

# Web API Design with SpringBoot Week 2 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. Lastly, in the Learning Management System, click the "Add Submission" button and paste the URL to your GitHub repository.

## Coding Steps:

Follow the tutorial in the homework section to build an API. Paste screenshots of your code and your postman requests and responses to show the API works. Push your project to GitHub and paste the link below.

## Screenshots of Code:

```
application.properties  log4j2.xml  App.java  ⌵
1  package com.cuevasprojects.inventoryManagement;
2
3
4+ import org.springframework.boot.SpringApplication;
7
8  @ComponentScan("com.cuevasprojects.inventoryManagement")
9  @SpringBootApplication
10 public class App
11 {
12     public static void main( String[] args )
13     {
14         SpringApplication.run(App.class, args);
15     }
16 }
17
```

```

application.properties  log4j2.xml  App.java  CustomerController.java
1 package com.cuevasprojects.inventoryManagement.controller;
2
3 import org.springframework.beans.factory.annotation.Autowired;
4
14 @RestController
15 @RequestMapping("/customers")
16 public class CustomerController {
17
18     @Autowired
19     private CustomerService service;
20
21     @RequestMapping(value="/{id}", method=RequestMethod.GET)
22     public ResponseEntity<Object> getCustomer(@PathVariable Long id){
23         try {
24             return new ResponseEntity<Object>(service.getCustomerById(id), HttpStatus.OK);
25         } catch (Exception e) {
26             return new ResponseEntity<Object>(e.getMessage(), HttpStatus.NOT_FOUND);
27         }
28     }
29
30     @RequestMapping(method=RequestMethod.GET)
31     public ResponseEntity<Object> getCustomers(){
32         return new ResponseEntity<Object>(service.getCustomers(), HttpStatus.OK);
33     }
34
35     @RequestMapping(method=RequestMethod.POST)
36     public ResponseEntity<Object> createCustomer(@RequestBody Customer customer){
37         return new ResponseEntity<Object>(service.createCustomer(customer), HttpStatus.CREATED);
38     }
39
40     @RequestMapping(value="/{id}", method=RequestMethod.PUT)
41     public ResponseEntity<Object> updateCustomer(@RequestBody Customer customer, @PathVariable Long id){
42         try {
43             return new ResponseEntity<Object>(service.updateCustomer(customer, id), HttpStatus.CREATED);
44         } catch (Exception e) {
45             return new ResponseEntity<Object>(e.getMessage(), HttpStatus.NOT_FOUND);
46         }
47     }
48
49     @RequestMapping(value="/{id}", method=RequestMethod.DELETE)
50     public ResponseEntity<Object> deleteCustomer(@PathVariable Long id){
51         try {
52             service.deleteCustomer(id);
53             return new ResponseEntity<Object>("Successfully deleted customer with id: " + id, HttpStatus.OK);
54         } catch (Exception e) {
55             return new ResponseEntity<Object>(e.getMessage(), HttpStatus.NOT_FOUND);
56         }
57     }
58 }
59
60 }

```

```
application.properties  log4j2.xml  App.java  CustomerController.java  OrderController.java
1 package com.cuevasprojects.inventoryManagement.controller;
2
3 import java.util.Set;
4
5
6
7
8
9
10
11
12
13
14
15 @RestController
16 @RequestMapping("customers/{id}/orders")
17 public class OrderController {
18
19     @Autowired
20     private OrderService service;
21
22
23     @RequestMapping(method=RequestMethod.POST)
24     public ResponseEntity<Object> createCustomer(@RequestBody Set<Long> productIds, @PathVariable Long id){
25         try {
26             return new ResponseEntity<Object>(service.submitNewOrder(productIds, id), HttpStatus.CREATED);
27         } catch (Exception e) {
28             return new ResponseEntity<Object>(e, HttpStatus.BAD_REQUEST);
29         }
30     }
31
32
33     @RequestMapping(value="/{orderId}", method=RequestMethod.PUT)
34     public ResponseEntity<Object> updateOrder(@RequestBody Order order, @PathVariable Long orderId){
35         try {
36             if (order.getStatus().equals(OrderStatus.CANCELED)) {
37                 return new ResponseEntity<Object>(service.cancelOrder(orderId), HttpStatus.OK);
38             } else if (order.getStatus().equals(OrderStatus.DELIVERED)) {
39                 return new ResponseEntity<Object>(service.completeOrder(orderId), HttpStatus.OK);
40             }
41             return new ResponseEntity<Object>("Invalid update request.", HttpStatus.BAD_REQUEST);
42         } catch (Exception e) {
43             return new ResponseEntity<Object>(e.getMessage(), HttpStatus.NOT_FOUND);
44         }
45     }
46 }
47
48
```

```
application.properties  log4j2.xml  App.java  CustomerController.java  OrderController.java  ProductController.java
1 package com.cuevasprojects.inventoryManagement.controller;
2
3 import org.springframework.beans.factory.annotation.Autowired;
4
14
15 @RestController
16 @RequestMapping("/products")
17 public class ProductController {
18
19     @Autowired
20     private ProductService service;
21
22     @RequestMapping(method=RequestMethod.GET)
23     public ResponseEntity<Object> getProducts(){
24         return new ResponseEntity<Object>(service.getProducts(), HttpStatus.OK);
25     }
26
27     @RequestMapping(method=RequestMethod.POST)
28     public ResponseEntity<Object> createProduct(@RequestBody Product product){
29         return new ResponseEntity<Object>(service.createProduct(product), HttpStatus.CREATED);
30     }
31
32     @RequestMapping(value =("/{id}", method=RequestMethod.PUT)
33     public ResponseEntity<Object> updateProduct(@RequestBody Product product, @PathVariable Long id){
34         try {
35             return new ResponseEntity<Object>(service.updateProduct(product, id), HttpStatus.OK);
36         } catch (Exception e) {
37             return new ResponseEntity<Object>("Unable to update product.", HttpStatus.BAD_REQUEST);
38         }
39     }
40
41     @RequestMapping(value =("/{id}", method=RequestMethod.DELETE)
42     public ResponseEntity<Object> deleteProduct(@PathVariable Long id){
43         try {
44             service.removeProduct(id);
45             return new ResponseEntity<Object>("Successfully deleted product with id: " + id, HttpStatus.OK);
46         } catch (Exception e) {
47             return new ResponseEntity<Object>("Unable to delete product.", HttpStatus.BAD_REQUEST);
48         }
49     }
50 }
51
```

application.properties log4j2.xml App.java CustomerController.java OrderContro

```
1 package com.cuevasprojects.inventoryManagement.entity;
2
3 import javax.persistence.Entity;
10
11 @Entity
12 public class Address {
13
14     private Long id;
15     private String street;
16     private String city;
17     private String state;
18     private String zip;
19
20     @JsonIgnore
21     private Customer customer;
22
23     @Id
24     @GeneratedValue(strategy = GenerationType.AUTO)
25     public Long getId() {
26         return id;
27     }
28
29     public void setId(Long id) {
30         this.id = id;
31     }
32
33     public String getStreet() {
34         return street;
35     }
36
37     public void setStreet(String street) {
38         this.street = street;
39     }
40
41     public String getCity() {
42         return city;
43     }
44
45     public void setCity(String city) {
46         this.city = city;
47     }
48
49     public String getState() {
50         return state;
51     }
52
53     public void setState(String state) {
54         this.state = state;
55     }
56
57     public String getZip() {
58         return zip;
59     }
60
61     public void setZip(String zip) {
62         this.zip = zip;
63     }
64 }
```

```
64  
65 @OneToOne(mappedBy = "address")  
66 public Customer getCustomer() {  
67     return customer;  
68 }  
69  
70 public void setCustomer(Customer customer) {  
71     this.customer = customer;  
72 }  
73  
74 }  
75
```

```
application.properties  log4j2.xml  Customer.java  ⌕
1 package com.cuevasprojects.inventoryManagement.entity;
2
3 import java.util.Set;
4
15
16 @Entity
17 public class Customer {
18
19     private Long id;
20     private String firstName;
21     private String lastName;
22     private Address address;
23     private MembershipLevel level;
24     private Set<Order> orders;
25
26     @Id
27     @GeneratedValue(strategy = GenerationType.AUTO)
28     public Long getId() {
29         return id;
30     }
31
32     public void setId(Long id) {
33         this.id = id;
34     }
35
36     public String getFirstName() {
37         return firstName;
38     }
39
40     public void setFirstName(String firstName) {
41         this.firstName = firstName;
42     }
43
44     public String getLastName() {
45         return lastName;
46     }
47
48     public void setLastName(String lastName) {
49         this.lastName = lastName;
50     }
51
52     @OneToOne(cascade = CascadeType.ALL)
53     @JoinColumn(name = "id")
54     public Address getAddress() {
55         return address;
56     }
57
58     public void setAddress(Address address) {
59         this.address = address;
60     }
61
62     public MembershipLevel getLevel() {
63         return level;
64     }
65
66     public void setLevel(MembershipLevel level) {
67         this.level = level;
68     }
69 }
```



```
69
70  @OneToMany(mappedBy = "customer")
71  public Set<Order> getOrders(){
72      return orders;
73  }
74
75  public void setOrders(Set<Order> orders) {
76      this.orders = orders;
77  }
78
79  }
80
```

```
application.properties  log4j2.xml  Customer.java  Order.java
1 package com.cuevasprojects.inventoryManagement.entity;
2
3 import java.time.LocalDate;
4
16
17 @Entity
18 public class Order {
19
20     private Long id;
21     private LocalDate ordered;
22     private LocalDate estimatedDelivery;
23     private LocalDate delivered;
24     private double invoiceAmount;
25     private OrderStatus status;
26     private Set<Product> products;
27
28     @JsonIgnore
29     private Customer customer;
30
31     @Id
32     @GeneratedValue(strategy = GenerationType.AUTO)
33     public Long getId() {
34         return id;
35     }
36
37     public void setId(Long id) {
38         this.id = id;
39     }
40
41     public LocalDate getOrdered() {
42         return ordered;
43     }
44
45     public void setOrdered(LocalDate ordered) {
46         this.ordered = ordered;
47     }
48
49     public LocalDate getEstimatedDelivery() {
50         return estimatedDelivery;
51     }
52
53     public void setEstimatedDelivery(LocalDate estimatedDelivery) {
54         this.estimatedDelivery = estimatedDelivery;
55     }
56
57     public LocalDate getDelivered() {
58         return delivered;
59     }
60
61     public void setDelivered(LocalDate delivered) {
62         this.delivered = delivered;
63     }
64
65     public double getInvoiceAmount() {
66         return invoiceAmount;
67     }
68
69     public void setInvoiceAmount(double invoiceAmount) {
```

```
69 public void setInvoiceAmount(double invoiceAmount) {
70     this.invoiceAmount = invoiceAmount;
71 }
72
73 @ManyToMany(mappedBy = "orders")
74 public Set<Product> getProducts(){
75     return products;
76 }
77
78 public void setProducts(Set<Product> products) {
79     this.products = products;
80 }
81
82 @ManyToOne
83 @JoinColumn(name = "customerId")
84 public Customer getCustomer() {
85     return customer;
86 }
87
88 public void setCustomer(Customer customer) {
89     this.customer = customer;
90 }
91
92 public OrderStatus getStatus() {
93     return status;
94 }
95
96 public void setStatus(OrderStatus status) {
97     this.status = status;
98 }
99 }
```

```
application.properties  log4j2.xml  Customer.java  Order.java  Product.java  ⌵
1 package com.cuevasprojects.inventoryManagement.entity;
2
3 import java.util.Set;
15
16 @Entity
17 public class Product {
18
19     private Long id;
20     private String name;
21     private String description;
22     private double price;
23
24     @JsonIgnore
25     private Set<Order> orders;
26
27     @Id
28     @GeneratedValue(strategy = GenerationType.AUTO)
29     public Long getId() {
30         return id;
31     }
32
33     public void setId(Long id) {
34         this.id = id;
35     }
36
37     public String getName() {
38         return name;
39     }
40
41     public void setName(String name) {
42         this.name = name;
43     }
44
45     public String getDescription() {
46         return description;
47     }
48
49     public void setDescription(String description) {
50         this.description = description;
51     }
52
53     public double getPrice() {
54         return price;
55     }
56
57     public void setPrice(double price) {
58         this.price = price;
59     }
60
61     @ManyToMany(cascade = CascadeType.ALL)
62     @JoinTable(name = "product_order",
63         joinColumns = @JoinColumn(name = "orderedId", referencedColumnName = "id"),
64         inverseJoinColumns = @JoinColumn(name = "productId", referencedColumnName = "id"))
65     public Set<Order> getOrders() {
66         return orders;
67     }
68 }
```

```
69     public void setOrders(Set<Order> orders) {
70         this.orders = orders;
71     }
72 }
73
```

```
application.properties  log4j2.xml  Customer.java  AddressRepository.java
1 package com.cuevasprojects.inventoryManagement.repository;
2
3 import org.springframework.data.repository.CrudRepository;
4
5
6
7 public interface AddressRepository extends CrudRepository<Address, Long> {
8
9 }
10
```

```
application.properties  log4j2.xml  Customer.java  AddressRepository.java  CustomerR
1 package com.cuevasprojects.inventoryManagement.repository;
2
3 import org.springframework.data.repository.CrudRepository;
4
5
6
7 public interface CustomerRepository extends CrudRepository<Customer, Long> {
8
9 }
10
```

```
application.properties  log4j2.xml  Customer.java  OrderRepository.java
1 package com.cuevasprojects.inventoryManagement.repository;
2
3 import org.springframework.data.repository.CrudRepository;
4
5
6
7 public interface OrderRepository extends CrudRepository<Order, Long>{
8
9 }
10
```

```
application.properties  log4j2.xml  Customer.java  OrderRepository.java  Product
1 package com.cuevasprojects.inventoryManagement.repository;
2
3 import org.springframework.data.repository.CrudRepository;
4
5
6
7 public interface ProductRepository extends CrudRepository<Product, Long>{
8
9 }
10
```

```
application.properties  log4j2.xml  CustomerService.java  ⌵
1 package com.cuevasprojects.inventoryManagement.service;
2
3 import org.apache.logging.log4j.LogManager;
10
11 @Service
12 public class CustomerService {
13
14     private static final Logger logger = LogManager.getLogger(CustomerService.class);
15
16     @Autowired
17     private CustomerRepository repo;
18
19     public Customer getCustomerById(Long id) throws Exception{
20         try {
21             return repo.findOne(id);
22         } catch (Exception e) {
23             logger.error("Exception occurred while trying to retrieve customer: " + id, e);
24             throw e;
25         }
26     }
27
28     public Iterable<Customer> getCustomers(){
29         return repo.findAll();
30     }
31
32     public Customer createCustomer(Customer customer) {
33         return repo.save(customer);
34     }
35
36     public Customer updateCustomer(Customer customer, Long id) throws Exception{
37         try {
38             Customer oldCustomer = repo.findOne(id);
39             oldCustomer.setAddress(customer.getAddress());
40             oldCustomer.setFirstName(customer.getFirstName());
41             oldCustomer.setLastName(customer.getLastName());
42             oldCustomer.setLevel(customer.getLevel());
43             return repo.save(oldCustomer);
44         } catch (Exception e) {
45             logger.error("Exception occurred while trying to update customer:" + id, e);
46             throw new Exception("Unable to update customer.");
47         }
48     }
49
50     public void deleteCustomer(Long id) throws Exception{
51         try {
52             repo.delete(id);
53         } catch (Exception e) {
54             logger.error("Exception occurred while trying to delete customer: " + id, e);
55             throw new Exception("Unable to delete customer.");
56         }
57     }
58
59 }
```

```
application.properties  log4j2.xml  CustomerService.java  OrderService.java
1 package com.cuevasprojects.inventoryManagement.service;
2
3 import java.time.LocalDate;
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21 @Service
22 public class OrderService {
23
24     private static final Logger logger = LogManager.getLogger(OrderService.class);
25     private final int DELIVERY_DAYS = 7;
26
27     @Autowired
28     private OrderRepository repo;
29
30     @Autowired
31     private CustomerRepository customerRepo;
32
33     @Autowired
34     private ProductRepository productRepo;
35
36     public Order submitNewOrder(Set<Long> productIds, Long customerId) throws Exception{
37         try {
38             Customer customer = customerRepo.findOne(customerId);
39             Order order = initializeNewOrder(productIds, customer);
40             return repo.save(order);
41         } catch (Exception e) {
42             logger.error("Exception occurred while trying to create new order for customer:" + customerId, e);
43             throw e;
44         }
45     }
46
47     public Order cancelOrder(Long orderId) throws Exception{
48         try {
49             Order order = repo.findOne(orderId);
50             order.setStatus(OrderStatus.CANCELED);
51             return repo.save(order);
52         } catch (Exception e) {
53             logger.error("Exception occurred while trying to cancel order:" + orderId, e);
54             throw new Exception("Unable to update order.");
55         }
56     }
57
58     public Order completeOrder(Long orderId) throws Exception{
59         try {
60             Order order = repo.findOne(orderId);
61             order.setStatus(OrderStatus.DELIVERED);
62             return repo.save(order);
63         } catch (Exception e) {
64             logger.error("Exception occurred while trying to complete order:" + orderId, e);
65             throw new Exception("Unable to update order.");
66         }
67     }
68 }
```

```

69 private Order initializeNewOrder(Set<Long> productIds, Customer customer) {
70     Order order = new Order();
71     order.setProducts(convertToProductSet(productRepo.findAll(productIds)));
72     order.setOrdered(LocalDate.now());
73     order.setEstimatedDelivery(LocalDate.now().plusDays(DELIVERY_DAYS));
74     order.setCustomer(customer);
75     order.setInvoiceAmount(calculateOrderTotal(order.getProducts(), customer.getLevel()));
76     order.setStatus(OrderStatus.ORDERED);
77     addOrderToProducts(order);
78     return order;
79 }
80
81 private void addOrderToProducts(Order order) {
82     Set<Product> products = order.getProducts();
83     for (Product product : products) {
84         product.getOrders().add(order);
85     }
86 }
87
88
89 private Set<Product> convertToProductSet(Iterable<Product> iterable){
90     Set<Product> set = new HashSet<Product>();
91     for(Product product : iterable) {
92         set.add(product);
93     }
94     return set;
95 }
96
97 private double calculateOrderTotal(Set<Product> products, MembershipLevel level) {
98     double total = 0;
99     for (Product product : products) {
100         total += product.getPrice();
101     }
102     return total - total * level.getDiscount();
103 }
104 }
105

```



```
application.properties  log4j2.xml  CustomerService.java  ProductService.java
1 package com.cuevasprojects.inventoryManagement.service;
2
3 import org.apache.logging.log4j.LogManager;
10
11 @Service
12 public class ProductService {
13
14     private static final Logger logger = LogManager.getLogger(ProductService.class);
15
16     @Autowired
17     private ProductRepository repo;
18
19     public Iterable<Product> getProducts(){
20         return repo.findAll();
21     }
22
23     public Product createProduct(Product product) {
24         return repo.save(product);
25     }
26
27     public Product updateProduct(Product product, Long id) throws Exception{
28         try {
29             Product oldProduct = repo.findOne(id);
30             oldProduct.setDescription(product.getDescription());
31             oldProduct.setName(product.getName());
32             oldProduct.setPrice(product.getPrice());
33             return repo.save(oldProduct);
34         } catch (Exception e) {
35             logger.error("Exception occurred while trying to update product: " + id, e);
36             throw new Exception ("Unable to update product.");
37         }
38     }
39
40     public void removeProduct(Long id) throws Exception{
41         try {
42             repo.delete(id);
43         } catch (Exception e) {
44             logger.error("Exception occurred while trying to delete product: " + id, e);
45             throw new Exception ("Unable to delete product.");
46         }
47     }
48
49 }
50
```

```
application.properties  log4j2.xml  CustomerService.java  P
1 package com.cuevasprojects.inventoryManagement.util;
2
3 public enum MembershipLevel {
4
5     SILVER(.05),
6     GOLD(.10),
7     DIAMOND(.15),
8     PLATINUM(.20);
9
10    private double discount;
11
12    MembershipLevel(double discount){
13        this.discount = discount;
14    }
15
16    public double getDiscount() {
17        return discount;
18    }
19 }
20
21
```

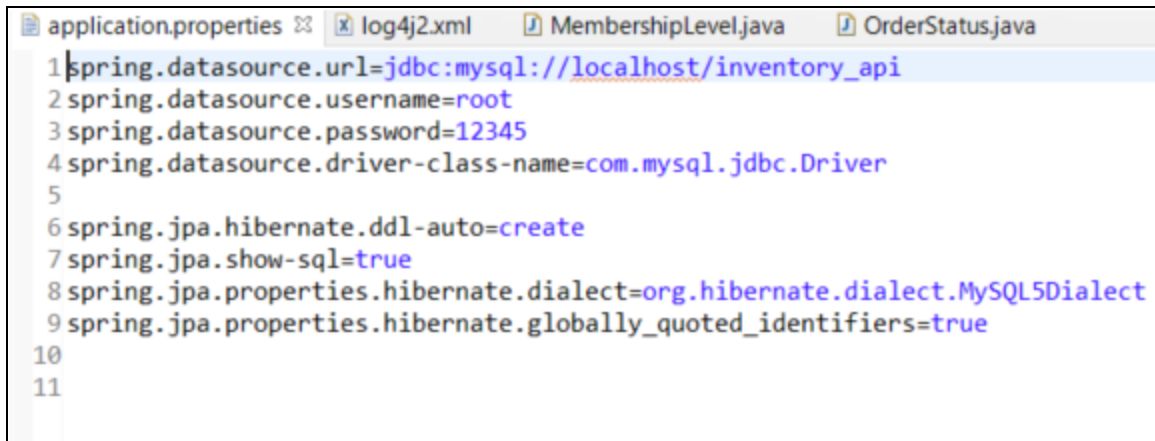
```
application.properties  log4j2.xml  MembershipLevel.java  C
1 package com.cuevasprojects.inventoryManagement.util;
2
3 public enum OrderStatus {
4
5     ORDERED,
6     DELIVERED,
7     CANCELED;
8
9 }
```

application.properties log4j2.xml MembershipLevel.java OrderStatus.java

```

1 <?xml version="1.0" encoding="UTF-8"?>
2 <Configuration status="WARN" monitorInterval="30">
3   <Properties>
4     <Property name="LOG_PATTERN">
5       %d{yyyy-MM-dd HH:mm:ss.SS} %5p ${hostName}
6       ---[%15.15t] %-40.40c{1.} : %m%n%ex
7     </Property>
8   </Properties>
9   <Appenders>
10    <Console name="ConsoleAppender" target="SYSTEM_OUT"
11      follow="true">
12      <PatternLayout pattern="${LOG_PATTERN}"/>
13    </Console>
14    <RollingFile name="FileAppender"
15      fileName="logs/log4j2-example.log"
16      filePattern="logs/log4j2-example.%d{yyyy-MM-dd}-%i.log">
17      <PatternLayout>
18        <Pattern>${LOG_PATTERN}</Pattern>
19      </PatternLayout>
20      <Policies>
21        <sizeBasedTriggeringPolicy size="10MB" />
22      </Policies>
23      <DefaultRolloverStrategy max="10MB"/>
24    </RollingFile>
25  </Appenders>
26  <Loggers>
27    <Logger name="com.example.log4j2example" level="debug"
28      additivity="false">
29      <AppenderRef ref="ConsoleAppender" />
30    </Logger>
31    <Root level="info">
32      <AppenderRef ref="ConsoleAppender"/>
33      <AppenderRef ref="FileAppender" />
34    </Root>
35  </Loggers>
36 </Configuration>

```

A screenshot of an IDE window showing a code editor with the file 'application.properties' selected. The editor contains configuration for a Spring application connecting to a MySQL database. The code is as follows:

```
1 spring.datasource.url=jdbc:mysql://localhost/inventory_api
2 spring.datasource.username=root
3 spring.datasource.password=12345
4 spring.datasource.driver-class-name=com.mysql.jdbc.Driver
5
6 spring.jpa.hibernate.ddl-auto=create
7 spring.jpa.show-sql=true
8 spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.MySQL5Dialect
9 spring.jpa.properties.hibernate.globally_quoted_identifiers=true
10
11
```

**Screenshots of Running Application:**

☐ none ☐ form-data ☐ x-www-form-urlencoded ☒ raw ☐ binary ☐ Graph

```
1 {
2   | "id": 1,
3   | "firstName": "John",
4   | "lastName": "Smith",
5   | "address": {
6   |   "street": "500 N Thomas Rd",
7   |   "city": "Phoenix",
8   |   "state": "AZ",
9   |   "zip": "85008"
10  | },
11  | "level": "GOLD",
12  | "orders": []
13 }
```

Body Cookies Headers (4) Test Results

Pretty Raw Preview Visualize

JSON ▼



```
1 {
2   | "id": 1,
3   | "firstName": "John",
4   | "lastName": "Smith",
5   | "address": {
6   |   "id": 1,
7   |   "street": "500 N Thomas Rd",
8   |   "city": "Phoenix",
9   |   "state": "AZ",
10  |   "zip": "85008"
11  | },
12  | "level": "GOLD",
13  | "orders": []
14 }
```

POST

http://localhost:8080/products

Params

Authorization

Headers (8)

Body ●

Pre-request Script

☐ none

☐ form-data

☐ x-www-form-urlencoded

☒ raw

☐ binary

```
1 {  
2   "name": "Soccer Ball",  
3   "description": "Red Nike Soccer Ball",  
4   "price": "25.00"  
5 }
```

Body

Cookies

Headers (4)

Test Results

Pretty

Raw

Preview

Visualize

JSON



```
1 {  
2   "id": 1,  
3   "name": "Soccer Ball",  
4   "description": "Red Nike Soccer Ball",  
5   "price": 25.0  
6 }
```

POST

http://localhost:8080/customers/1/orders

Params

Authorization

Headers (8)

Body ●

Pre-request Script

Tests

● none

● form-data

● x-www-form-urlencoded

● raw

● binary

● GraphQL

1 [1]

Body

Cookies

Headers (4)

Test Results

Pretty

Raw

Preview

Visualize

JSON



```
1 {
2   "id": 1,
3   "ordered": {
4     "year": 2020,
5     "month": "NOVEMBER",
6     "era": "CE",
7     "dayOfMonth": 22,
8     "dayOfWeek": "SUNDAY",
9     "dayOfYear": 327,
10    "leapYear": true,
11    "monthValue": 11,
12    "chronology": {
13      "calendarType": "iso8601",
14      "id": "ISO"
15    }
16  },
17  "estimatedDelivery": {
18    "year": 2020,
19    "month": "NOVEMBER",
20    "era": "CE",
21    "dayOfMonth": 29,
22    "dayOfWeek": "SUNDAY",
23    "dayOfYear": 334,
24    "leapYear": true,
25    "monthValue": 11,
26    "chronology": {
27      "calendarType": "iso8601",
28      "id": "ISO"
29    }
30  }
31 }
```

```
29     }
30   },
31   "delivered": null,
32   "invoiceAmount": 22.5,
33   "status": "ORDERED",
34   "products": [
35     {
36       "id": 1,
37       "name": "Soccer Ball",
38       "description": "Red Nike Soccer Ball",
39       "price": 25.0
40     }
41   ]
42 }
```

Body Cookies Headers (4) Test Results

Pretty Raw Preview Visualize JSON ↺

```
1 {
2   "id": 2,
3   "name": "Soccer Cleats",
4   "description": "Black Nike Soccer Cleats",
5   "price": 65.0
6 }
```



POST

http://localhost:8080/customers/1/orders

Params

Authorization

Headers (8)

Body ●

Pre-request Script

● none

● form-data

● x-www-form-urlencoded

● raw

● binary

1 [1,2]

Body

Cookies

Headers (4)

Test Results

Pretty

Raw

Preview

Visualize

JSON ▼



```
1 {
2   "id": 2,
3   "ordered": {
4     "year": 2020,
5     "month": "NOVEMBER",
6     "era": "CE",
7     "dayOfMonth": 22,
8     "dayOfWeek": "SUNDAY",
9     "dayOfYear": 327,
10    "leapYear": true,
11    "monthValue": 11,
12    "chronology": {
13      "calendarType": "iso8601",
14      "id": "ISO"
15    }
16  },
17  "estimatedDelivery": {
18    "year": 2020,
19    "month": "NOVEMBER",
20    "era": "CE",
21    "dayOfMonth": 29,
22    "dayOfWeek": "SUNDAY",
23    "dayOfYear": 334,
24    "leapYear": true,
25    "monthValue": 11,
26    "chronology": {
```

```
27         "calendarType": "iso8601",
28         "id": "ISO"
29     }
30 },
31 "delivered": null,
32 "invoiceAmount": 81.0,
33 "status": "ORDERED",
34 "products": [
35     {
36         "id": 2,
37         "name": "Soccer Cleats",
38         "description": "Black Nike Soccer Cleats",
39         "price": 65.0
40     },
41     {
42         "id": 1,
43         "name": "Soccer Ball",
44         "description": "Red Nike Soccer Ball",
45         "price": 25.0
46     }
47 ]
48 }
```

**URL to GitHub Repository:**

**<https://github.com/lcuevas6/week-2-springboot.git>**