

```

1 package com.promineotech.inventoryManagement.entity;
2
3 import java.util.Set;
4
5
6 @Entity
7 public class Customer {
8
9     private Long id;
10    private String firstName;
11    private String lastName;
12    private Address address;
13    private MembershipLevel level;
14    private Set<Orders> orders;
15
16    @Id
17    @GeneratedValue(strategy = GenerationType.AUTO)
18    public Long getId() {
19        return id;
20    }
21    public void setId(Long id) {
22        this.id = id;
23    }
24    public String getFirstName() {
25        return firstName;
26    }
27    public void setFirstName(String firstName) {
28        this.firstName = firstName;
29    }
30    public String getLastName() {
31        return lastName;
32    }
33    public void setLastName(String lastName) {
34        this.lastName = lastName;
35    }
36
37    @OneToOne(cascade = CascadeType.ALL)
38    @JoinColumn(name = "id")
39    public Address getAddress() {
40        return address;
41    }
42    public void setAddress(Address address) {
43        this.address = address;
44    }
45    public MembershipLevel getLevel() {
46        return level;
47    }
48    public void setLevel(MembershipLevel level) {
49        this.level = level;
50    }
51
52    @OneToMany(mappedBy = "customer")
53    public Set<Orders> getOrders() {
54        return orders;
55    }
56    public void setOrders(Set<Orders> orders) {
57        this.orders = orders;
58    }
59 }
60
61
62

```

```

1 package com.promineotech.inventoryManagement.entity;
2
3 import java.time.LocalDate;
4
5
6 @Entity
7 public class Orders {
8
9     private Long id;
10    private LocalDate ordered;
11    private LocalDate estimatedDelivery;
12    private LocalDate delivered;
13    private double invoiceAmount;
14    private OrderStatus status;
15    private Set<Product> products;
16
17    @JsonIgnore
18    private Customer customer;
19
20    @Id
21    @GeneratedValue(strategy = GenerationType.AUTO)
22    public Long getId() {
23        return id;
24    }
25    public void setId(Long id) {
26        this.id = id;
27    }
28    public LocalDate getOrdered() {
29        return ordered;
30    }
31    public void setOrdered(LocalDate ordered) {
32        this.ordered = ordered;
33    }
34    public LocalDate getEstimatedDelivery() {
35        return estimatedDelivery;
36    }
37    public void setEstimatedDelivery(LocalDate estimatedDelivery) {
38        this.estimatedDelivery = estimatedDelivery;
39    }
40    public LocalDate getDelivered() {
41        return delivered;
42    }
43    public void setDelivered(LocalDate delivered) {
44        this.delivered = delivered;
45    }
46    public double getInvoiceAmount() {
47        return invoiceAmount;
48    }
49    public void setInvoiceAmount(double invoiceAmount) {
50        this.invoiceAmount = invoiceAmount;
51    }
52    public OrderStatus getStatus() {
53        return status;
54    }
55    public void setStatus(OrderStatus status) {
56        this.status = status;
57    }
58
59    @ManyToMany(mappedBy = "orders")
60    public Set<Product> getProducts() {
61        return products;
62    }
63
64
65

```

Address.java

```

1 package com.promineotech.inventoryManagement.entity;
2
3 import javax.persistence.Entity;
4
5 @Entity
6 public class Address {
7
8     private Long id;
9     private String street;
10    private String city;
11    private String state;
12    private String zip;
13
14    @JsonIgnore
15    private Customer customer;
16
17    @Id
18    @GeneratedValue(strategy = GenerationType.AUTO)
19    public Long getId() {
20        return id;
21    }
22
23    public void setId(Long id) {
24        this.id = id;
25    }
26
27    public String getStreet() {
28        return street;
29    }
30
31    public void setStreet(String street) {
32        this.street = street;
33    }
34
35    public String getCity() {
36        return city;
37    }
38
39    public void setCity(String city) {
40        this.city = city;
41    }
42
43    public String getState() {
44        return state;
45    }
46
47    public void setState(String state) {
48        this.state = state;
49    }
50
51    public String getZip() {
52        return zip;
53    }
54
55    public void setZip(String zip) {
56        this.zip = zip;
57    }
58
59    @OneToOne(mappedBy = "address")
60    public Customer getCustomer() {
61        return customer;
62    }
63
64    public void setCustomer(Customer customer) {
65        this.customer = customer;
66    }
67
68 }

```

Product.java

```

1 package com.promineotech.inventoryManagement.entity;
2
3 import java.util.Set;
4
5 @Entity
6 public class Product {
7
8     private Long id;
9     private String name;
10    private String description;
11    private double price;
12
13    @JsonIgnore
14    private Set<Orders> orders;
15
16    @Id
17    @GeneratedValue(strategy = GenerationType.AUTO)
18    public Long getId() {
19        return id;
20    }
21
22    public void setId(Long id) {
23        this.id = id;
24    }
25
26    public String getName() {
27        return name;
28    }
29
30    public void setName(String name) {
31        this.name = name;
32    }
33
34    public String getDescription() {
35        return description;
36    }
37
38    public void setDescription(String description) {
39        this.description = description;
40    }
41
42    public double getPrice() {
43        return price;
44    }
45
46    public void setPrice(double price) {
47        this.price = price;
48    }
49
50    @ManyToMany(cascade = CascadeType.ALL)
51    @JoinTable(name = "product_order",
52        joinColumns = @JoinColumn(name = "productId", referencedColumnName = "id"),
53        inverseJoinColumns = @JoinColumn(name = "orderId", referencedColumnName = "id"))
54    public Set<Orders> getOrders() {
55        return orders;
56    }
57
58    public void setOrders(Set<Orders> orders) {
59        this.orders = orders;
60    }
61
62    public void setProducts(Set<Product> products) {
63        this.products = products;
64    }
65
66    @ManyToOne
67    @JoinColumn(name = "customerId")
68    public Customer getCustomer() {
69        return customer;
70    }
71
72    public void setCustomer(Customer customer) {
73        this.customer = customer;
74    }
75
76 }
77
78
79
80
81
82
83
84
85
86
87
88
89

```

CustomerService.java

```
1 package com.promineotech.inventoryManagement.service;
2
3 import java.util.NoSuchElementException;
4
5 @Service
6 public class CustomerService {
7
8     private static final Logger logger = LogManager.getLogger(CustomerService.class);
9
10    @Autowired
11    private CustomerRepository repo;
12
13    public Customer getCustomerById(Long id) {
14        logger.info("Retreiving customer by ID ={}", id);
15        Customer customer = repo.findById(id).orElseThrow();
16
17        if (customer == null) {
18            throw new NoSuchElementException("Customer with ID=" + id + " is not found.");
19        }
20
21        return customer;
22    }
23
24    public Iterable<Customer> getCustomers() {
25        return repo.findAll();
26    }
27
28    public Customer createCustomer (Customer customer) {
29        return repo.save(customer);
30    }
31
32    public Customer updateCustomer (Customer customer, Long id) throws Exception {
33        try {
34            Customer oldCustomer = repo.findById(id).orElseThrow();
35            oldCustomer.setAddress(customer.getAddress());
36            oldCustomer.setFirstName(customer.getFirstName());
37            oldCustomer.setLastName(customer.getLastName());
38            oldCustomer.setLevel(customer.getLevel());
39            return repo.save(oldCustomer);
40        } catch (Exception e) {
41            logger.error("Exception occurred while trying to update customer: " + id, e);
42            throw new Exception("Unable to update customer.");
43        }
44    }
45
46    public void deleteCustomer(Long id) throws Exception {
47        try {
48            repo.deleteById(id);
49        } catch (Exception e) {
50            logger.error("Exception occurred while trying to delete customer: " + id, e);
51            throw new Exception("Unable to delete customer.");
52        }
53    }
54
55 }
```

ProductService.java

```
1 package com.promineotech.inventoryManagement.service;
2
3 import org.apache.logging.log4j.LogManager;
4
5 @Service
6 public class ProductService {
7
8     private static final Logger logger = LogManager.getLogger(ProductService.class);
9
10    @Autowired
11    private ProductRepository repo;
12
13    public Iterable<Product> getProducts() {
14        return repo.findAll();
15    }
16
17    public Product createProduct(Product product) {
18        return repo.save(product);
19    }
20
21    public Product updateProduct(Product product, Long id) throws Exception {
22        try {
23            Product oldProduct = repo.findById(id).orElseThrow();
24            oldProduct.setDescription(product.getDescription());
25            oldProduct.setName(product.getName());
26            oldProduct.setPrice(product.getPrice());
27            return repo.save(oldProduct);
28        } catch (Exception e) {
29            logger.error("Exception occurred while trying to update product: " + id, e);
30            throw new Exception("Unable to update product.");
31        }
32    }
33
34    public void removeProduct(Long id) throws Exception {
35        try {
36            repo.deleteById(id);
37        } catch (Exception e) {
38            logger.error("Exception occurred while trying to delete product: " + id, e);
39            throw new Exception("Unable to delete product.");
40        }
41    }
42
43 }
```

```

OrderService.java 38
1 package com.promineotech.inventoryManagement.service;
2
3 import java.time.LocalDate;
4
5 @Service
6 public class OrderService {
7
8     private static final Logger Logger = LogManager.getLogger(OrderService.class);
9     private final int DELIVERY_DAYS = 7;
10
11     @Autowired
12     private OrderRepository repo;
13
14     @Autowired
15     private CustomerRepository customerRepo;
16
17     @Autowired
18     private ProductRepository productRepo;
19
20     public Orders submitNewOrder(Set<Long> productIds, Long customerId) throws Exception {
21         try {
22             Customer customer = customerRepo.findById(customerId).orElseThrow();
23             Orders order = initializeNewOrder(productIds, customer);
24             return repo.save(order);
25         } catch (Exception e) {
26             Logger.error("Exception occurred while trying to create a new order for customer: " + customerId, e);
27             throw e;
28         }
29     }
30
31     public Orders cancelOrder(Long orderId) throws Exception {
32         try {
33             Orders order = repo.findById(orderId).orElseThrow();
34             order.setStatus(OrderStatus.CANCELED);
35             return repo.save(order);
36         } catch (Exception e) {
37             Logger.error("Exception occurred while trying to cancel order: " + orderId, e);
38             throw new Exception("Unable to update order.");
39         }
40     }
41
42     public Orders completeOrder(Long orderId) throws Exception {
43         try {
44             Orders order = repo.findById(orderId).orElseThrow();
45             order.setStatus(OrderStatus.DELIVERED);
46             return repo.save(order);
47         } catch (Exception e) {
48             Logger.error("Exception occurred while trying to complete order: " + orderId, e);
49             throw new Exception("Unable to update order.");
50         }
51     }
52
53     private Orders initializeNewOrder(Set<Long> productIds, Customer customer) {
54         Orders order = new Orders();
55         order.setProducts(convertToProductSet(productRepo.findAllById(productIds)));
56         order.setOrdered(LocalDate.now());
57         order.setEstimatedDelivery(LocalDate.now().plusDays(DELIVERY_DAYS));
58         order.setCustomer(customer);
59         order.setInvoiceAmount(calculateOrderTotal(order.getProducts(), customer.getLevel()));
60         order.setStatus(OrderStatus.ORDERED);
61         addOrderToProducts(order);
62         return order;
63     }
64
65     private void addOrderToProducts(Orders order) {
66         Set<Product> products = order.getProducts();
67         for (Product product : products) {
68             product.getOrders().add(order);
69         }
70     }
71
72     private double calculateOrderTotal(Set<Product> products, MembershipLevel level) {
73         double total = 0;
74         for (Product product : products) {
75             total += product.getPrice();
76         }
77         return total - total * level.getDiscount();
78     }
79
80     private Set<Product> convertToProductSet(Iterable<Product> iterable) {
81         Set<Product> set = new HashSet<Product>();
82         for (Product product : iterable) {
83             set.add(product);
84         }
85         return set;
86     }
87 }

```

```

1 package com.promineotech.inventoryManagement.repository;
2
3 import org.springframework.data.jpa.repository.JpaRepository;
4
5 import com.promineotech.inventoryManagement.entity.Orders;
6
7 public interface OrderRepository extends JpaRepository <Orders, Long> {
8 }
9
10

```

```

1 package com.promineotech.inventoryManagement.repository;
2
3 import org.springframework.data.jpa.repository.JpaRepository;
4
5
6 import com.promineotech.inventoryManagement.entity.Customer;
7
8 public interface CustomerRepository extends JpaRepository <Customer, Long>{
9 }
10
11

```

```

1 package com.promineotech.inventoryManagement.repository;
2
3 import org.springframework.data.jpa.repository.JpaRepository;
4
5 import com.promineotech.inventoryManagement.entity.Product;
6
7 public interface ProductRepository extends JpaRepository <Product, Long> {
8 }
9
10

```

```

1 package com.promineotech.inventoryManagement.repository;
2
3 import org.springframework.data.jpa.repository.JpaRepository;
4
5
6 import com.promineotech.inventoryManagement.entity.Address;
7
8 public interface AddressRepository extends JpaRepository <Address, Long> {
9 }
10
11

```

```

1 package com.promineotech.inventoryManagement.util;
2
3 public enum OrderStatus {
4
5     ORDERED,
6     DELIVERED,
7     CANCELED;
8 }
9
10

```

```

1 package com.promineotech.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

```

```

1 package com.promineotech.inventoryManagement.controller;
2
3 import java.util.Set;
4
5 @RestController
6 @RequestMapping("/customers/{id}/orders")
7 public class OrderController {
8
9     @Autowired
10     private OrderService service;
11
12     @RequestMapping(method=RequestMethod.POST)
13     public ResponseEntity<Object> createOrder (@RequestBody Set<Long> productIds, @PathVariable Long id) {
14         try {
15             return new ResponseEntity<Object>(service.submitNewOrder(productIds, id), HttpStatus.CREATED);
16         } catch (Exception e) {
17             return new ResponseEntity<Object>(e, HttpStatus.BAD_REQUEST);
18         }
19     }
20
21     @RequestMapping(value="/{orderId}", method=RequestMethod.PUT)
22     public ResponseEntity<Object> updateOrder (@RequestBody Orders order, @PathVariable Long orderId) {
23         try {
24             if (order.getStatus().equals(OrderStatus.CANCELED)) {
25                 return new ResponseEntity<Object>(service.cancelOrder(orderId), HttpStatus.OK);
26             } else if (order.getStatus().equals(OrderStatus.DELIVERED)) {
27                 return new ResponseEntity<Object>(service.completeOrder(orderId), HttpStatus.OK);
28             }
29             return new ResponseEntity<Object>("Invalid update request.", HttpStatus.BAD_REQUEST);
30         } catch (Exception e) {
31             return new ResponseEntity<Object>(e.getMessage(), HttpStatus.NOT_FOUND);
32         }
33     }
34 }

```

```

1 package com.promineotech.inventoryManagement.controller;
2
3 import org.springframework.beans.factory.annotation.Autowired;
4
5 @RestController
6 @RequestMapping("/customers")
7 public class CustomerController {
8
9     @Autowired
10     private CustomerService service;
11
12     @RequestMapping(value="/{id}", method=RequestMethod.GET)
13     @ResponseStatus(value = HttpStatus.OK)
14     public Customer getCustomer (@PathVariable Long id) {
15         return service.getCustomerById(id);
16     }
17
18     @RequestMapping(method=RequestMethod.GET)
19     public ResponseEntity<Object> getCustomers() {
20         return new ResponseEntity<Object>(service.getCustomers(), HttpStatus.OK);
21     }
22
23     @RequestMapping(method=RequestMethod.POST)
24     public ResponseEntity<Object> createCustomer (@RequestBody Customer customer) {
25         return new ResponseEntity<Object>(service.createCustomer(customer), HttpStatus.CREATED);
26     }
27
28     @RequestMapping(value="/{id}", method=RequestMethod.PUT)
29     public ResponseEntity<Object> updateCustomer (@RequestBody Customer customer, @PathVariable Long id) {
30         try {
31             return new ResponseEntity<Object>(service.updateCustomer(customer, id), HttpStatus.OK);
32         } catch (Exception e) {
33             return new ResponseEntity<Object>(e.getMessage(), HttpStatus.NOT_FOUND);
34         }
35     }
36
37     @RequestMapping(value="/{id}", method=RequestMethod.DELETE)
38     public ResponseEntity<Object> deleteCustomer (@PathVariable Long id) {
39         try {
40             service.deleteCustomer(id);
41             return new ResponseEntity<Object>("Successfully deleted customer with id: " + id, HttpStatus.OK);
42         } catch (Exception e) {
43             return new ResponseEntity<Object>(e.getMessage(), HttpStatus.NOT_FOUND);
44         }
45     }
46 }

```

```

1 package com.promineotech.inventoryManagement.controller;
2
3 import org.springframework.beans.factory.annotation.Autowired;
4
5 @RestController
6 @RequestMapping("/products")
7 public class ProductController {
8
9     @Autowired
10     private ProductService service;
11
12     @RequestMapping(method=RequestMethod.GET)
13     public ResponseEntity<Object> getProducts() {
14         return new ResponseEntity<Object>(service.getProducts(), HttpStatus.OK);
15     }
16
17     @RequestMapping(method = RequestMethod.POST)
18     public ResponseEntity<Object> createProduct (@RequestBody Product product) {
19         return new ResponseEntity<Object>(service.createProduct(product), HttpStatus.CREATED);
20     }
21
22     @RequestMapping(value = "/{id}", method = RequestMethod.PUT)
23     public ResponseEntity<Object> updateProduct (@RequestBody Product product, @PathVariable Long id) {
24         try {
25             return new ResponseEntity<Object>(service.updateProduct(product, id), HttpStatus.OK);
26         } catch (Exception e) {
27             return new ResponseEntity<Object>("Unable to update product.", HttpStatus.BAD_REQUEST);
28         }
29     }
30
31     @RequestMapping(value = "/{id}", method=RequestMethod.DELETE)
32     public ResponseEntity<Object> deleteProduce (@PathVariable Long id) {
33         try {
34             service.removeProduct(id);
35             return new ResponseEntity<Object>("Successfully deleted product with id: " + id, HttpStatus.OK);
36         } catch (Exception e) {
37             return new ResponseEntity<Object>("Unable to delete product.", HttpStatus.BAD_REQUEST);
38         }
39     }
40 }

```

```

1 package com.promineotech.inventoryManagement;
2
3 import org.springframework.boot.SpringApplication;
4
5 @ComponentScan("com.promineotech.inventoryManagement")
6 @SpringBootApplication
7 public class App
8 {
9     public static void main( String[] args )
10     {
11         SpringApplication.run(App.class, args);
12     }
13 }

```


POST http://localhost:8080/customers/1

Params Authorization Headers (8) **Body** Pre-request Script Tests Settings

☐ none ☐ form-data ☐ x-www-form-urlencoded ☒ raw ☐ binary ☐ GraphQL **JSON** ▾

```
1 {
2   ... "firstName": "Kip",
3   ... "lastName": "Smith",
4   ... "address": {
5     ... "city": "Phoenix",
6     ... "state": "AZ",
7     ... "street": "1245 Culver St",
8     ... "zip": "95623"
9   },
10  ... "level": "GOLD"
11 }
```

Body Cookies Headers (5) Test Results

🌐 Status: 200

Pretty Raw Preview Visualize **JSON** ▾

```
1 {
2   ... "id": 1,
3   ... "firstName": "Kip",
4   ... "lastName": "Smith",
5   ... "address": {
6     ... "id": 3,
7     ... "street": "1245 Culver St",
8     ... "city": "Phoenix",
9     ... "state": "AZ",
10    ... "zip": "95623"
11  },
12  ... "level": "GOLD",
13  ... "orders": []
14 }
```

POST

http://localhost:8080/products

Send

Params

Authorization

Headers (8)

Body

Pre-request Script

Tests

Settings

Cookies

Beautiful

none

form-data

x-www-form-urlencoded

raw

binary

GraphQL

JSON

1

2

3

4

5

...."name": "7 Wonders",

...."description": "Draft cards to develop your ancient civilization and build its Wonder of the World.",

...."price": 49.99

Body

Cookies

Headers (5)

Test Results

Status: 201 Created

Time: 54 ms

Size: 310 B

Save Response

Pretty

Raw

Preview

Visualize

JSON

1

2

3

4

5

6

...."id": 4,







...."name": "7 Wonders",

...."description": "Draft cards to develop your ancient civilization and build its Wonder of the World.",

...."price": 49.99

HomeWorkspaces ▾ReportsExplore

Search Postman



Upgrade



Overview

POST http://localhost:8080/products

No Environment ▾

http://localhost:8080/products

Save ▾



POST ▾ http://localhost:8080/products

Send ▾

ParamsAuthorizationHeaders (8)Body ●Pre-request ScriptTestsSettings

● none ● form-data ● x-www-form-urlencoded ● raw ● binary ● GraphQL JSON ▾

Cookies


Beautify

12345

....."name": "Red Rising",
....."description": "Craft a hand of powerful characters from the dystopian world of Red Rising",
....."price": 59.99

BodyCookiesHeaders (5)Test Results

Status: 201 CreatedTime: 38 msSize: 302 BSave Response ▾

PrettyRawPreviewVisualizeJSON ▾

123456

....."id": 6,
....."name": "Red Rising",
....."description": "Craft a hand of powerful characters from the dystopian world of Red Rising",
....."price": 59.99

GET

http://localhost:8080/products

Send

ParamsAuthorizationHeaders (6)BodyPre-request ScriptTestsSettingsCookies

none

form-data

x-www-form-urlencoded

raw

binary

GraphQL

This request does not have a body

BodyCookiesHeaders (5)Test Results

Status: 200 OKTime: 41 msSize: 729 BSave Response

PrettyRawPreviewVisualizeJSON

```
1  [
2    {
3      "id": 4,
4      "name": "7 Wonders",
5      "description": "Draft cards to develop your ancient civilization and build its Wonder of the World.",
6      "price": 49.99
7    },
8    {
9      "id": 5,
10     "name": "Root",
11     "description": "Decide the fate of the forest as woodland factions fight for contrasting goal",
12     "price": 39.99
13   },
14   {
15     "id": 6,
16     "name": "Red Rising",
17     "description": "Craft a hand of powerful characters from the dystopian world of Red Rising",
18     "price": 59.99
19   },
20   {
21     "id": 7,
22     "name": "Gloomhaven: Jaws of the Lion",
23     "description": "Vanquish monsters with strategic cardplay in a 25-scenario Gloomhaven campaign.",
24     "price": 49.99
25   }
26 ]
```

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

Params Authorization Headers (8) Body Pre-request Script Tests Settings

none form-data x-www-form-urlencoded raw binary GraphQL JSON

1 [[5,7]]

Body Cookies Headers (5) Test Results Status: 201 Created Time: 252 ms Size: 593 B

Pretty Raw Preview Visualize JSON

```
1 {
2   "id": 8,
3   "ordered": "2021-02-04",
4   "estimatedDeilvery": "2021-02-11",
5   "delivered": null,
6   "invoiceAmount": 80.982,
7   "status": "ORDERED",
8   "products": [
9     {
10      "id": 7,
11      "name": "Gloomhaven: Jaws of the Lion",
12      "description": "Vanquish monsters with strategic cardplay in a 25-scenario Gloomhaven campaign.",
13      "price": 49.99
14    },
15    {
16      "id": 5,
17      "name": "Root",
18      "description": "Decide the fate of the forest as woodland factions fight for contrasting goal",
19      "price": 39.99
20    }
21  ]
22 }
```

<https://github.com/Christinalytle/InventoryManagementAPI.git>