jeepSalesService).fetchJeeps(
133             model,trim);
134     }
135 }

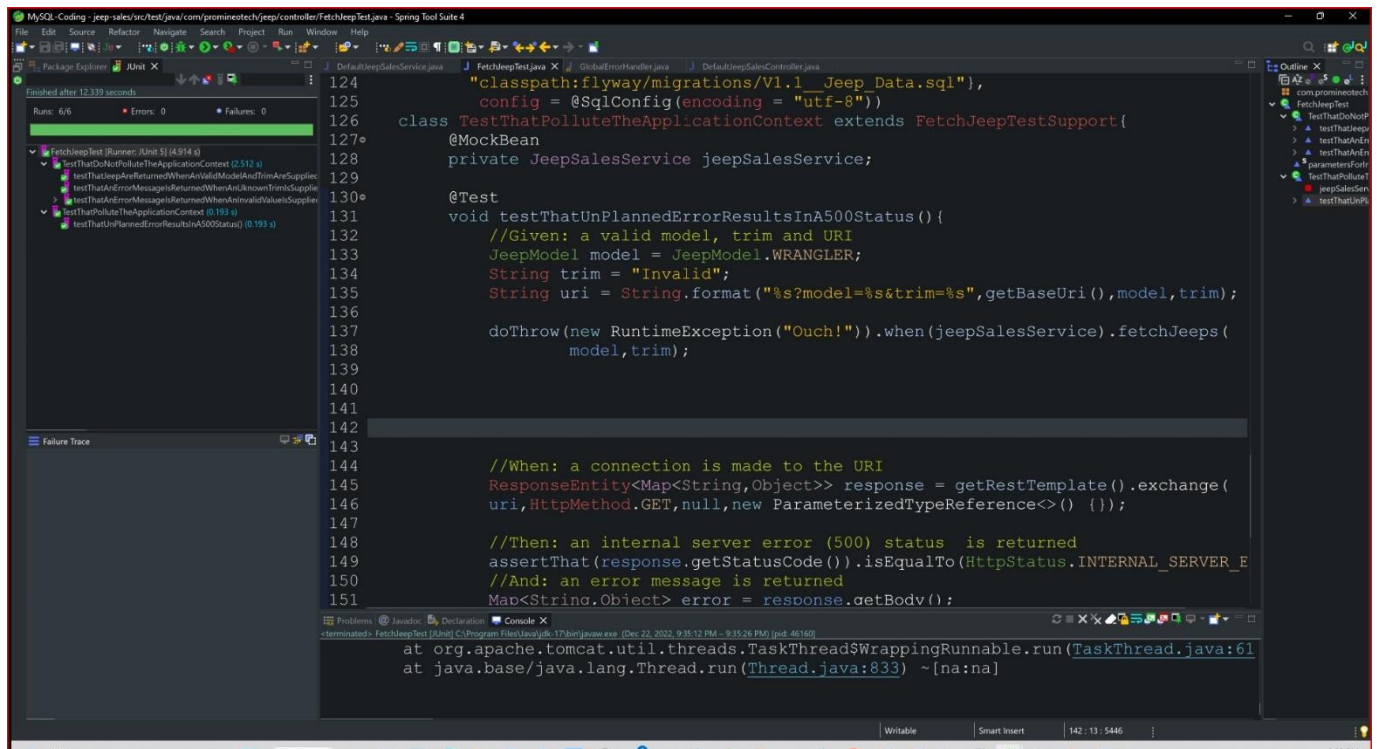
```

```

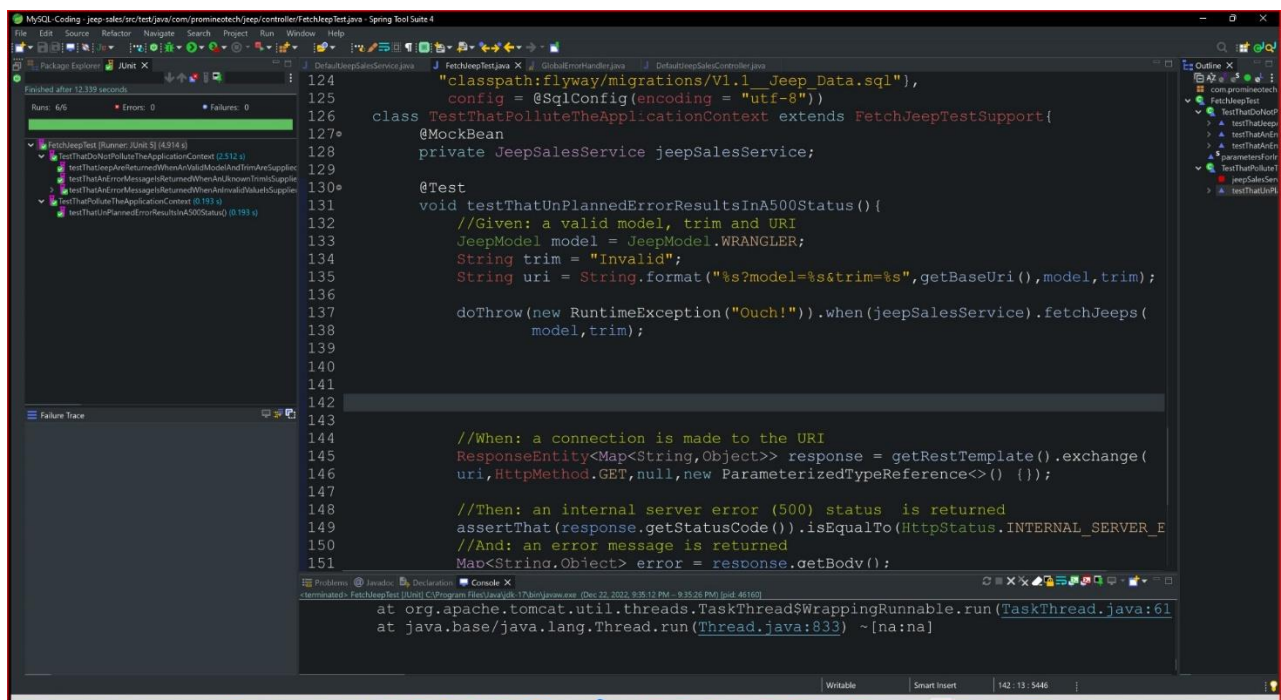
J BaseTest.java X
1 package com.promineotech.jeep.controller.support;
2
3 import org.springframework.beans.factory.annotation.Autowired;
4
5
6
7
8
9 public class BaseTest {
10
11
12     @LocalServerPort
13     private int serverPort;
14
15     @Autowired
16     @Getter
17     private TestRestTemplate restTemplate;
18     protected String getBaseUri() {
19         return String.format("http://localhost:%d/jeeps",serverPort);
20     }
21 }
22
23
24

```





## Screenshots of Running Application:



**URL to GitHub Repository: <https://github.com/ghide/week14-jEEP-sales-spring-boot-project>.git**