

# Web API Design with Spring Boot Week 16 Coding Assignment

**Points possible: 75**


**URL to GitHub Repository:** <https://github.com/Anghel-Valdehueza/Jeep-Sales.git>

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


---

## 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 16 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:

For this week's homework you need to copy source code from the supplied resources.


For this week's homework you need to copy source code from the Source folder in the supplied resources. Wait until the instructions tell you to copy the resources or you will get errors.

- 1) Select some options for a Jeep order:
  - a) Use the `data.sql` file or the jeep database tables to select options for a Jeep order. Select any one of each of the following for the order:
    - i) color
    - ii) customer
    - iii) engine
    - iv) model
    - v) tire(s)
  - b) Select one or more options from the options table as well. Keep in mind that some options may work better than others – but if you want to put 37-inch tires on your Jeep Renegade, so be it!
- 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) Add the Spring Boot Test annotations: `@SpringBootTest`, `@ActiveProfiles`, and `@Sql`. They should have the same parameters as the test created in weeks 1 and 2.
  - b) Create a test method (annotated with `@Test`) named `testCreateOrderReturnsSuccess201`.
  - c) In the test class, create a method named `createOrderBody`. This method returns a type of `String`. In this method, return a JSON object with the IDs that you picked in Step 1a and 1b. For example:

## Web API Design with Spring Boot Week 16 Coding Assignment

```
{
  "customer": "MORISON_LINA",
  "model": "WRANGLER",
  "trim": "Sport Altitude",
  "doors": 4,
  "color": "EXT_NACHO",
  "engine": "2_0_TURBO",
  "tire": "35_TOYO",
  "options": [
    "DOOR_QUAD_4",
    "EXT_AEV_LIFT",
    "EXT_WARN_WINCH",
    "EXT_WARN BUMPER_FRONT",
    "EXT_WARN BUMPER_REAR",
    "EXT_ARB_COMPRESSOR"
  ]
}
```

Make sure that the JSON is correct! If necessary, use a JSON formatter/validator like the one here: <https://jsonformatter.curiousconcept.com/>.

Produce a screenshot of the `createOrderBody()` method. 

In the test method, assign the return value of the `createOrderBody()` method to a variable named `body`.

## Web API Design with Spring Boot Week 16 Coding Assignment

```
65 private String createOrderBody() {
66     return "{\n" +
67         "    \"customer\": \"MORISON_LINA\", \n" +
68         "    \"model\": \"WRANGLER\", \n" +
69         "    \"trim\": \"Sport Altitude\", \n" +
70         "    \"doors\": 4, \n" +
71         "    \"color\": \"EXT_NACHO\", \n" +
72         "    \"engine\": \"2_0_TURBO\", \n" +
73         "    \"tire\": \"35_TOYO\", \n" +
74         "    \"options\": [\n" +
75         "        \"DOOR_QUAD_4\", \n" +
76         "        \"EXT_AEV_LIFT\", \n" +
77         "        \"EXT_WARN_WINCH\", \n" +
78         "        \"EXT_WARN BUMPER_FRONT\", \n" +
79         "        \"EXT_WARN BUMPER_REAR\", \n" +
80         "        \"EXT_ARB_COMPRESSOR\" \n" +
81         "    ] \n" +
82         "}\n";
83 }
84 }
```

- d) In the test class, add an instance variable named `serverPort` to hold the port that Tomcat is listening on in the test. Annotate the variable with `@LocalServerPort`.
- e) Add another instance variable for an injected `TestRestTemplate` named `restTemplate`.
- f) In the test method, assign a value to a local variable named `uri` as follows:

```
String uri = String.format("http://localhost:%d/orders", serverPort);
```

- g) In the test method, create an `HttpHeaders` object and set the content type to "application/json" like this:

```
HttpHeaders headers = new HttpHeaders();
```

```
headers.setContentType(MediaType.APPLICATION_JSON);
```

Make sure to import the package `org.springframework.http.HttpHeaders`.

- h) Create an `HttpEntity` object and set the request body and headers:

```
HttpEntity<String> bodyEntity = new HttpEntity<>(body, headers);
```

- i) Send the request body and headers to the server. The `Order` class should have been copied earlier from the supplied resources. Ensure that you import `com.promineotech.jeepp.entity.Order` and not some other `Order` class.

```
ResponseEntity<Order> response = restTemplate.exchange(uri,
    HttpMethod.POST, bodyEntity, Order.class);
```

## Web API Design with Spring Boot Week 16 Coding Assignment

- j) Add the AssertJ assertions to ensure that the response is correct. Replace the expected values to match the JSON in step 2c.

```
assertThat(response.getStatusCode()).isEqualTo(HttpStatus.CREATED);

assertThat(response.getBody()).isNotNull();

Order order = response.getBody();

assertThat(order.getCustomer().getCustomerId()).isEqualTo("MORISON_LINA");

assertThat(order.getModel().getModelId()).isEqualTo(JeepModel.WRANGLER);

assertThat(order.getModel().getTrimLevel()).isEqualTo("Sport Altitude");

assertThat(order.getModel().getNumDoors()).isEqualTo(4);

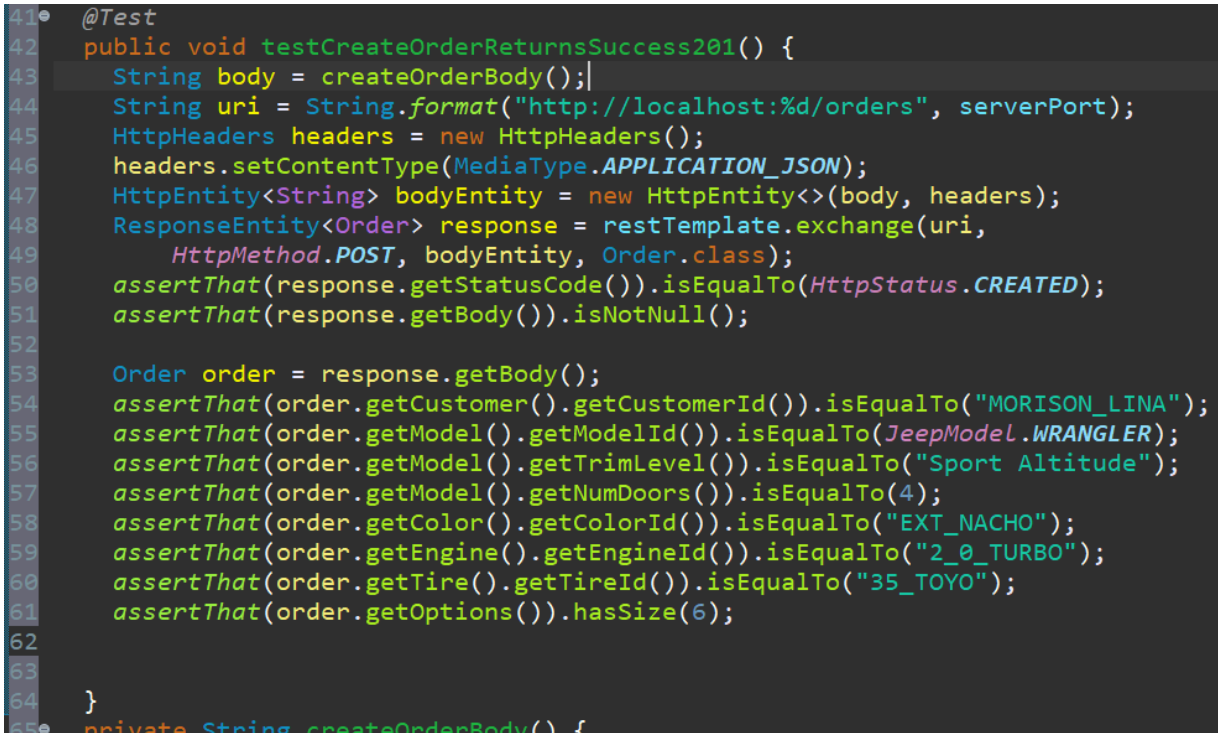
assertThat(order.getColor().getColorId()).isEqualTo("EXT_NACHO");

assertThat(order.getEngine().getEngineId()).isEqualTo("2_0_TURBO");

assertThat(order.getTire().getTireId()).isEqualTo("35_TOYO");

assertThat(order.getOptions()).hasSize(6);
```


- k) Produce a screenshot of the test method. 



```
41 @Test
42 public void testCreateOrderReturnsSuccess201() {
43     String body = createOrderBody();
44     String uri = String.format("http://localhost:%d/orders", serverPort);
45     HttpHeaders headers = new HttpHeaders();
46     headers.setContentType(MediaType.APPLICATION_JSON);
47     HttpEntity<String> bodyEntity = new HttpEntity<>(body, headers);
48     ResponseEntity<Order> response = restTemplate.exchange(uri,
49         HttpMethod.POST, bodyEntity, Order.class);
50     assertThat(response.getStatusCode()).isEqualTo(HttpStatus.CREATED);
51     assertThat(response.getBody()).isNotNull();
52
53     Order order = response.getBody();
54     assertThat(order.getCustomer().getCustomerId()).isEqualTo("MORISON_LINA");
55     assertThat(order.getModel().getModelId()).isEqualTo(JeepModel.WRANGLER);
56     assertThat(order.getModel().getTrimLevel()).isEqualTo("Sport Altitude");
57     assertThat(order.getModel().getNumDoors()).isEqualTo(4);
58     assertThat(order.getColor().getColorId()).isEqualTo("EXT_NACHO");
59     assertThat(order.getEngine().getEngineId()).isEqualTo("2_0_TURBO");
60     assertThat(order.getTire().getTireId()).isEqualTo("35_TOYO");
61     assertThat(order.getOptions()).hasSize(6);
62
63 }
64
65 private String createOrderBody() {
```

- 3) In the controller sub-package in src/main/java, create an interface named JeepOrderController. Add @RequestMapping("/orders") as a class-level annotation.

## Web API Design with Spring Boot Week 16 Coding Assignment


- Create a method in the interface to create an order (createOrder). It should return an object of type Order (see below). It should accept a single parameter of type OrderRequest as described in the video. Make sure it accepts an HTTP POST request and returns a status code of 201 (created).
- Add the @RequestBody annotation to the orderRequest parameter. Make sure to add the RequestBody annotation from the org.springframework.web.bind.annotation package.
- Produce a screenshot of the finished JeepOrderController interface showing no compile errors. 

```
1 // **
2
3 package com.promineotech.jeep.controller;
4
5
6 import org.springframework.http.HttpStatus;
7 import org.springframework.validation.annotation.Validated;
8 import org.springframework.web.bind.annotation.PostMapping;
9 import org.springframework.web.bind.annotation.RequestBody;
10 import org.springframework.web.bind.annotation.RequestMapping;
11 import org.springframework.web.bind.annotation.ResponseStatus;
12 import com.promineotech.jeep.entity.Order;
13 import com.promineotech.jeep.entity.OrderRequest;
14 import io.swagger.v3.oas.annotations.Operation;
15 import io.swagger.v3.oas.annotations.Parameter;
16 import io.swagger.v3.oas.annotations.media.Content;
17 import io.swagger.v3.oas.annotations.media.Schema;
18 import io.swagger.v3.oas.annotations.responses.ApiResponse;
19
20 // **
21 * @author angva
22 *
23 */
```

```
24 @RequestMapping("/orders")
25 @Validated
26 public interface JeepOrderController {
27     // @formatter:off
28     @Operation(
29         summary = "Create an order for a Jeep",
30         description = "Returns the created Jeep",
31         responses = {
32             @ApiResponse(responseCode = "201",
33                 description = "The created Jeep is returned",
34                 content = @Content(
35                     mediaType = "application/json",
36                     schema = @Schema(implementation = Order.class))),
37             @ApiResponse(responseCode = "400",
38                 description = "The request parameters are invalid",
39                 content = @Content(mediaType = "application/json")),
40             @ApiResponse(responseCode = "404",
41                 description = "No jeeps were found with the input criteria",
42                 content = @Content(mediaType = "application/json")),
43             @ApiResponse(responseCode = "500",
44                 description = "An unplanned error occurred.",
```

## Web API Design with Spring Boot Week 16 Coding Assignment

```
45         content = @Content(mediaType = "application/json"))
46     },
47     parameters = {
48         @Parameter(name = "orderRequest",
49             required = true, description = "The order as JSON")
50     }
51 )
52 @PostMapping
53 @ResponseStatus(code = HttpStatus.CREATED)
54 public Order createOrder(@RequestBody OrderRequest orderRequest);
55 // @formatter:on
56 }
57
```

- 4) Create a class that implements JeepOrderController named DefaultJeepOrderController.
- Add `@RestController` as a class-level annotation.
  - Add a log line to the implementing controller method showing the input request body (orderRequest)
  - 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. 

```
41 @Test
42 public void testCreateOrderReturnsSuccess201() {
43     String body = createOrderBody();
44     String uri = String.format("http://localhost:%d/orders", serverPort);
45
46     HttpHeaders headers = new HttpHeaders();
47     headers.setContentType(MediaType.APPLICATION_JSON);
48     HttpEntity<String> bodyEntity = new HttpEntity<>(body, headers);
49
50     ResponseEntity<Order> response = restTemplate.exchange(uri,
51         HttpMethod.POST, bodyEntity, Order.class);
52
53     assertThat(response.getStatusCode()).isEqualTo(HttpStatus.CREATED);
54     assertThat(response.getBody()).isNotNull();
55 }
```

Package Explorer JUnit X

Finished after 22.003 seconds

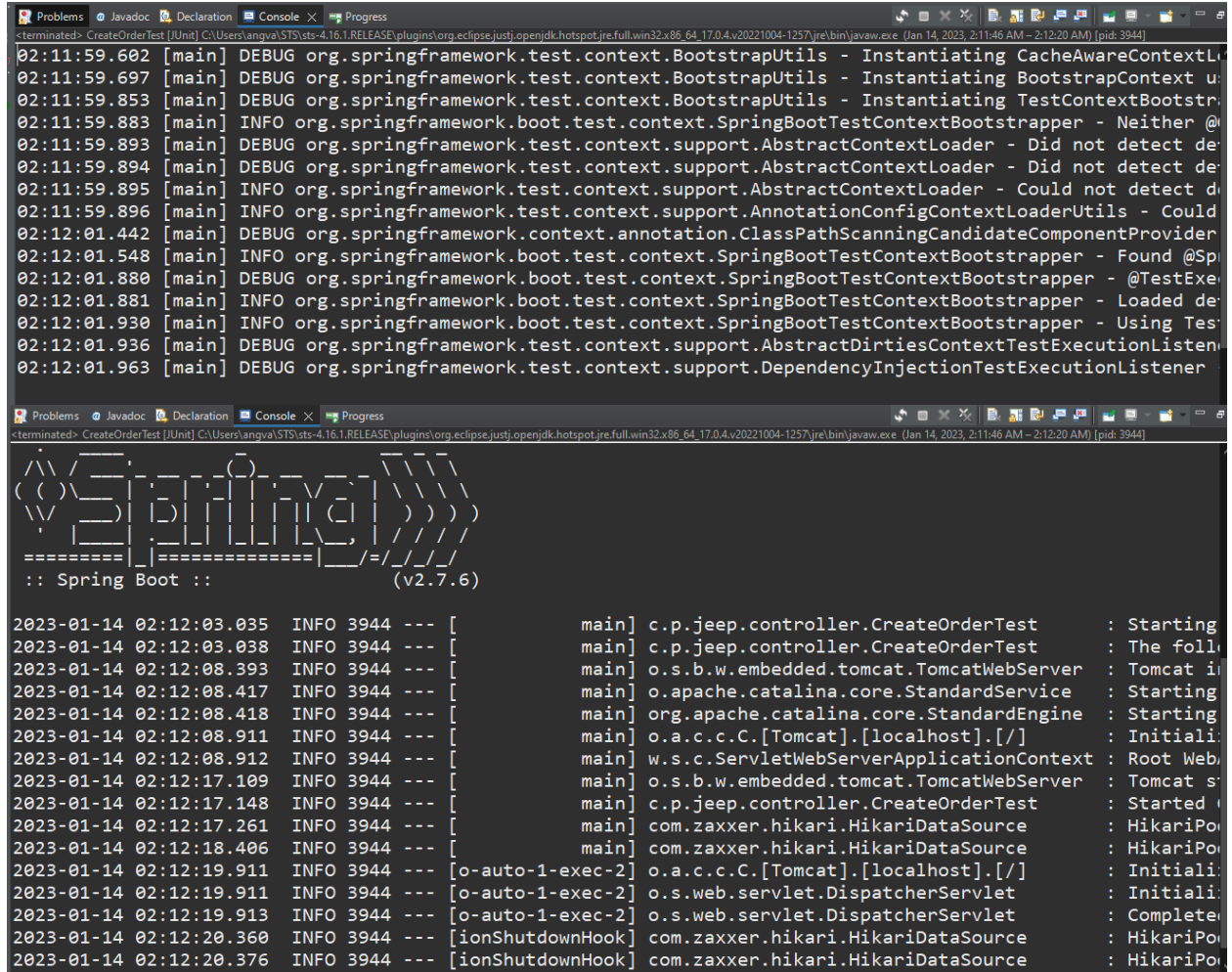
Runs: 1/1    ✖ Errors: 0    ✖ Failures: 1

Failure Trace

- ! java.lang.AssertionError:  
Expecting actual not to be null  
at com.promineotech.jeep.controller.CreateOrderTest.testCreateOrderReturnsSuccess201(CreateOrderTest.java:54)  
at java.base/java.util.ArrayList.forEach(ArrayList.java:1511)  
at java.base/java.util.ArrayList.forEach(ArrayList.java:1511)



# Web API Design with Spring Boot Week 16 Coding Assignment



```
<terminated> CreateOrderTest [JUnit] C:\Users\angva\STS\sts-4.16.1.RELEASE\plugins\org.eclipse.justi.openjdk.hotspot.jre.full.win32.x86_64.17.0.4.v20221004-1257\jre\bin\javaw.exe (Jan 14, 2023, 2:11:46 AM - 2:12:20 AM) [pid: 3944]
02:11:59.602 [main] DEBUG org.springframework.test.context.BootstrapUtils - Instantiating CacheAwareContextLoaderDelegate
02:11:59.697 [main] DEBUG org.springframework.test.context.BootstrapUtils - Instantiating BootstrapContext using bootstrapUtils
02:11:59.853 [main] DEBUG org.springframework.test.context.BootstrapUtils - Instantiating TestContextBootstrapper for org.springframework.test.context.support.AnnotationConfigContextLoaderUtils
02:11:59.883 [main] INFO org.springframework.boot.test.context.SpringBootTestContextBootstrapper - Neither @ContextConfiguration nor @ContextInitializer found on class org.springframework.test.context.support.AnnotationConfigContextLoaderUtils
02:11:59.893 [main] DEBUG org.springframework.test.context.support.AbstractContextLoader - Did not detect default context location org.springframework.test.context.support.AnnotationConfigContextLoaderUtils for class org.springframework.test.context.support.AnnotationConfigContextLoaderUtils
02:11:59.894 [main] DEBUG org.springframework.test.context.support.AbstractContextLoader - Did not detect default context location org.springframework.test.context.support.AnnotationConfigContextLoaderUtils for class org.springframework.test.context.support.AnnotationConfigContextLoaderUtils
02:11:59.895 [main] INFO org.springframework.test.context.support.AnnotationConfigContextLoaderUtils - Could not detect default context location org.springframework.test.context.support.AnnotationConfigContextLoaderUtils for class org.springframework.test.context.support.AnnotationConfigContextLoaderUtils
02:11:59.896 [main] INFO org.springframework.test.context.support.AnnotationConfigContextLoaderUtils - Could not detect default context location org.springframework.test.context.support.AnnotationConfigContextLoaderUtils for class org.springframework.test.context.support.AnnotationConfigContextLoaderUtils
02:12:01.442 [main] DEBUG org.springframework.context.annotation.AnnotationConfigApplicationContext - Loading bean definitions from URL [file:/C:/Users/angva/.m2/repository/org/springframework/spring-test/5.3.12/spring-test-5.3.12.jar!/org/springframework/test/context/annotation/ContextConfiguration.class]
02:12:01.548 [main] INFO org.springframework.boot.test.context.SpringBootTestContextBootstrapper - Found @SpringBootTest on class org.springframework.test.context.support.AnnotationConfigContextLoaderUtils
02:12:01.880 [main] DEBUG org.springframework.boot.test.context.SpringBootTestContextBootstrapper - @TestExecutionListeners: [org.springframework.test.context.support.AnnotationConfigContextLoaderUtils]
02:12:01.881 [main] INFO org.springframework.boot.test.context.SpringBootTestContextBootstrapper - Loaded default TestExecutionListener class names from location [META-INF/spring.factories]: [org.springframework.test.context.support.AnnotationConfigContextLoaderUtils]
02:12:01.930 [main] INFO org.springframework.boot.test.context.SpringBootTestContextBootstrapper - Using TestExecutionListeners: [org.springframework.test.context.support.AnnotationConfigContextLoaderUtils]
02:12:01.936 [main] DEBUG org.springframework.test.context.support.AbstractContextLoader - Did not detect default context location org.springframework.test.context.support.AnnotationConfigContextLoaderUtils for class org.springframework.test.context.support.AnnotationConfigContextLoaderUtils
02:12:01.963 [main] DEBUG org.springframework.test.context.support.DependencyInjectionTestExecutionListener - Loading bean definitions from URL [file:/C:/Users/angva/.m2/repository/org/springframework/spring-test/5.3.12/spring-test-5.3.12.jar!/org/springframework/test/context/annotation/ContextConfiguration.class]

:: Spring Boot :: (v2.7.6)

2023-01-14 02:12:03.035 INFO 3944 --- [main] c.p.jee.controller.CreateOrderTest : Starting
2023-01-14 02:12:03.038 INFO 3944 --- [main] c.p.jee.controller.CreateOrderTest : The following profiles are active: []
2023-01-14 02:12:08.393 INFO 3944 --- [main] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat initialized with 1 thread(s)
2023-01-14 02:12:08.417 INFO 3944 --- [main] o.apache.catalina.core.StandardService : Starting
2023-01-14 02:12:08.418 INFO 3944 --- [main] org.apache.catalina.core.StandardEngine : Starting
2023-01-14 02:12:08.911 INFO 3944 --- [main] o.a.c.c.C.[Tomcat].[localhost].[/] : Initializing
2023-01-14 02:12:08.912 INFO 3944 --- [main] w.s.c.ServletWebServerApplicationContext : Root WebApplicationContext: initialization completed
2023-01-14 02:12:17.109 INFO 3944 --- [main] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat started on port(s) 8080 (http)
2023-01-14 02:12:17.148 INFO 3944 --- [main] c.p.jee.controller.CreateOrderTest : Started CreateOrderTest
2023-01-14 02:12:17.261 INFO 3944 --- [main] com.zaxxer.hikari.HikariDataSource : HikariPool-1 started
2023-01-14 02:12:18.406 INFO 3944 --- [main] com.zaxxer.hikari.HikariDataSource : HikariPool-2 started
2023-01-14 02:12:19.911 INFO 3944 --- [o-auto-1-exec-2] o.a.c.c.C.[Tomcat].[localhost].[/] : Initializing
2023-01-14 02:12:19.911 INFO 3944 --- [o-auto-1-exec-2] o.s.web.servlet.DispatcherServlet : Initializing
2023-01-14 02:12:19.913 INFO 3944 --- [o-auto-1-exec-2] o.s.web.servlet.DispatcherServlet : Complete
2023-01-14 02:12:20.360 INFO 3944 --- [ionShutdownHook] com.zaxxer.hikari.HikariDataSource : HikariPool-1 shutdown
2023-01-14 02:12:20.376 INFO 3944 --- [ionShutdownHook] com.zaxxer.hikari.HikariDataSource : HikariPool-2 shutdown
```

- 5) Find the Maven dependency spring-boot-starter-validation by looking it up at <https://mvnrepository.com/>. Add this repository to the project POM file (pom.xml).
- 6) Add the class-level annotation @Validated to the JeepOrderController interface.
- 7) Add Bean Validation annotations to the OrderRequest class as shown in the video.
  - a) Use these annotations for String types:
    - i) @NotNull
    - ii) @Length(max = 30)
    - iii) @Pattern(regexp = "[\\w\\s]\*")
  - b) Use these annotations for integer types:




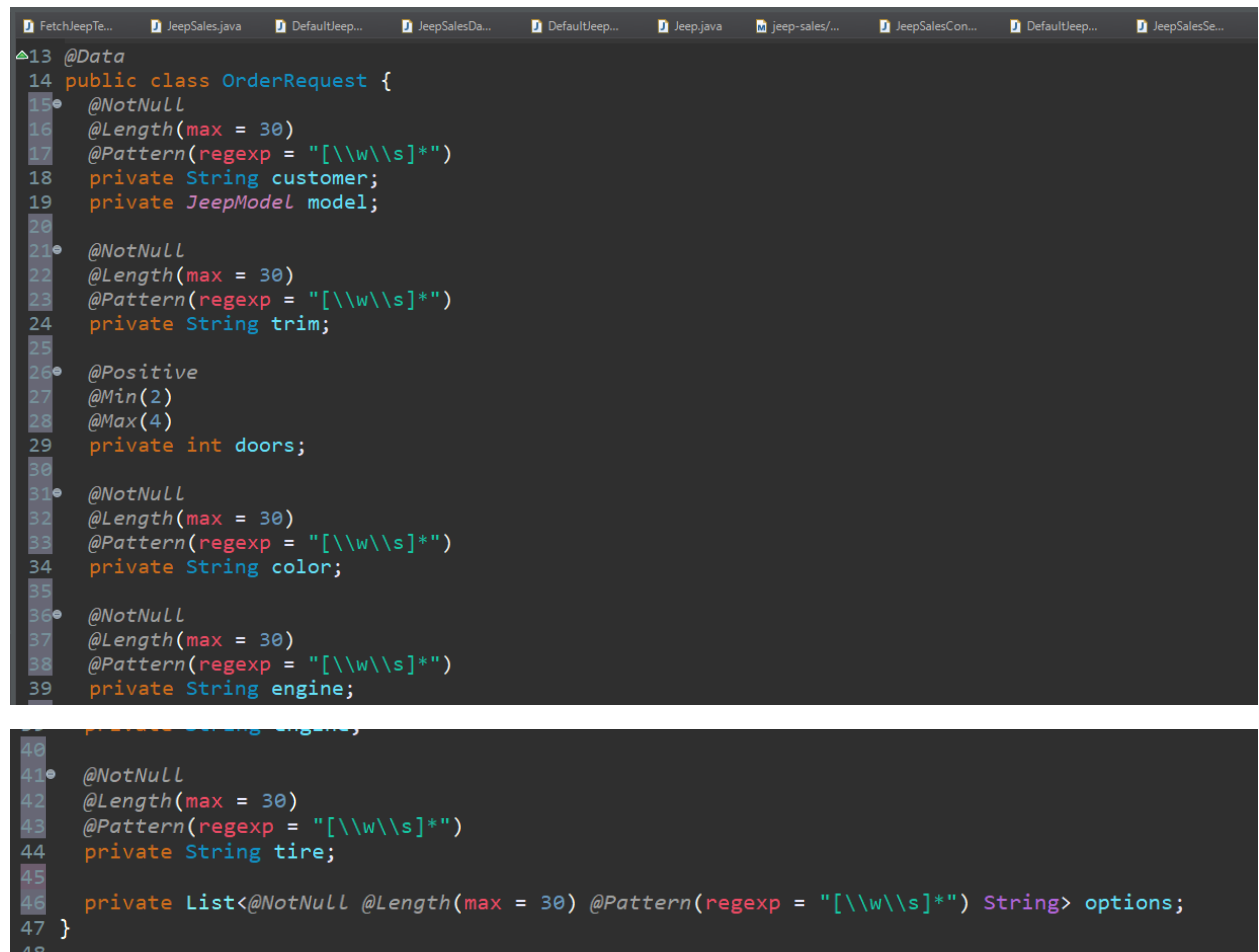
## Web API Design with Spring Boot Week 16 Coding Assignment

- i) @Positive
  - ii) @Min(2)
  - iii) @Max(4)
- c) Add @NotNull to the enum type.
- d) Add validation to the list element (type String) by adding the validation annotations *inside* the generic definition. So, to add the String validation to the options, you would do this:

```
private List<@NotNull @Length(max = 30) @Pattern(regexp = "[\\w\\s]*") String> options;
```

Do not apply a @NotNull annotation to the List because if you have no options the List may be null.

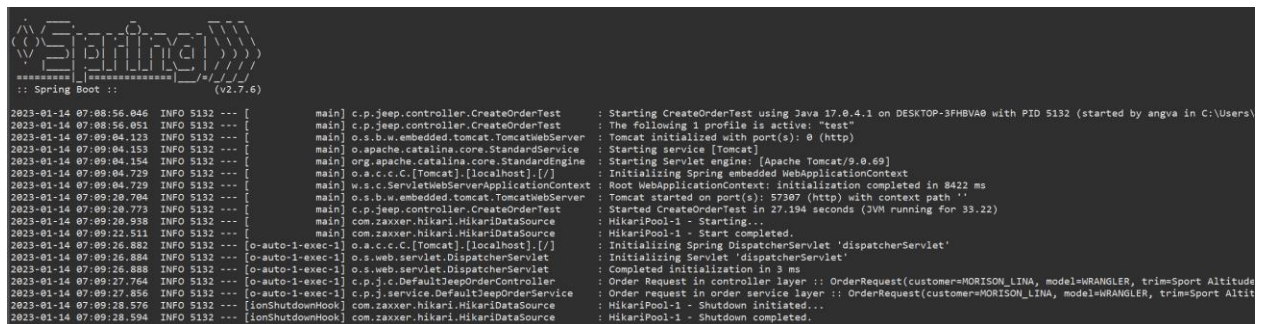
- e) Produce a screenshot of this class with the annotations. 



```
13 @Data
14 public class OrderRequest {
15     @NotNull
16     @Length(max = 30)
17     @Pattern(regexp = "[\\w\\s]*")
18     private String customer;
19     private JeepModel model;
20
21     @NotNull
22     @Length(max = 30)
23     @Pattern(regexp = "[\\w\\s]*")
24     private String trim;
25
26     @Positive
27     @Min(2)
28     @Max(4)
29     private int doors;
30
31     @NotNull
32     @Length(max = 30)
33     @Pattern(regexp = "[\\w\\s]*")
34     private String color;
35
36     @NotNull
37     @Length(max = 30)
38     @Pattern(regexp = "[\\w\\s]*")
39     private String engine;
40
41     @NotNull
42     @Length(max = 30)
43     @Pattern(regexp = "[\\w\\s]*")
44     private String tire;
45
46     private List<@NotNull @Length(max = 30) @Pattern(regexp = "[\\w\\s]*") String> options;
47 }
```

## Web API Design with Spring Boot Week 16 Coding Assignment

- 8) In the `jeep.service` sub-package, create the empty (no methods yet) order service interface (named `JeepOrderService`) and implementation (named `DefaultJeepOrderService`).
  - a) Inject the interface into the order controller implementation class.
  - b) Add the `@Service` annotation to the service implementation class.
  - c) Create the `createOrder` method in the interface and implementing service. The method signature should look like this:  
`Order createOrder(OrderRequest orderRequest);`
  - d) Call the `createOrder` method from the controller and return the value returned by the service.
  - e) Add a log line in the `createOrder` method and log the `orderRequest` parameter.
  - 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).



```
Spring
:: Spring Boot ::
(v2.7.6)

2023-01-14 07:08:56.046 INFO 5132 --- [main] c.p.jeep.controller.CreateOrderTest : Starting CreateOrderTest using Java 17.0.4.1 on DESKTOP-3FHBVA0 with PID 5132 (started by angva in C:\Users\
2023-01-14 07:08:56.051 INFO 5132 --- [main] c.p.jeep.controller.CreateOrderTest : The following 1 profile is active: "test"
2023-01-14 07:09:04.123 INFO 5132 --- [main] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat initialized with port(s): 0 (http)
2023-01-14 07:09:04.153 INFO 5132 --- [main] org.apache.catalina.core.StandardService : Starting service [Tomcat]
2023-01-14 07:09:04.154 INFO 5132 --- [main] org.apache.catalina.core.StandardEngine : Starting Servlet engine: [Apache Tomcat/9.0.69]
2023-01-14 07:09:04.729 INFO 5132 --- [main] o.a.c.c.C.[Tomcat].[localhost].[/] : Initializing Spring embedded WebApplicationContext
2023-01-14 07:09:04.729 INFO 5132 --- [main] w.s.c.ServletWebServerApplicationContext : Root WebApplicationContext: initialization completed in 8422 ms
2023-01-14 07:09:20.704 INFO 5132 --- [main] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat started on port(s): 57367 (http) with context path '/'
2023-01-14 07:09:20.773 INFO 5132 --- [main] c.p.jeep.controller.CreateOrderTest : Started CreateOrderTest in 27.194 seconds (2VM running for 33.22)
2023-01-14 07:09:20.938 INFO 5132 --- [main] com.zaxxer.hikari.HikariDataSource : HikariPool-1 - Starting...
2023-01-14 07:09:22.511 INFO 5132 --- [main] com.zaxxer.hikari.HikariDataSource : HikariPool-1 - Start completed.
2023-01-14 07:09:26.882 INFO 5132 --- [o-auto-1-exec-1] o.a.c.c.C.[Tomcat].[localhost].[/] : Initializing Spring DispatcherServlet 'dispatcherServlet'
2023-01-14 07:09:26.884 INFO 5132 --- [o-auto-1-exec-1] o.s.web.servlet.DispatcherServlet : Initializing Servlet 'dispatcherServlet'
2023-01-14 07:09:26.888 INFO 5132 --- [o-auto-1-exec-1] o.s.web.servlet.DispatcherServlet : Completed initialization in 3 ms
2023-01-14 07:09:27.764 INFO 5132 --- [o-auto-1-exec-1] c.p.j.c.DefaultJeepOrderController : Order Request in controller layer :: OrderRequest(customer=MORISON_LINA, model=WRANGLER, trim=Sport Altitude
2023-01-14 07:09:27.856 INFO 5132 --- [o-auto-1-exec-1] c.p.j.service.DefaultJeepOrderService : Order Request in order service layer :: OrderRequest(customer=MORISON_LINA, model=WRANGLER, trim=Sport Altit
2023-01-14 07:09:28.576 INFO 5132 --- [ionShutdownHook] com.zaxxer.hikari.HikariDataSource : HikariPool-1 - Shutdown initiated...
2023-01-14 07:09:28.594 INFO 5132 --- [ionShutdownHook] com.zaxxer.hikari.HikariDataSource : HikariPool-1 - Shutdown completed.
```

- 9) In the `jeep.dao` sub-package, create the empty (no methods yet) DAO interface (named `JeepOrderDao`) and implementation (named `DefaultJeepOrderDao`).
  - a) Inject the DAO interface into the order service implementation class.
  - b) Add the `@Component` annotation to the DAO implementation class.
- 10) Replace the entire content of `JeepOrderDao.java` with the source found in `JeepOrderDao.source`. The source file is found in the `Source` folder of the supplied project resources.
- 11) \*\*\* The next steps require you to copy source code from the `Source` directory in the supplied resources. Please follow the instructions EXACTLY. Some steps require you to replace ALL the source in a file. Some steps require you to ADD source to a file.

## Web API Design with Spring Boot Week 16 Coding Assignment

- 12) Copy the *contents* of the file `DefaultJeepOrderDao.source` *into* `DefaultJeepOrderDao.java`. The source file is found in the Source folder of the supplied project resources.

In Eclipse, click the "Source" menu and select "Organize Imports". Pick packages from your project where applicable. Make sure you pick the import `java.util.Optional`, `java.util.List`, and `org.springframework.jdbc.core.RowMapper`.

- 13) Copy the *contents* of the file `DefaultJeepOrderService.source` *into* `DefaultJeepOrderService.java`. Add the source after the `createOrder()` method, but *inside* the class body. The source file is found in the Source folder of the supplied project resources.


In Eclipse, click the "Source" menu and select "Organize Imports". Pick packages from your project where applicable.

- 14) In `DefaultJeepOrderService.java`, work with the method `createOrder`.

- Add the `@Transactional` annotation to the `createOrder` method.
- In the `createOrder` method call the copied methods: `getCustomer`, `getModel`, `getColor`, `getEngine`, `getTire` and `getOption`, assigning the return values of these methods to variables of the appropriate types.
- Calculate the price, including all options.

- 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);
```

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



```
1  /**
2
3
4  package com.promineotech.jeep.service;
5
6  import com.promineotech.jeep.entity.Order;
7  import com.promineotech.jeep.entity.OrderRequest;
8
9  /**
10 * @author angva
11 *
12 */
13 public interface JeepOrderService {
14     Order createOrder(OrderRequest orderRequest);
15 }
16
17
```

## Web API Design with Spring Boot Week 16 Coding Assignment

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

i) Call the supplied `generateInsertSql` method, passing in the customer, jeep, color, engine, tire and price. Assign the return value of the method to a `SqlParams` object.

ii) Call the `update` method on the `NamedParameterJdbcTemplate` object, passing in a `KeyHolder` object as shown in the video. Create the `KeyHolder` like this:

```
KeyHolder keyHolder = new GeneratedKeyHolder();
```


Be sure to extract the order primary key from the `KeyHolder` object into a variable of type `Long` named `orderPK`.

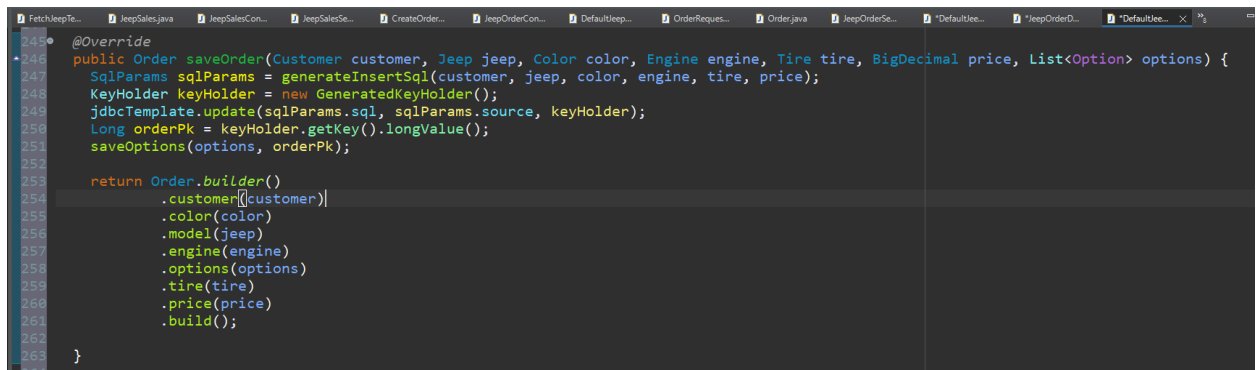
iii) Write a method named `saveOptions` as shown in the video. This method should have the following method signature:

```
private void saveOptions(List<Option> options, Long orderPK)
```

For each option in the `Options` list, call the supplied `generateInsertSql` method passing the parameters `option` and order primary key (`orderPK`). Call the `update` method on the `NamedParameterJdbcTemplate` object.


iv) In the `saveOrder` method in the DAO implementation, return an `Order` object using the `Order.builder`. The `Order` should include `orderPK`, customer, jeep (model), color, engine, tire, options and price.

v) Produce a screenshot of the `saveOrder` method. 

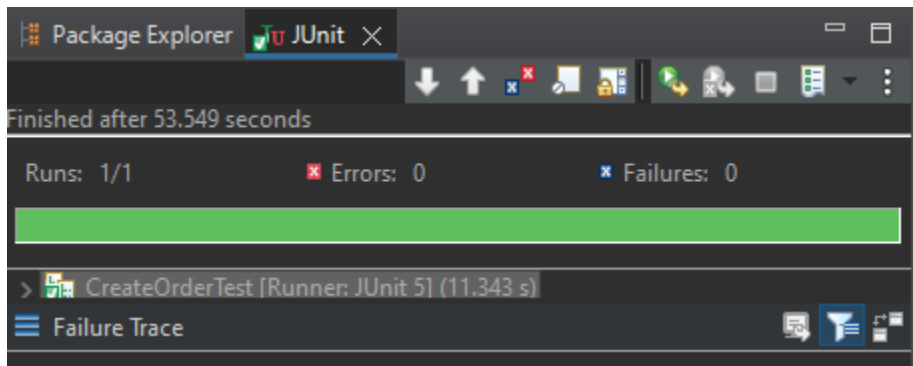


```
246 @Override
247 public Order saveOrder(Customer customer, Jeep jeep, Color color, Engine engine, Tire tire, BigDecimal price, List<Option> options) {
248     SqlParams sqlParams = generateInsertSql(customer, jeep, color, engine, tire, price);
249     KeyHolder keyHolder = new GeneratedKeyHolder();
250     jdbcTemplate.update(sqlParams.sql, sqlParams.source, keyHolder);
251     Long orderPk = keyHolder.getKey().longValue();
252     saveOptions(options, orderPk);
253
254     return Order.builder()
255         .customer(customer)
256         .color(color)
257         .model(jeep)
258         .engine(engine)
259         .options(options)
260         .tire(tire)
261         .price(price)
262         .build();
263 }
```

# Web API Design with Spring Boot Week 16 Coding Assignment

- 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. 

```
41 • @Test
42 public void testCreateOrderReturnsSuccess201() {
43     String body = createOrderBody();
44     String uri = String.format("http://localhost:%d/orders", serverPort);
45
46     HttpHeaders headers = new HttpHeaders();
47     headers.setContentType(MediaType.APPLICATION_JSON);
48     HttpEntity<String> bodyEntity = new HttpEntity<>(body, headers);
49
50     ResponseEntity<Order> response = restTemplate.exchange(uri,
51         HttpMethod.POST, bodyEntity, Order.class);
52
53     assertThat(response.getStatusCode()).isEqualTo(HttpStatus.CREATED);
54     assertThat(response.getBody()).isNotNull();
55
56     Order order = response.getBody();
57     assertThat(order.getCustomerId()).isEqualTo("MORISON_LINA");
58     assertThat(order.getModel().getModelId()).isEqualTo(JeepModel.WRANGLER);
59     assertThat(order.getModel().getTrimLevel()).isEqualTo("Sport Altitude");
60     assertThat(order.getModel().getNumDoors()).isEqualTo(4);
61     assertThat(order.getColor().getColorId()).isEqualTo("EXT_NACHO");
62     assertThat(order.getEngine().getEngineId()).isEqualTo("2_0_TURBO");
63     assertThat(order.getTire().getTireId()).isEqualTo("35_TOYO");
64     assertThat(order.getOptions()).hasSize(6);
65
66
67 }
```



```
08:09:44.464 [main] INFO org.springframework.test.context.support.AbstractContextLoader - Could not detect default resource locations for test class [com.promineotech.jee...
08:09:44.464 [main] INFO org.springframework.test.context.support.AnnotationConfigContextLoaderUtils - Could not detect default configuration classes for test class [com.promineotech.jee...
08:09:45.519 [main] DEBUG org.springframework.test.context.annotation.ClassPathScanningCandidateComponentProvider - Identified candidate component class: file [C:\Users\angva\Documents\workspace-spring-tool-suite-...
08:09:45.523 [main] INFO org.springframework.test.context.SpringBootContextBootstrapper - Found @SpringBootConfiguration com.promineotech.jee.JeeSales for test class com.promineotech.jee.controller...
08:09:46.315 [main] DEBUG org.springframework.test.context.SpringBootContextBootstrapper - @TestExecutionListeners is not present for class [com.promineotech.jee.controller.CreateOrderTest]; using d...
08:09:46.317 [main] INFO org.springframework.test.context.SpringBootContextBootstrapper - Loaded default TestExecutionListener class names from location [META-INF/spring.factories]: [org.springframework...
08:09:46.525 [main] INFO org.springframework.test.context.SpringBootContextBootstrapper - Using TestExecutionListeners: [org.springframework.test.context.web.ServletTestExecutionListener, org.spr...
08:09:46.540 [main] DEBUG org.springframework.test.context.support.AbstractDirtiesContextTestExecutionListener - Before test class: context [DefaultTestContext@24bdb479 testClass = CreateOrderTest, testInstan...
08:09:46.667 [main] DEBUG org.springframework.test.context.support.DependencyInjectionTestExecutionListener - Performing dependency injection for test context [[DefaultTestContext@24bdb479 testClass = CreateO...

Spring
=====
:: Spring Boot ::
(v2.7.6)

2023-01-14 08:09:49.596 INFO 8456 --- [main] c.p.jee.controller.CreateOrderTest : Starting CreateOrderTest using Java 17.0.4.1 on DESKTOP-3PHBVAB with PID 8456 (started by angva in C:\Users\...
2023-01-14 08:09:49.598 INFO 8456 --- [main] c.p.jee.controller.CreateOrderTest : The following 1 profile is active: "test"
2023-01-14 08:10:00.997 INFO 8456 --- [main] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat initialized with port(s): 0 (http)
2023-01-14 08:10:01.044 INFO 8456 --- [main] org.apache.catalina.core.StandardService : Starting service [Tomcat]
2023-01-14 08:10:01.045 INFO 8456 --- [main] o.s.c.c.C.[Tomcat].[localhost].[/] : Starting Servlet engine: [Apache Tomcat/9.0.69]
2023-01-14 08:10:01.780 INFO 8456 --- [main] w.s.c.ServletWebServerApplicationContext : Initializing Spring embedded WebApplicationContext
2023-01-14 08:10:01.781 INFO 8456 --- [main] w.s.c.ServletWebServerApplicationContext : Root WebApplicationContext: initialization completed in 1992 ms
2023-01-14 08:10:23.432 INFO 8456 --- [main] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat started on port(s): 57578 (http) with context path ''
2023-01-14 08:10:23.688 INFO 8456 --- [main] c.p.jee.controller.CreateOrderTest : Started CreateOrderTest in 36.79 seconds (JVM running for 44.536)
2023-01-14 08:10:24.729 INFO 8456 --- [main] com.zaxxer.hikari.HikariDataSource : HikariPool-1 - Starting...
2023-01-14 08:10:27.044 INFO 8456 --- [main] com.zaxxer.hikari.HikariDataSource : HikariPool-1 - Start completed.
2023-01-14 08:10:32.459 INFO 8456 --- [o-auto-1-exec-1] o.a.c.c.C.[Tomcat].[localhost].[/] : Initializing Spring DispatcherServlet 'dispatcherServlet'
2023-01-14 08:10:33.459 INFO 8456 --- [o-auto-1-exec-1] o.s.web.servlet.DispatcherServlet : Initializing Servlet 'dispatcherServlet'
2023-01-14 08:10:33.467 INFO 8456 --- [o-auto-1-exec-1] o.s.web.servlet.DispatcherServlet : Completed initialization in 7 ms
2023-01-14 08:10:34.602 INFO 8456 --- [o-auto-1-exec-1] c.p.j.c.DefaultJeeOrderController : Order request in controller layer :: OrderRequest(customer=MORISON_LINA, model=WRANGLER, trim=Sport Altitude)
2023-01-14 08:10:34.708 INFO 8456 --- [o-auto-1-exec-1] c.p.j.service.DefaultJeeOrderService : Order request in order service layer :: OrderRequest(customer=MORISON_LINA, model=WRANGLER, trim=Sport Altitude)
2023-01-14 08:10:36.224 INFO 8456 --- [o-auto-1-exec-1] com.zaxxer.hikari.HikariDataSource : HikariPool-1 - Shutdown initiated...
2023-01-14 08:10:36.251 INFO 8456 --- [o-auto-1-exec-1] com.zaxxer.hikari.HikariDataSource : HikariPool-1 - Shutdown completed.
```