


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:

- 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.jeepp.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:
 - i) `@NotNull`
 - ii) `@Length(max = 30)`
 - iii) `@Pattern(regexp = "[\\w\\s]*")`
 - b) Use these annotations for integer types:
 - 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. 
- 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). 
- 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) 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`.
- 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.

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. 

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. 

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:

2c

```
30
31 @Test
32 void testCreateOrderReturnsSuccess201() {
33     // Given: an order as JSON
34     String body = createOrderBody();
35
36     // When: the order is sent
37
38     // Then: a 201 status is returned
39
40     // And: the returned order is correct
41 }
42
43 protected String createOrderBody() {
44     // formatter:off
45     return "{\n"
46         + "  \"customer\": \"ATTAWAY_HECTOR\", \n"
47         + "  \"model\": \"WRANGLER\", \n"
48         + "  \"trim\": \"Sport Altitude\", \n"
49         + "  \"doors\": \"4\", \n"
50         + "  \"color\": \"EXT_NACHO\", \n"
51         + "  \"engine\": \"2_0_TURBO\", \n"
52         + "  \"tire\": \"35_TOYO\", \n"
53         + "  \"options\": [\n"
54         + "    \"DOOR_QUAD_4\", \n"
55         + "    \"EXT_AEV_LIFT\", \n"
56         + "    \"EXT_WARN_WINCH\", \n"
57         + "    \"EXT_WARN BUMPER_FRONT\", \n"
58         + "    \"EXT_WARN BUMPER REAR\", \n"
59         + "    \"EXT_ARB_COMPRESSOR\" \n"
60         + "  ] \n"
61         + "}";
62     // formatter:on
63 }
64 }
```

2k

```
*CreateOrderTest.java x DefaultJeeOrderDao.java Order.java
38
39 @Test
40 void testCreateOrderReturnsSuccess201() {
41     // Given: an order as JSON
42     String body = createOrderBody();
43     String uri = String.format("http://localhost:%d/orders", serverPort);
44
45     HttpHeaders headers = new HttpHeaders();
46     headers.setContentType(MediaType.APPLICATION_JSON);
47
48     HttpEntity<String> bodyEntity = new HttpEntity<>(body, headers);
49
50     // When: the order is sent
51     ResponseEntity<Order> response = restTemplate.exchange(uri,
52         HttpMethod.POST, bodyEntity, Order.class);
53
54     // Then: a 201 status is returned
55     assertThat(response.getStatusCode()).isEqualTo(HttpStatus.CREATED);
56
57     // And: the returned order is correct
58
59     assertThat(response.getBody()).isNotNull();
60
61     Order order = response.getBody();
62     assertThat(order.getCustomer().getCustomerId()).isEqualTo("ATTAWAY_HECTOR");
63     assertThat(order.getModel().getModelId()).isEqualTo(JeepModel.WRANGLER);
64     assertThat(order.getModel().getTrimLevel()).isEqualTo("Sport Altitude");
65     assertThat(order.getModel().getNumDoors()).isEqualTo(4);
66     assertThat(order.getColor().getColorId()).isEqualTo("EXT_NACHO");
67     assertThat(order.getEngine().getEngineId()).isEqualTo("2_0_TURBO");
68     assertThat(order.getTire().getTireId()).isEqualTo("35_TOYO");
69     assertThat(order.getOptions()).hasSize(6);
70 }
71 }
```

3c

```
Problems Javadoc Declaration Console
<terminated> CreateOrderTest [JUnit] C:\Users\Promineo\Desktop\Promineo\STS\sts-4.14.1.RELEASE\plugins\org.eclipse.justi.openjdk.hotspot.jre.full.win32.x86_64_17.0.2.v20220201-1208\jre\bin\javaw.exe (Jun 13, 2022, 10:42:20 PM)
22:42:22.464 [main] DEBUG org.springframework.boot.test.context.SpringBootTestContextBootstrapper - @TestExecutionListeners is not present for class [c
22:42:22.464 [main] INFO org.springframework.boot.test.context.SpringBootTestContextBootstrapper - Loaded default TestExecutionListener class names from
22:42:22.479 [main] INFO org.springframework.boot.test.context.SpringBootTestContextBootstrapper - Using TestExecutionListeners: [org.springframework.te
22:42:22.481 [main] DEBUG org.springframework.test.context.support.AbstractDirtiesContextTestExecutionListener - Before test class: context [DefaultTest
22:42:22.490 [main] DEBUG org.springframework.test.context.support.DependencyInjectionTestExecutionListener - Performing dependency injection for test c

:: Spring Boot ::
(v2.7.0)

2022-06-13 22:42:22.778 INFO 24528 --- [main] c.p.jeepp.controller.CreateOrderTest : Starting CreateOrderTest using Java 17.0.2 on Khou-
2022-06-13 22:42:22.779 INFO 24528 --- [main] c.p.jeepp.controller.CreateOrderTest : The following 1 profile is active: "test"
2022-06-13 22:42:23.590 INFO 24528 --- [main] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat initialized with port(s): 0 (http)
2022-06-13 22:42:23.598 INFO 24528 --- [main] o.apache.catalina.core.StandardService : Starting service [Tomcat]
2022-06-13 22:42:23.599 INFO 24528 --- [main] org.apache.catalina.core.StandardEngine : Starting Servlet engine: [Apache Tomcat/9.0.63]
2022-06-13 22:42:23.752 INFO 24528 --- [main] o.a.c.c.C.[Tomcat].[localhost].[/] : Initializing Spring embedded WebApplicationContext
2022-06-13 22:42:23.752 INFO 24528 --- [main] w.s.c.ServletWebServerApplicationContext : Root WebApplicationContext: initialization complete
2022-06-13 22:42:24.952 INFO 24528 --- [main] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat started on port(s): 58471 (http) with context
2022-06-13 22:42:24.964 INFO 24528 --- [main] c.p.jeepp.controller.CreateOrderTest : Started CreateOrderTest in 2.454 seconds (JVM runni
2022-06-13 22:42:24.994 INFO 24528 --- [main] com.zaxxer.hikari.HikariDataSource : HikariPool-1 - Starting...
2022-06-13 22:42:25.179 INFO 24528 --- [main] com.zaxxer.hikari.HikariDataSource : HikariPool-1 - Start completed.
2022-06-13 22:42:25.631 INFO 24528 --- [o-auto-1-exec-1] o.a.c.c.C.[Tomcat].[localhost].[/] : Initializing Spring DispatcherServlet 'dispatcherSe
2022-06-13 22:42:25.631 INFO 24528 --- [o-auto-1-exec-1] o.s.web.servlet.DispatcherServlet : Initializing Servlet 'dispatcherServlet'
2022-06-13 22:42:25.632 INFO 24528 --- [o-auto-1-exec-1] o.s.web.servlet.DispatcherServlet : Completed initialization in 1 ms
2022-06-13 22:42:25.768 INFO 24528 --- [ionShutdownHook] com.zaxxer.hikari.HikariDataSource : HikariPool-1 - Shutdown initiated...
2022-06-13 22:42:25.769 INFO 24528 --- [ionShutdownHook] com.zaxxer.hikari.HikariDataSource : HikariPool-1 - Shutdown completed.
```

4c

Package Explorer JUnit x

Finished after 2.075 seconds

Runs: 1/1 Errors: 1 Failures: 0

CreateOrderTest [Runner: JUnit 5] (0.001 s)

testCreateOrderReturnsSuccess201() (0.001 s)

```
Problems Javadoc Declaration Search Console
<terminated> CreateOrderTest [JUnit] C:\Users\Promineo\Desktop\Promineo\STS\sts-4.14.1.RELEASE\plugins\org.eclipse.justi.openjdk.hotspot.jre.full.win32.x86_64_17.0.2.v20220201-1208\jre\bin\javaw.exe (Jun 14,
:: Spring Boot ::
(v2.7.0)

2022-06-14 17:46:44.796 INFO 36244 --- [main] c.p.jeepp.controller.CreateOrderTest : Starting CreateOrderTest using Java 17.0
2022-06-14 17:46:44.797 INFO 36244 --- [main] c.p.jeepp.controller.CreateOrderTest : The following 1 profile is active: "test"
2022-06-14 17:46:45.583 INFO 36244 --- [main] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat initialized with port(s): 0 (http
2022-06-14 17:46:45.591 INFO 36244 --- [main] o.apache.catalina.core.StandardService : Starting service [Tomcat]
2022-06-14 17:46:45.591 INFO 36244 --- [main] org.apache.catalina.core.StandardEngine : Starting Servlet engine: [Apache Tomcat/
2022-06-14 17:46:45.755 INFO 36244 --- [main] o.a.c.c.C.[Tomcat].[localhost].[/] : Initializing Spring embedded WebApplicat
2022-06-14 17:46:45.755 INFO 36244 --- [main] w.s.c.ServletWebServerApplicationContext : Root WebApplicationContext: initializati
2022-06-14 17:46:46.958 INFO 36244 --- [main] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat started on port(s): 53913 (http)
2022-06-14 17:46:46.966 INFO 36244 --- [main] c.p.jeepp.controller.CreateOrderTest : Started CreateOrderTest in 2.446 seconds
2022-06-14 17:46:46.989 INFO 36244 --- [main] com.zaxxer.hikari.HikariDataSource : HikariPool-1 - Starting...
2022-06-14 17:46:47.175 INFO 36244 --- [main] com.zaxxer.hikari.HikariDataSource : HikariPool-1 - Start completed.
2022-06-14 17:46:47.684 INFO 36244 --- [o-auto-1-exec-1] o.a.c.c.C.[Tomcat].[localhost].[/] : Initializing Spring DispatcherServlet 'd
2022-06-14 17:46:47.685 INFO 36244 --- [o-auto-1-exec-1] o.s.web.servlet.DispatcherServlet : Initializing Servlet 'dispatcherServlet'
2022-06-14 17:46:47.685 INFO 36244 --- [o-auto-1-exec-1] o.s.web.servlet.DispatcherServlet : Completed initialization in 0 ms
2022-06-14 17:46:47.714 WARN 36244 --- [o-auto-1-exec-1] .w.s.m.s.DefaultHandlerExceptionResolver : Resolved [org.springframework.http.conve
2022-06-14 17:46:47.810 INFO 36244 --- [ionShutdownHook] com.zaxxer.hikari.HikariDataSource : HikariPool-1 - Shutdown initiated...
2022-06-14 17:46:47.812 INFO 36244 --- [ionShutdownHook] com.zaxxer.hikari.HikariDataSource : HikariPool-1 - Shutdown completed.
```


7e

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

8

```
Console × Problems Debug Shell
<terminated> CreateOrderTest [JUnit] C:\Users\Promineo\Desktop\Promineo\STS\sts-4.14.1.RELEASE\plugins\org.eclipse.justi.openjdk.hotspot.jre.full.win32.x86_64.17.0.2.v20220201-1208\jre\bin\javaw.exe (Jun 15, 2022, 6:33:34 PM
2022-06-15 18:33:37.689 INFO 2348 --- [main] c.p.jeep.controller.CreateOrderTest : Started CreateOrderTest in 2.538 seconds (JVM running
for 3.413)
2022-06-15 18:33:37.710 INFO 2348 --- [main] com.zaxxer.hikari.HikariDataSource : HikariPool-1 - Starting...
2022-06-15 18:33:37.867 INFO 2348 --- [main] com.zaxxer.hikari.HikariDataSource : HikariPool-1 - Start completed.
2022-06-15 18:33:38.323 INFO 2348 --- [o-auto-1-exec-1] o.a.c.c.C.[Tomcat].[localhost].[/] : Initializing Spring DispatcherServlet
'dispatcherServlet'
2022-06-15 18:33:38.323 INFO 2348 --- [o-auto-1-exec-1] o.s.web.servlet.DispatcherServlet : Initializing Servlet 'dispatcherServlet'
2022-06-15 18:33:38.324 INFO 2348 --- [o-auto-1-exec-1] o.s.web.servlet.DispatcherServlet : Completed initialization in 1 ms
2022-06-15 18:33:38.454 INFO 2348 --- [o-auto-1-exec-1] c.p.j.service.DefaultJeepOrderService : Order=OrderRequest(customer=ATTAWAY_HECKTOR,
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])
_
```

15a

```
CreateOrderTest.java JeepOrderDao.java JeepSalesDao.java DefaultJeepOrderDao.java DefaultJeepOrderService.java ×
25 public class DefaultJeepOrderService implements JeepOrderService {
26
27     @Autowired
28     private JeepOrderDao jeepOrderDao;
29
30     @Transactional
31     @Override
32     public Order createOrder(OrderRequest orderRequest) {
33         log.info("Order={}", orderRequest);
34         Customer customer = getCustomer(orderRequest);
35         Jeep jeep = getModel(orderRequest);
36         Color color = getColor(orderRequest);
37         Engine engine = getEngine(orderRequest);
38         Tire tire = getTire(orderRequest);
39         List<Option> options = getOption(orderRequest);
40
41         BigDecimal price = jeep.getBasePrice().add(color.getPrice())
42             .add(engine.getPrice()).add(tire.getPrice());
43
44         for (Option option : options) {
45             price = price.add(option.getPrice());
46         }
47
48         return jeepOrderDao.saveOrder(customer, jeep, color, engine, tire, price,
49             options);
50     }
51 }
```

15bv

```
38     @Override
39     public Order saveOrder(Customer customer, Jeep jeep, Color color, Engine engine, Tire tire, BigDecimal price,
40         List<Option> options) {
41         SqlParams params =
42             generateInsertSql(customer, jeep, color, engine, tire, price);
43
44         KeyHolder keyHolder = new GeneratedKeyHolder();
45         jdbcTemplate.update(params.sql, params.source, keyHolder);
46
47         Long orderPK = keyHolder.getKey().longValue();
48         saveOptions(options, orderPK);
49
50         // formatter:off
51         return Order.builder()
52             .orderPK(orderPK)
53             .customer(customer)
54             .model(jeep)
55             .color(color)
56             .engine(engine)
57             .tire(tire)
58             .options(options)
59             .price(price)
60             .build();
61         // formatter:on
62     }
63 }
```

15c

```
Finished after 3.867 seconds
Runs: 1/1 Errors: 0 Failures: 0
> CreateOrderTest [Runner: JUnit 5] (0.878 s)
```

Screenshots of Running Application:



```
Console x Problems Debug Shell Search
jeep-sales - JeepSales [Spring Boot App] C:\Users\Promineo\Desktop\Promineo\STS\sts-4.14.1.RELEASE\plugins\org.eclipse.justi.openjdk.hotspot.jre.full.win32.x86_64_17.0.2.v20220201-1208\jre\bin\javaw.exe (Jun 15, 2022, 7:18:02 PM)

:: Spring Boot ::
(v2.7.0)

2022-06-15 19:18:03.967 INFO 31876 --- [ restartedMain] com.promineotech.jee... : Starting JeepSales using Java 17.0.2 on Khou-laptop w
2022-06-15 19:18:03.967 INFO 31876 --- [ restartedMain] com.promineotech.jee... : No active profile set, falling back to 1 default prof
2022-06-15 19:18:03.993 INFO 31876 --- [ restartedMain] .e.DevToolsPropertyDefaultsPostProcessor : Devtools property defaults active! Set 'spring.devto
2022-06-15 19:18:03.993 INFO 31876 --- [ restartedMain] .e.DevToolsPropertyDefaultsPostProcessor : For additional web related logging consider setting t
2022-06-15 19:18:04.626 INFO 31876 --- [ restartedMain] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat initialized with port(s): 8080 (http)
2022-06-15 19:18:04.631 INFO 31876 --- [ restartedMain] o.apache.catalina.core.StandardService : Starting service [Tomcat]
2022-06-15 19:18:04.631 INFO 31876 --- [ restartedMain] org.apache.catalina.core.StandardEngine : Starting Servlet engine: [Apache Tomcat/9.0.63]
2022-06-15 19:18:04.737 INFO 31876 --- [ restartedMain] o.a.c.c.C.[Tomcat].[localhost].[/] : Initializing Spring embedded WebApplicationContext
2022-06-15 19:18:04.737 INFO 31876 --- [ restartedMain] w.s.c.ServletWebServerApplicationContext : Root WebApplicationContext: initialization completed
2022-06-15 19:18:05.292 INFO 31876 --- [ restartedMain] o.s.b.d.a.OptionalLiveReloadServer : LiveReload server is running on port 35729
2022-06-15 19:18:05.314 INFO 31876 --- [ restartedMain] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat started on port(s): 8080 (http) with context p
2022-06-15 19:18:05.321 INFO 31876 --- [ restartedMain] com.promineotech.jee... : Started JeepSales in 1.607 seconds (JVM running for 2
```

URL to GitHub Repository:

<https://github.com/kvang789/spring-boot-week-16-hw.git>