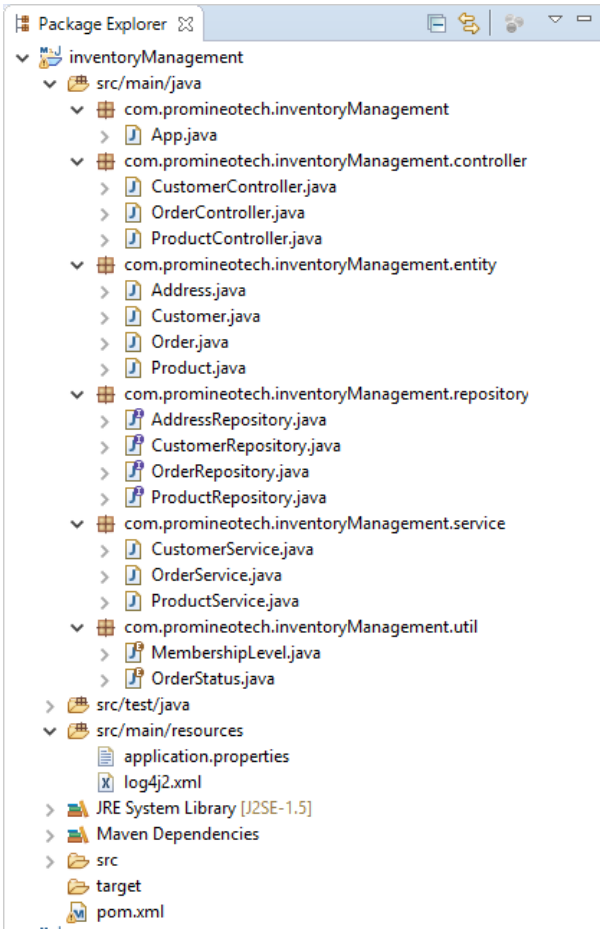
Inventory Management API Tutorial

Create a new maven project in Eclipse. If you don't remember how to do that, refer to last week's tutorial. In this tutorial, we will be building an API that allows users to create and manage customers, products, and orders. We will be introduced to a few new annotations as well as logging. We will take the approach of working from the data layer (the entities and repositories), towards the endpoints. To do this, we will create the different layers of our application in the following order:

1. Entities
2. Repositories
3. Services
4. Controllers

Here is what the folder structure will look like when complete:



Start by creating all the packages, classes, and interfaces (the repositories) we see. The MembershipLevel.java and OrderStatus.java files are **Enums**, which we'll talk more about later. For now, just make sure we select Enum from the list when we create them. Once we have everything created, we can start writing our code. Let's start by making the pom.xml and entities look like the following:

```xml
<project xmlns="http://maven.apache.org/POM/4.0.0"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>

    <groupId>com.promineotech</groupId>
    <artifactId>inventoryManagement</artifactId>
    <version>0.0.1-SNAPSHOT</version>
    <packaging>jar</packaging>

    <name>inventoryManagement</name>
    <url>http://maven.apache.org</url>

    <parent>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-parent</artifactId>
        <version>1.2.6.RELEASE</version>
    </parent>

    <properties>
        <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
    </properties>

    <dependencies>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-web</artifactId>
            <exclusions>
                <exclusion>
                    <groupId>org.springframework.boot</groupId>
                    <artifactId>spring-boot-starter-logging</artifactId>
                </exclusion>
            </exclusions>
        </dependency>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-log4j2</artifactId>
        </dependency>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-test</artifactId>
            <scope>test</scope>
        </dependency>
        <dependency>
            <groupId>mysql</groupId>
            <artifactId>mysql-connector-java</artifactId>
            <scope>runtime</scope>
        </dependency>
        <dependency>
            <groupId>com.jayway.jsonpath</groupId>
            <artifactId>json-path</artifactId>
            <scope>test</scope>
        </dependency>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-data-jpa</artifactId>
        </dependency>
        <dependency>
            <groupId>org.springframework.security</groupId>
            <artifactId>spring-security-crypto</artifactId>
            <version>5.1.2.RELEASE</version>
        </dependency>
        <dependency>
            <groupId>junit</groupId>
            <artifactId>junit</artifactId>
            <version>3.8.1</version>
            <scope>test</scope>
        </dependency>
    </dependencies>
</project>
```

```java
package com.promineotech.inventoryManagement.entity;

import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.OneToOne;

import com.fasterxml.jackson.annotation.JsonIgnore;

@Entity
public class Address {

    private Long id;
    private String street;
    private String city;
    private String state;
    private String zip;

    @JsonIgnore
    private Customer customer;

    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    public Long getId() {
        return id;
    }

    public void setId(Long id) {
        this.id = id;
    }

    public String getStreet() {
        return street;
    }

    public void setStreet(String street) {
        this.street = street;
    }

    public String getCity() {
        return city;
    }

    public void setCity(String city) {
        this.city = city;
    }

    public String getState() {
        return state;
    }

    public void setState(String state) {
        this.state = state;
    }

    public String getZip() {
        return zip;
    }

    public void setZip(String zip) {
        this.zip = zip;
    }

    @OneToOne(mappedBy = "address")
    public Customer getCustomer() {
        return customer;
    }

    public void setCustomer(Customer customer) {
        this.customer = customer;
    }

}
```

```java
package com.promineotech.inventoryManagement.entity;

import java.util.Set;

import javax.persistence.CascadeType;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.JoinColumn;
import javax.persistence.OneToMany;
import javax.persistence.OneToOne;

import com.promineotech.inventoryManagement.util.MembershipLevel;

@Entity
public class Customer {

    private Long id;
    private String firstName;
    private String lastName;
    private Address address;
    private MembershipLevel level;
    private Set<Order> orders;

    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    public Long getId() {
        return id;
    }

    public void setId(Long id) {
        this.id = id;
    }

    public String getFirstName() {
        return firstName;
    }

    public void setFirstName(String firstName) {
        this.firstName = firstName;
    }

    public String getLastName() {
        return lastName;
    }

    public void setLastName(String lastName) {
        this.lastName = lastName;
    }

    @OneToOne(cascade = CascadeType.ALL)
    @JoinColumn(name = "id")
    public Address getAddress() {
        return address;
    }

    public void setAddress(Address address) {
        this.address = address;
    }

    public MembershipLevel getLevel() {
        return level;
    }

    public void setLevel(MembershipLevel level) {
        this.level = level;
    }

    @OneToMany(mappedBy = "customer")
    public Set<Order> getOrders() {
        return orders;
    }

    public void setOrders(Set<Order> orders) {
        this.orders = orders;
    }

}
```

```java
package com.promineotech.inventoryManagement.entity;

import java.time.LocalDate;
import java.util.Set;

import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.JoinColumn;
```

```java
import javax.persistence.ManyToMany;
import javax.persistence.ManyToOne;

import com.fasterxml.jackson.annotation.JsonIgnore;
import com.promineotech.inventoryManagement.util.OrderStatus;

@Entity
public class Order {

    private Long id;
    private LocalDate ordered;
    private LocalDate estimatedDelivery;
    private LocalDate delivered;
    private double invoiceAmount;
    private OrderStatus status;
    private Set<Product> products;

    @JsonIgnore
    private Customer customer;

    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    public Long getId() {
        return id;
    }

    public void setId(Long id) {
        this.id = id;
    }

    public LocalDate getOrdered() {
        return ordered;
    }

    public void setOrdered(LocalDate ordered) {
        this.ordered = ordered;
    }

    public LocalDate getEstimatedDelivery() {
        return estimatedDelivery;
    }

    public void setEstimatedDelivery(LocalDate estimatedDelivery) {
        this.estimatedDelivery = estimatedDelivery;
    }

    public LocalDate getDelivered() {
        return delivered;
    }

    public void setDelivered(LocalDate delivered) {
        this.delivered = delivered;
    }

    public double getInvoiceAmount() {
        return invoiceAmount;
    }

    public void setInvoiceAmount(double invoiceAmount) {
        this.invoiceAmount = invoiceAmount;
    }

    @ManyToMany(mappedBy = "orders")
    public Set<Product> getProducts() {
        return products;
    }

    public void setProducts(Set<Product> products) {
        this.products = products;
    }

    @ManyToOne
    @JoinColumn(name = "customerId")
    public Customer getCustomer() {
        return customer;
    }

    public void setCustomer(Customer customer) {
        this.customer = customer;
    }

    public OrderStatus getStatus() {
        return status;
    }

    public void setStatus(OrderStatus status) {
        this.status = status;
    }

}
```

```java
package com.promineotech.inventoryManagement.entity;

import java.util.Set;

import javax.persistence.CascadeType;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.JoinColumn;
import javax.persistence.JoinTable;
import javax.persistence.ManyToMany;

import com.fasterxml.jackson.annotation.JsonIgnore;

@Entity
public class Product {

    private Long id;
    private String name;
    private String description;
    private double price;

    @JsonIgnore
    private Set<Order> orders;

    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    public Long getId() {
        return id;
    }

    public void setId(Long id) {
        this.id = id;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public String getDescription() {
        return description;
    }

    public void setDescription(String description) {
        this.description = description;
    }

    public double getPrice() {
        return price;
    }

    public void setPrice(double price) {
        this.price = price;
    }

    @ManyToMany(cascade = CascadeType.ALL)
    @JoinTable(name = "product_order",
        joinColumns = @JoinColumn(name = "orderId", referencedColumnName = "id"),
        inverseJoinColumns = @JoinColumn(name = "productId", referencedColumnName = "id"))
    public Set<Order> getOrders() {
        return orders;
    }

    public void setOrders(Set<Order> orders) {
        this.orders = orders;
    }

}

```

You will also need to create the enums that these entities use:

```java
package com.promineotech.inventoryManagement.util;

public enum MembershipLevel {

    SILVER(.05),
    GOLD(.10),
    DIAMOND(.15),
    PLATINUM(.20);

    private double discount;

    MembershipLevel(double discount) {
        this.discount = discount;
    }

    public double getDiscount() {
        return discount;
    }

}
```

```java
package com.promineotech.inventoryManagement.util;

public enum OrderStatus {

    ORDERED,
    DELIVERED,
    CANCELED;

}
```

Enums allow us to create constant values that can be used to ensure other values not defined are excluded from possible values. For example, an enum called DaysOfWeek may have the values SUNDAY, MONDAY, TUESDAY, WEDNESDAY, THURSDAY, FRIDAY, and SATURDAY, and nothing more, which restricts any variable of type DaysOfWeek from using any other values. We have an OrderStatus enum, that tells the status of an order in our system (ORDERED, DELIVERED, or CANCELLED), and we have a MembershipLevel enum that contains values related to the status of a customer in our system. Enums can have fields, as shown in MembershipLevel, where we have a double that represents the discount percent customers get at each level. We will use this to calculate discounts on order totals later on in our service layer.

In the entities, we also see some new annotations such as **@OneToOne** and **@ManyToMany**. These JPA relationship annotations are a little more complex than the previous annotations we've used. In one class, you need to identify the columns to join on and which class member (field) they relate to, and on the other class you only need to identify which field in the related class maps to the field you are annotating. If this seems a little complicated, don't worry, with practice and more learning it will become easier.

Now that we have the entities completed, let's continue moving towards the endpoints. The next step is to create the repositories. In each of your repository interfaces, extend CrudRepository using the entity name for the first generic parameter and Long for the second. For example, the AddressRepository would look like this:

public interface AddressRepository extends CrudRepository<Address, Long> {}

With the repositories completed, we can now move on to the service layer. Remember, the service layer is where all the heavy lifting is done; it is where we put all the business logic. Business logic is the rules of what our application does. For example, rules like applying discounts based on MembershipLevel, or adding up the total cost of an order. Any real logic should live in this layer. Let's make our service classes look like the following.

```java
package com.promineotech.inventoryManagement.service;

import org.apache.logging.log4j.LogManager;
import org.apache.logging.log4j.Logger;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

import com.promineotech.inventoryManagement.entity.Product;
import com.promineotech.inventoryManagement.repository.ProductRepository;

@Service
public class ProductService {

    private static final Logger logger = LogManager.getLogger(ProductService.class);

    @Autowired
    private ProductRepository repo;

    public Iterable<Product> getProducts() {
        return repo.findAll();
    }

    public Product createProduct(Product product) {
        return repo.save(product);
    }

    public Product updateProduct(Product product, Long id) throws Exception {
        try {
            Product oldProduct = repo.findOne(id);
            oldProduct.setDescription(product.getDescription());
            oldProduct.setName(product.getName());
            oldProduct.setPrice(product.getPrice());
            return repo.save(oldProduct);
        } catch (Exception e) {
            logger.error("Exception occurred while trying to update product: " + id, e);
            throw new Exception("Unable to update product.");
        }
    }

    public void removeProduct(Long id) throws Exception {
        try {
            repo.delete(id);
        } catch (Exception e) {
            logger.error("Exception occurred while trying to delete product: " + id, e);
            throw new Exception("Unable to delete product.");
        }
    }

}
```

The ProductService is the simplest of the three service classes in this API. We can create, update, and remove products from our database. Something new that we notice in this class is the use of the logger. We will create a file that sets all the configurations for our logger later, but for now, let's look at how we can use loggers. Sometimes there is information that is useful to log and can help troubleshoot defects (bugs/errors) that may occur. We can log out this information to a file so that we can review it and help fix defects if/when they occur. That is why in our example the logging occurs in our catch blocks, but we can log out any information we feel is useful no matter where it is.

```java
package com.promineotech.inventoryManagement.service;

import org.apache.logging.log4j.LogManager;
import org.apache.logging.log4j.Logger;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

import com.promineotech.inventoryManagement.entity.Customer;
import com.promineotech.inventoryManagement.repository.CustomerRepository;

@Service
public class CustomerService {

    private static final Logger logger = LogManager.getLogger(CustomerService.class);

    @Autowired
    private CustomerRepository repo;

    public Customer getCustomerById(Long id) throws Exception {
        try {
            return repo.findOne(id);
        } catch (Exception e) {
            logger.error("Exception occurred while trying to retrieve customer: " + id, e);
            throw e;
        }
    }

    public Iterable<Customer> getCustomers() {
        return repo.findAll();
    }

    public Customer createCustomer(Customer customer) {
        return repo.save(customer);
    }

    public Customer updateCustomer(Customer customer, Long id) throws Exception {
        try {
            Customer oldCustomer = repo.findOne(id);
            oldCustomer.setAddress(customer.getAddress());
            oldCustomer.setFirstName(customer.getFirstName());
            oldCustomer.setLastName(customer.getLastName());
            oldCustomer.setLevel(customer.getLevel());
            return repo.save(oldCustomer);
        } catch (Exception e) {
            logger.error("Exception occurred while trying to update customer: " + id, e);
            throw new Exception("Unable to update customer.");
        }
    }

    public void deleteCustomer(Long id) throws Exception {
        try {
            repo.delete(id);
        } catch (Exception e) {
            logger.error("Exception occurred while trying to delete customer: " + id, e);
            throw new Exception("Unable to delete customer.");
        }
    }

}
```

The CustomerService is pretty straight forward, without any new concepts introduced.

```java
package com.promineotech.inventoryManagement.service;

import java.time.LocalDate;
import java.util.HashSet;
import java.util.Set;

import org.apache.logging.log4j.LogManager;
import org.apache.logging.log4j.Logger;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

import com.promineotech.inventoryManagement.entity.Customer;
import com.promineotech.inventoryManagement.entity.Order;
import com.promineotech.inventoryManagement.entity.Product;
import com.promineotech.inventoryManagement.repository.CustomerRepository;
import com.promineotech.inventoryManagement.repository.OrderRepository;
import com.promineotech.inventoryManagement.repository.ProductRepository;
import com.promineotech.inventoryManagement.util.MembershipLevel;
import com.promineotech.inventoryManagement.util.OrderStatus;

@Service
public class OrderService {

    private static final Logger logger = LogManager.getLogger(OrderService.class);
    private final int DELIVERY_DAYS = 7;

    @Autowired
    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
48      public Order cancelOrder(Long orderId) throws Exception {
49          try {
50              Order order = repo.findOne(orderId);
51              order.setStatus(OrderStatus.CANCELED);
52              return repo.save(order);
53          } catch (Exception e) {
54              logger.error("Exception occurred while trying to cancel order: " + orderId, e);
55              throw new Exception("Unable to update order.");
56          }
57      }
58
59      public Order completeOrder(Long orderId) throws Exception {
60          try {
61              Order order = repo.findOne(orderId);
62              order.setStatus(OrderStatus.DELIVERED);
63              return repo.save(order);
64          } catch (Exception e) {
65              logger.error("Exception occurred while trying to complete order: " + orderId, e);
66              throw new Exception("Unable to update order.");
67          }
68      }
69
70      private Order initializeNewOrder(Set<Long> productIds, Customer customer) {
71          Order order = new Order();
72          order.setProducts(convertToProductSet(productRepo.findAll(productIds)));
73          order.setOrdered(LocalDate.now());
74          order.setEstimatedDelivery(LocalDate.now().plusDays(DELIVERY_DAYS));
75          order.setCustomer(customer);
76          order.setInvoiceAmount(calculateOrderTotal(order.getProducts(), customer.getLevel()));
77          order.setStatus(OrderStatus.ORDERED);
78          addOrderToProducts(order);
79          return order;
80      }
81
82      private void addOrderToProducts(Order order) {
83          Set<Product> products = order.getProducts();
84          for (Product product : products) {
85              product.getOrders().add(order);
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 }
```

This OrderService is the big boy. Let's break down some of the items in this class to get a better idea of what's going on. We have a few public methods, which are accessible to be called from other files. Remember, our public methods are what we want other classes to use to interact with this class; it's like saying "here is the functionality that I provide to you as a class". We then have some private methods that are used internally. The public methods show what this class is in charge of doing, submitting a new order, canceling an order, and completing an order. The submitNewOrder method calls the private method, **initializeNewOrder**, which is where the bulk of the work in this class is done.

In **initializeNewOrder**, we create a new order and set all the fields that can be set with default values, such as the dates, the customer we pulled from the database that this order is being created for, and the initial status of the new order. We then call the **convertToProductSet** method and pass all the products retrieved from the databases into it so that we can convert the Iterable<Product> to a Set<Product> so it can be set on the Order object. Now that

we have all the products pulled from the database and added to the order, we calculate the invoice amount which gets the price from each product, adds them all together, and then subtracts the discount derived from the MembershipLevel of the customer placing the order, as seen in the **calculateOrderTotal** method.

The last item of business we need to take care of is, even though we've added all the products to the order, we need to add the order to all the products so that the join table will be populated when we save the order. We do that in the **addOrderToProducts** method, which iterates over the products we added to the order, and adds the order to each of those products.

With all this work finished, we are finally ready to move to the controller layer and create our endpoints.

```
package com.promineotech.inventoryManagement.controller;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.bind.annotation.RestController;

import com.promineotech.inventoryManagement.entity.Customer;
import com.promineotech.inventoryManagement.service.CustomerService;

@RestController
@RequestMapping("/customers")
public class CustomerController {

    @Autowired
    private CustomerService service;

    @RequestMapping(value="/{id}", method=RequestMethod.GET)
    public ResponseEntity<Object> getCustomer(@PathVariable Long id) {
        try {
            return new ResponseEntity<Object>(service.getCustomerById(id), HttpStatus.OK);
        } catch (Exception e) {
            return new ResponseEntity<Object>(e.getMessage(), HttpStatus.NOT_FOUND);
        }
    }

    @RequestMapping(method=RequestMethod.GET)
    public ResponseEntity<Object> getCustomers() {
        return new ResponseEntity<Object>(service.getCustomers(), HttpStatus.OK);
    }

    @RequestMapping(method=RequestMethod.POST)
    public ResponseEntity<Object> createCustomer(@RequestBody Customer customer) {
        return new ResponseEntity<Object>(service.createCustomer(customer), HttpStatus.CREATED);
    }

    @RequestMapping(value="/{id}", method=RequestMethod.PUT)
    public ResponseEntity<Object> updateCustomer(@RequestBody Customer customer, @PathVariable Long id) {
        try {
            return new ResponseEntity<Object>(service.updateCustomer(customer, id), HttpStatus.OK);
        } catch (Exception e) {
            return new ResponseEntity<Object>(e.getMessage(), HttpStatus.NOT_FOUND);
        }
    }

    @RequestMapping(value="/{id}", method=RequestMethod.DELETE)
    public ResponseEntity<Object> deleteCustomer(@PathVariable Long id) {
        try {
            service.deleteCustomer(id);
            return new ResponseEntity<Object>("Successfully deleted customer with id: " + id, HttpStatus.OK);
        } catch (Exception e) {
            return new ResponseEntity<Object>(e.getMessage(), HttpStatus.NOT_FOUND);
        }
    }
}
```

```java
package com.promineotech.inventoryManagement.controller;

import java.util.Set;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.bind.annotation.RestController;

import com.promineotech.inventoryManagement.entity.Order;
import com.promineotech.inventoryManagement.service.OrderService;
import com.promineotech.inventoryManagement.util.OrderStatus;

@RestController
@RequestMapping("customers/{id}/orders")
public class OrderController {

    @Autowired
    private OrderService service;

    @RequestMapping(method=RequestMethod.POST)
    public ResponseEntity<Object> createCustomer(@RequestBody Set<Long> productIds, @PathVariable Long id) {
        try {
            return new ResponseEntity<Object>(service.submitNewOrder(productIds, id), HttpStatus.CREATED);
        } catch (Exception e) {
            return new ResponseEntity<Object>(e, HttpStatus.BAD_REQUEST);
        }
    }

    @RequestMapping(value="/{orderId}", method=RequestMethod.PUT)
    public ResponseEntity<Object> updateOrder(@RequestBody Order order, @PathVariable Long orderId) {
        try {
            if (order.getStatus().equals(OrderStatus.CANCELED)) {
                return new ResponseEntity<Object>(service.cancelOrder(orderId), HttpStatus.OK);
            } else if (order.getStatus().equals(OrderStatus.DELIVERED)) {
                return new ResponseEntity<Object>(service.completeOrder(orderId), HttpStatus.OK);
            }
            return new ResponseEntity<Object>("Invalid update request.", HttpStatus.BAD_REQUEST);
        } catch (Exception e) {
            return new ResponseEntity<Object>(e.getMessage(), HttpStatus.NOT_FOUND);
        }
    }

}
```

```java
package com.promineotech.inventoryManagement.controller;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.bind.annotation.RestController;

import com.promineotech.inventoryManagement.entity.Product;
import com.promineotech.inventoryManagement.service.ProductService;

@RestController
@RequestMapping("/products")
public class ProductController {

    @Autowired
    private ProductService service;

    @RequestMapping(method=RequestMethod.GET)
    public ResponseEntity<Object> getProducts() {
        return new ResponseEntity<Object>(service.getProducts(), HttpStatus.OK);
    }

    @RequestMapping(method=RequestMethod.POST)
    public ResponseEntity<Object> createProduct(@RequestBody Product product) {
        return new ResponseEntity<Object>(service.createProduct(product), HttpStatus.CREATED);
    }

    @RequestMapping(value="/{id}", method=RequestMethod.PUT)
    public ResponseEntity<Object> updateProduct(@RequestBody Product product, @PathVariable Long id) {
        try {
            return new ResponseEntity<Object>(service.updateProduct(product, id), HttpStatus.OK);
        } catch (Exception e) {
            return new ResponseEntity<Object>("Unable to update product.", HttpStatus.BAD_REQUEST);
        }
    }

    @RequestMapping(value="/{id}", method=RequestMethod.DELETE)
    public ResponseEntity<Object> deleteProduct(@PathVariable Long id) {
        try {
            service.removeProduct(id);
            return new ResponseEntity<Object>("Successfully deleted product with id: " + id, HttpStatus.OK);
        } catch (Exception e) {
            return new ResponseEntity<Object>("Unable to delete product.", HttpStatus.BAD_REQUEST);
        }
    }

}
```

The last class we have to add is the App class, which is the same as the last API we built, just need to tell Spring Boot where to start up and where to look for all the components to inject.

```java
package com.promineotech.inventoryManagement;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.context.annotation.ComponentScan;

@ComponentScan("com.promineotech.inventoryManagement")
@SpringBootApplication
public class App
{
    public static void main( String[] args )
    {
        SpringApplication.run(App.class, args);
    }
}
```

Here is what our application.properties file needs to look like (make sure we put the credentials for our database server, and make sure you create the database in MySQL so it exists).

```
spring.datasource.url=jdbc:mysql://localhost/inventory_api
spring.datasource.username=root
spring.datasource.password=root
spring.datasource.driver-class-name=com.mysql.jdbc.Driver

spring.jpa.hibernate.ddl-auto=create
spring.jpa.show-sql=true
spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.MySQL5Dialect
spring.jpa.properties.hibernate.globally_quoted_identifiers=true
```

Lastly, let's add our logger configuration.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<Configuration status="WARN" monitorInterval="30">
    <Properties>
        <Property name="LOG_PATTERN">
            %d{yyyy-MM-dd HH:mm:ss.SSS} %5p ${hostName}
            --- [%15.15t] %-40.40c{1.} : %m%n%ex
        </Property>
    </Properties>
    <Appenders>
        <Console name="ConsoleAppender" target="SYSTEM_OUT"
            follow="true">
            <PatternLayout pattern="${LOG_PATTERN}" />
        </Console>
        <RollingFile name="FileAppender"
            fileName="logs/log4j2-example.log"
            filePattern="logs/log4j2-example-%d{yyyy-MM-dd}-%i.log">
            <PatternLayout>
                <Pattern>${LOG_PATTERN}</Pattern>
            </PatternLayout>
            <Policies>
                <SizeBasedTriggeringPolicy size="10MB" />
            </Policies>
            <DefaultRolloverStategy max="10" />
        </RollingFile>
    </Appenders>
    <Loggers>
        <Logger name="com.example.log4j2example" level="debug"
            additivity="false">
            <AppenderRef ref="ConsoleAppender" />
        </Logger>
        <Root level="info">
            <AppenderRef ref="ConsoleAppender" />
            <AppenderRef ref="FileAppender" />
        </Root>
    </Loggers>
</Configuration>
```

At this point, we can run our application by running **mvn spring-boot:run** in the project directory. Make sure to test the endpoints using postman to ensure everything is working properly. If we run into any errors, we should see the logs created in a log folder. We may need to refresh our project in Eclipse by right-clicking it and selecting refresh to see the log folder and files appear. Good luck!

Last modified: Thursday, 18 April 2019, 5:26 PM

◄ Week 2 Coding Assignment

Jump to...

LinkedIn Posts Week 2 ►

NAVIGATION

Dashboard
- Site home
- Site pages
- My courses
  - Java
  - MySQL
  - spring boot
    - Participants
    - Badges
    - Competencies
    - Grades
    - General
    - Topic 1
    - Topic 2
      - Videos
      - Week 2 Research Discussion
      - Week 2 Coding Assignment
      - **Inventory Management API Tutorial**

- LinkedIn Posts Week 2
- Topic 3
- Topic 4
- Topic 5
- Topic 6
- career services

You are logged in as Renee Dubuc (Log out)

English (en)
አማርኛ (am)
Afaan Oromoo (om)
Afrikaans (af)
Aragonés (an)
Aranés (oc_es)
Asturianu (ast)
Azərbaycanca (az)
Bahasa Melayu (ms)
Bairisch (bar)
Bamanankan (bm)
Bislama (bi)
Bosanski (bs)
Breizh (br)
Català (ca)
Català (Valencià) (ca_valencia)
Català per a Workplace (ca_wp)
Čeština (cs)
Crnogorski (mis)
Cymraeg (cy)
Dansk (da)
Dansk (kursus) (da_kursus)
Dansk Rum (da_rum)
Davvisámegiella (se)
Deutsch - Du (de_du)
Deutsch - Kids (de_kids)
Deutsch - Schweiz (de_ch)
Deutsch (de_wp)
Deutsch (de)
Deutsch community (de_comm)
Dolnoserbski (dsb)
Ebon (mh)
eesti (et)
English - Pirate (en_ar)

English - United States (en_us)
English (en)
English for kids (en_kids)
English for Workplace - United States (en_us_wp)
English for Workplace (en_wp)
English US - K12 (en_us_k12)
Español - Colombia (es_co)
Español - Internacional (es)
Español - México (es_mx)
Español - México para niños (es_mx_kids)
Español - Venezuela (es_ve)
Español para la Empresa (es_wp)
Esperanto (eo)
Euskara (eu)
Ɛʋɛgbe (ee)
Filipino (fil)
Finlandssvenska (sv_fi)
Føroyskt (fo)
Français - Canada (fr_ca)
Français (écriture inclusive) (fr_incl)
Français (fr)

Français (fr)
Français pour Workplace (fr_wp)
Gaeilge (ga)
Gàidhlig (gd)
Galego (gl)
Gascon (oc_gsc)
Hausa (ha)
Hrvatski (hr_schools)
Hrvatski (hr)
ʻŌlelo Hawaiʻi (haw)
Igbo (ig)
Indonesian (id)
isiZulu (zu)
Íslenska (is)
Italiano (it)
Italiano per Workplace (it_wp)
Japanese - kids (ja_kids)
Kalaallisut (kl)
Kinyarwanda (rw)
Kiswahili (sw)
Kreyòl Ayisyen (hat)
Kurmanji (kmr)
Kvääni (fkv)
Laotian (lo)
Latin (la)
Latviešu (lv)
Lengadocian (oc_lnc)
Lëtzebuergesch (lb)
Lietuvių (lt)
Lithuanian (university) (lt_uni)
Lulesamisk (smj)
magyar (hu)
Malagasy (mg)
Māori - Tainui (mi_tn)
Māori - Waikato (mi_wwow)
Māori (Te Reo) (mi)
Mapudungún (arn)
Mirandés (mwl)
Mongolian (mn_mong)
Nederlands (nl)
Nederlands Workplace (nl_wp)
Norsk - bokmål (no)
Norsk - nynorsk (nn)
Norsk (no_gr)
Norsk bokmål (nb)
Norsk Workplace (no_wp)
O'zbekcha (uz)
Pidgin (pcm)
Polski (pl)
Português - Brasil (pt_br)
Português - Portugal (pt)
Português para Workplace (pt_br_wp)
Qaraqalpaq tili (kaa)
Română (ro)
Romansh Sursilvan (rm_surs)
Samoan (sm)
Setswana (tn)
Shqip (sq)
Sicilianu (scn)
Slovenčina (sk)
Slovenščina (sl)
Soomaali (so)
Sørsamisk (sma)
Srpski (sr_lt)
Suomi (fi)
Suomi+ (fi_co)
Svenska (sv)
Tagalog (tl)
Tamil (ta)

Tamil (ta)
Taqbaylit (kab)
Thai (th)
Tok Pisin (tpi)
Tongan (to)
Türkçe (tr)
Turkmen (tk)
Uyghur - latin (ug_lt)
VakaViti (fj)
Valencià_RACV (ca_valencia_racv)
Vietnamese (vi)
Wolof (wo)
Workplace în limba română (ro_wp)
Workplace 简体中文版 (zh_cn_wp)
Workplace 繁體中文版 (zh_tw_wp)
Workplace準拠の日本語 (ja_wp)
Ελληνικά (el)
Ελληνικά για παιδιά (el_kids)
Ελληνικά για σχολές (el_uni)
Ελληνικά για Χώρους Εργασίας (el_wp)
Башкорт теле (ba)
Беларуская (be)
Български (bg)
Кыргызча (ky)
Қазақша (kk)
Македонски (mk)
Монгол (mn)
Русский (ru)
Српски (sr_cr)
татар теле (tt)
Тоҷикӣ (tg)
Українська (uk)
ქართული (ka)
Հայերեն (hy)
עברית (he)
עברית בתי־ספר (he_kids)
Workplace עברית עבור (he_wp)
ئۇيغۇرچە (ug_ug)
اردو (ur)
workplace بالعربي (ar_wp)
پښتو (ps)
دری (prs)
سنڌي (sd_ap)
سۆرانی (ckb)
عربي (ar)
فارسی (fa)
ދިވެހި (dv)
ⵜⴰⵎⴰⵣⵉⵖⵜ (zgh)
ትግርኛ (ti)
नेपाली (ne)
बच्चों के लिए हिंदी (hi_kids)
मराठी (mr)
वर्कप्लेस के लिए हिंदी (hi_wp)
हिंदी (hi)
বাংলা (bn)
ਪੰਜਾਬੀ (pan)
ગુજરાતી (gu)
ଓଡ଼ିଆ (or)
தமிழ் (ta_lk)
తెలుగు (te)
ಕನ್ನಡ (kn)
മലയാളം (ml)
සිංහල (si)
ᨅᨔᨗ (xct)
རྫོང་ཁ (dz)
ဗမာစာ (my)
ខ្មែរ (km)
한국어 (ko)
日本語 (ja)

Português (pt)
正體中文 (zh_tw)
简体中文 (zh_cn)

Data retention summary
Get the mobile app
Policies