

# 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. 

```
14 @Sql
15 class CreateOrderTest {
16
17     @Test
18     void testCreateOrderReturnsSuccess201() {
19         fail("Not yet implemented");
20     }
21 }
22
23 protected String createOrderBody() {
24     // @formatter: off
25     return "{\n"
26         + "  \"customer\": \"MORISON_LINA\", \n"
27         + "  \"model\": \"WRANGLER\", \n"
28         + "  \"trim\": \"Sport Altitude\", \n"
29         + "  \"doors\": 4, \n"
30         + "  \"color\": \"EXT_NACHO\", \n"
31         + "  \"engine\": \"2_0_TURBO\", \n"
32         + "  \"tire\": \"35_TOYO\", \n"
33         + "  \"options\": [\n"
34         + "    \"DOOR_QUAD_4\", \n"
35         + "    \"EXT_AEV_LIFT\", \n"
36         + "    \"EXT_WARN_WINCH\", \n"
37         + "    \"EXT_WARN BUMPER_FRONT\", \n"
38         + "    \"EXT_WARN BUMPER_REAR\", \n"
39         + "    \"EXT_ARB_COMPRESSOR\", \n"
40         + "  ] \n"
41         + "}";
42     // @formatter: on
43 }
44
45 }
```

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.

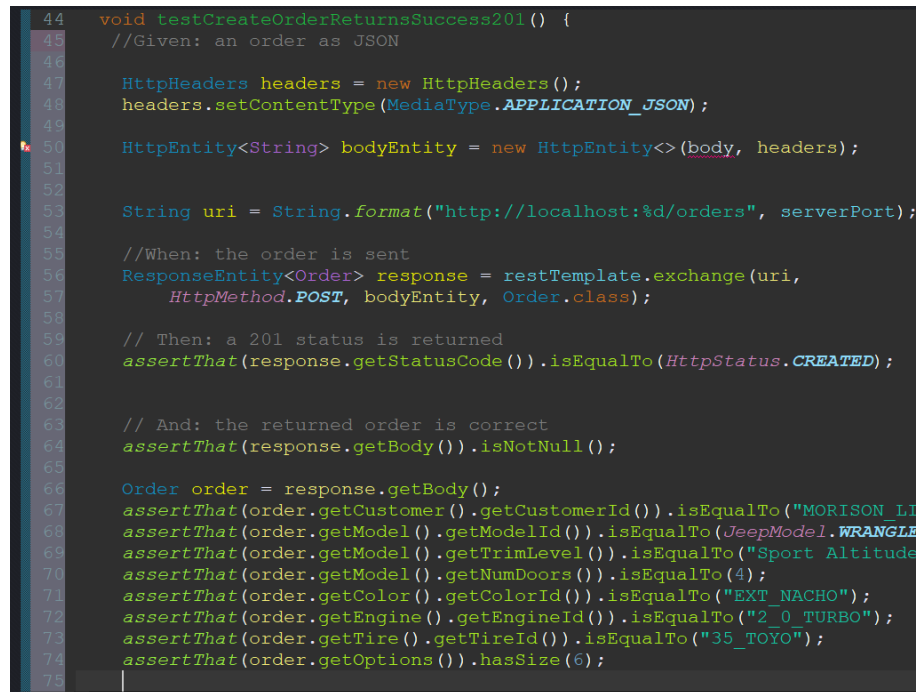
```
ResponseEntity<Order> response = restTemplate.exchange(uri,
    HttpMethod.POST, bodyEntity, Order.class);
```

- 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. 



```
44 void testCreateOrderReturnsSuccess201() {
45     //Given: an order as JSON
46
47     HttpHeaders headers = new HttpHeaders();
48     headers.setContentType(MediaType.APPLICATION_JSON);
49
50     HttpEntity<String> bodyEntity = new HttpEntity<>(body, headers);
51
52
53     String uri = String.format("http://localhost:%d/orders", serverPort);
54
55     //When: the order is sent
56     ResponseEntity<Order> response = restTemplate.exchange(uri,
57         HttpMethod.POST, bodyEntity, Order.class);
58
59     // Then: a 201 status is returned
60     assertThat(response.getStatusCode()).isEqualTo(HttpStatus.CREATED);
61
62
63     // And: the returned order is correct
64     assertThat(response.getBody()).isNotNull();
65
66     Order order = response.getBody();
67     assertThat(order.getCustomer().getCustomerId()).isEqualTo("MORISON_LINA");
68     assertThat(order.getModel().getModelId()).isEqualTo(JeepModel.WRANGLER);
69     assertThat(order.getModel().getTrimLevel()).isEqualTo("Sport Altitude");
70     assertThat(order.getModel().getNumDoors()).isEqualTo(4);
71     assertThat(order.getColor().getColorId()).isEqualTo("EXT_NACHO");
72     assertThat(order.getEngine().getEngineId()).isEqualTo("2_0_TURBO");
73     assertThat(order.getTire().getTireId()).isEqualTo("35_TOYO");
74     assertThat(order.getOptions()).hasSize(6);
75 }
```

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

- c) Produce a screenshot of the finished JeepOrderController interface showing no compile errors.

The screenshot shows an IDE with the following components:

- Project Explorer:** Shows a package structure with `CreateOrderTest` and `testCreateOrderReturnsSuccess201`.
- Code Editor:** Displays the `JeepOrderController` interface with annotations like `@Validated`, `@RequestMapping("/orders")`, `@OpenAPIDefinition`, and `@ApiResponse`.
- Failure Trace:** Shows a message: `org.opentest4j.AssertionFailedError: expected: 201 CREATED but was: 404 NOT_FOUND`.
- Console:** Displays the Spring Boot logo and logs indicating the application is running successfully.

- 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.

The screenshot shows an IDE with the following components:

- Project Explorer:** Shows the same package structure as the previous screenshot.
- Code Editor:** Displays the `DefaultJeepOrderController` class implementing `JeepOrderController`. It includes the `@RestController` annotation and a `createOrder` method with a `log.debug` statement.
- Failure Trace:** Shows a message: `org.opentest4j.AssertionFailedError: expected: 201 CREATED but was: 400 BAD_REQUEST`.
- Console:** Displays the Spring Boot logo and logs indicating the application is running, but with a warning message: `[o-auto-1-exec-1] .w.s.m.s.DefaultHandlerExceptionHandlerResolver : Resolve`.

- 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 `createorder@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.

- 

```

7 import javax.validation.constraints.Pattern;
8 import javax.validation.constraints.Positive;
9 import org.hibernate.validator.constraints.Length;
10 import lombok.Data;
11
12 @Data
13 public class OrderRequest {
14     @NotNull
15     @Length(max = 30)
16     @Pattern(regexp = "A-Z0-9_.*")
17     private String customer;
18
19     @NotNull
20     private JeepModel model;
21
22     @NotNull
23     @Length(max = 30)
24     @Pattern(regexp = "[\\w\\s]*")
25     private String trim;
26
27     @Positive
28     @Min(2)
29     @Max(4)
30     private int doors;
31
32     @NotNull
33     @Length(max = 30)
34     @Pattern(regexp = "[\\w\\s]*")
35     private String color;
36
37     @NotNull
38     @Length(max = 30)
39     @Pattern(regexp = "[\\w\\s]*")
40     private String engine;
41
42     @NotNull
43     @Length(max = 30)
44     @Pattern(regexp = "[\\w\\s]*")
45     private String tire;
46
47     private List<@NotNull @Length(max = 30)@Pattern(
48         regexp = "A-Z0-9_.*") String> options;
49 }
50
51

```

- OrderRequest in the console from within the Service Layer). 

```
<terminated> CreateOrderTest [JUnit] C:\Users\Rainsy\Pluto-4.16.0.RELEASE\plugins\org.eclipse.justi.openjdk.hotspot.jre.full.win32.x86_64_17.0.4.v20220903-1038\jre\bin\java.exe (Nov 12, 2022, 3:40:31 AM) [pid: 2316]
  2
  3
  4
  5
  6
  7
  8
  9
 10
 11
 12
 13
 14
 15
 16
 17
 18
 19
 20
 21
 22
 23
 24
 25
 26
 27
 28
 29
 30
 31
 32
 33
 34
 35
 36
 37
 38
 39
 40
 41
 42
 43
 44
 45
 46
 47
 48
 49
 50
 51
 52
 53
 54
 55
 56
 57
 58
 59
 60
 61
 62
 63
 64
 65
 66
 67
 68
 69
 70
 71
 72
 73
 74
 75
 76
 77
 78
 79
 80
 81
 82
 83
 84
 85
 86
 87
 88
 89
 90
 91
 92
 93
 94
 95
 96
 97
 98
 99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
9
```



- 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.

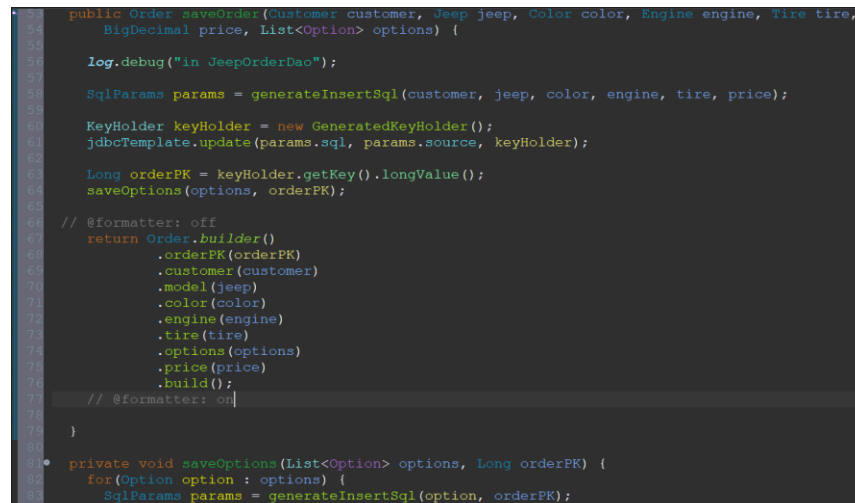
- iii) Write a method named saveOptions as shown in the video. This method should have the following method signature:

```
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. 



```
53 public Order saveOrder(Customer customer, Jeep jeep, Color color, Engine engine, Tire tire,
54     BigDecimal price, List<Option> options) {
55
56     log.debug("in JeepOrderDao");
57
58     SqlParams params = generateInsertSql(customer, jeep, color, engine, tire, price);
59
60     KeyHolder keyHolder = new GeneratedKeyHolder();
61     jdbcTemplate.update(params.sql, params.source, keyHolder);
62
63     Long orderPK = keyHolder.getKey().longValue();
64     saveOptions(options, orderPK);
65
66     // @formatter: off
67     return Order.builder()
68         .orderPK(orderPK)
69         .customer(customer)
70         .model(jeep)
71         .color(color)
72         .engine(engine)
73         .tire(tire)
74         .options(options)
75         .price(price)
76         .build();
77     // @formatter: on
78 }
79
80
81 private void saveOptions(List<Option> options, Long orderPK) {
82     for (Option option : options) {
83         SqlParams params = generateInsertSql(option, orderPK);
```

- 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:

```

1 package com.jeeptest.controller;
2
3 import static org.assertj.core.api.Assertions.assertThat;
4
5 @SpringBootTest(webEnvironment = WebEnvironment.RANDOM_PORT)
6 @ActiveProfiles("test")
7 @Sql(scripts = {
8     "classpath:flyway/migrations/V1.0_Jeep_Schema.sql",
9     "classpath:flyway/migrations/V1.1_Jeep_Data.sql"),
10 config = @SqlConfig(encoding = "UTF-8"))
11 class CreateOrderTest extends CreateOrderTestSupport {
12     @Autowired
13     @Setter
14     private TestRestTemplate restTemplate;
15
16     @LocalServerPort
17     private int serverPort;
18
19     protected String getUriForOrders() {
20         return String.format("http://localhost:%d/orders", serverPort);
21     }
22
23     @Autowired
24     private JdbcTemplate jdbcTemplate;
25
26     /**
27      *
28      */
29     @Test
30     void testCreateOrderReturnsSuccess201() {
31         //Given: an order as JSON
32         String body = createOrderBody();
33         String uri = String.format("http://localhost:%d/orders", serverPort);
34
35         int numRowsOrders = JdbcTestUtils.countRowsInTable(jdbcTemplate, "orders");
36         int numRowsOptions = JdbcTestUtils.countRowsInTable(jdbcTemplate, "order_options");
37
38         //When: the order is sent
39         ResponseEntity<Order> response = getRestTemplate().exchange(uri, HttpMethod.POST, bodyEntity, Order.class);
40
41         //Then: a 201 status is returned
42         assertThat(response.getStatusCode()).isEqualTo(HttpStatus.CREATED);
43
44         //And: the returned order is correct
45         assertThat(response.getBody()).isNotNull();
46
47         Order order = response.getBody();
48         assertThat(order.getCustomerId()).isEqualTo("MORISON LINA");
49         assertThat(order.getModel().getModelId()).isEqualTo(JeepModel.WRANGLER);
50         assertThat(order.getModel().getTrimLevel()).isEqualTo("Sport Altitude");
51         assertThat(order.getModel().getNumDoors()).isEqualTo(4);
52         assertThat(order.getColor().getColorId()).isEqualTo("EXT_NACHO");
53         assertThat(order.getEngine().getEngineId()).isEqualTo("2.0_TURBO");
54         assertThat(order.getTire().getTireId()).isEqualTo("35_TOYO");
55         assertThat(order.getOptions()).hasSize(6);
56
57         assertThat(JdbcTestUtils.countRowsInTable(jdbcTemplate, "orders")).isEqualTo(numRowsOrders + 1);
58         assertThat(JdbcTestUtils.countRowsInTable(jdbcTemplate, "order_options")).isEqualTo(numRowsOptions + 6);
59
60     }
61
62     protected String createOrderBody() {
63         // @formatter: off
64         return "\\c\\n"
65             + "\\customer\\:\\\"MORISON LINA\\\"\\r\\n"
66             + "\\model\\:\\\"WRANGLER\\\"\\r\\n"
67             + "\\trim\\:\\\"Sport Altitude\\\"\\r\\n"
68             + "\\doors\\:4\\r\\n"
69             + "\\color\\:\\\"EXT_NACHO\\\"\\r\\n"
70             + "\\engine\\:\\\"2.0_TURBO\\\"\\r\\n"
71             + "\\tire\\:\\\"35_TOYO\\\"\\r\\n"
72             + "\\options\\:\\\"{\\r\\n"
73             + "\\DOOR_QUAD_4\\\"\\r\\n"
74             + "\\EXT_AEV_LIFT\\\"\\r\\n"
75             + "\\EXT_WARN_WINCH\\\"\\r\\n"
76             + "\\EXT_WARN BUMPER_FRONT\\\"\\r\\n"
77             + "\\EXT_WARN BUMPER_REAR\\\"\\r\\n"
78             + "\\EXT_ARB_COMPRESSOR\\\"\\r\\n"
79             + "\\}\\r\\n"
80             + "\\\"";
81         // @formatter: on
82     }
83 }
84 // end of class

```

```

58
59 int numRowsOrders = JdbcTestUtils.countRowsInTable(jdbcTemplate, "orders");
60 int numRowsOptions = JdbcTestUtils.countRowsInTable(jdbcTemplate, "order_options");
61
62 HttpHeaders headers = new HttpHeaders();
63 headers.setContentType(MediaType.APPLICATION_JSON);
64
65
66 HttpEntity<String> bodyEntity = new HttpEntity<>(body, headers);
67 //When: the order is sent
68 ResponseEntity<Order> response = getRestTemplate().exchange(uri, HttpMethod.POST, bodyEntity, Order.class);
69
70 //Then: a 201 status is returned
71 assertThat(response.getStatusCode()).isEqualTo(HttpStatus.CREATED);
72
73 //And: the returned order is correct
74 assertThat(response.getBody()).isNotNull();
75
76 Order order = response.getBody();
77 assertThat(order.getCustomerId()).isEqualTo("MORISON LINA");
78 assertThat(order.getModel().getModelId()).isEqualTo(JeepModel.WRANGLER);
79 assertThat(order.getModel().getTrimLevel()).isEqualTo("Sport Altitude");
80 assertThat(order.getModel().getNumDoors()).isEqualTo(4);
81 assertThat(order.getColor().getColorId()).isEqualTo("EXT_NACHO");
82 assertThat(order.getEngine().getEngineId()).isEqualTo("2.0_TURBO");
83 assertThat(order.getTire().getTireId()).isEqualTo("35_TOYO");
84 assertThat(order.getOptions()).hasSize(6);
85
86 assertThat(JdbcTestUtils.countRowsInTable(jdbcTemplate, "orders")).isEqualTo(numRowsOrders + 1);
87 assertThat(JdbcTestUtils.countRowsInTable(jdbcTemplate, "order_options")).isEqualTo(numRowsOptions + 6);
88
89 }
90
91
92 protected String createOrderBody() {
93     // @formatter: off

```

```

91
92 protected String createOrderBody() {
93     // @formatter: off
94     return "\\c\\n"
95         + "\\customer\\:\\\"MORISON LINA\\\"\\r\\n"
96         + "\\model\\:\\\"WRANGLER\\\"\\r\\n"
97         + "\\trim\\:\\\"Sport Altitude\\\"\\r\\n"
98         + "\\doors\\:4\\r\\n"
99         + "\\color\\:\\\"EXT_NACHO\\\"\\r\\n"
100        + "\\engine\\:\\\"2.0_TURBO\\\"\\r\\n"
101        + "\\tire\\:\\\"35_TOYO\\\"\\r\\n"
102        + "\\options\\:\\\"{\\r\\n"
103        + "\\DOOR_QUAD_4\\\"\\r\\n"
104        + "\\EXT_AEV_LIFT\\\"\\r\\n"
105        + "\\EXT_WARN_WINCH\\\"\\r\\n"
106        + "\\EXT_WARN BUMPER_FRONT\\\"\\r\\n"
107        + "\\EXT_WARN BUMPER_REAR\\\"\\r\\n"
108        + "\\EXT_ARB_COMPRESSOR\\\"\\r\\n"
109        + "\\}\\r\\n"
110        + "\\\"";
111    // @formatter: on
112 }
113 // end of class
114

```

**Screenshots of Running Application:**

**URL to GitHub Repository:**

[RPador \(Ranon Pador\) \(github.com\)](#)