Documentation of Shopify Developer Intern Challenge Project

Wei Wei

**0. Before the introduction**

I am an international CS student from University of Ottawa, last term our professor asked all the students in CSI 5380 to form a team of 4 people to build a online bookstore website, which have functions like user register and login, querying for all products, purchase products.

At that time I was responsible for **database design and the development of Service layer and DAO layer.** Instead of using API, I used normal Java class as Service, which resulted in a successful project:

**https://github.com/SevenWG/BookStore\_Parto.git**

Then I used Jersey to rebuild the Service layer in that project, and also I uploaded it to Github:

**https://github.com/SevenWG/BookStoreAPI.git**

After I saw the Summer Intern developer challenge, I found that the only difference is that there is no “inventory count” in my BOOK table, so I altered my database table and modified some functions in the Second project to meet the requirements in Intern challenge.

I also uploaded my project for Intern challenge to Github:

**<https://github.com/SevenWG/InternChallengeQuestion.git>**

**(The database file is in the Root directory)**

I didn’t delete other functions and files which are not related to this challenge just because I want to show you the project I developed before. Sorry about the inconvenience .

**1. Development Environment**

System: Mac OS 10.13.6

IDE: IntelliJ IDEA 2018.2.4 (Ultimate Edition)

Database: MySQL 8.0.12

Programming Language : Java 10 + Hibernate + REST API + Maven

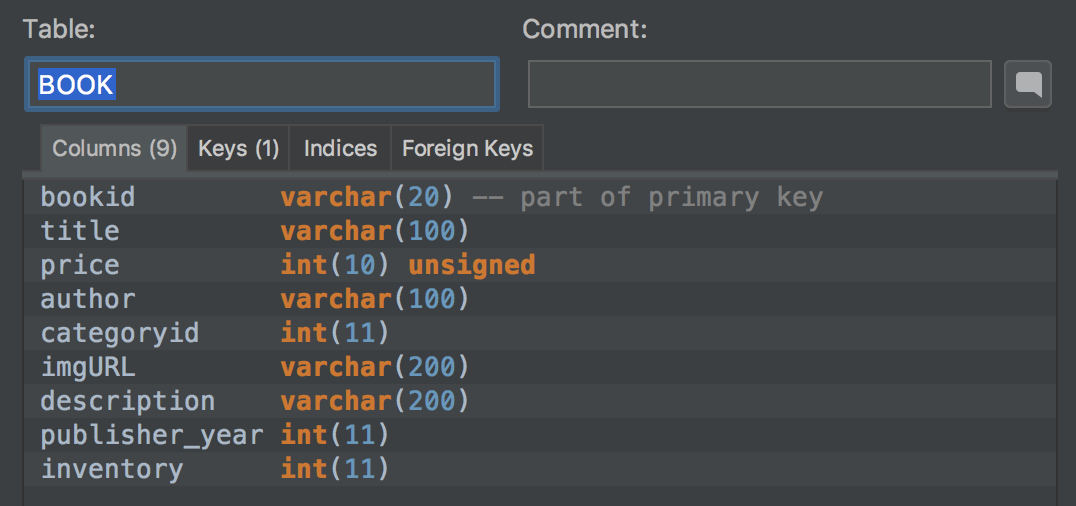
**2. Database Design**

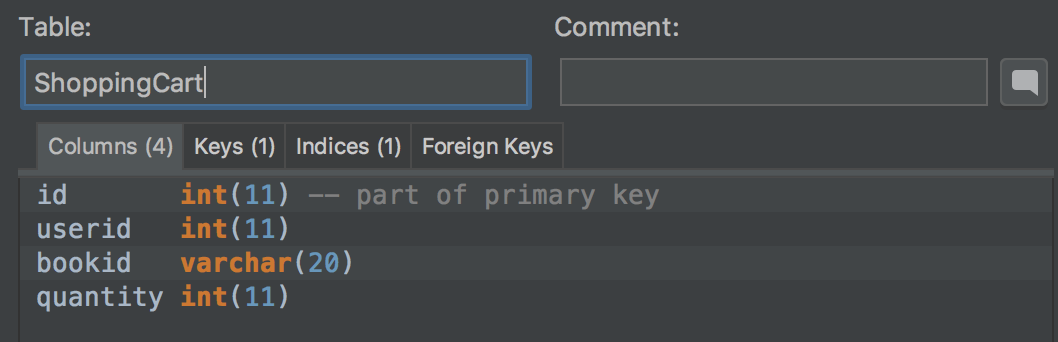
Product Table: BOOK

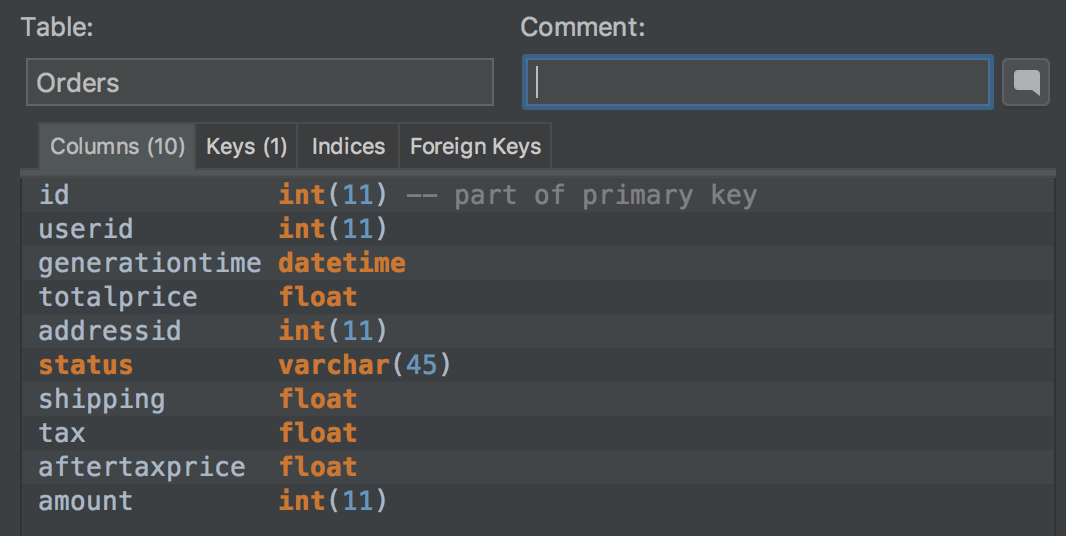
Primary key: bookid

title: title

price: price

inventory\_count: inventory

Shopping cart Table: ShoppingCart

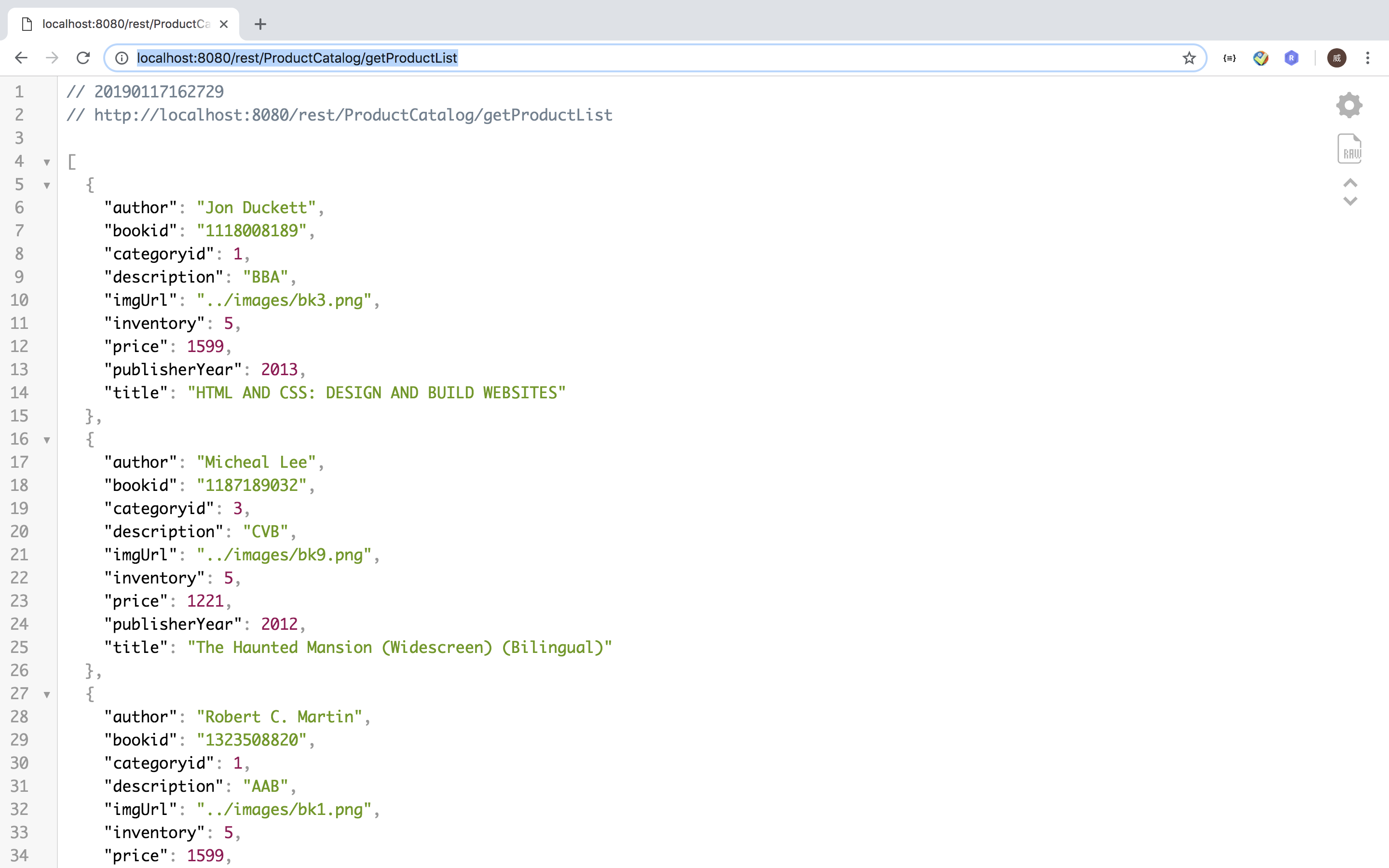
Orders Table: Orders

(This part my design is a little different with the requirement. I will explain in the following parts)

**3. Methods Design**

① Querying for all products:

Set up and run the server

input "**<http://localhost:8080/rest/ProductCatalog/getProductList>**" in the browser, and results will be output by JSON format:

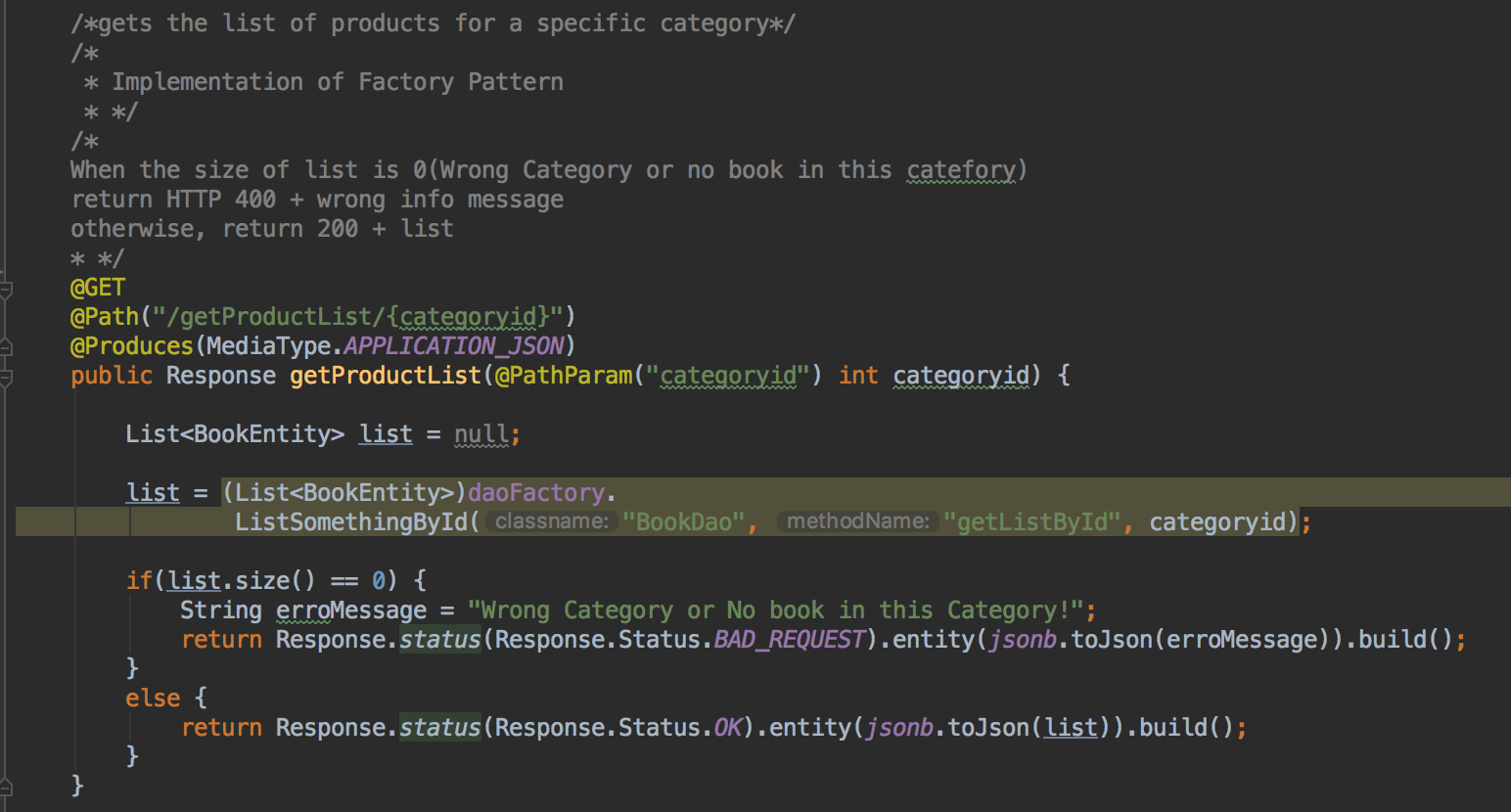
Or you can input "**<http://localhost:8080/rest/ProductCatalog/getProductList>**" **+** **specific category id**, then it will only output products in that category

Note that Only those products whose inventory value is more than 0 will be output.

Source code:

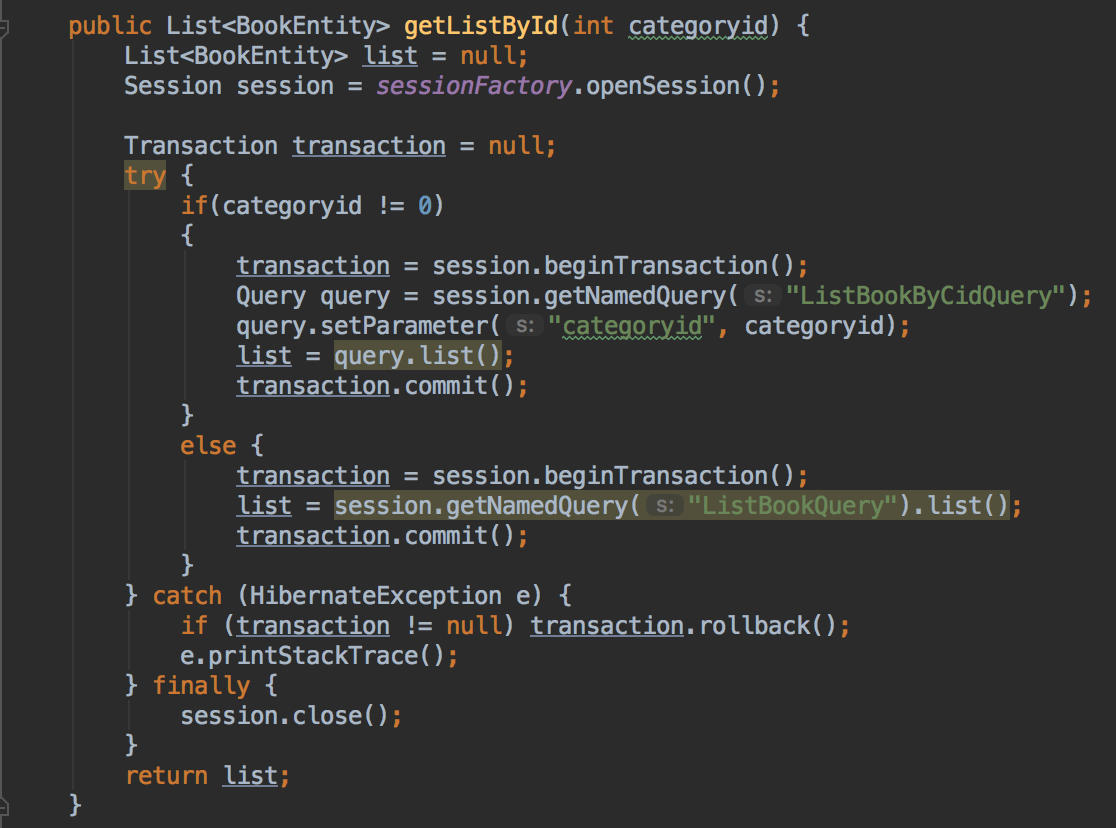
Service layer:

**com.team404.bookstore.service.ProductCatalogAPI.java**



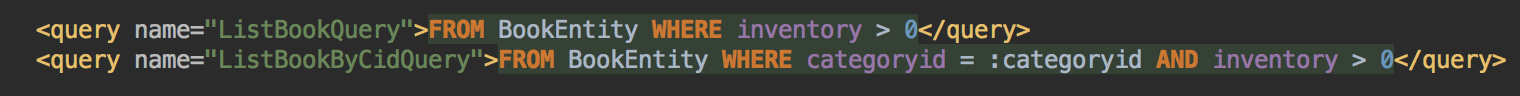
DAO layer:

**com.team404.bookstore.dao.BookDao.java**



HQL query:

**com.team404.bookstore.entity.BookEntity.hbm.xml**

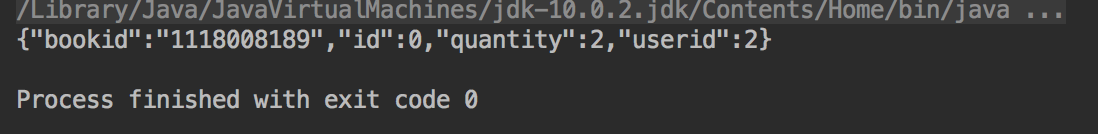
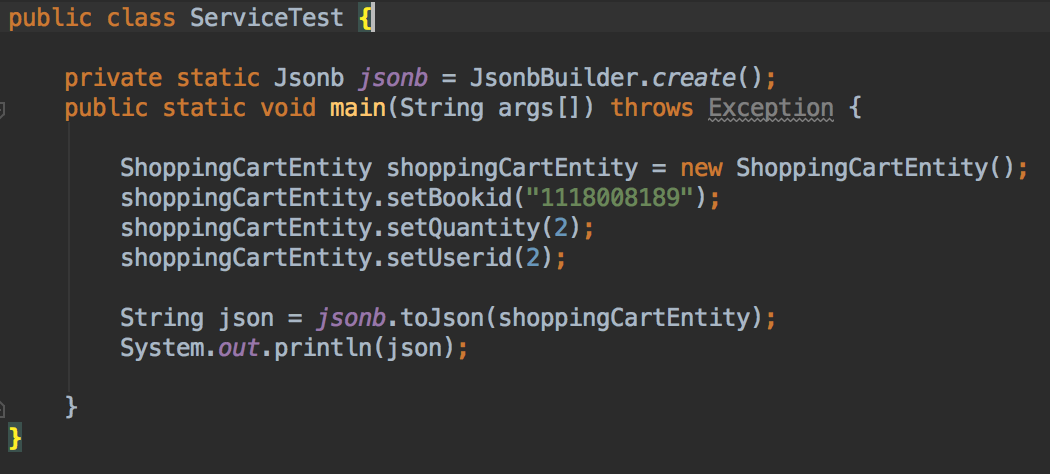


② Create shopping cart and add items into it

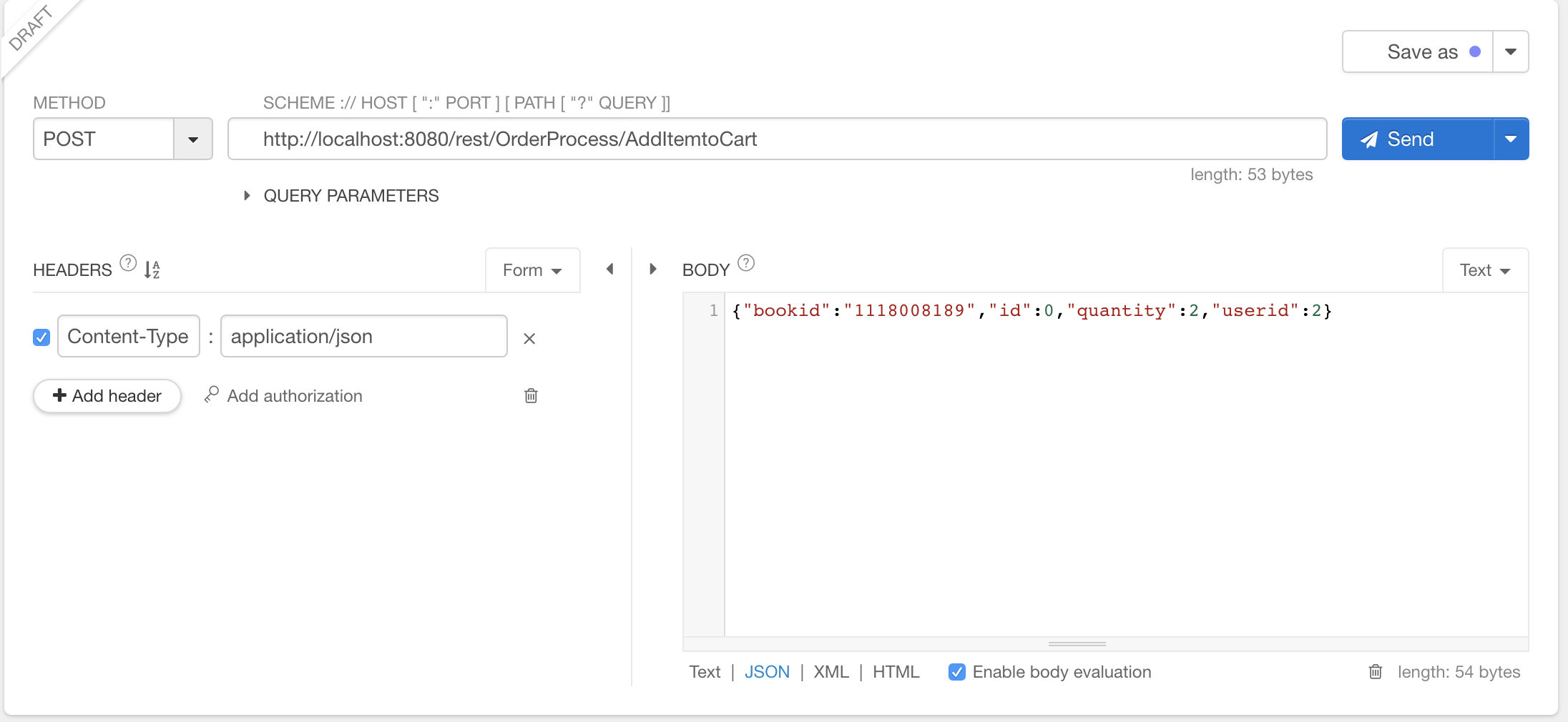
Because in this project shopping cart is also a data table in database, it also has Hibernate entity class and mapping file.

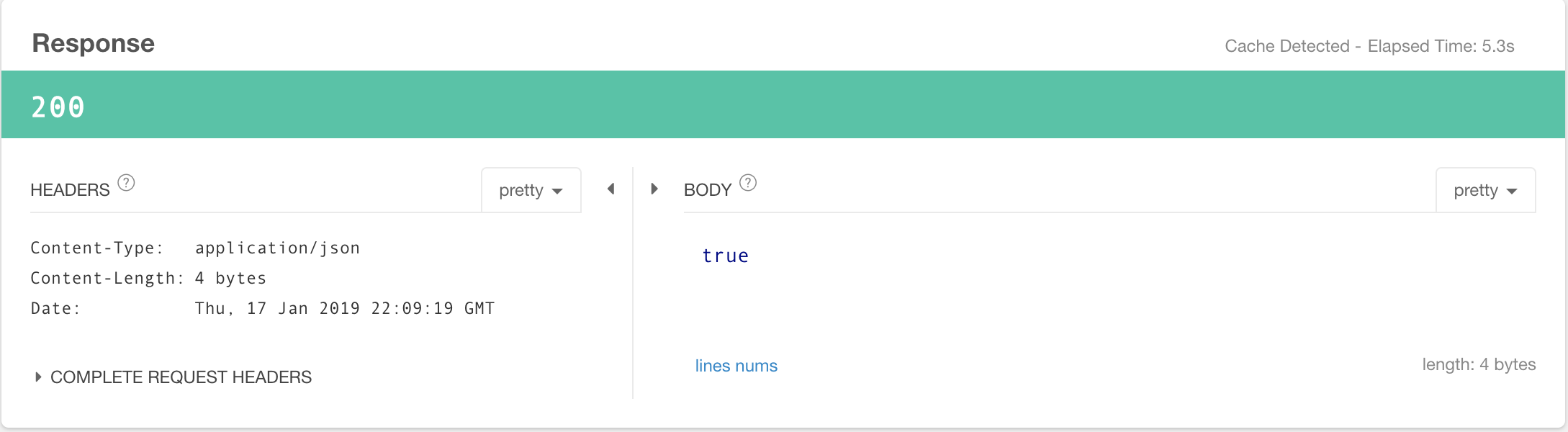
In service layer, the method receive a String by POST, which is a shopping cart object in JSON format, then the method will transform it back to a shopping cart object, then use the DAO method to store it in database.

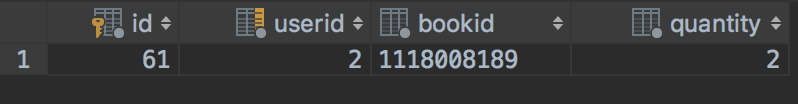
Because we don’t have a front end. Firstly we need initiate a shopping cart entity(object), then transform it into JSON string, then we use **Restlet Client** to do the POST action.

Create JSON string: com.team404.bookstore.service.ServiceTest

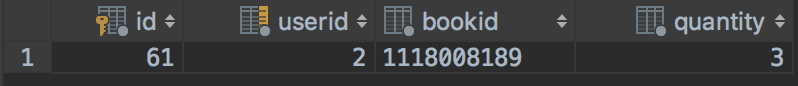
Then we paste it into Restlet Client and input url:

**"http://localhost:8080/rest/OrderProcess/AddItemtoCart"**

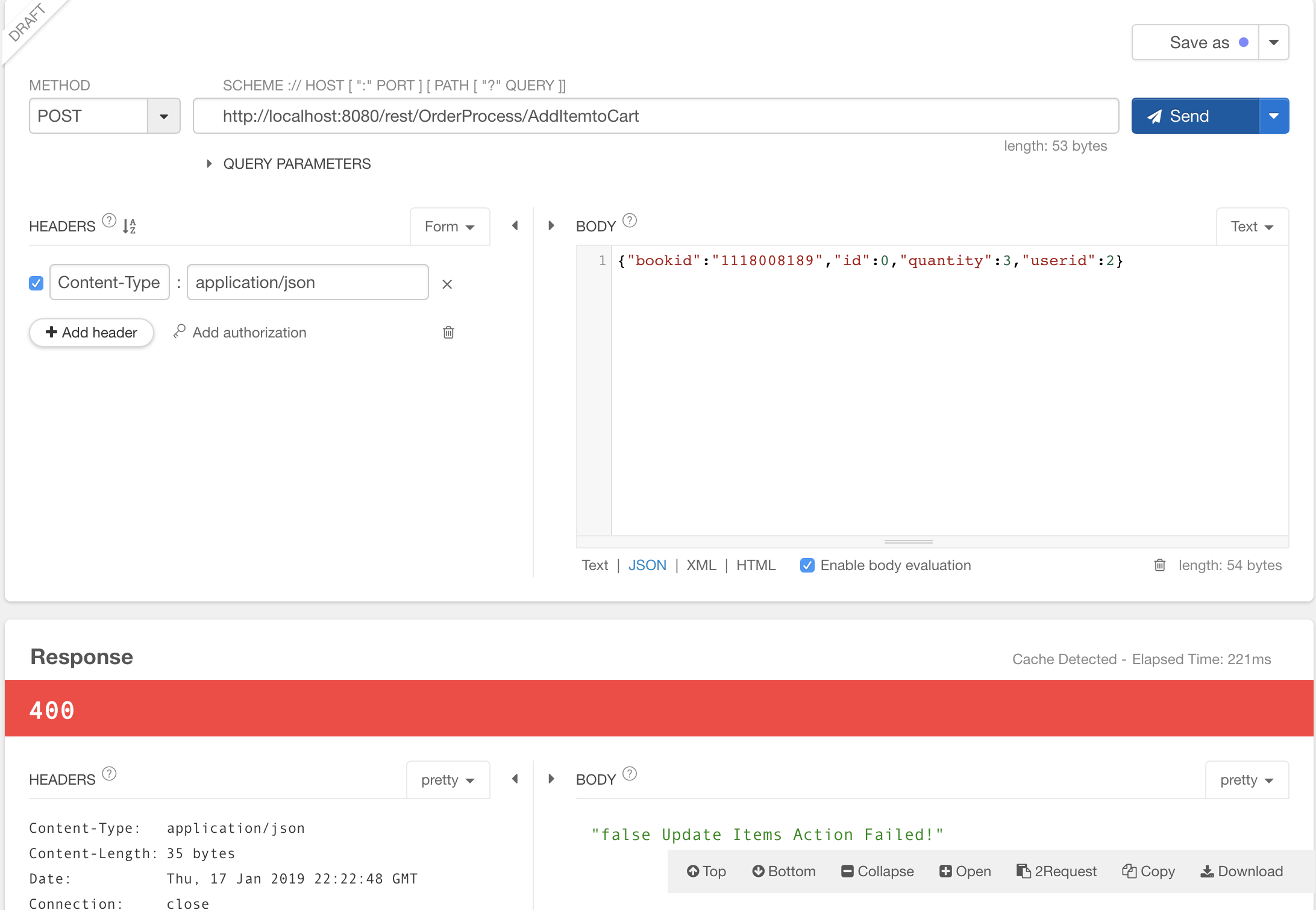
After clicking “Send”, the Response will be shown below:

And the shopping cart object will be added into ShoppingCart table:

If the same user add same book again before he check out (assume this time he add 1 book with same Book id, so the json string is {“bookid":"1118008189","id":0,"quantity":1,"userid":2} )

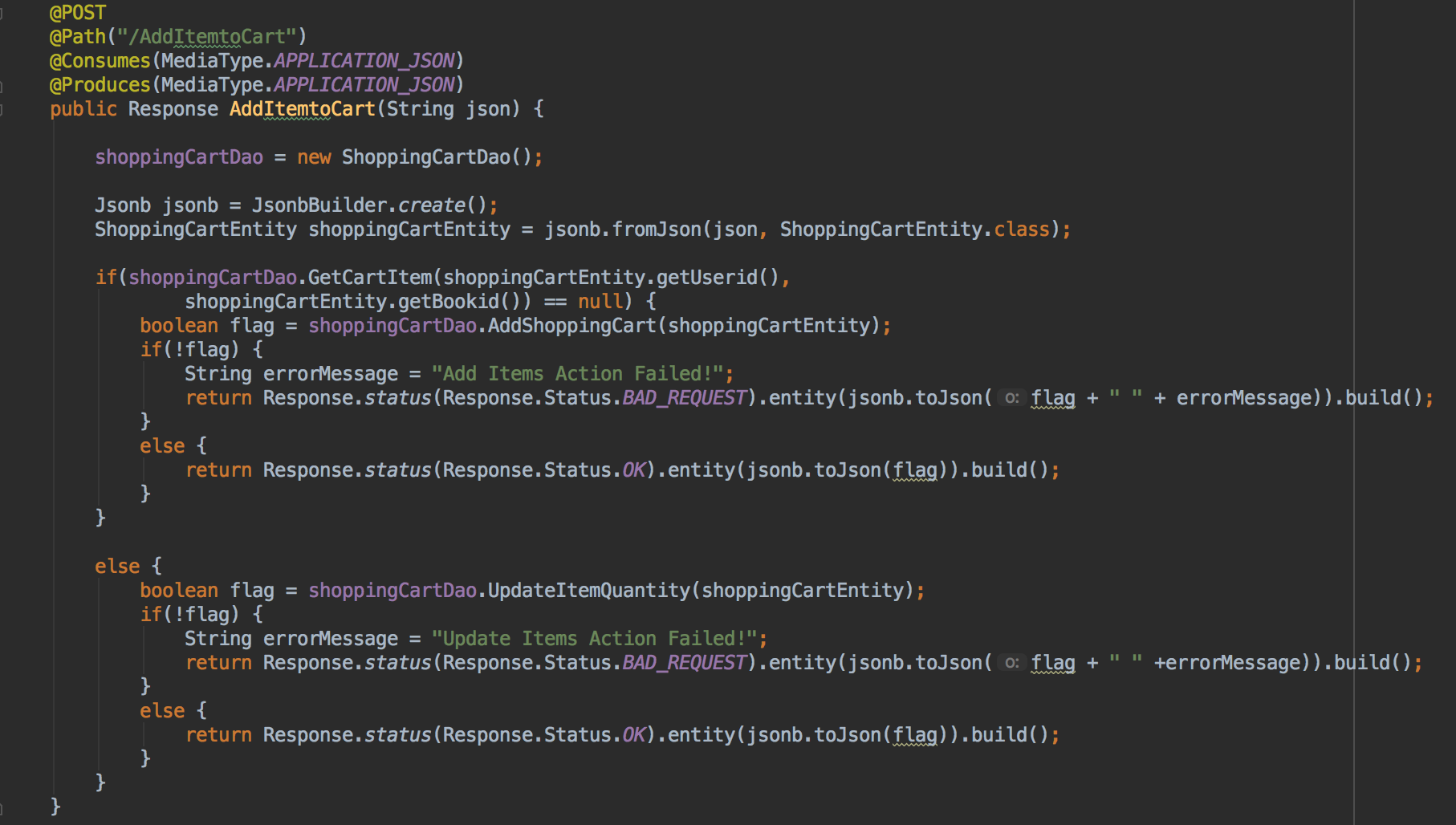
Then it will simply update the previous row instead add a new row.

If the total number of books a user wants to buy is greater than the inventory, then the add or update action will fail.

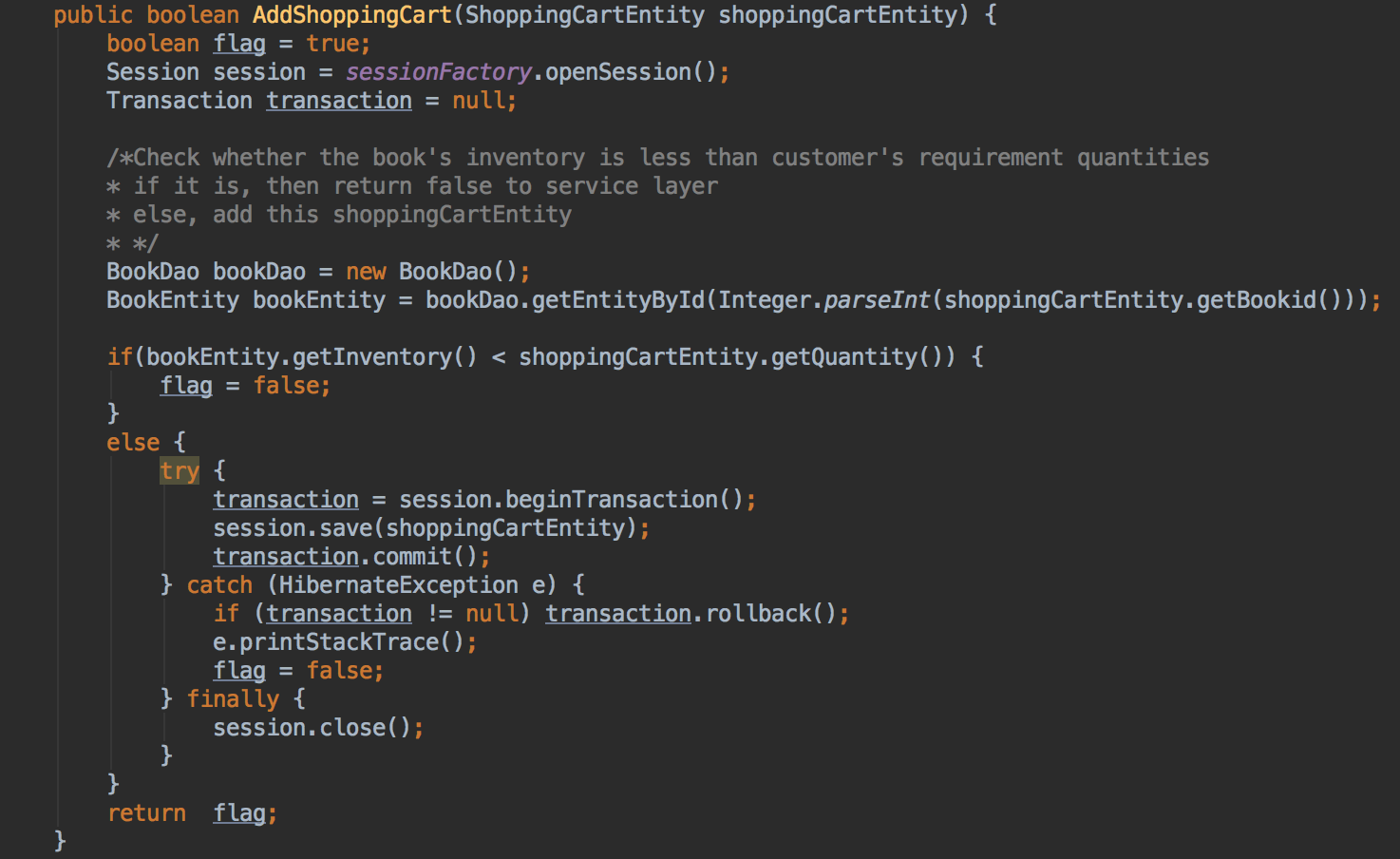
Assume now there is 3 book in this user’s shopping cart, if he still wants to buy 3 more books with the same book id but the inventory count is 5, then this action will fail:

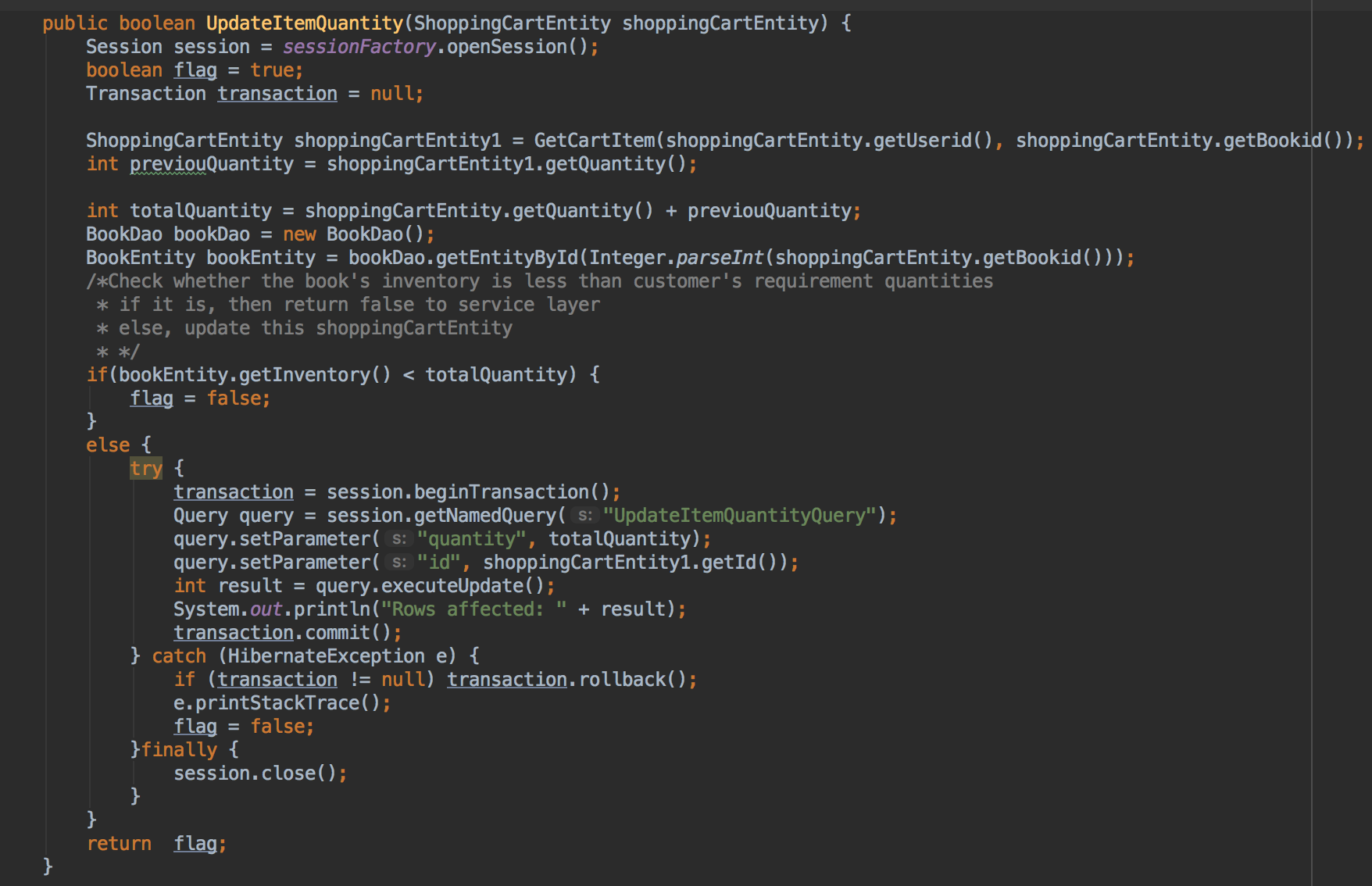
Source Code:

**Service layer: com.team404.bookstore.service.OrderProcessAPI.java**

Method: public Response AddItemtoCart(String json)

**Dao layer: com.team404.bookstore.dao.ShoppingCartDao.java**

Method: public boolean AddShoppingCart

Method: public boolean UpdateItemQuantity

③Check out

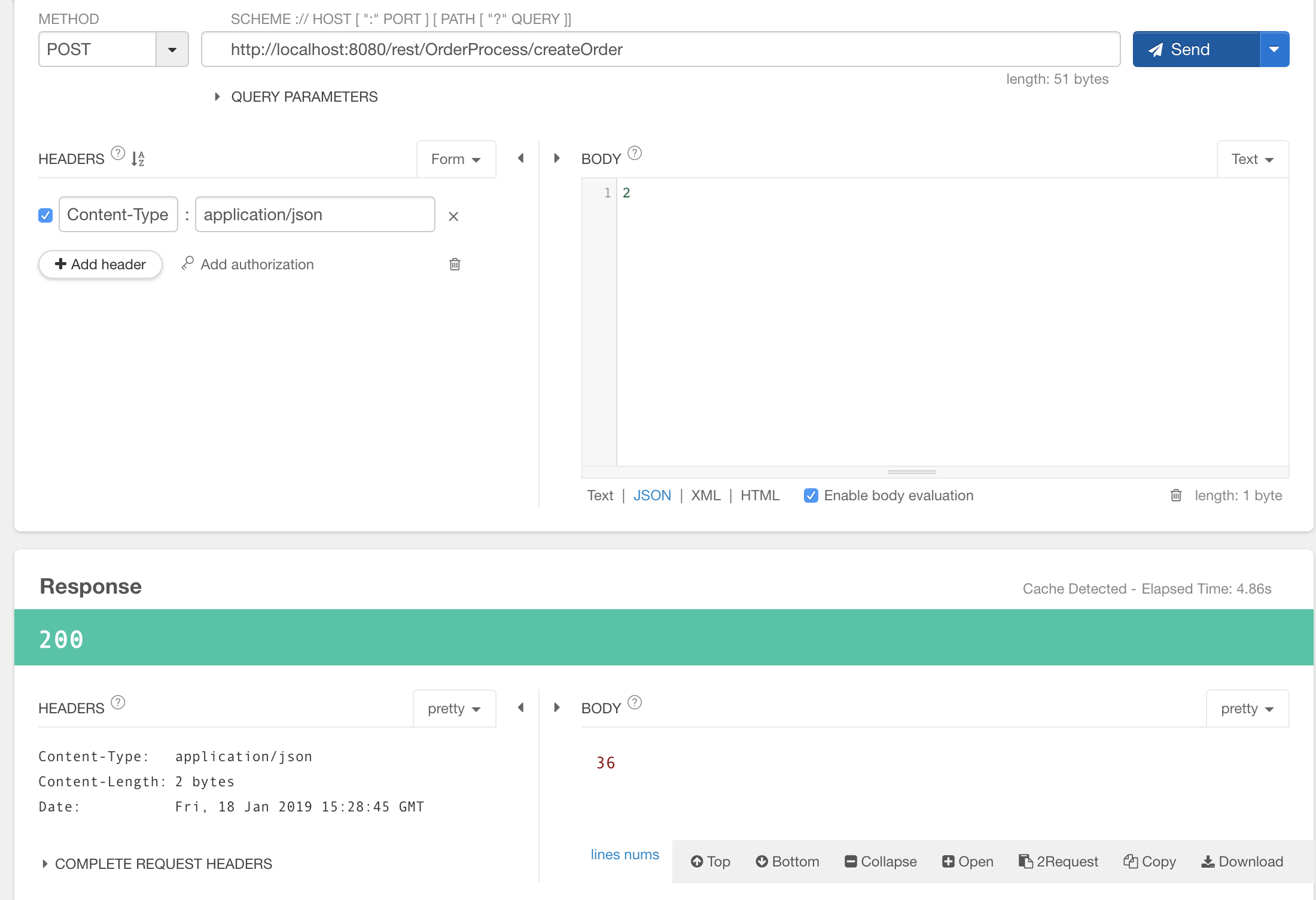
Check out is divided into 2 parts: Create Order and Confirm Order

In Create Order part, the createOrder method in service layer will receive an userid by POST, then it will generate a order object according to user id and all the rows in ShoppingCart Table which have the same user id.

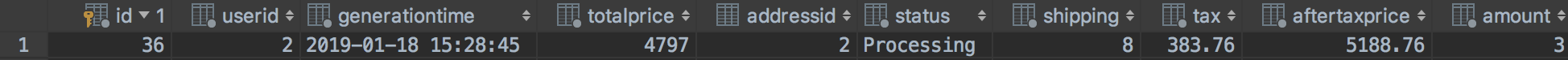
The order object contains total amount of products, total price (before and after tax), generation time and some other values. The total price, total amount of products and other values are generated by different specific methods.

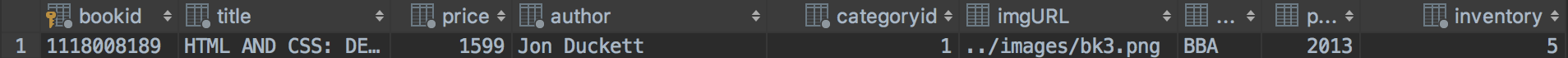
Then a method in DAO layer will store this object into database and return this order’s id(primary key) to Service layer.

Go back to Restlet Client, and input the url: **“http://localhost:8080/rest/OrderProcess/createOrder”** and the userid.

After clicking “Send”, then it will get the response, which is the order’s id

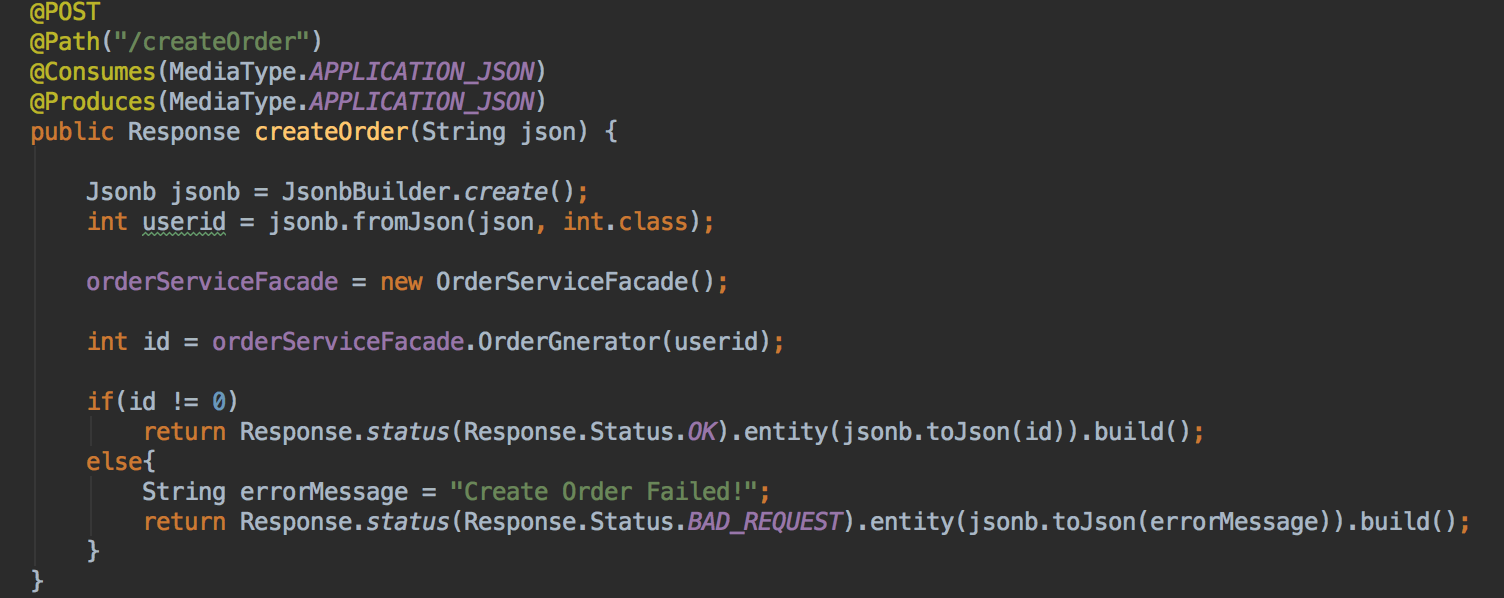
In Orders Table, we can see that this order object has been added into the table. But now it status is still “Processing”, and the book’s inventory count in BOOK table has not been changed yet.

Orders Table:

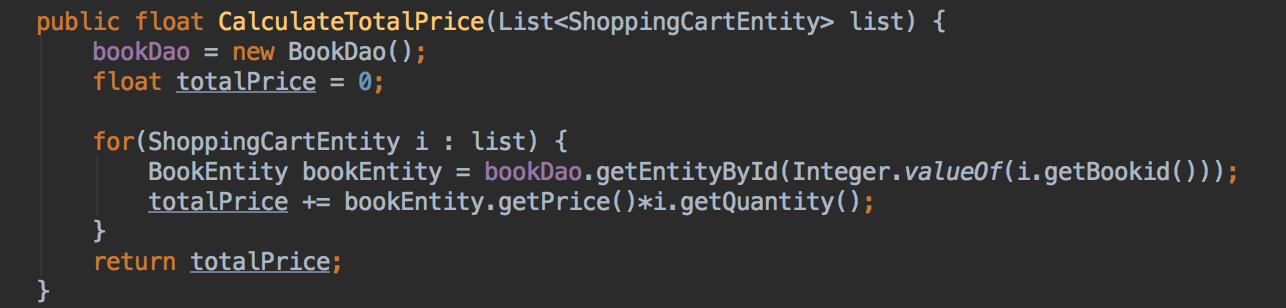
BOOK Table:

Source Code

Service layer: com.team404.bookstore.service.OrderProcessAPI

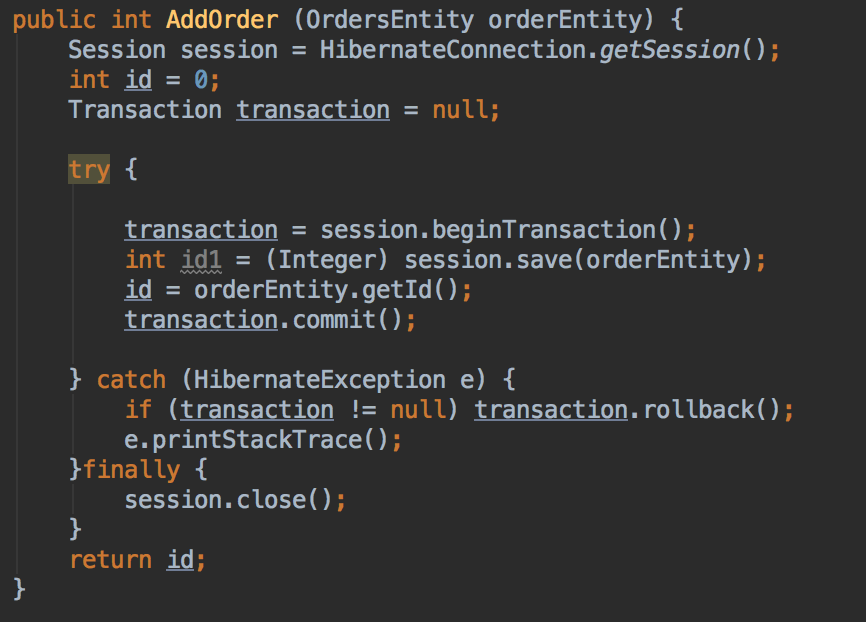
Method: public Response createOrder(String json)

Service layer: com.team404.bookstore.service.PriceCalculator

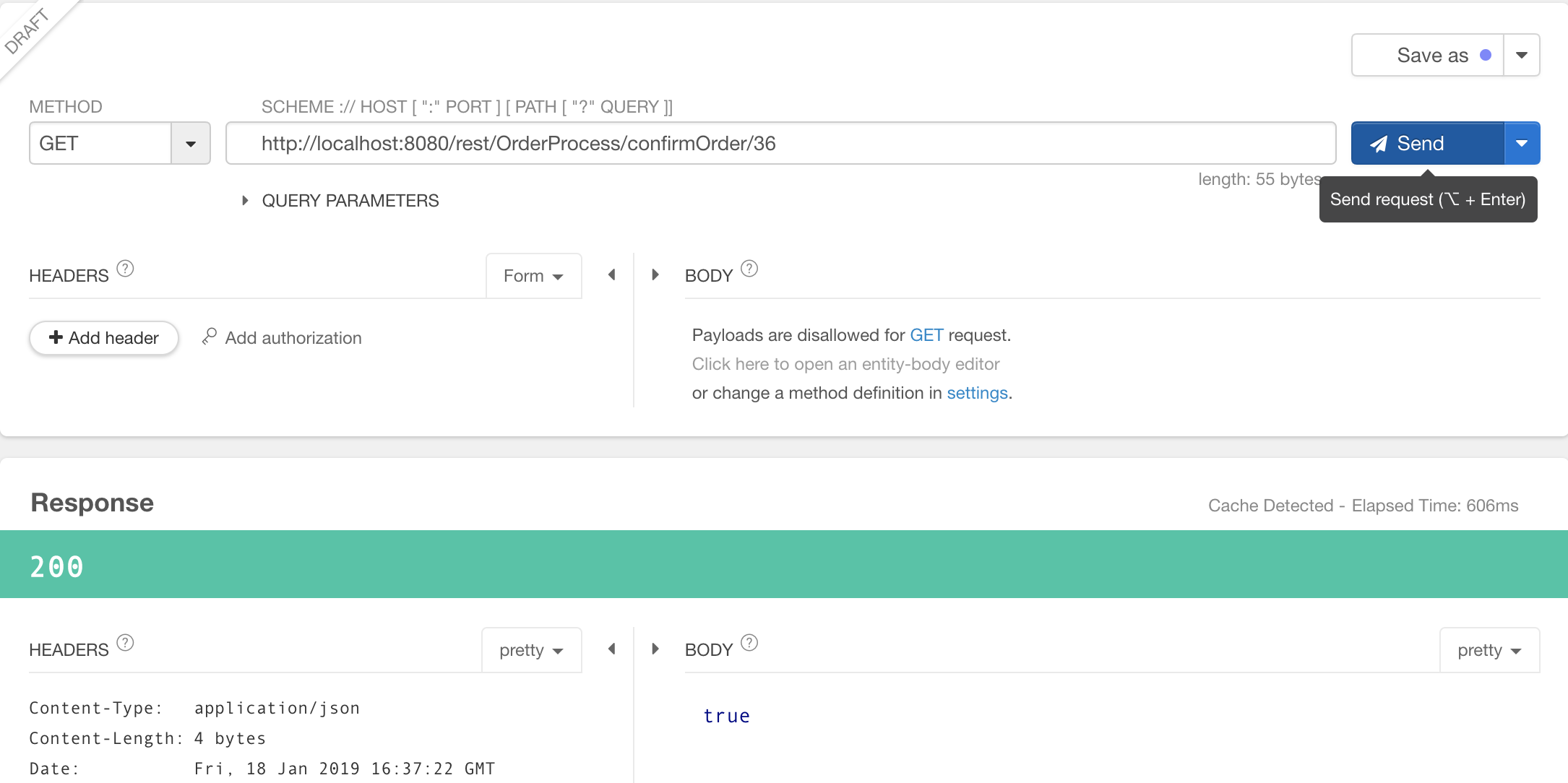
Method: public float CalculateTotalPrice

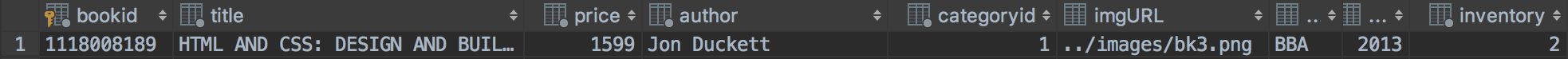
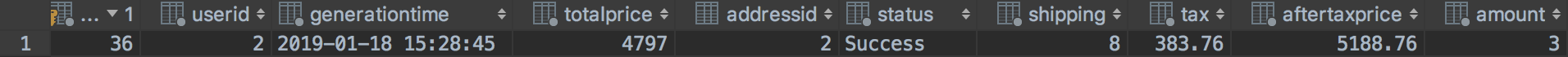
DAO layer: com.team404.bookstore.dao.OrderDao

Method: public int AddOrder

In Confirm Order Part, the confirmOrder method in service layer will receive a order id by GET, then it will call a method in DAO lay to update the order’s status from “Processing” to “Success” or “Failed”.

After the update action, if the order’s status is Success, then another method in DAO layer will update(reduce) the inventory count of the book(s) in this orders.

Go back to Restlet Client, select “GET” method and input the url: **“http://localhost:8080/rest/OrderProcess/confirmOrder/” + Order id**, then click “Send”, the response will be shown below:

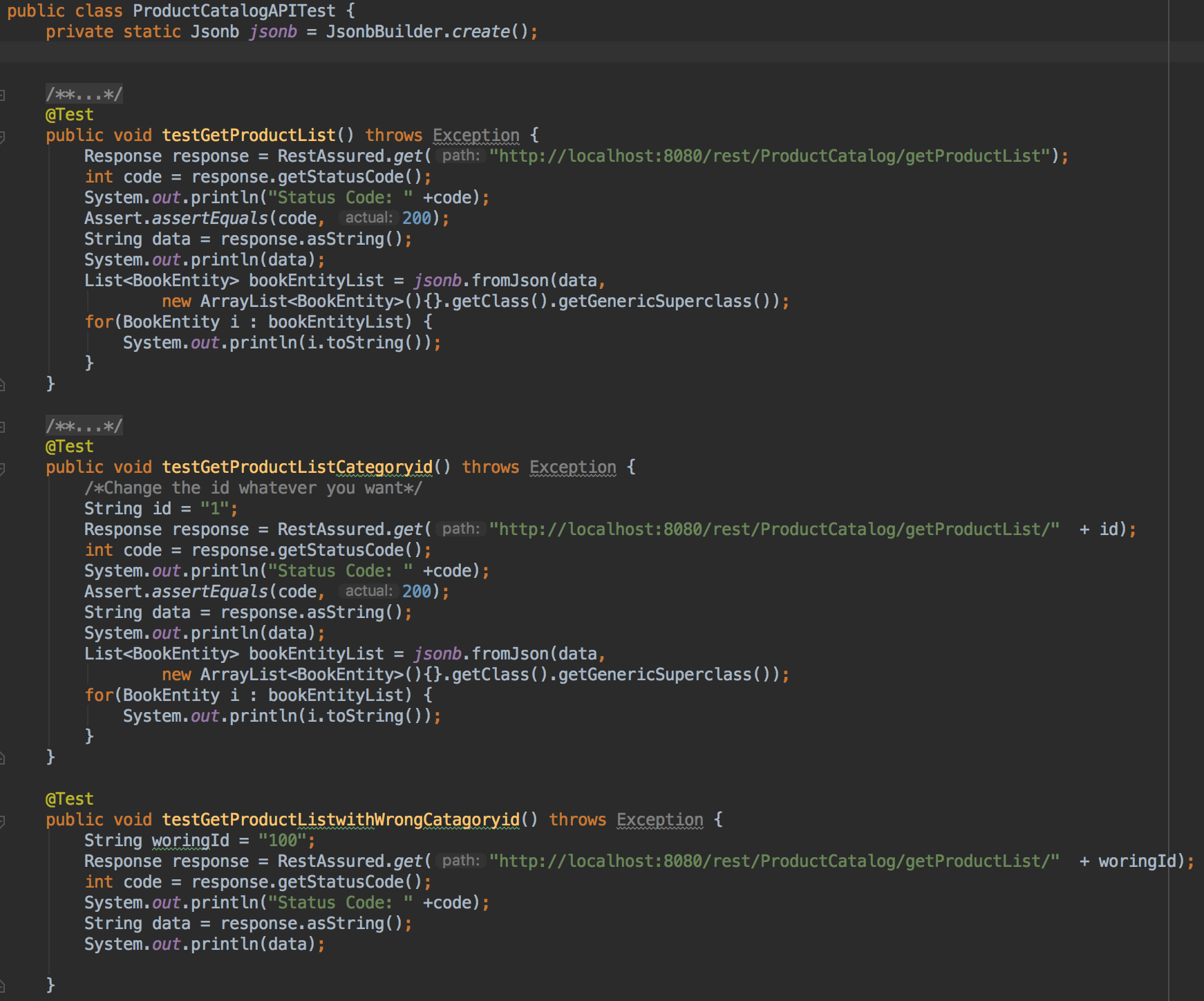
Check the Orders Table and BOOK Table, the order status and inventory have been updated:

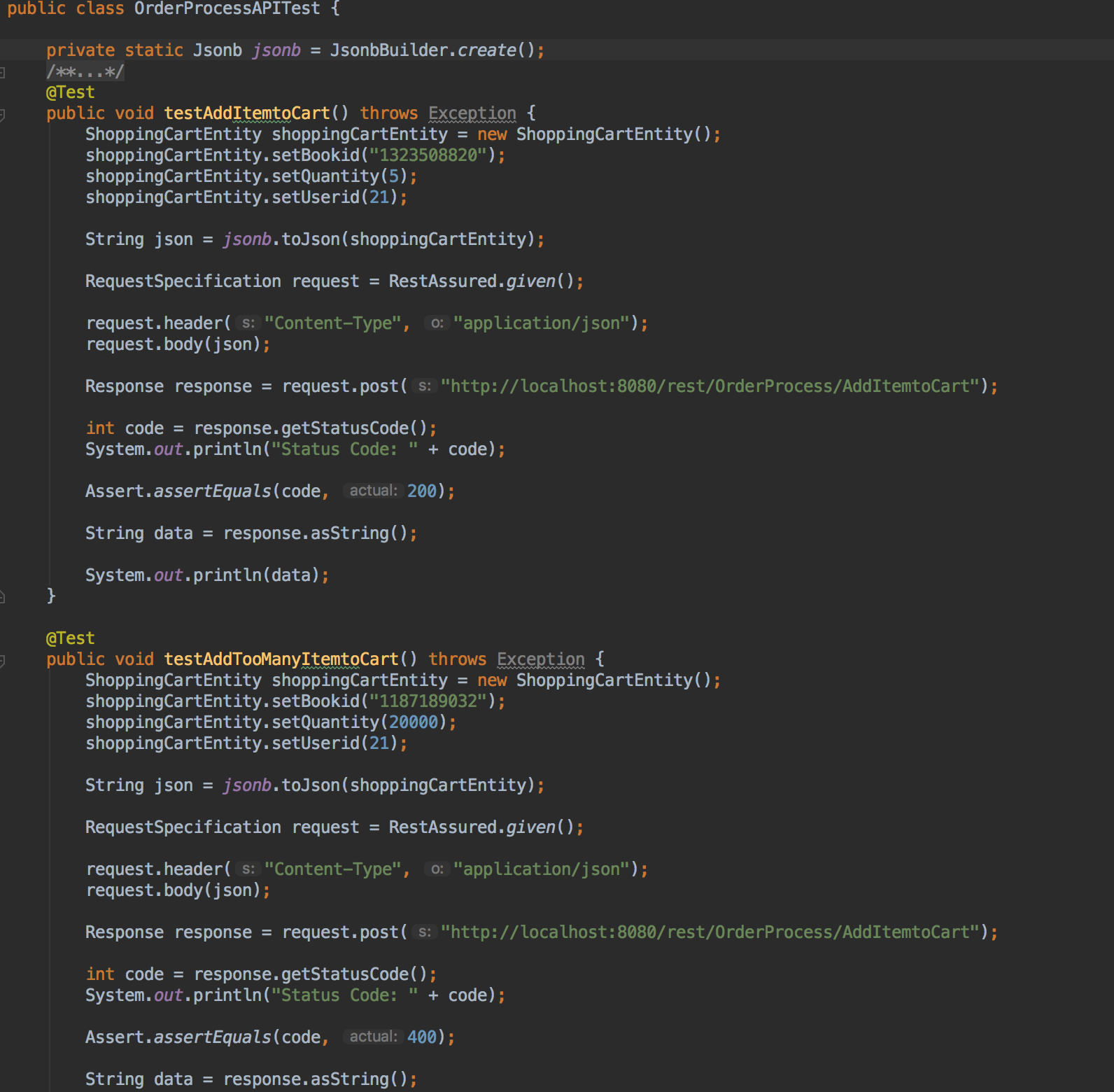
4. Unit Test:

Test Tools: **Junit 4 + Rest-Assured**

Test Files:

**/src/resources/java/ProductCatalogAPITest.java**



**/src/resources/java/O****rderProcessAPITest.java**