


Web API Design with Spring Boot Week 4 Coding Assignment

- 1)
- 2) Create a new integration test class to test a Jeep order named `CreateOrderTest.java`. Create this class in `src/test/java` in the `com.promineotech.jeep.controller` package.
 - a)
 - b)
 - c) Produce a screenshot of the `createOrderBody()` method. 

```

45 @Test
46 void testCreateOrderReturnsSuccess201() {
47
48     HttpHeaders headers = new HttpHeaders();
49     headers.setContentType(MediaType.APPLICATION_JSON);
50
51     String body = createOrderBody();
52     String uri = String.format("http://localhost:%d/orders", serverPort);
53     HttpEntity<String> bodyEntity = new HttpEntity<>(body, headers);
54     ParameterizedTypeReference<Order> typeRef =
55         new ParameterizedTypeReference<>() {};
56
57     ResponseEntity<Order> response =
58         restTemplate.exchange(uri, HttpMethod.POST, bodyEntity, typeRef);
59     //Did the POST call succeed?
60     assertThat(response.getStatusCode()).isEqualTo(HttpStatus.CREATED);
61     //Did the POST call actually create what was requested?
62     //    assertThat(response.getBody()).isNotNull();
63     //    Order order = response.getBody();
64     //    assertThat(order.getCustomer().getCustomerId()).isEqualTo("STERN_TORO");
65     //    assertThat(order.getModel().getModelID()).isEqualTo("GRAND_CHEROKEE");
66     //    assertThat(order.getModel().getTrimLevel()).isEqualTo("Summit");
67     //    assertThat(order.getModel().getNumDoors()).isEqualTo(4);
68     //    assertThat(order.getColor().getColorId()).isEqualTo("EXT_JETSET_BLUE");
69     //    assertThat(order.getEngine().getEngineId()).isEqualTo("6_4_GAS");
70     //    assertThat(order.getTire().getTireId()).isEqualTo("265_GOODYEAR");
71     //    assertThat(order.getOptions()).hasSize(3);
72 }
73
74
75 public String createOrderBody() {
76     // @formatter:off
77     String body = "{"
78         + "\n" + "  \"customer\": \"STERN_TORO\", \"
79         + "\n" + "  \"model\": \"GRAND_CHEROKEE\", \"
80         + "\n" + "  \"trim\": \"Summit\", \"
81         + "\n" + "  \"doors\": \"4\", \"
82         + "\n" + "  \"color\": \"EXT_JETSET_BLUE\", \"
83         + "\n" + "  \"engine\": \"6_4_GAS\", \"
84         + "\n" + "  \"tire\": \"265_GOODYEAR\", \"
85         + "\n" + "  \"options\": [\"
86         + "\n" + "    \"DOOR_MOPAR_REINFORCE\", \"
87         + "\n" + "    \"EXT_MOPAR_HEAD_LED\", \"
88         + "\n" + "    \"INT_MOPAR_COLR\" \"
89         + "\n" + "  ]\n"
90         + "}";
91     // @formatter:on
92
93     //printing body yields :  formatted for JSON in K:V pairs
94     //    {
95     //        "customer": "STERN_TORO",
96     //        "model": "GRAND_CHEROKEE",
97     //        "trim": "Summit",
98     //        "doors": "4",
99     //        "color": "EXT_JETSET_BLUE",
100    //        "engine": "6_4_GAS",
101    //        "tire": "265_GOODYEAR",
102    //        "options": [
103    //            "DOOR_MOPAR_REINFORCE",
104    //            "EXT_MOPAR_HEAD_LED",
105    //            "INT_MOPAR_COLR"]
106    //    }

```

d)

e)


f)

g)

h)

i)

j)

k) Produce a screenshot of the test method. 

```

@Test
void testCreateOrderReturnsSuccess201() {

    HttpHeaders headers = new HttpHeaders();
    headers.setContentType(MediaType.APPLICATION_JSON);

    String body = createOrderBody();
    String uri = String.format("http://localhost:%d/orders", serverPort);
    HttpEntity<String> bodyEntity = new HttpEntity<>(body, headers);
    ParameterizedTypeReference<Order> typeRef =
        new ParameterizedTypeReference<>() {};

    ResponseEntity<Order> response =
        restTemplate.exchange(uri, HttpMethod.POST, bodyEntity, typeRef);
    //Did the POST call succeed?
    assertThat(response.getStatusCode()).isEqualTo(HttpStatus.CREATED);
    // Did the POST call actually create what was requested?
    assertThat(response.getBody()).isNotNull();
    Order order = response.getBody();
    assertThat(order.getCustomer().getCustomerId()).isEqualTo("STERN_TORO");
    assertThat(order.getModel().getModelID()).isEqualTo("GRAND_CHEROKEE");
    assertThat(order.getModel().getTrimLevel()).isEqualTo("Summit");
    assertThat(order.getModel().getNumDoors()).isEqualTo(4);
    assertThat(order.getColor().getColorId()).isEqualTo("EXT_JETSET_BLUE");
    assertThat(order.getEngine().getEngineId()).isEqualTo("6_4_GAS");
    assertThat(order.getTire().getTireId()).isEqualTo("265_GOODYEAR");
    assertThat(order.getOptions()).hasSize(3);
}


```

```

public String createOrderBody() {
    // @formatter:off
    String body = "{
        +\"\\n\"+ \"customer\\\":\\\"STERN_TORO\\\",
        +\"\\n\"+ \"model\\\":\\\"GRAND_CHEROKEE\\\",
        +\"\\n\"+ \"trim\\\":\\\"Summit\\\",
        +\"\\n\"+ \"doors\\\":\\\"4\\\",
        +\"\\n\"+ \"color\\\":\\\"EXT_JETSET_BLUE\\\",
        +\"\\n\"+ \"engine\\\":\\\"6_4_GAS\\\",
        +\"\\n\"+ \"tire\\\":\\\"265_GOODYEAR\\\",
        +\"\\n\"+ \"options\\\":[
        +\"\\n\"+ \"DOOR_MOPAR_REINFORCE\\\",
        +\"\\n\"+ \"EXT_MOPAR_HEAD_LED\\\",
        +\"\\n\"+ \"INT_MOPAR_COLR\\\"
        +\"\\n\"
        +\"}";
    // @formatter:on

    //printing body yields : formatted for JSON in K:V pairs
    // {
    //     "customer":"STERN_TORO",
    //     "model":"GRAND_CHEROKEE",
    //     "trim":"Summit",
    //     "doors":"4",
    //     "color":"EXT_JETSET_BLUE",
    //     "engine":"6_4_GAS",
    //     "tire":"265_GOODYEAR",
    //     "options":[
    //         "DOOR_MOPAR_REINFORCE",
    //         "EXT_MOPAR_HEAD_LED",
    //         "INT_MOPAR_COLR"]
    // }
}

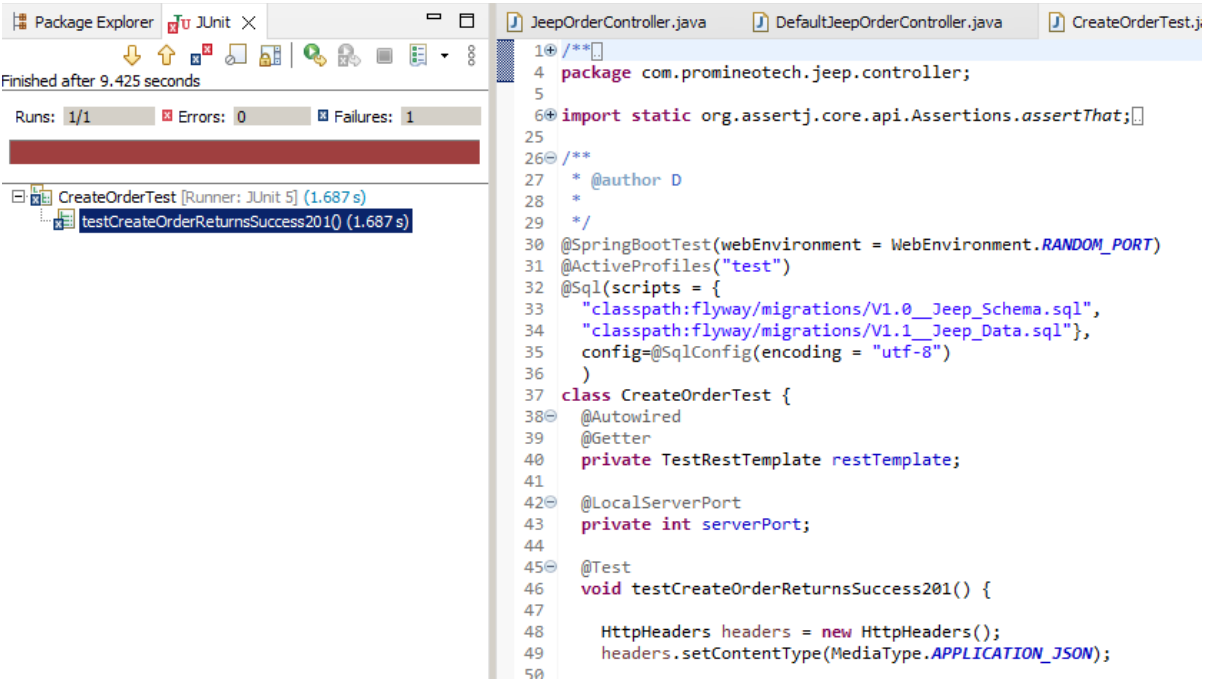
```


- 3) In the controller sub-package in src/main/java, create an interface named JeepOrderController. Add @RequestMapping("/orders") as a class-level annotation.
- a)
 - b)
 - c) Produce a screenshot of the finished JeepOrderController interface showing no compile errors. 

```
JeepOrderController.java X
1  /**
2  package com.promineotech.jeep.controller;
3
4  import org.springframework.http.HttpStatus;
5
6  /**
7   * @author D
8   *
9   */
10 @Validated
11 @RequestMapping("/orders")
12
13 @OpenAPIDefinition(info = @Info(title = "Jeep Order Service"), servers = {
14     @Server(url = "http://localhost:8080", description = "Local server.")})
15
16 public interface JeepOrderController {
17     @Operation(
18         summary = "Create an order for a Jeep",
19         description = "Returns the created Jeep , with options", //time 20:07
20         responses = {
21             @ApiResponse(responseCode = "201", description = "Success, A Jeep Order was created",
22                 content = @Content(
23                     mediaType = "application/json",
24                     schema = @Schema(implementation = Order.class))),
25             @ApiResponse(responseCode = "400", description = "Request parameters invlaid",
26                 content = @Content(mediaType = "application/json")),
27             @ApiResponse(responseCode = "404", description = "A Jeep component wa not found with the input criteria",
28                 content = @Content(mediaType = "application/json")),
29             @ApiResponse(responseCode = "500", description = "Error 500 ",
30                 content = @Content(mediaType = "application/json"))},
31         parameters = {
32             @Parameter(name = "orderRequest" , required = true , description = "The JSON formatted order"),
33         }
34     )
35
36     @PostMapping
37     @ResponseStatus(code=HttpStatus.CREATED)
38     Order createOrder(@RequestBody OrderRequest orderRequest);
39 }


```

- 4) Create a class that implements JeepOrderController named DefaultJeepOrderController.

- a)
- b)
- c) Run the test to show a red status bar. Produce a screenshot that shows the test method, the log line, and the red JUnit status bar. 

- 5)
- 6)
- 7) Add Bean Validation annotations to the OrderRequest class as shown in the video.

- a)
- b)
- c)
- d)

e) Produce a screenshot of this class with the annotations. 



```
JeepOrderCont...  DefaultJeepOr...  CreateOrderTe...  ComponentSca...  DefaultJeepSa...  
  
package com.promineotech.jeep.entity;  
  
import java.util.List;  
  
@Data  
public class OrderRequest {  
    @NotNull  
    @Length(max = 30)  
    @Pattern(regexp = "[\\w\\s]*")  
    private String customer;  
  
    @NotNull  
    private JeepModel model;  
  
    @NotNull  
    @Length(max = 30)  
    @Pattern(regexp = "[\\w\\s]*")  
    private String trim;  
  
    @Positive  
    @Min(2)  
    @Max(4)  
    private int doors;  
  
    @NotNull  
    @Length(max = 30)  
    @Pattern(regexp = "[\\w\\s]*")  
    private String color;  
  
    @NotNull  
    @Length(max = 30)  
    @Pattern(regexp = "[\\w\\s]*")  
    private String engine;  
  
    @NotNull  
    @Length(max = 30)  
    @Pattern(regexp = "[\\w\\s]*")  
    private String tire;  
  
    private List<@NotNull @Length(max = 30) @Pattern(regexp = "[\\w\\s]*") String> options;  
}
```


8) In the `jeep.service` sub-package, create the empty (no methods yet) order service interface (named `JeepOrderService`) and implementation (named `DefaultJeepOrderService`).

a)

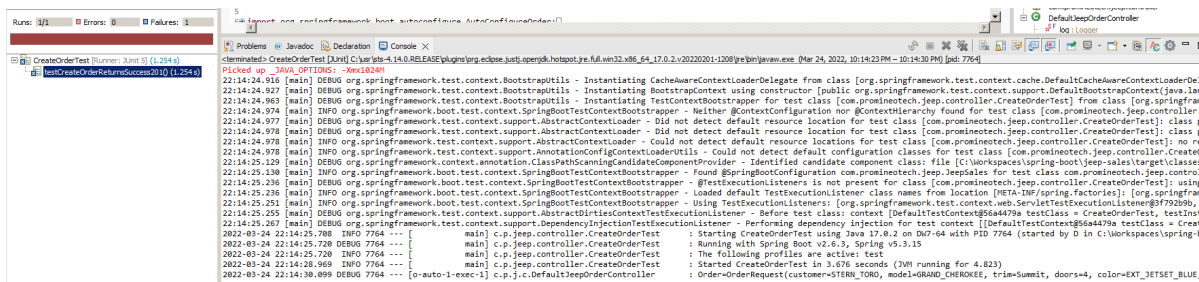
b)

c)

d)

e)

f) Run the test `CreateOrderTest` again. Produce a screenshot showing that the service layer `createOrder` method correctly prints the log line in the console. (e.g. prints out the `OrderRequest` in the console from within the Service Layer).



```
Run: 1/1 | Errors: 0 | Failures: 1
CreateOrderTest (Runner: JUnit 5) (1.264 s)
22:14:24.916 [main] DEBUG org.springframework.test.context.BootstrapUtils - Instantiating CacheAwareContextLoaderDelegate from class [org.springframework.test.context.cache.DefaultCacheAwareContextLoaderDele
22:14:24.927 [main] DEBUG org.springframework.test.context.BootstrapUtils - Instantiating BootstrapContext using constructor [public org.springframework.test.context.BootstrapContext(java.lar
22:14:24.963 [main] DEBUG org.springframework.test.context.BootstrapUtils - Instantiating TestContextBootstrapper for test class [com.promineotech.jee.controller.CreateOrderTest] from class [org.springfram
22:14:24.974 [main] INFO org.springframework.test.context.BootstrapUtils - Neither @ContextConfiguration nor @ContextHierarchy found for test class [com.promineotech.jee.controller.
22:14:24.977 [main] DEBUG org.springframework.test.context.support.AbstractContextLoader - Did not detect default resource location for test class [com.promineotech.jee.controller.CreateOrderTest]: class p
22:14:24.978 [main] DEBUG org.springframework.test.context.support.AbstractContextLoader - Did not detect default resource location for test class [com.promineotech.jee.controller.CreateOrderTest]: class p
22:14:24.978 [main] INFO org.springframework.test.context.support.AbstractContextLoader - Could not detect default resource locations for test class [com.promineotech.jee.controller.CreateOrderTest]: no re
22:14:24.978 [main] INFO org.springframework.test.context.support.AnnotationConfigContextLoaderUtils - Could not detect default configuration classes for test class [com.promineotech.jee.controller.CreateO
22:14:25.129 [main] DEBUG org.springframework.context.annotation.ClassPathScanningCandidateComponentProvider - Identified candidate component class: file [C:\Workspaces\spring-boot\jeep-sales\target\classes
22:14:25.130 [main] INFO org.springframework.test.context.BootstrapUtils - Found @SpringBootConfiguration com.promineotech.jee.JeeSales for test class com.promineotech.jee.control
22:14:25.136 [main] DEBUG org.springframework.test.context.BootstrapUtils - @TestExecutionListeners is not present for class [com.promineotech.jee.controller.CreateOrderTest]: using
22:14:25.236 [main] INFO org.springframework.test.context.BootstrapUtils - Loaded default TestExecutionListener class names from location [META-INF/spring.factories]: [org.springfram
22:14:25.251 [main] INFO org.springframework.test.context.BootstrapUtils - Using TestExecutionListeners: [org.springframework.test.context.web.ServletTestExecutionListener, org.springframework
22:14:25.255 [main] DEBUG org.springframework.test.context.support.AbstractJUnit4TestExecutionListener - Before test class: context [DefaultTestContext@564479a testClass = CreateOrderTest, testInst
22:14:25.267 [main] INFO org.springframework.test.context.support.DependencyInjectionTestExecutionListener - Performing dependency injection for test context [[DefaultTestContext@564479a testClass = Creat
2022-03-24 22:14:25.788 INFO 7764 --- [main] c.p.jee.controller.CreateOrderTest : Starting CreateOrderTest using Java 17.0.2 on DM7-64 with PID 7764 (started by D in C:\Workspaces\spring-b
2022-03-24 22:14:25.728 INFO 7764 --- [main] c.p.jee.controller.CreateOrderTest : Running with Spring Boot v2.6.1, Spring v5.3.15
2022-03-24 22:14:25.728 INFO 7764 --- [main] c.p.jee.controller.CreateOrderTest : The following profiles are active: test
2022-03-24 22:14:26.969 INFO 7764 --- [main] c.p.jee.controller.CreateOrderTest : Started CreateOrderTest in 0.676 seconds (JVM running for 4.823)
2022-03-24 22:14:30.099 DEBUG 7764 --- [o-auto-1-exec-1] c.p.jee.DefaultJeepOrderController : OrderRequest(customer=STERN_TORO, model=GRAND_CHEROKEE, trim=Summit, doors=4, color=EXT_3ETSET_BLUE,
```

9)

10)

11)


12)

13)

14)

15) In JeepOrderDao.java and DefaultJeepOrderDao.java, add the method:

Order saveOrder(Customer customer, Jeep jeep, Color color, Engine engine, Tire tire, BigDecimal price, List<Option> options);


- a) Call the jeepOrder.Dao.saveOrder method from the jeepOrderSalesService.createOrder service. Produce a screenshot of the jeepOrderSalesService.createOrder method. 



```
22 @Service
23 public class DefaultJeepOrderService implements JeepOrderService {
24
25     @Autowired
26     private JeepOrderDao jeepOrderDao;
27
28     @Override
29     @Transactional
30     public Order createOrder(OrderRequest orderRequest) {
31         List<Option> options = getOption(orderRequest);
32         Jeep jeep = getModel(orderRequest);
33         Color color = getColor(orderRequest);
34         Engine engine = getEngine(orderRequest);
35         Tire tire = getTire(orderRequest);
36         Customer customer = getCustomer(orderRequest);
37         BigDecimal price = jeep.getBasePrice().add(color.getPrice())
38             .add(engine.getPrice().add(tire.getPrice()));
39         BigDecimal accum = BigDecimal.ZERO;
40         for (Option option : options) {accum = accum.add(option.getPrice());}
41         price = price.add(accum);
42
43         // return jeepOrderDao.createOrder(orderRequest);
44         return jeepOrderDao.saveOrder(customer, jeep, color, engine, tire, price, options);
45     }
}
```

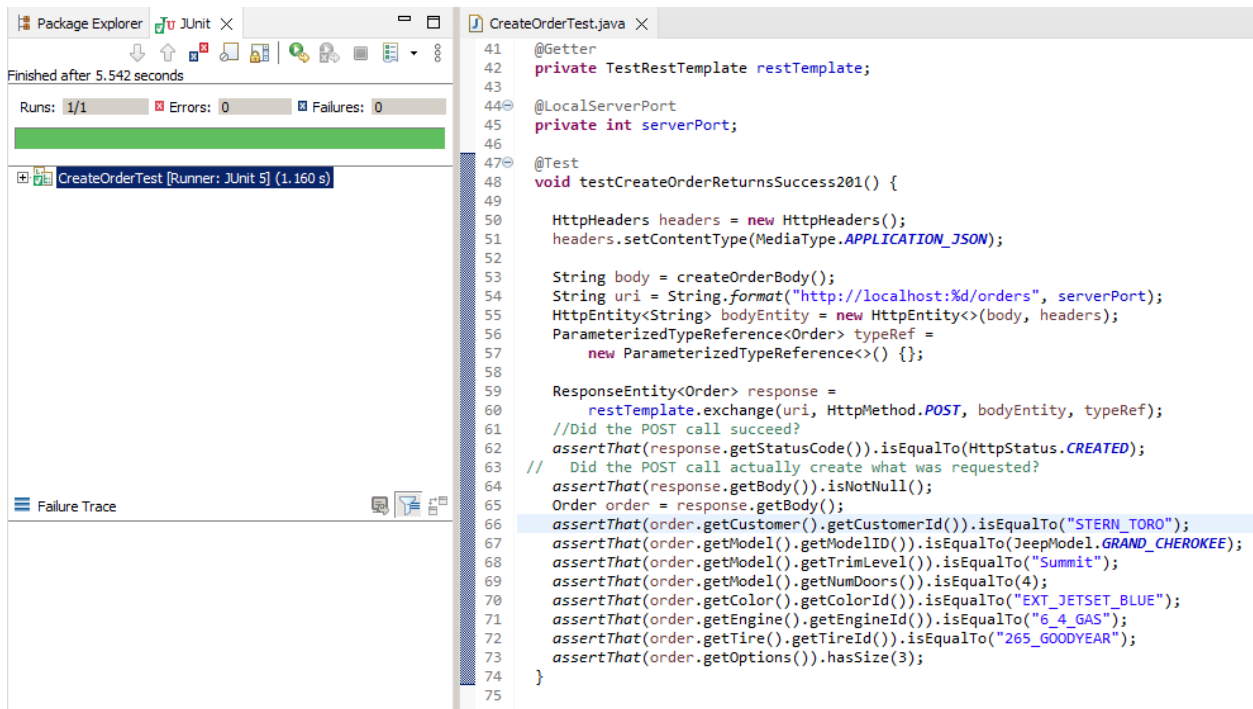
- b) Write the implementation of the saveOrder method in the DAO.

- i)
- ii)
- iii)
- iv)

v) Produce a screenshot of the saveOrder method. 

```
1 *DefaultJeepOrderDao.java X
372     }
373
374 @Override
375 public Order saveOrder(Customer customer, Jeep jeep, Color color, Engine engine, Tire tire, BigDecimal price,
376     List<Option> options) {
377     SqlParams params = generateInsertSql(customer, jeep, color, engine, tire, price);
378
379     KeyHolder keyHolder = new GeneratedKeyHolder();
380     jdbcTemplate.update(params.sql, params.source, keyHolder);
381     Long orderPK = keyHolder.getKey().longValue();
382     saveOptions(options, orderPK);
383
384     // @formatter:off
385     return Order.builder()
386         .orderPK(orderPK)
387         .customer(customer)
388         .model(jeep)
389         .color(color)
390         .engine(engine)
391         .tire(tire)
392         .options(options)
393         .price(price)
394         .build();
395     // @formatter:on
396 }
397
398 private void saveOptions(List<Option> options, Long orderPK) {
399     for (Option option : options) {
400         SqlParams params = generateInsertSql(option, orderPK);
401         jdbcTemplate.update(params.sql, params.source);
402     }
403 }
404
405
406
```

- c) Run the integration test in CreateOrderTest. Produce a screenshot of the test method that shows the green JUnit status bar, the console output, and the test class. 🖥️



URL to GitHub Repository:

<https://github.com/david2joh/nccspringboot/tree/main/Week04>