

Web API Design with SpringBoot Week 2 Coding Assignment

Points possible: 70

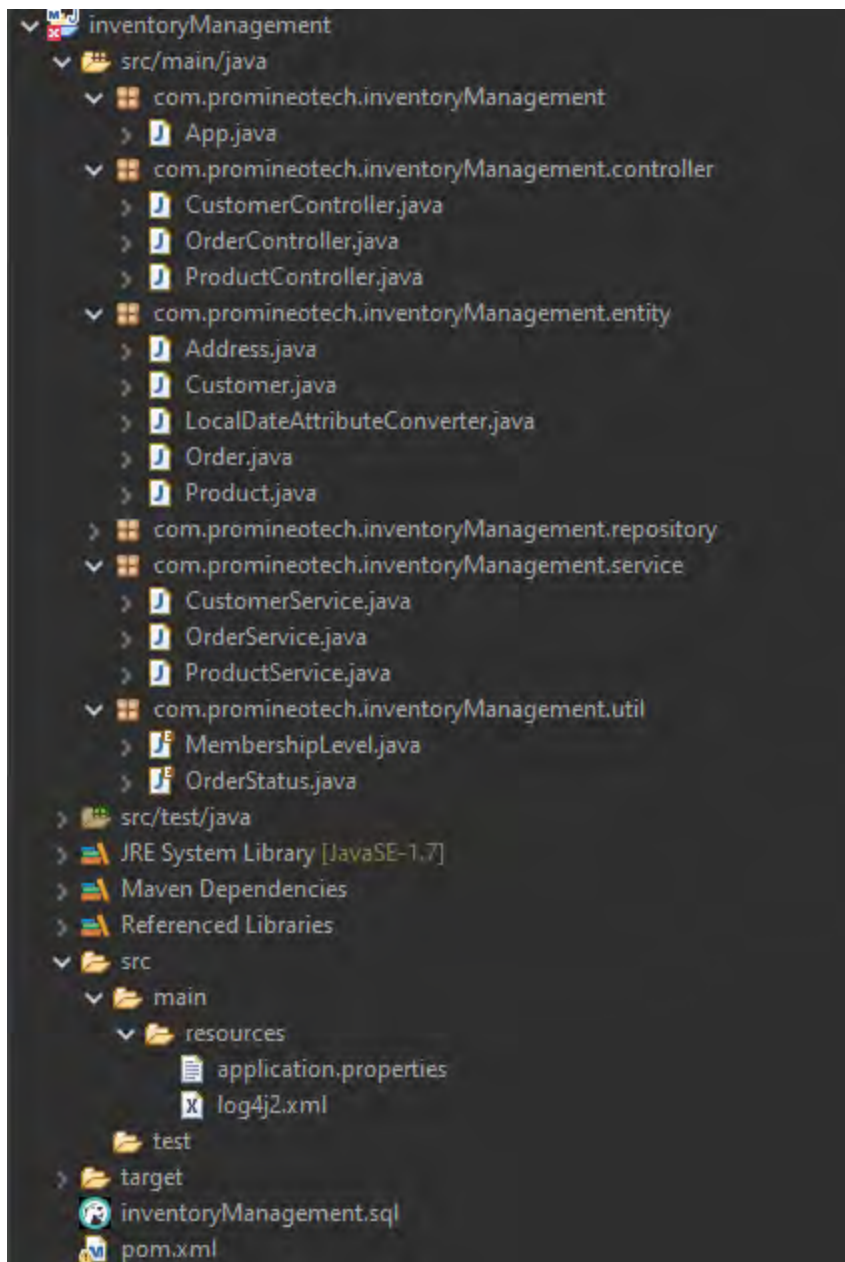
Category	Criteria	% of Grade
Functionality	Does the code work?	25
Organization	Is the code clean and organized? Proper use of white space, syntax, and consistency are utilized. Names and comments are concise and clear.	25
Creativity	Student solved the problems presented in the assignment using creativity 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. Add the URL for this week's repository to this document where instructed and submit this document to your instructor when complete.

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:



com.promineotech.inventoryManagement
App.java

```
App.java X
1 package com.promineotech.inventoryManagement;
2
3 import org.springframework.boot.SpringApplication;
4
5
6
7 @ComponentScan("com.promineotech.inventoryManagement")
8 @SpringBootApplication
9 public class App
10 {
11     public static void main( String[] args )
12     {
13         SpringApplication.run(App.class, args);
14     }
15 }
16
```

com.promineotech.inventoryManagement.controller
CustomerController.java
OrderController.java
ProductController.java

```
CustomerController.java X
1 package com.promineotech.inventoryManagement.controller;
2
3
4 import org.springframework.beans.factory.annotation.Autowired;
5
6
7 @RestController
8 @RequestMapping("/customer")
9 public class CustomerController {
10
11     @Autowired
12     private CustomerService service;
13
14     @RequestMapping(value="/{id}", method=RequestMethod.GET)
15     public ResponseEntity<Object> getCustomer(@PathVariable Long id) {
16         try {
17             return new ResponseEntity<Object>(service.getCustomerById(id), HttpStatus.OK);
18         } catch (Exception e) {
19             return new ResponseEntity<Object>(e.getMessage(), HttpStatus.NOT_FOUND);
20         }
21     }
22
23     @RequestMapping(method=RequestMethod.GET)
24     public ResponseEntity<Object> getCustomer() {
25         return new ResponseEntity<Object>(service.getCustomer(), HttpStatus.OK);
26     }
27
28     @RequestMapping(method=RequestMethod.POST)
29     public ResponseEntity<Object> createCustomer(@RequestBody Customer customer) {
30         return new ResponseEntity<Object>(service.createCustomer(customer), HttpStatus.CREATED);
31     }
32
33     @RequestMapping(value="/{id}", method=RequestMethod.PUT)
34     public ResponseEntity<Object> updateCustomer(@RequestBody Customer customer, @PathVariable Long id) {
35         try {
36             return new ResponseEntity<Object>(service.updateCustomer(customer, id), HttpStatus.OK);
37         } catch (Exception e) {
38             return new ResponseEntity<Object>(e.getMessage(), HttpStatus.NOT_FOUND);
39         }
40     }
41
42     @RequestMapping(value="/{id}", method=RequestMethod.DELETE)
43     public ResponseEntity<Object> deleteCustomer(@PathVariable Long id) {
44         try {
45             service.deleteCustomer(id);
46             return new ResponseEntity<Object>("Successfully deleted customer with id: " + id, HttpStatus.OK);
47         } catch (Exception e) {
48             return new ResponseEntity<Object>(e.getMessage(), HttpStatus.NOT_FOUND);
49         }
50     }
51 }
52
```

```

1 package com.promineotech.inventoryManagement.controller;
2
3 import java.util.Set;
4
5 @RestController
6 @RequestMapping("customer/{id}/orders")
7 public class OrderController {
8
9     @Autowired
10     private OrderService service;
11
12     @RequestMapping(method=RequestMethod.POST)
13     public ResponseEntity<Object> createCustomer(@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 Order 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 import org.springframework.http.HttpStatus;
5 import org.springframework.http.ResponseEntity;
6 import org.springframework.web.bind.annotation.PathVariable;
7 import org.springframework.web.bind.annotation.RequestBody;
8 import org.springframework.web.bind.annotation.RequestMapping;
9 import org.springframework.web.bind.annotation.RequestMethod;
10 import org.springframework.web.bind.annotation.RestController;
11
12 import com.promineotech.inventoryManagement.entity.Product;
13 import com.promineotech.inventoryManagement.service.ProductService;
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: ", HttpStatus.OK);
46         } catch (Exception e) {
47             return new ResponseEntity<Object>("Unable to delete product.", HttpStatus.BAD_REQUEST);
48         }
49     }
50 }

```

- ▼ com.promineotech.inventoryManagement.entity
 - > Address.java
 - > Customer.java
 - > LocalDateAttributeConverter.java
 - > Order.java
 - > Product.java

Address.java X

```

1 package com.promineotech.inventoryManagement.entity;
2
3 import javax.persistence.Entity;
4 import javax.persistence.GeneratedValue;
5 import javax.persistence.GenerationType;
6 import javax.persistence.Id;
7 import javax.persistence.OneToOne;
8
9 import com.fasterxml.jackson.annotation.JsonIgnore;
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
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

```



```

Customer.java x
1 package com.promineotech.inventoryManagement.entity;
2
3 import java.util.Set;
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     public void setId(Long id) {
32         this.id = id;
33     }
34     public String getFirstName() {
35         return firstName;
36     }
37     public void setFirstName(String firstName) {
38         this.firstName = firstName;
39     }
40     public String getLastName() {
41         return lastName;
42     }
43     public void setLastName(String lastName) {
44         this.lastName = lastName;
45     }
46     @OneToOne(cascade = CascadeType.ALL)
47     @JoinColumn(name = "address_id")
48     public Address getAddress() {
49         return address;
50     }
51     public void setAddress(Address address) {
52         this.address = address;
53     }
54     public MembershipLevel getLevel() {
55         return level;
56     }
57     public void setLevel(MembershipLevel level) {
58         this.level = level;
59     }
60     @OneToMany(mappedBy = "customer")
61     public Set<Order> getOrders() {
62         return orders;
63     }
64     public void setOrders(Set<Order> orders) {
65         this.orders = orders;
66     }
67
68
69
70 }
71

```

```

1 package com.promineotech.inventoryManagement.entity;
2
3 import java.sql.Date;
4
5
6 @Converter(autoApply = true)
7 public class LocalDateAttributeConverter implements AttributeConverter<LocalDate, Date> {
8
9     // @Override
10    public Date convertToDatabaseColumn(LocalDate locDate) {
11        return (locDate == null ? null : Date.valueOf(locDate));
12    }
13
14    // @Override
15    public LocalDate convertToEntityAttribute(Date sqlDate) {
16        return (sqlDate == null ? null : sqlDate.toLocalDate());
17    }
18 }
19
20
21
22
23
24

```

```

1 package com.promineotech.inventoryManagement.entity;
2
3 import java.time.LocalDate;
4 import java.util.Set;
5
6 import javax.persistence.Entity;
7 import javax.persistence.GeneratedValue;
8 import javax.persistence.GenerationType;
9 import javax.persistence.Id;
10 import javax.persistence.JoinColumn;
11 import javax.persistence.ManyToMany;
12 import javax.persistence.ManyToOne;
13 import javax.persistence.Table;
14
15 import com.fasterxml.jackson.annotation.JsonIgnore;
16 import com.promineotech.inventoryManagement.util.OrderStatus;
17
18 @Entity
19 @Table(name = "orders")
20 public class Order {
21
22     private Long id;
23     private LocalDate ordered;
24     private LocalDate estimatedDelivery;
25     private LocalDate delivered;
26     private double invoiceAmount;
27     private OrderStatus status;
28     private Set<Product> products;
29
30     @JsonIgnore
31     private Customer customer;
32
33     @Id
34     @GeneratedValue(strategy = GenerationType.AUTO)
35     public Long getId() {
36         return id;
37     }
38     public void setId(Long id) {
39         this.id = id;
40     }
41     public LocalDate getOrdered() {
42         return ordered;
43     }
44     public void setOrdered(LocalDate ordered) {
45         this.ordered = ordered;
46     }
47     public LocalDate getEstimatedDelivery() {
48         return estimatedDelivery;
49     }
50     public void setEstimatedDelivery(LocalDate estimatedDelivery) {
51         this.estimatedDelivery = estimatedDelivery;
52     }
53     public LocalDate getDelivered() {
54         return delivered;
55     }
56     public void setDelivered(LocalDate delivered) {
57         this.delivered = delivered;
58     }
59
60     public double getInvoiceAmount() {
61         return invoiceAmount;
62     }
63     public void setInvoiceAmount(double invoiceAmount) {
64         this.invoiceAmount = invoiceAmount;
65     }
66
67     // It's plain an order can contain many products, and a product can be in many orders.
68     // This is the "owned" side.
69     @ManyToMany(mappedBy = "orders")
70     public Set<Product> getProducts() {
71         return products;
72     }
73     public void setProducts(Set<Product> products) {
74         this.products = products;
75     }
76
77     @ManyToOne
78     @JoinColumn(name = "customerId")
79     public Customer getCustomer() {
80         return customer;
81     }
82     public void setCustomer(Customer customer) {
83         this.customer = customer;
84     }
85
86     public OrderStatus getStatus() {
87         return status;
88     }
89     public void setStatus(OrderStatus status) {
90         this.status = status;
91     }
92 }
93

```

```

1 package com.promineotech.inventoryManagement.entity;
2
3 import java.util.Set;
4
5 import javax.persistence.CascadeType;
6 import javax.persistence.Entity;
7 import javax.persistence.GeneratedValue;
8 import javax.persistence.GenerationType;
9 import javax.persistence.Id;
10 import javax.persistence.JoinColumn;
11 import javax.persistence.JoinTable;
12 import javax.persistence.ManyToMany;
13
14 import com.fasterxml.jackson.annotation.JsonIgnore;
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 = "productId", referencedColumnName = "id"),
64         inverseJoinColumns = @JoinColumn(name = "orderId", 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
74
75 }
76

```


▼ com.promineotech.inventoryManagement.service
 > CustomerService.java
 > OrderService.java
 > ProductService.java

AddressRepository.java X

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

CustomerRepository.java X

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

OrderRepository.java X

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

ProductRepository.java X

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

```
▼ com.promineotech.inventoryManagement.service
  > CustomerService.java
  > OrderService.java
  > ProductService.java
```

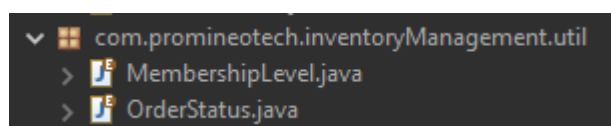
```
CustomerService.java X
1 package com.promineotech.inventoryManagement.service;
2
3
4 import org.apache.logging.log4j.LogManager;
5 import org.apache.logging.log4j.Logger;
6 import org.springframework.beans.factory.annotation.Autowired;
7 import org.springframework.stereotype.Service;
8
9 import com.promineotech.inventoryManagement.entity.Customer;
10 import com.promineotech.inventoryManagement.repository.CustomerRepository;
11
12 @Service
13 public class CustomerService {
14
15     private static final Logger logger = LogManager.getLogger(CustomerService.class);
16
17     @Autowired
18     private CustomerRepository repo;
19
20     public Customer getCustomerById(Long id) throws Exception {
21         try {
22             return repo.findOne(id);
23         } catch (Exception e) {
24             logger.error("Exception occurred while trying to retrieve customer: " + id, e);
25             throw e;
26         }
27     }
28
29     public Iterable<Customer> getCustomer() {
30         return repo.findAll();
31     }
32
33     public Customer createCustomer(Customer customer) {
34         return repo.save(customer);
35     }
36
37     public Customer updateCustomer(Customer customer, Long id) throws Exception {
38         try {
39             Customer oldCustomer = repo.findOne(id);
40             oldCustomer.setAddress(customer.getAddress());
41             oldCustomer.setFirstName(customer.getFirstName());
42             oldCustomer.setLastName(customer.getLastName());
43             oldCustomer.setLevel(customer.getLevel());
44             return repo.save(oldCustomer);
45         } catch (Exception e) {
46             logger.error("Exception occurred while trying to update customer: " + id, e);
47             throw new Exception("unable to update customer.");
48         }
49     }
50
51     public void deleteCustomer(Long id) throws Exception {
52         try {
53             repo.delete(id);
54         } catch (Exception e) {
55             logger.error("Exception occurred while trying to delete customer: " + id, e);
56             throw new Exception("unable to delete customer.");
57         }
58     }
59 }
60 }
```

```

OrderService.java x
1 package com.promineotech.inventoryManagement.service;
2
3 import java.time.LocalDate;
4 import java.util.HashSet;
5 import java.util.Set;
6
7 import org.apache.logging.log4j.LogManager;
8 import org.apache.logging.log4j.Logger;
9 import org.springframework.beans.factory.annotation.Autowired;
10 import org.springframework.stereotype.Service;
11
12 import com.promineotech.inventoryManagement.entity.Customer;
13 import com.promineotech.inventoryManagement.entity.Order;
14 import com.promineotech.inventoryManagement.entity.Product;
15 import com.promineotech.inventoryManagement.repository.CustomerRepository;
16 import com.promineotech.inventoryManagement.repository.OrderRepository;
17 import com.promineotech.inventoryManagement.repository.ProductRepository;
18 import com.promineotech.inventoryManagement.util.MembershipLevel;
19 import com.promineotech.inventoryManagement.util.OrderStatus;
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);

```

```
ProductService.java x
1 package com.promineotech.inventoryManagement.service;
2
3
4 import org.apache.logging.log4j.LogManager;
11
12 @Service
13 public class ProductService {
14
15     private static final Logger logger = (Logger) LogManager.getLogger(ProductService.class);
16
17     @Autowired
18     private ProductRepository repo;
19
20     public Iterable<Product> getProducts() {
21         return repo.findAll();
22     }
23
24     public Product createProduct(Product product) {
25         return repo.save(product);
26     }
27
28     public Product updateProduct(Product product, Long id) throws Exception {
29         try {
30             Product oldProduct = repo.findOne(id);
31             oldProduct.setDescription(product.getDescription());
32             oldProduct.setName(product.getName());
33             oldProduct.setPrice(product.getPrice());
34             return repo.save(oldProduct);
35         } catch (Exception e) {
36             logger.error("Exception occurred while trying to update product: " + id, e);
37             throw new Exception("unable to update product.");
38         }
39     }
40
41     public void removeProduct(Long id) throws Exception {
42         try {
43             repo.delete(id);
44         } catch (Exception e) {
45             logger.error("Exception occurred while trying to delete product: " + id, e);
46             throw new Exception("unable to delete product.");
47         }
48     }
49 }
50 }
51
```



```
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
```

OrderStatus.java X

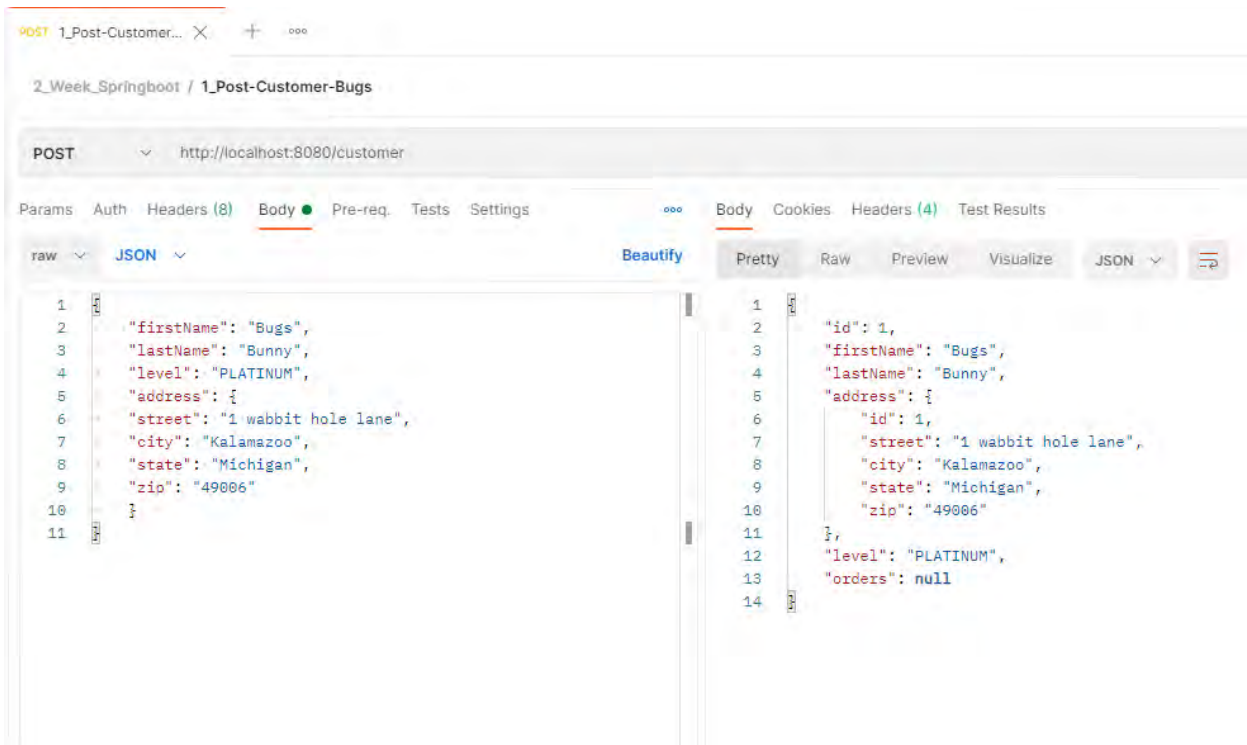
```
1 package com.promineotech.inventoryManagement.util;
2
3 public enum OrderStatus {
4
5     ORDERED,
6     DELIVERED,
7     CANCELED;
8 }
9
```


Screenshots of Running Application:

Text document titled “JeffBeck_Week_2_CodingAssignment_Postman_Tests” saved to Git details data used and steps for testing code in postman. Each step in text document corresponds with steps listed below:

Text document titled “CommandPrompt-mvn spring-boot run” is the CLI documentations showing the spring-boot API was able to make establish the necessary endpoints in order to carry out and test the API in Postman...

(1) POST (body/raw/JSON) http://localhost:8080/customer



(2) //2_Post-Customer-Elmer

POST http://localhost:8080/customer

Params Auth Headers (8) Body Pre-req. Tests Settings

raw JSON Beautify

```
1 {
2   "firstName": "Elmer",
3   "lastName": "Fudd",
4   "level": "PLATINUM",
5   "address": {
6     "street": "123 Hunters Rd.",
7     "city": "Battle Creek",
8     "state": "Michigan",
9     "zip": "49015"
10  }
11 }
```

Body Cookies Headers (4) Test Results

Pretty Raw Preview Visualize JSON

```
1 {
2   "id": 2,
3   "firstName": "Elmer",
4   "lastName": "Fudd",
5   "address": {
6     "id": 2,
7     "street": "123 Hunters Rd.",
8     "city": "Battle Creek",
9     "state": "Michigan",
10    "zip": "49015"
11  },
12  "level": "PLATINUM",
13  "orders": null
14 }
```

(3) //3_Post-Customer-Wile

2_Week_Springboot / 3_Post-Customer-Wile

POST http://localhost:8080/customer

Params Auth Headers (8) Body Pre-req. Tests Settings

raw JSON Beautify

```
1 {
2   "firstName": "Wile E.",
3   "lastName": "Coyote",
4   "level": "SILVER",
5   "address": {
6     "street": "2020 Acme Lane",
7     "city": "Portage",
8     "state": "Michigan",
9     "zip": "49002"
10  }
11 }
```

Body Cookies Headers (4) Test Results

Pretty Raw Preview Visualize JSON

```
1 {
2   "id": 3,
3   "firstName": "Wile E.",
4   "lastName": "Coyote",
5   "address": {
6     "id": 3,
7     "street": "2020 Acme Lane",
8     "city": "Portage",
9     "state": "Michigan",
10    "zip": "49002"
11  },
12  "level": "SILVER",
13  "orders": null
14 }
```

(4) //4_Post-Customer-Tasmanian

The screenshot displays a REST client interface with a POST request to `http://localhost:8080/customer`. The request body is a JSON object representing a customer. The response body is a JSON object representing the created customer, including an assigned ID and a null orders field.

Request Body:

```
1  {
2    "firstName": "Tasmanian",
3    "lastName": "Devil",
4    "level": "SILVER",
5    "address": {
6      "street": "1010 Circle DR.",
7      "city": "Kalamazoo",
8      "state": "Michigan",
9      "zip": "49005"
10   }
11 }
```

Response Body:

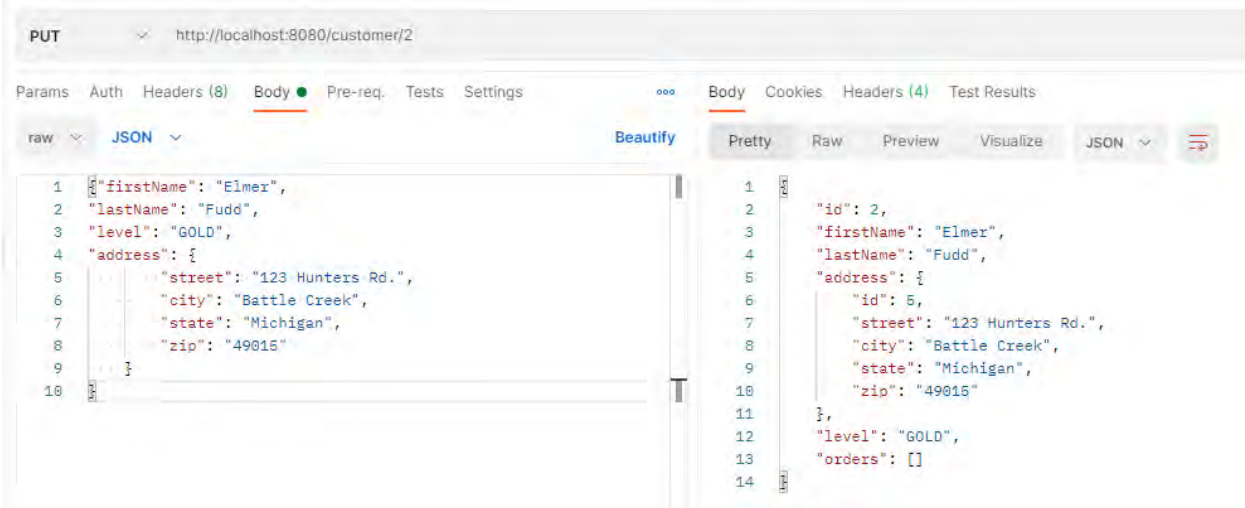
```
1  {
2    "id": 4,
3    "firstName": "Tasmanian",
4    "lastName": "Devil",
5    "address": {
6      "id": 4,
7      "street": "1010 Circle DR.",
8      "city": "Kalamazoo",
9      "state": "Michigan",
10     "zip": "49005"
11   },
12   "level": "SILVER",
13   "orders": null
14 }
```

(5) GET (body/raw/JSON) <http://localhost:8080/customer>

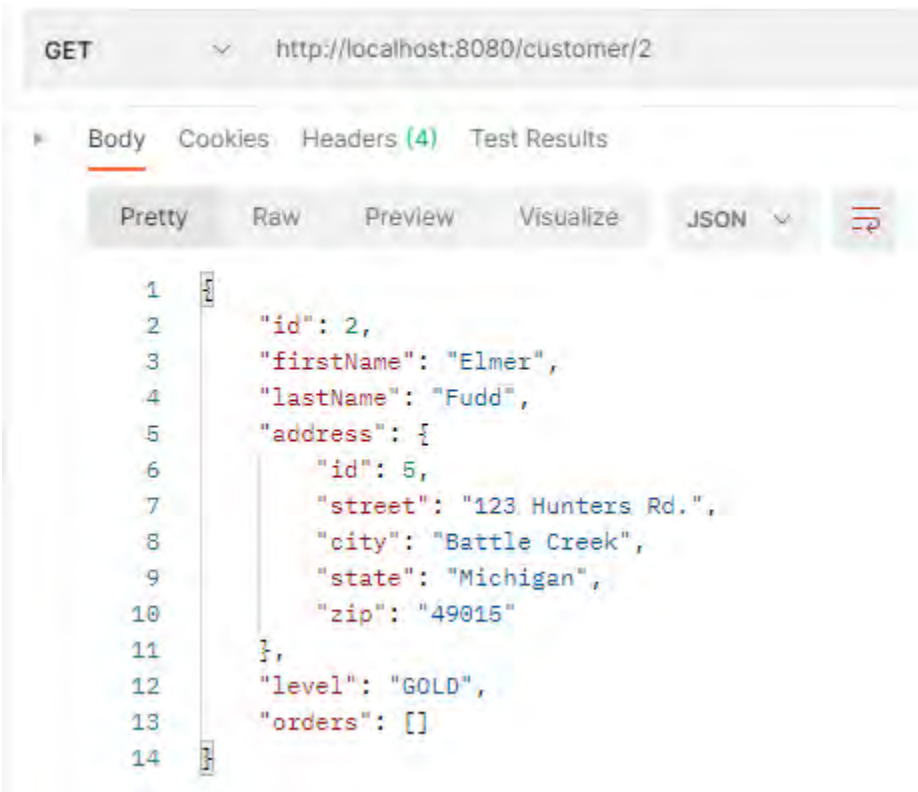


```
1  [
2    {
3      "id": 1,
4      "firstName": "Bugs",
5      "lastName": "Bunny",
6      "address": {
7        "id": 1,
8        "street": "1 wabbit hole lane",
9        "city": "Kalamazoo",
10       "state": "Michigan",
11       "zip": "49006"
12     },
13     "level": "PLATINUM",
14     "orders": []
15   },
16   {
17     "id": 2,
18     "firstName": "Elmer",
19     "lastName": "Fudd",
20     "address": {
21       "id": 2,
22       "street": "123 Hunters Rd.",
23       "city": "Battle Creek",
24       "state": "Michigan",
25       "zip": "49015"
26     },
27     "level": "PLATINUM",
28     "orders": []
29   },
30   {
31     "id": 3,
32     "firstName": "Wile E.",
33     "lastName": "Coyote",
34     "address": {
35       "id": 3,
36       "street": "2020 Acme Lane",
37       "city": "Portage",
38       "state": "Michigan",
39       "zip": "49002"
40     },
41     "level": "SILVER",
42     "orders": []
43   },
44   {
45     "id": 4,
46     "firstName": "Tasmanian",
47     "lastName": "Devil",
48     "address": {
49       "id": 4,
50       "street": "1010 Circle DR.",
51       "city": "Kalamazoo",
52       "state": "Michigan",
53       "zip": "49005"
54     },
55     "level": "SILVER",
56     "orders": []
57   }
58 ]
```

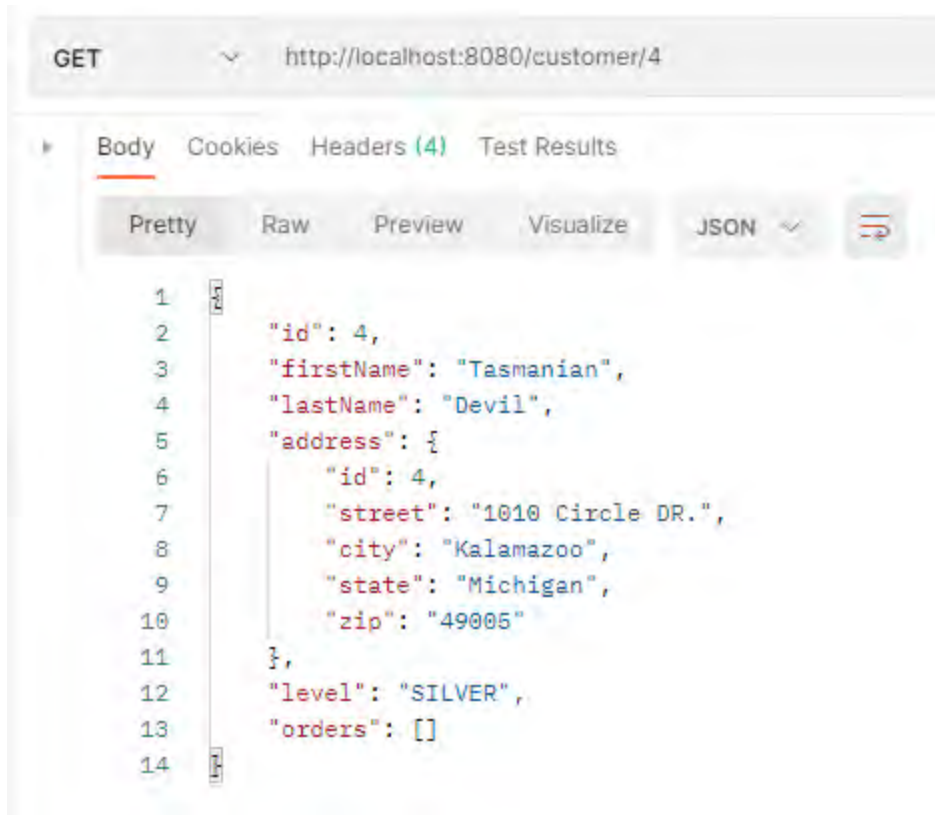
(6) PUT (body/raw/JSON) <http://localhost:8080/customer/2>



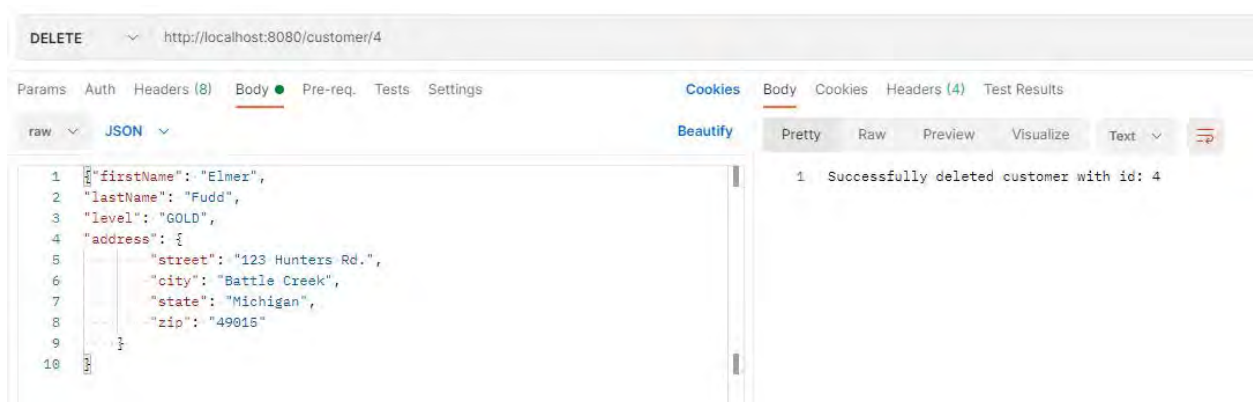
(7) GET (body/raw/JSON) <http://localhost:8080/customer/2>



(8) GET (body/raw/JSON) <http://localhost:8080/customer/4>



(9) DELETE (body/raw/JSON) <http://localhost:8080/customer/4>



(10) GET (body/raw/JSON) <http://localhost:8080/customer/>



GET <http://localhost:8080/customer/>

Body Cookies Headers (4) Test Results

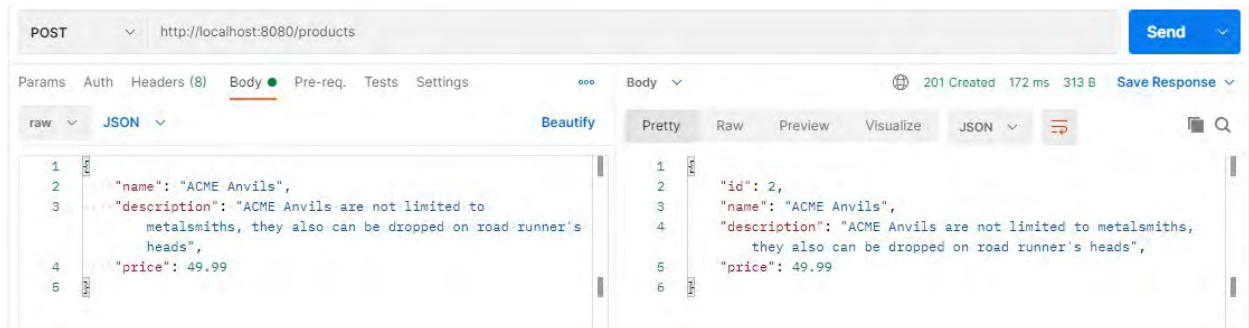
Pretty Raw Preview Visualize JSON

```
1  [
2    {
3      "id": 1,
4      "firstName": "Bugs",
5      "lastName": "Bunny",
6      "address": {
7        "id": 1,
8        "street": "1 wabbit hole lane",
9        "city": "Kalamazoo",
10       "state": "Michigan",
11       "zip": "49006"
12     },
13     "level": "PLATINUM",
14     "orders": []
15   },
16   {
17     "id": 2,
18     "firstName": "Elmer",
19     "lastName": "Fudd",
20     "address": {
21       "id": 5,
22       "street": "123 Hunters Rd.",
23       "city": "Battle Creek",
24       "state": "Michigan",
25       "zip": "49015"
26     },
27     "level": "GOLD",
28     "orders": []
29   },
30   {
31     "id": 3,
32     "firstName": "Wile E.",
33     "lastName": "Coyote",
34     "address": {
35       "id": 3,
36       "street": "2020 Acme Lane",
37       "city": "Portage",
38       "state": "Michigan",
39       "zip": "49002"
40     },
41     "level": "SILVER",
42     "orders": []
43   }
44 ]
```

(11) POST (body/raw/JSON) <http://localhost:8080/products>



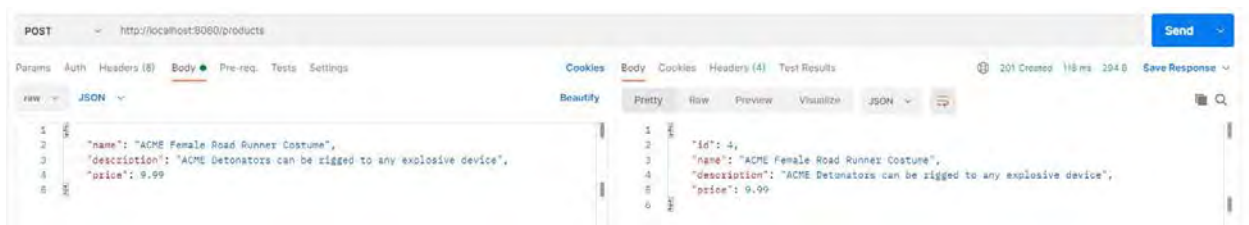
(12) // POST-Products-Anvils



(13) // POST-Products-Detonator



(14) // POST-Products-FemaleRoadRunnerCostume



(15) // POST-Products-BuckShot

POST http://localhost:8080/products

Params Auth Headers (8) Body Pre-req. Tests Settings

raw JSON Beautify

```
1 {
2   "name": "ACME Buck Shot",
3   "description": "ACME Buck Shot can be used in any
4     situation where weapons are involved and is so tiny
5     even spiders can use it to catch elusive insects.",
6   "price": 14.75
7 }
```

Body 201 Created 99 ms 360 B Save Response

Pretty Raw Preview Visualize JSON

```
1 {
2   "id": 5,
3   "name": "ACME Buck Shot",
4   "description": "ACME Buck Shot can be used in any situation
5     where weapons are involved and is so tiny even spiders
6     can use it to catch elusive insects.",
7   "price": 14.75
8 }
```

(16) // GET (body/raw/JSON) <http://localhost:8080/products>

GET http://localhost:8080/products

Body Cookies Headers (4) Test Results Status: 200 OK Time: 40 ms

Pretty Raw Preview Visualize JSON

```
1 [
2   {
3     "id": 1,
4     "name": "ACME Boomerang",
5     "description": "The ACME boomerangs are guaranteed to always return to Wile E. Coyote when he is not looking...",
6     "price": 24.99
7   },
8   {
9     "id": 2,
10    "name": "ACME Anvils",
11    "description": "ACME Anvils are not limited to metalsmiths, they also can be dropped on road runner's heads",
12    "price": 49.99
13  },
14  {
15    "id": 3,
16    "name": "ACME Detonator",
17    "description": "ACME Detonators can be rigged to any explosive device",
18    "price": 9.99
19  },
20  {
21    "id": 5,
22    "name": "ACME Buck Shot",
23    "description": "ACME Buck Shot can be used in any situation where weapons are involved and is so tiny even spiders can use it to catch elusive insects.",
24    "price": 14.75
25  },
26  {
27    "id": 6,
28    "name": "ACME Female Road Runner Costume",
29    "description": "The ACME Female Road Runner Costume is used by cross dressing animal lovers everywhere",
30    "price": 79.99
31  }
32 ]
```

(17) //PUT /products/1

PUT http://localhost:8080/products/1

Params Auth Headers (8) Body Pre-req. Tests Settings

raw JSON Beautify

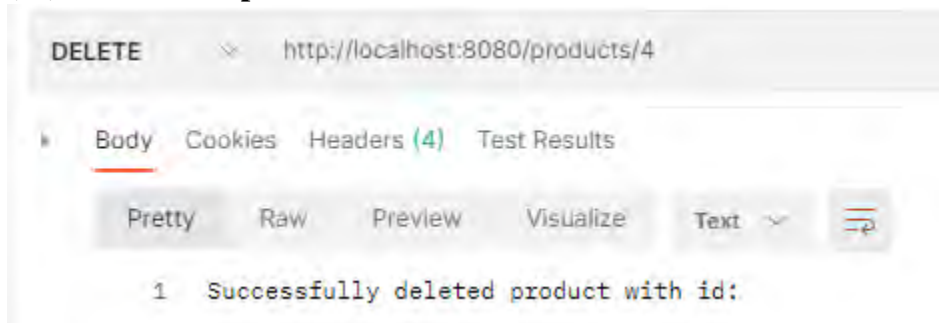
```
1 {
2   "name": "ACME Boomerang",
3   "description": "The ACME boomerangs are guaranteed to
4     always return to Wile E. Coyote when he is not
5     looking...",
6   "price": 24.99
7 }
```

Body 200 OK 95 ms 315 B Save Response

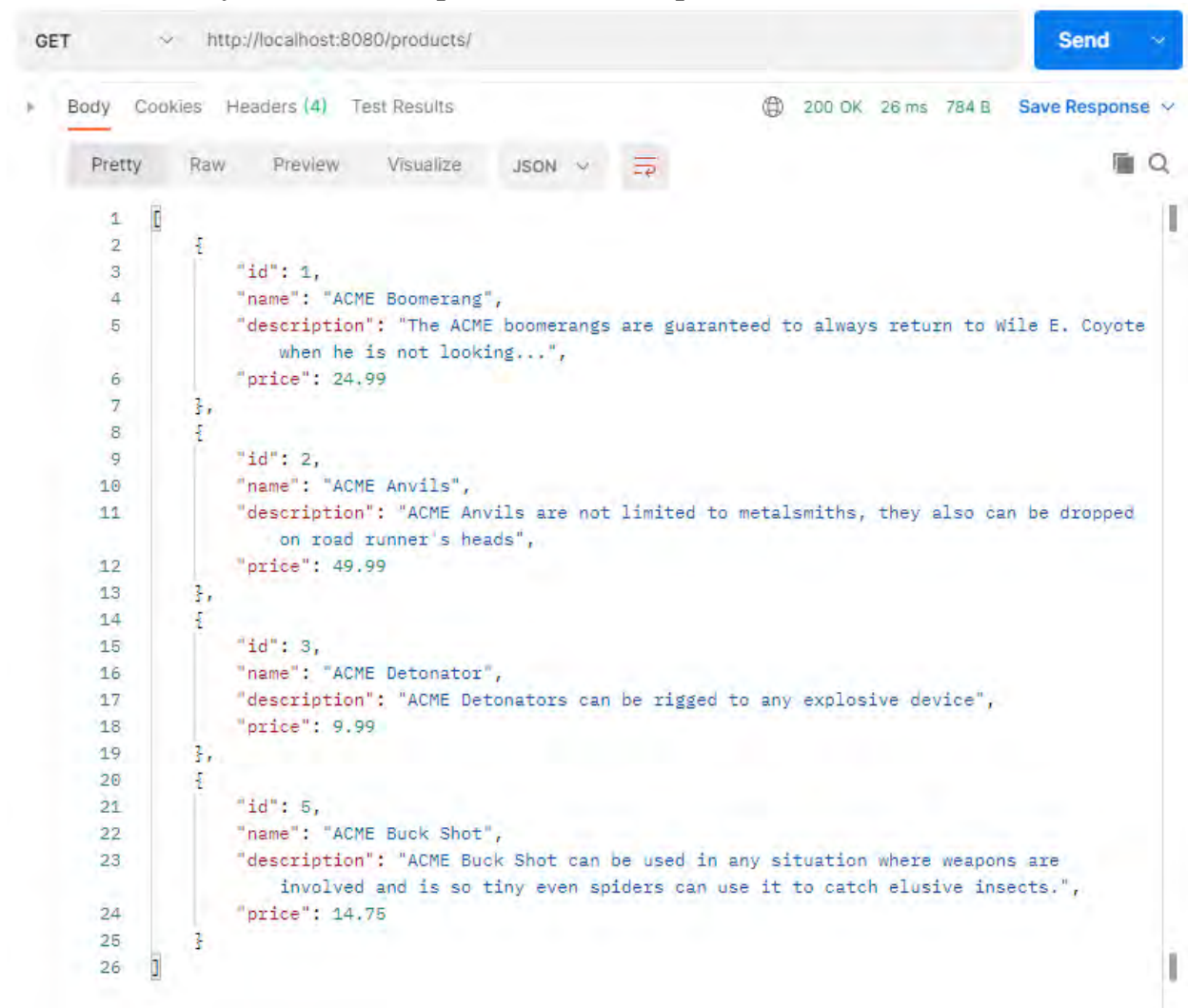
Pretty Raw Preview Visualize JSON

```
1 {
2   "id": 1,
3   "name": "ACME Boomerang",
4   "description": "The ACME boomerangs are guaranteed to
5     always return to Wile E. Coyote when he is not looking..
6   "price": 24.99
7 }
```

(18) // DELETE /products/4



(18) // GET (body/raw/JSON) http://localhost:8080/products



(19) // POST /customers/3/orders -- Wile E. Coyote(3) - Detonator (3) –

The screenshot displays a REST client interface with a POST request to `http://localhost:8080/customer/3/orders`. The request body is a JSON object with the following structure:

```
1  [
2    "id": 1,
3    "ordered": {
4      "year": 2021,
5      "month": "APRIL",
6      "era": "CE",
7      "dayOfMonth": 18,
8      "dayOfWeek": "SUNDAY",
9      "dayOfYear": 108,
10     "leapYear": false,
11     "monthValue": 4,
12     "chronology": {
13       "id": "ISO",
14       "calendarType": "iso8601"
15     }
16   },
17   "estimatedDelivery": {
18     "year": 2021,
19     "month": "APRIL",
20     "era": "CE",
21     "dayOfMonth": 25,
22     "dayOfWeek": "SUNDAY",
23     "dayOfYear": 115,
24     "leapYear": false,
25     "monthValue": 4,
26     "chronology": {
27       "id": "ISO",
28       "calendarType": "iso8601"
29     }
30   },
31   "delivered": null,
32   "invoiceAmount": 9.4905,
33   "status": "ORDERED",
34   "products": [
35     {
36       "id": 3,
37       "name": "ACME Detonator",
38       "description": "ACME Detonators can be rigged to any explosive device",
39       "price": 9.99
40     }
41   ]
42 ]
```

(20) POST /customer/2/orders – Elmer Fudd (2) – Anvils (5)

The screenshot displays a REST client interface with a POST request to `http://localhost:8080/customer/2/orders`. The request body is a JSON object. The response status is 201 Created.

Request Body:

```
1  {
2    "id": 3,
3    "ordered": {
4      "year": 2021,
5      "month": "APRIL",
6      "era": "CE",
7      "dayOfMonth": 18,
8      "dayOfWeek": "SUNDAY",
9      "dayOfYear": 108,
10     "leapYear": false,
11     "monthValue": 4,
12     "chronology": {
13       "id": "ISO",
14       "calendarType": "iso8601"
15     }
16   },
17   "estimatedDelivery": {
18     "year": 2021,
19     "month": "APRIL",
20     "era": "CE",
21     "dayOfMonth": 25,
22     "dayOfWeek": "SUNDAY",
23     "dayOfYear": 115,
24     "leapYear": false,
25     "monthValue": 4,
26     "chronology": {
27       "id": "ISO",
28       "calendarType": "iso8601"
29     }
30   },
31   "delivered": null,
32   "invoiceAmount": 58.266000000000005,
33   "status": "ORDERED",
34   "products": [
35     {
36       "id": 2,
37       "name": "ACME Anvils",
38       "description": "ACME Anvils are not limited to metalsmiths, they also can be
39         dropped on road runner's heads",
40       "price": 49.99
41     },
42     {
43       "id": 5,
44       "name": "ACME Buck Shot",
45       "description": "ACME Buck Shot can be used in any situation where weapons are
46         involved and is so tiny even soiders can use it to catch elusive insects.",
47       "price": 14.75
48     }
49   ]
50 }
```

Response: 201 Created, 335 ms, 989 B. Save Response.

(21) // PUT /customer/2/orders/1_CANCELED

The screenshot displays a REST client interface with a PUT request to `http://localhost:8080/customer/2/orders/1`. The request body is a JSON object with the property `"status": "CANCELED"`. The response body is a detailed JSON object representing an order.

Request Body:

```
{
  "status": "CANCELED"
}
```

Response Body:

```
{
  "id": 1,
  "ordered": {
    "year": 2021,
    "month": "APRIL",
    "era": "CE",
    "dayOfMonth": 18,
    "dayOfWeek": "SUNDAY",
    "dayOfYear": 108,
    "leapYear": false,
    "monthValue": 4,
    "chronology": {
      "id": "ISO",
      "calendarType": "iso8601"
    }
  },
  "estimatedDelivery": {
    "year": 2021,
    "month": "APRIL",
    "era": "CE",
    "dayOfMonth": 25,
    "dayOfWeek": "SUNDAY",
    "dayOfYear": 115,
    "leapYear": false,
    "monthValue": 4,
    "chronology": {
      "id": "ISO",
      "calendarType": "iso8601"
    }
  },
  "delivered": null,
  "invoiceAmount": 9.4905,
  "status": "CANCELED",
  "products": [
    {
      "id": 3,
      "name": "ACME Detonator",
      "description": "ACME Detonators can be rigged to any explosive device",
      "price": 9.99
    }
  ]
}
```

(22) // PUT /customers/2/orders/5_DELIVERED

The screenshot displays a REST client interface with two panels. The top panel shows a PUT request to `http://localhost:8080/customer/2/orders/2` with a JSON body: `{ "status": "DELIVERED" }`. The bottom panel shows the response body, which is a JSON object containing order details, delivery estimates, and a list of products.

```
PUT http://localhost:8080/customer/2/orders/2
{
  "status": "DELIVERED"
}
```

```
PUT http://localhost:8080/customer/2/orders/2
{
  "id": 2,
  "ordered": {
    "year": 2021,
    "month": "APRIL",
    "era": "CE",
    "dayOfMonth": 18,
    "dayOfWeek": "SUNDAY",
    "dayOfYear": 108,
    "leapYear": false,
    "monthValue": 4,
    "chronology": {
      "id": "ISO",
      "calendarType": "iso8601"
    }
  },
  "estimatedDelivery": {
    "year": 2021,
    "month": "APRIL",
    "era": "CE",
    "dayOfMonth": 26,
    "dayOfWeek": "SUNDAY",
    "dayOfYear": 116,
    "leapYear": false,
    "monthValue": 4,
    "chronology": {
      "id": "ISO",
      "calendarType": "iso8601"
    }
  },
  "delivered": null,
  "invoiceAmount": 58.266000000000005,
  "status": "DELIVERED",
  "products": [
    {
      "id": 5,
      "name": "ACME Buck Shot",
      "description": "ACME Buck Shot can be used in any situation where weapons are involved and is so tiny even spiders can use it to catch elusive insects.",
      "price": 14.75
    },
    {
      "id": 2,
      "name": "ACME Anvils",
      "description": "ACME Anvils are not limited to metalsmiths, they also can be dropped on road runner's heads",
      "price": 49.99
    }
  ]
}
```

URL to GitHub Repository: https://github.com/becje/Springboot_Coding_Assignments