


# Web API Design with Spring Boot Week 4 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:

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:

- a.i) color
  - a.ii) customer
  - a.iii) engine
  - a.iv) model
  - a.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 b. For example:

```
{
  "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`.

- 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);
```

- 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. 
- 3) In the controller sub-package in src/main/java, create an interface named JeepOrderController. Add @RequestMapping("/orders") as a class-level annotation.
- a) 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).
  - b) Add the @RequestBody annotation to the orderRequest parameter. Make sure to add the RequestBody annotation from the org.springframework.web.bind.annotation package.
  - c) Produce a screenshot of the finished JeepOrderController interface showing no compile errors. 
- 4) Create a class that implements JeepOrderController named DefaultJeepOrderController.
- a) Add @RestController as a class-level annotation.
  - b) Add a log line to the implementing controller method showing the input request body (orderRequest)
  - 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) 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:
    - a.i) @NotNull
    - a.ii) @Length(max = 30)
    - a.iii) @Pattern(regexp = "[\\w\\s]\*")
  - b) Use these annotations for integer types:
    - b.i) @Positive

b.ii) @Min(2)


b.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. 

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 createOrder method in the service was called in the service class. 

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.**

- 12) Replace the entire contents of `DefaultJeepOrderDao.java` with the source found in `DefaultJeepOrderDao.source`. The source file is found in the Source folder of the supplied project resources. After this step you will see errors in `DefaultJeepOrderDao`. This will be fixed shortly.
- 13) 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`.

- 14) 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.

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

- a) Add the `@Transactional` annotation to the `createOrder` method.
- b) 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.
- c) Calculate the price, including all options.

- 16) 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 method from the order service. Produce a screenshot of the service method. 
- b) Write the implementation of the `saveOrder` method in the DAO.
  - b.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.
  - b.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();
```

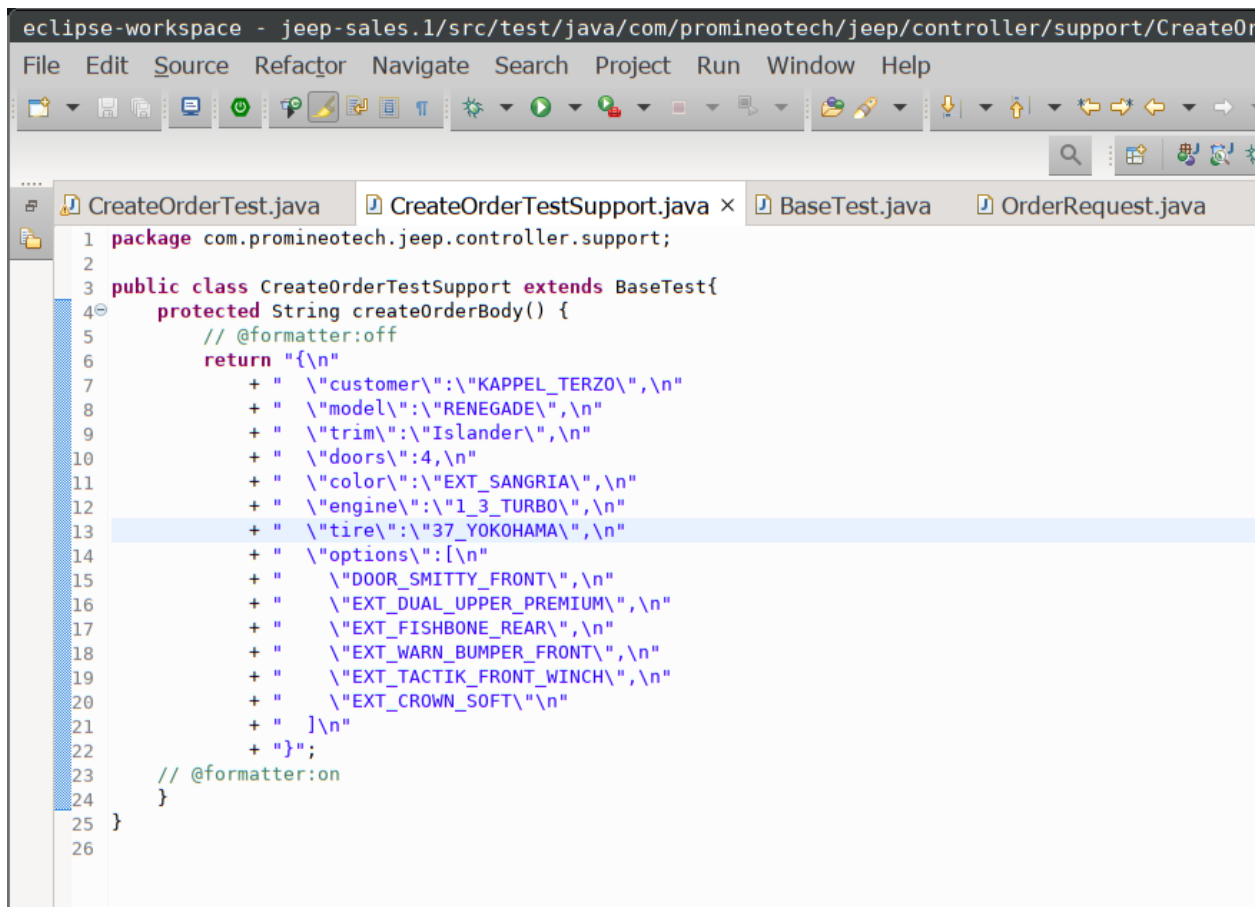
Be sure to extract the order primary key from the `KeyHolder` object into a variable of type `Long` named `orderPK`.
  - b.iii) Write a method named `saveOptions` as shown in the video. This method should have the following method signature:

```
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.

- b.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.
- b.v) Produce a screenshot of the saveOrder method. 
- 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. 

## Screenshots of Code:



```
eclipse-workspace - jeep-sales.1/src/test/java/com/promineotech/jeep/controller/support/CreateOrderTestSupport.java
File Edit Source Refactor Navigate Search Project Run Window Help
CreateOrderTestSupport.java x BaseTest.java OrderRequest.java
1 package com.promineotech.jeep.controller.support;
2
3 public class CreateOrderTestSupport extends BaseTest{
4     protected String createOrderBody() {
5         // @formatter:off
6         return "{\n"
7             + "  \"customer\": \"KAPPEL_TERZO\", \n"
8             + "  \"model\": \"RENEGADE\", \n"
9             + "  \"trim\": \"Islander\", \n"
10            + "  \"doors\": 4, \n"
11            + "  \"color\": \"EXT_SANGRIA\", \n"
12            + "  \"engine\": \"1_3_TURBO\", \n"
13            + "  \"tire\": \"37_YOKOHAMA\", \n"
14            + "  \"options\": [\n"
15            + "    \"DOOR_SMITTY_FRONT\", \n"
16            + "    \"EXT_DUAL_UPPER_PREMIUM\", \n"
17            + "    \"EXT_FISHBONE_REAR\", \n"
18            + "    \"EXT_WARN BUMPER_FRONT\", \n"
19            + "    \"EXT_TACTIK_FRONT_WINCH\", \n"
20            + "    \"EXT_CROWN_SOFT\" \n"
21            + "  ] \n"
22            + "}";
23         // @formatter:on
24     }
25 }
26
```

```

@Autowired
private JdbcTemplate jdbcTemplate;

@Test
void testCreateOrderReturnsSuccess201() {

    //Given; an order as JSON
    String body = createOrderBody();
    String uri = getBaseUriForOrders();

    int numRowsOrder = JdbcTestUtils.countRowsInTable(jdbcTemplate, "orders");
    int numRowsOptions = JdbcTestUtils.countRowsInTable(jdbcTemplate, "order_options");

    HttpHeaders headers = new HttpHeaders();
    headers.setContentType(MediaType.APPLICATION_JSON);
    HttpEntity<String> bodyEntity = new HttpEntity<>(body, headers);

    //When: the order is sent
    ResponseEntity<Order> response = getRestTemplate().exchange(uri, HttpMethod.POST, bodyEntity,
        Order.class);

    //Then: a 201 status is returned
    assertThat(response.getStatusCode()).isEqualTo(HttpStatus.CREATED);

    //And: the returned order is correct
    assertThat(response.getBody()).isNotNull();
    Order order = response.getBody();
    assertThat(order.getCustomer().getCustomerId()).isEqualTo("KAPPEL_TERZO");
    assertThat(order.getModel().getModelId()).isEqualTo(JeepModel.RENEGADE);
    assertThat(order.getModel().getTrimLevel()).isEqualTo("Islander");
    assertThat(order.getModel().getNumDoors()).isEqualTo(4);
    assertThat(order.getColor().getColorId()).isEqualTo("EXT_SANGRIA");
    assertThat(order.getEngine().getEngineId()).isEqualTo("1_3_TURBO");
    assertThat(order.getTire().getTireId()).isEqualTo("37_YOKOHAMA");
    assertThat(order.getOptions()).hasSize(6);
    assertThat(JdbcTestUtils.countRowsInTable(jdbcTemplate, "orders")).isEqualTo(numRowsOrder + 1);
}

```



```
eclipse-workspace - jeep-sales.1/src/main/java/
File Edit Source Refactor Navigate Search Project Run Window Help

Package Explorer BasicJeepOrderController.java DefaultJeepOrderService.java x

jeep-sales.1 src/main/java com.promineotech.jeepp.service DefaultJeepOrderSer

1 package com.promineotech.jeepp.service;
2
3 import java.math.BigDecimal;
22 @Service
23 @Slf4j
24 public class DefaultJeepOrderService implements JeepOrderService {
25
26     @Autowired
27     private JeepOrderDao jeepOrderDao;
28
29
30     @Transactional
31     @Override
32     public Order createOrder(OrderRequest orderRequest) {
33
34         Customer customer = getCustomer(orderRequest);
35         Jeeps model = getModel(orderRequest);
36         Color color = getColor(orderRequest);
37         Engine engine = getEngine(orderRequest);
38         Tire tire = getTire(orderRequest);
39         List<Option> options = getOption(orderRequest);
40         BigDecimal price = model.getBasePrice().add(color.getPrice())
41             .add(engine.getPrice()).add(tire.getPrice());
42
43         for(Option option : options) {
44             price = price.add(option.getPrice());
45         }
46
47         return jeepOrderDao.saveOrder(customer, model,
48             color, engine, tire, price, options);
49     }
50 }
```

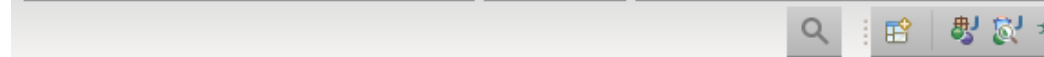
```
eclipse-workspace - jeep-sales.1/src/main/java/com/promineotech/jeep/controller/JeepOrderController - + x
File Edit Source Refactor Navigate Search Project Run Window Help

JeepOrderController.java x
1 package com.promineotech.jeep.controller;
2
3 import org.springframework.http.HttpStatus;
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22 @Validated
23 @RequestMapping(value = "/orders", method = RequestMethod.POST)
24 @OpenAPIDefinition(info = @Info(title = "Jeep Order Service"), servers = {
25     @Server(url = "http://localhost:8080", description = "Local server.")
26 })
27
28
29 public interface JeepOrderController {
30
31
32 //formatter:off
33 @Operation (
34     summary = "Create an order for a Jeep",
35     description = "Returns a created jeep object",
36     responses = {
37
38         @ApiResponse(responseCode = "201",
39             description = "The created jeep is returned",
40             content = @Content (mediaType = "application/json",
41                 schema = @Schema(implementation = Order.class)),
42
43         @ApiResponse(responseCode = "400",
44             description = "The request parameters are invalid",
45             content = @Content (mediaType = "application/json")),
46
47         @ApiResponse(responseCode = "404",
48             description = "A jeep component was not found with the input criteria",
49             content = @Content (mediaType = "application/json")),
50
51         @ApiResponse(responseCode = "500",
52             description = "An unplanned error occurred.",
53             content = @Content (mediaType = "application/json")),
54     },
55     parameters = {
56         @Parameter(
57             name = "orderRequest",
58             required = true,
59             description = "The order as JSON")
60     }
61 )
62 @GetMapping()
63 @ResponseStatus(code = HttpStatus.CREATED)
64
65     Order createOrder (@RequestBody OrderRequest orderRequest);
66 //formatter:on
67 }
68
```

```
eclipse-workspace - jeep-sales.1/src/main/java/com/promineotech/jeep/dao/JeepOrderDao.java
File Edit Source Refactor Navigate Search Project Run Window Help
[Icons] [Icons] [Icons] [Icons] [Icons] [Icons] [Icons] [Icons] [Icons] [Icons] [Icons] [Icons] [Icons] [Icons] [Icons] [Icons]
[Icons] [Icons] [Icons] [Icons] [Icons] [Icons] [Icons] [Icons] [Icons] [Icons] [Icons] [Icons] [Icons] [Icons] [Icons] [Icons]
DefaultJeepOrderDao.java JeepOrderDao.java x
JeepOrderDao fetchCustomer(String) : Optional<Customer>
1 package com.promineotech.jeep.dao;
2
3 import java.math.BigDecimal;
14
15 public interface JeepOrderDao {
16     List<Option> fetchOptions(List<String> optionIds);
17     Optional<Customer> fetchCustomer(String customerId);
18     Optional<Jeeps> fetchModel(JeepModel model, String trim, int doors);
19     Optional<Color> fetchColor(String colorId);
20     Optional<Engine> fetchEngine(String engineId);
21     Optional<Tire> fetchTire(String tireId);
22
23
24 Order saveOrder(Customer customer, Jeeps jeep, Color color, Engine engine,
25                 Tire tire, BigDecimal price, List<Option> options);
26 }
```

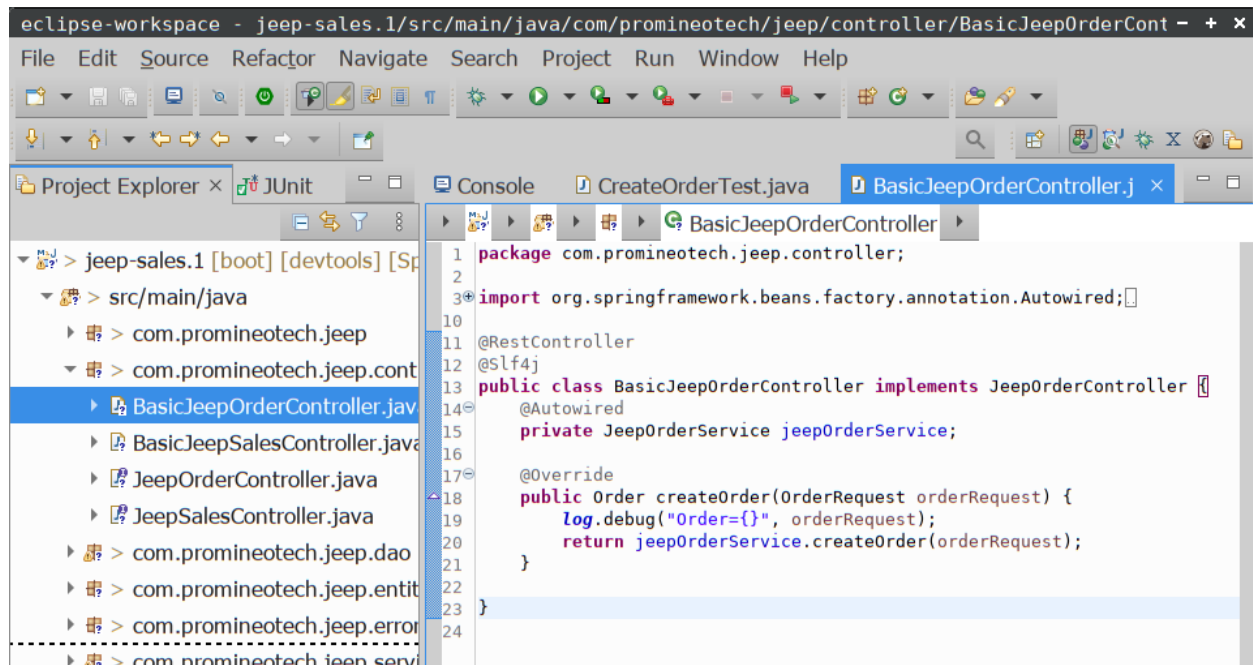
/src/main/java/com/promineotech/jeep/entity/OrderRequest.java -

ate Search Project Run Window Help



OrderRequest.java ×

```
1 package com.promineotech.jee.entity;  
2  
3 import java.util.List;  
4  
5  
6  
7  
8  
9  
10  
11  
12  
13  
14 @Data  
15 public class OrderRequest {  
16     @NotNull  
17     @Length(max = 30)  
18     @Pattern(regexp = "[\\w\\s]*")  
19     private String customer;  
20  
21     @NotNull  
22     private JeepModel model;  
23  
24     @NotNull  
25     @Length(max = 30)  
26     @Pattern(regexp = "[\\w\\s]*")  
27     private String trim;  
28  
29     @Positive  
30     @Min(2)  
31     @Max(4)  
32     private String doors;  
33  
34     @NotNull  
35     @Length(max = 30)  
36     @Pattern(regexp = "[\\w\\s]*")  
37     private String color;  
38  
39     @NotNull  
40     @Length(max = 30)  
41     @Pattern(regexp = "[\\w\\s]*")  
42     private String engine;  
43  
44     @NotNull  
45     @Length(max = 30)  
46     @Pattern(regexp = "[\\w\\s]*")  
47     private String tire;  
48  
49     private List<@NotNull @Length(max = 30)  
50         @Pattern(regexp = "[\\w\\s]*") String> Options;  
51  
52  
53  
54  
55 }  
56
```



```
eclipse-workspace - jeep-sales.1/src/main/java/com/promineotech/jeep/dao/DefaultJeepOrderDao.java - + x
File Edit Source Refactor Navigate Search Project Run Window Help

DefaultJeepOrderDao.java x
saveOrder(Customer, Jeeps, Color, Engine, Tire, BigDecimal, List<Option>) : Order
35 @Slf4j
36 public class DefaultJeepOrderDao implements JeepOrderDao{
37
38     @Autowired
39     private NamedParameterJdbcTemplate jdbcTemplate;
40
41
42
43
44     @Override
45     public Order saveOrder(Customer customer, Jeeps jeep, Color color, Engine engine, Tire tire, BigDecimal price,
46         List<Option> options) {
47         SqlParams params = generateInsertSql(customer, jeep, color, engine, tire, price);
48         KeyHolder keyHolder = new GeneratedKeyHolder();
49         jdbcTemplate.update(params.sql, params.source, keyHolder);
50         Long orderPK = keyHolder.getKey().longValue();
51         saveOptions(options, orderPK);
52         return Order.builder()
53             .orderPK(orderPK)
54             .customer(customer)
55             .model(jeep)
56             .color(color)
57             .engine(engine)
58             .tire(tire)
59             .options(options)
60             .price(price)
61             .build();
62     }
63
64
65
66
67
```

```
eclipse-workspace - jeep-sales.1/src/test/java/com/promineotech/jeep/controller/CreateOrderTest.java - Eclipse
File Edit Source Refactor Navigate Search Project Run Window Help

CreateOrderTest.java x JUnit
jeep-sales.1 src/test/java com.promineotech.jeep.controller CreateOrderTest testCreateOrderReturnsSuccess201() : void

1 package com.promineotech.jeep.controller;
2
3 import static org.assertj.core.api.Assertions.assertThat;
4
5 @SpringBootTest(classes = JeepSales.class, webEnvironment = WebEnvironment.RANDOM_PORT)
6 @ActiveProfiles("test")
7 @Sql(scripts = {"classpath:migrations/V1.0_Jeep_Schema.sql"},
8     config = @SqlConfig(encoding = "utf-8"))
9
10 class CreateOrderTest extends CreateOrderTestSupport {
11
12     @Autowired
13     private JdbcTemplate jdbcTemplate;
14
15     @Test
16     void testCreateOrderReturnsSuccess201() {
17
18         //Given: an order as JSON
19         String body = createOrderBody();
20         String uri = getBaseUrlForOrders();
21
22         int numRowsOrder = JdbcTestUtils.countRowsInTable(jdbcTemplate, "orders");
23         int numRowsOptions = JdbcTestUtils.countRowsInTable(jdbcTemplate, "order_options");
24
25         HttpHeaders headers = new HttpHeaders();
26         headers.setContentType(MediaType.APPLICATION_JSON);
27         HttpEntity<String> bodyEntity = new HttpEntity<>(body, headers);
28
29         //When: the order is sent
30         ResponseEntity<Order> response = getRestTemplate().exchange(uri, HttpMethod.POST, bodyEntity,
31             Order.class);
32
33         //Then: a 201 status is returned
34         assertThat(response.getStatusCode()).isEqualTo(HttpStatus.CREATED);
35
36         //And: the returned order is correct
37         assertThat(response.getBody()).isNotNull();
38         Order order = response.getBody();
39         assertThat(order.getCustomerId().getCustomerId()).isEqualTo("KAPPEL_TERZO");
40         assertThat(order.getModel().getModelId()).isEqualTo(JeepModel.RENEGADE);
41         assertThat(order.getModel().getTrimLevel()).isEqualTo("Islander");
42         assertThat(order.getModel().getNumDoors()).isEqualTo(4);
43         assertThat(order.getColor().getColorId()).isEqualTo("EXT_SANGRIA");
44         assertThat(order.getEngine().getEngineId()).isEqualTo("1_3_TURBO");
45         assertThat(order.getTire().getTireId()).isEqualTo("37_YOKOHAMA");
46         assertThat(order.getOptions()).hasSize(6);
47         assertThat(JdbcTestUtils.countRowsInTable(jdbcTemplate, "orders")).isEqualTo(numRowsOrder + 1);
48         assertThat(JdbcTestUtils.countRowsInTable(jdbcTemplate, "order_options")).isEqualTo(numRowsOptions + 6);
49     }
50 }
```

## Screenshots of Running Application:

```
eclipse-workspace - jeep-sales.1/src/test/java/com/promineotech/jeep/controller/CreateOrderTest.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help

JUnit x

Finished after 8.672 seconds
Runs: 1/1 Errors: 0 Failures: 0

Project Explorer Console x

<terminated> CreateOrderTest [JUnit] /usr/lib/jvm/java-14-openjdk-amd64/bin/java (Sep 30, 2021, 1:47:45 AM - 1:47:55 AM)
01:47:46.537 [main] DEBUG org.springframework.test.context.support.AbstractContextLoader - Did not detect default resource location for test class [com.promineotech.jee
01:47:46.538 [main] INFO org.springframework.test.context.support.AbstractContextLoader - Could not detect default resource locations for test class [com.promineotech.jee
01:47:46.825 [main] DEBUG org.springframework.boot.test.context.SpringBootTestContextBootstrapper - @TestExecutionListeners is not present for class [com.promineotech.jee
01:47:46.825 [main] INFO org.springframework.boot.test.context.SpringBootTestContextBootstrapper - Loaded default TestExecutionListener class names from location [META-INF/spring.factories]: [org.springframework.boot.test.mock.mockito
01:47:46.845 [main] INFO org.springframework.boot.test.context.SpringBootTestContextBootstrapper - Using TestExecutionListeners: [org.springframework.test.context.web.ServletTestExecutionListener@846041a4, org.springframework.test.conte
01:47:46.850 [main] DEBUG org.springframework.test.context.support.AbstractDirContextTestExecutionListener - Before test class: context [[DefaultTestContext@2e570ded testClass = CreateOrderTest, testInstance = [null], testMethod = [r
01:47:46.868 [main] DEBUG org.springframework.test.context.support.DependencyInjectionTestExecutionListener - Performing dependency injection for test context [[DefaultTestContext@2e570ded testClass = CreateOrderTest, testInstance = con
01:47:46.903 [main] DEBUG org.springframework.core.env.StandardEnvironment - Activating profiles [test]
01:47:46.911 [main] DEBUG org.springframework.test.context.support.TestPropertySourceUtils - Adding inlined properties to environment: {spring.jmx.enabled=false, org.springframework.boot.test.context.SpringBootTestContextBootstrapper=tr

:: Spring Boot ::
      ____ _
     / ___/| | | |
    / /   | | | |
   / /___| | | |
  /_____|_|_|_|
  (v2.5.4)

2021-09-30 01:47:47.429 INFO 30592 --- [main] c.p.jee
2021-09-30 01:47:47.430 DEBUG 30592 --- [main] c.p.jee
2021-09-30 01:47:47.431 INFO 30592 --- [main] c.p.jee
2021-09-30 01:47:48.900 INFO 30592 --- [main] .s.d.r.c
2021-09-30 01:47:48.920 INFO 30592 --- [main] .s.d.r.c
2021-09-30 01:47:49.399 INFO 30592 --- [main] trationDelegatesBeanPostProcessorChecker
2021-09-30 01:47:49.494 INFO 30592 --- [main] .w.s.a.s
2021-09-30 01:47:50.092 INFO 30592 --- [main] o.s.b.w
2021-09-30 01:47:50.119 INFO 30592 --- [main] o.a.pac
2021-09-30 01:47:50.120 INFO 30592 --- [main] org.apac
2021-09-30 01:47:50.264 INFO 30592 --- [main] o.a.c.c
2021-09-30 01:47:50.264 INFO 30592 --- [main] w.s.c
2021-09-30 01:47:52.695 INFO 30592 --- [main] com.zaxx
2021-09-30 01:47:52.968 INFO 30592 --- [main] com.zaxx
2021-09-30 01:47:53.259 INFO 30592 --- [main] o.s.b.w
2021-09-30 01:47:53.278 INFO 30592 --- [main] c.p.jee
2021-09-30 01:47:54.652 INFO 30592 --- [o-auto-1-exec-1] o.a.c.c
2021-09-30 01:47:54.653 INFO 30592 --- [o-auto-1-exec-1] o.s.web
2021-09-30 01:47:54.656 INFO 30592 --- [o-auto-1-exec-1] o.s.web
2021-09-30 01:47:54.770 DEBUG 30592 --- [o-auto-1-exec-1] c.p.j
2021-09-30 01:47:54.921 INFO 30592 --- [onShutdownHook] com.zaxx
2021-09-30 01:47:54.926 INFO 30592 --- [onShutdownHook] com.zaxx

: Starting CreateOrderTest using Java 14.0.2 on gvcxxy with PID 30592 (started by cheetah in /home/cheetah/eclipse-workspace/Spring-Boot-
: Running with Spring Boot v2.5.4, Spring v5.3.9
: The following profiles are active: test
: Bootstrapping Spring Data JDBC repositories in DEFAULT mode.
: Finished Spring Data repository scanning in 15 ms. Found 0 JDBC repository interfaces.
: Bean 'org.springframework.ws.config.annotation.DelegatingWsConfiguration' of type [org.springframework.ws.config.annotation.Delegating
: Supporting [WS-Addressing August 2004, WS-Addressing 1.0]
: Tomcat initialized with port(s): 0 (http)
: Starting service [Tomcat]
: Starting Servlet engine: [Apache Tomcat/9.0.52]
: Initializing Spring embedded WebApplicationContext
: Root WebApplicationContext: initialization completed in 2791 ms
: HikariPool-1 - Starting...
: HikariPool-1 - Start completed.
: Tomcat started on port(s): 34067 (http) with context path ''
: Started CreateOrderTest in 6.363 seconds (JVM running for 7.617)
: Initializing Spring DispatcherServlet 'dispatcherServlet'
: Initializing Servlet 'dispatcherServlet'
: Completed initialization in 3 ms
: Order= orderRequest(customer=KAPPEL_TERZO, model=RENEGADE, trim=Islander, doors=4, color=EXT_SANGRIA, engine=1_3_TURBO, tire=37_YOKOHAM
: HikariPool-1 - Shutdown initiated...
: HikariPool-1 - Shutdown completed.
```



eclipse-workspace - jeep-sales.1/src/test/java/com/promineotech/jeep/controller/CreateOrderTest.je - + x

File Edit Source Refactor Navigate Search Project Run Window Help

Projec Conso JUnit x

Finished after 9.332 seconds

Runs: 1/1 Errors: 0 Failures: 1

CreateOrderTest [Runner: JUnit 5]

testCreateOrderReturnsSuccess2

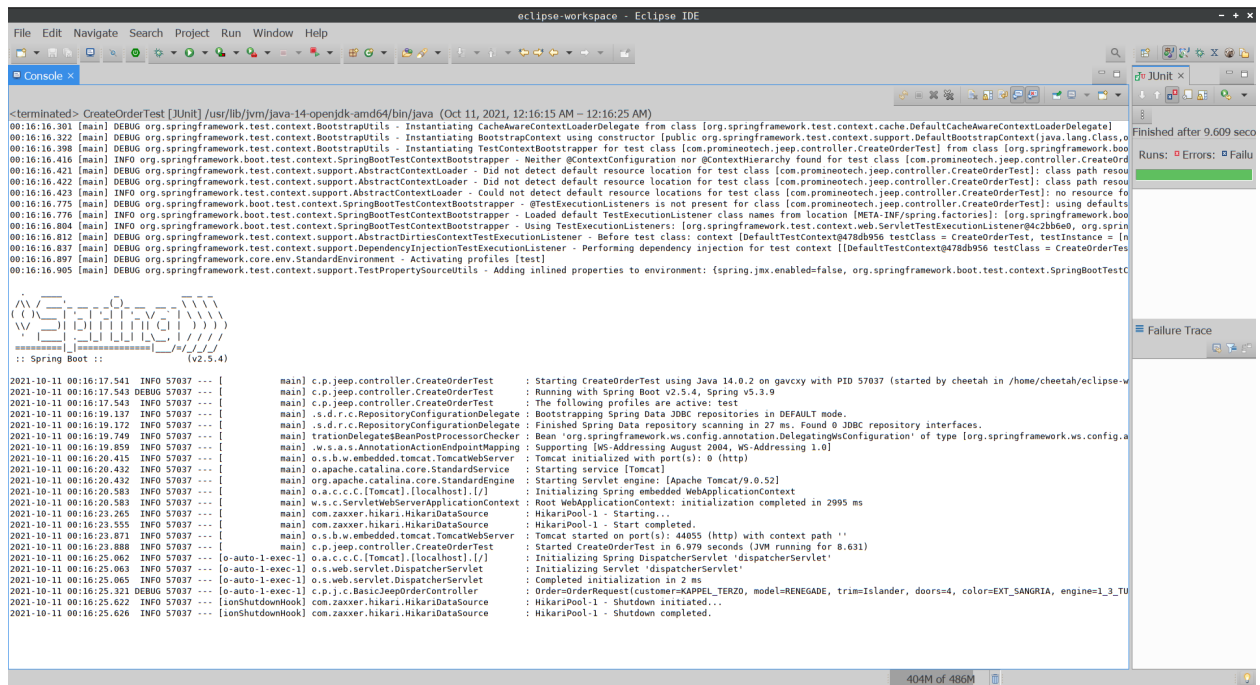
Failure Trace

java.lang.AssertionError:  
Expecting actual not to be null

at com.promineotech.jeep.controller.  
at java.base/jdk.internal.reflect.Nativ  
at java.base/jdk.internal.reflect.Nativ  
at java.base/jdk.internal.reflect.Deleg  
at java.base/java.lang.reflect.Method  
at org.junit.platform.commons.util.Re  
at org.junit.jupiter.engine.execution.  
at org.junit.jupiter.engine.execution.  
at org.junit.jupiter.engine.extension.  
at org.junit.jupiter.engine.extension.  
at org.junit.jupiter.engine.extension.

CreateOrderTest.java x

```
1 package com.promineotech.jeep.controller;
2
3 import static org.assertj.core.api.Assertions.assertThat;
4
5 @SpringBootTest(classes = JeepSales.class, webEnvironment = WebEnvironment
6 @ActiveProfiles("test")
7 @Sql(scripts = {"classpath:migrations/V1.0__Jeep_Schema.sql"},
8 config = @SqlConfig(encoding = "utf-8"))
9
10 class CreateOrderTest extends CreateOrderTestSupport {
11
12     @Test
13     void testCreateOrderReturnsSuccess201() {
14
15         //Given; an order as JSON
16         String body = createOrderBody();
17         String uri = getBaseUriForOrders();
18
19         HttpHeaders headers = new HttpHeaders();
20         headers.setContentType(MediaType.APPLICATION_JSON);
21         HttpEntity<String> bodyEntity = new HttpEntity<>(body, headers);
22
23         //When: the order is sent
24         ResponseEntity<Order> response = getRestTemplate().exchange(uri, HttpM
25         Order.class);
26
27         //Then: a 201 status is returned
28         assertThat(response.getStatusCode()).isEqualTo(HttpStatus.CREATED);
29
30         //And: the returned order is correct
31         assertThat(response.getBody()).isNotNull();
32         Order order = response.getBody();
33         assertThat(order.getCustomerId()).isEqualTo("KAPPEL_TERZ");
34         assertThat(order.getModel().getModelId()).isEqualTo(JeepModel.RENEGADE);
35         assertThat(order.getModel().getTrimLevel()).isEqualTo("Islander");
36         assertThat(order.getModel().getNumDoors()).isEqualTo(4);
37         assertThat(order.getColor().getColorId()).isEqualTo("EXT_SANGRIA");
38         assertThat(order.getEngine().getEngineId()).isEqualTo("1_3_TURBO");
39         assertThat(order.getTire().getTireId()).isEqualTo("37_YOKOHAMA");
40         assertThat(order.getOptions()).hasSize(6);
41     }
42 }
```



```
<terminated> CreateOrderTest [JUnit] /usr/lib/jvm/java-14-openjdk-amd64/bin/java (Oct 11, 2021, 12:16:15 AM - 12:16:25 AM)
00:16:16.301 [main] DEBUG org.springframework.test.context.BootstrapUtils - Instantiating CacheAwareContextLoaderDelegate from class [org.springframework.test.context.cache.DefaultCacheAwareContextLoaderDelegate]
00:16:16.322 [main] DEBUG org.springframework.test.context.BootstrapUtils - Instantiating BootstrapContext using constructor [public org.springframework.test.context.support.DefaultBootstrapContext(java.lang.Class,org.springframework.test.context.CacheAwareContextLoaderDelegate)]
00:16:16.398 [main] DEBUG org.springframework.test.context.BootstrapUtils - Instantiating TestContextBootstrapper for test class [com.pronineotech.jeepp.controller.CreateOrderTest] from class [org.springframework.test.context.support.DefaultTestContextBootstrapper]
00:16:16.416 [main] INFO org.springframework.boot.test.context.SpringBootTest - Neither @ContextConfiguration nor @ContextHierarchy found for test class [com.pronineotech.jeepp.controller.CreateOrderTest]: class path resource [META-INF/spring.factories]
00:16:16.421 [main] DEBUG org.springframework.test.context.support.AbstractContextLoader - Did not detect default resource location for test class [com.pronineotech.jeepp.controller.CreateOrderTest]: class path resource [META-INF/spring.factories]
00:16:16.422 [main] DEBUG org.springframework.test.context.support.AbstractContextLoader - Did not detect default resource location for test class [com.pronineotech.jeepp.controller.CreateOrderTest]: class path resource [META-INF/spring.factories]
00:16:16.423 [main] INFO org.springframework.test.context.support.AbstractContextLoader - Could not detect default resource locations for test class [com.pronineotech.jeepp.controller.CreateOrderTest]: no resource found for [com.pronineotech.jeepp.controller.CreateOrderTest]
00:16:16.775 [main] DEBUG org.springframework.boot.test.context.SpringBootTest - @TestExecutionListeners is not present for class [com.pronineotech.jeepp.controller.CreateOrderTest]: using defaults
00:16:16.776 [main] INFO org.springframework.boot.test.context.SpringBootTest - Loaded default TestExecutionListener class names from location [META-INF/spring.factories]: [org.springframework.boot.test.autoconfigure.TestExecutionListeners, org.springframework.test.context.support.DefaultTestExecutionListeners]
00:16:16.804 [main] INFO org.springframework.boot.test.context.SpringBootTest - Using TestExecutionListeners: [org.springframework.test.context.web.ServletTestExecutionListener@44c2bb0b, org.springframework.test.context.support.DefaultTestExecutionListeners@44c2bb0b]
00:16:16.812 [main] DEBUG org.springframework.test.context.support.AbstractDirtiesContextTestExecutionListener - Before test class: context [DefaultTestContext@478db956 testClass = CreateOrderTest, testInstance = [com.pronineotech.jeepp.controller.CreateOrderTest], testMethod = [CreateOrderTest], testArguments = [], testContextClassLoader = [com.pronineotech.jeepp.controller.CreateOrderTest.class]]
00:16:16.837 [main] DEBUG org.springframework.test.context.support.DefaultTestExecutionListener - Performing dependency injection for test context [DefaultTestContext@478db956 testClass = CreateOrderTest, testInstance = [com.pronineotech.jeepp.controller.CreateOrderTest], testMethod = [CreateOrderTest], testArguments = [], testContextClassLoader = [com.pronineotech.jeepp.controller.CreateOrderTest.class]]
00:16:16.897 [main] DEBUG org.springframework.core.env.StandardEnvironment - Activating profiles [test]
00:16:16.905 [main] DEBUG org.springframework.test.context.support.TestPropertySourceUtils - Adding inlined properties to environment: {spring.jmx.enabled=false, org.springframework.boot.test.context.SpringBootTestContextBootstrapper=true}

:: Spring Boot ::
(v2.5.4)

2021-10-11 00:16:17.541 INFO 57037 --- [main] c.p.jeepp.controller.CreateOrderTest : Starting CreateOrderTest using Java 14.0.2 on gacvxy with PID 57037 (started by cheetah in /home/cheetah/eclipse-workspace)
2021-10-11 00:16:17.543 DEBUG 57037 --- [main] c.p.jeepp.controller.CreateOrderTest : Running with Spring Boot v2.5.4, Spring v5.3.9
2021-10-11 00:16:17.543 INFO 57037 --- [main] c.p.jeepp.controller.CreateOrderTest : The following profiles are active: test
2021-10-11 00:16:19.137 INFO 57037 --- [main] s.d.r.c.RepositoryConfigurationDelegate : Bootstrapping Spring Data JDBC repositories in DEFAULT mode.
2021-10-11 00:16:19.172 INFO 57037 --- [main] s.d.r.c.RepositoryConfigurationDelegate : Finished Spring Data repository scanning in 27 ms. Found 0 JDBC repository interfaces.
2021-10-11 00:16:19.749 INFO 57037 --- [main] trationDelegateBeanPostProcessorChecker : Bean 'org.springframework.ws.config.annotation.DelegatingConfiguration' of type [org.springframework.ws.config.annotation.DelegatingConfiguration] is not eligible for processing: class is abstract
2021-10-11 00:16:19.859 INFO 57037 --- [main] s.a.s.AnnotationActionEndpointMapping : Supporting [WS-Addressing August 2004, WS-Addressing 1.0]
2021-10-11 00:16:20.415 INFO 57037 --- [main] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat initialized with port(s): 0 (http)
2021-10-11 00:16:20.432 INFO 57037 --- [main] o.apache.catalina.core.StandardEngine : Starting service [Tomcat]
2021-10-11 00:16:20.432 INFO 57037 --- [main] org.apache.catalina.core.StandardEngine : Starting Servlet engine: [Apache Tomcat/9.0.52]
2021-10-11 00:16:20.583 INFO 57037 --- [main] o.a.c.c.C.[Tomcat].[localhost].[/] : Initializing Spring embedded WebApplicationContext
2021-10-11 00:16:20.583 INFO 57037 --- [main] w.s.c.ServletWebServerApplicationContext : Root WebApplicationContext: initialization completed in 2995 ms
2021-10-11 00:16:23.265 INFO 57037 --- [main] com.zaxxer.hikari.HikariDataSource : HikariPool-1 - Starting...
2021-10-11 00:16:23.555 INFO 57037 --- [main] com.zaxxer.hikari.HikariDataSource : HikariPool-1 - Start completed.
2021-10-11 00:16:23.871 INFO 57037 --- [main] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat started on port(s): 44055 (http) with context path '/'
2021-10-11 00:16:23.888 INFO 57037 --- [main] c.p.jeepp.controller.CreateOrderTest : Started CreateOrderTest in 6.979 seconds (JVM running for 8.631s)
2021-10-11 00:16:25.062 INFO 57037 --- [o-auto-1-exec-1] o.a.c.c.C.[Tomcat].[localhost].[/] : Initializing Spring DispatcherServlet 'dispatcherServlet'
2021-10-11 00:16:25.063 INFO 57037 --- [o-auto-1-exec-1] o.s.web.servlet.DispatcherServlet : Initializing Servlet 'dispatcherServlet'
2021-10-11 00:16:25.065 INFO 57037 --- [o-auto-1-exec-1] o.s.web.servlet.DispatcherServlet : completed initialization in 2 ms
2021-10-11 00:16:25.321 DEBUG 57037 --- [o-auto-1-exec-1] c.p.j.c.BasicJeeppOrderController : Order=OrderRequest(customer=KAPPEL_TERZO, model=RENEGADE, trim=Islander, doors=4, color=EXT_SANGRIA, engine=1_3_TURBO)
2021-10-11 00:16:25.622 INFO 57037 --- [lonShutdownHook] com.zaxxer.hikari.HikariDataSource : HikariPool-1 - Shutdown initiated...
2021-10-11 00:16:25.626 INFO 57037 --- [lonShutdownHook] com.zaxxer.hikari.HikariDataSource : HikariPool-1 - Shutdown completed.
```

URL to GitHub Repository:

<https://github.com/villarr/Spring-Boot-Week-4-Coding-Assignment>