Tokyo Task Assignment

Project Description

We are developing an online store, making it possible for our customers to order different items. Each item has its name, description, image and price. For each of our customers, we keep some basic information: first and last name, and contact information: e-mail and phone number. Interaction with the customer is represented by an order. A different number of items may be added to an order. Whenever an item is added to the order, the overall price is updated.

Solution

A small Spring Boot application was created (IntelliJ IDEA was used). There are 3 different REST API's: CustomerController, ItemController and OrderController.

As requested, OrderController makes it possible for a customer to create, modify and place an order.

There are 4 different models: Customer, Item, Order and OrderedItem. OrderedItem is used to represent an item that has been added to an order, alongside the quantity.

Java Persistence API (JPA) was used to persist Java objects into relational database. For each table in the database a repository class was created, that extends JpaRepository.

In between API and Data Access Layer, there is a Service Layer. It consists of CustomerService, ItemService and OrderService and is used for defining "business logic".

Exceptions are being handled across whole application (ErrorCode enum is used to define different error codes, that are used to differentiate exceptions without the need to define a class for every exception that can occur – BaseException practically defines all of them.), RestExceptionAdvice class annotated with @RestContollerAdvice.

This solution assumes that there is a Client side already defined. Postman was used for testing purposes.

Sample Requests

Create order:

```
POST /api/v1/order/createOrder HTTP/1.1
Host: localhost:8080
Content-Type: application/json
Content-Length: 43

{
    "customerId": 3,
    "itemId": 4
}
```

List all orders:

```
GET /api/v1/order/orders HTTP/1.1
Host: localhost:8080
```

Get order by ID:

```
GET /api/v1/order/orders/2 HTTP/1.1
Host: localhost:8080
```

Get orders by customer:

```
GET /api/v1/order/getOrdersByCustomer/3 HTTP/1.1
Host: localhost:8080
```

Remove item from order:

```
DELETE /api/v1/order/removeItem HTTP/1.1
Host: localhost:8080
Content-Type: application/json
Content-Length: 40

{
    "orderId": 1,
    "itemId": 5
}
```

Add item to order:

```
PUT /api/v1/order/addItem HTTP/1.1
Host: localhost:8080
Content-Type: application/json
Content-Length: 41

{
    "orderId": 1,
    "itemId": 10
}
```

Place an order:

```
PUT /api/v1/order/placeOrder/1 HTTP/1.1
Host: localhost:8080
```