1. MyRetail REST API - Objectives

MyRetail RESTful service offers the following features
1. Retrieve the Product and Price information for a given product ID

2. Ability to modify the price information in the database

2. Frameworks/Tool/Technologies:

1. Spring Boot Rest Template: to expose REST Services, and to develop a REST client to consume the product information from external API.

2. MongoDB: To store the product price information.

3. Docker: To build and manage the deployment artifacts and to create/ manage the MongoDB

4. Mockito: Unit Testing

5. PostMan : Unit testing

6. Maven: Build Configuration

7. GIT: Source configuration

******Services:

3. Services - Implementation

3.1 Get Product Information:

The consumer can do a GET request at the path "/products/{id}" for a product detail to "redsky.target.com" and retrieves the product description, and is appended to the available price and name information. For a product with product id '13860428', the sample JSON output is as shown below

{"id":13860428,"name":"The Big Lebowski (Blu-ray) (Widescreen)","current_price":{"value": 13.49,"currency_code":"USD"}}

3. 2. Update Product Price in MongoDB:

The price informatio could be udpated with this API. The user/client application can do a PUT request with input similar to the response received in GET and should be able to modify the price in the datastore. The request is done at the same path "/products/{id}"
####Sample Input: JSON Body - {"value": 15.67,"currency_code":"USD"}

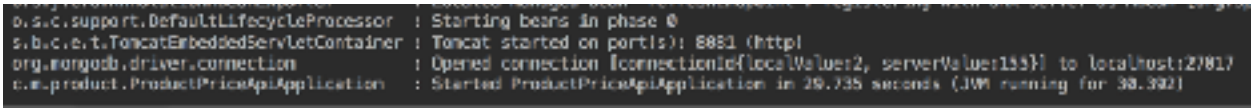
4. Instructions to setup the environment

- Clone the code from the git repository

The API can be deployed in dev environment with Intellij/Eclipse or commandline, or as a Docker container

4.1 Dev Environment:

- Install Maven, Docker
- Docker: 'docker-compose -f docker-mongo.yml up -d' (to Install mongoDB as docker container), or setup a standalone MongoDb
- maven clean package - to create Jar file for REST endpoints
- execute: mvn Sprint-boot:run. to bring up the server



- or run the API in docker container

docker-compose up -d

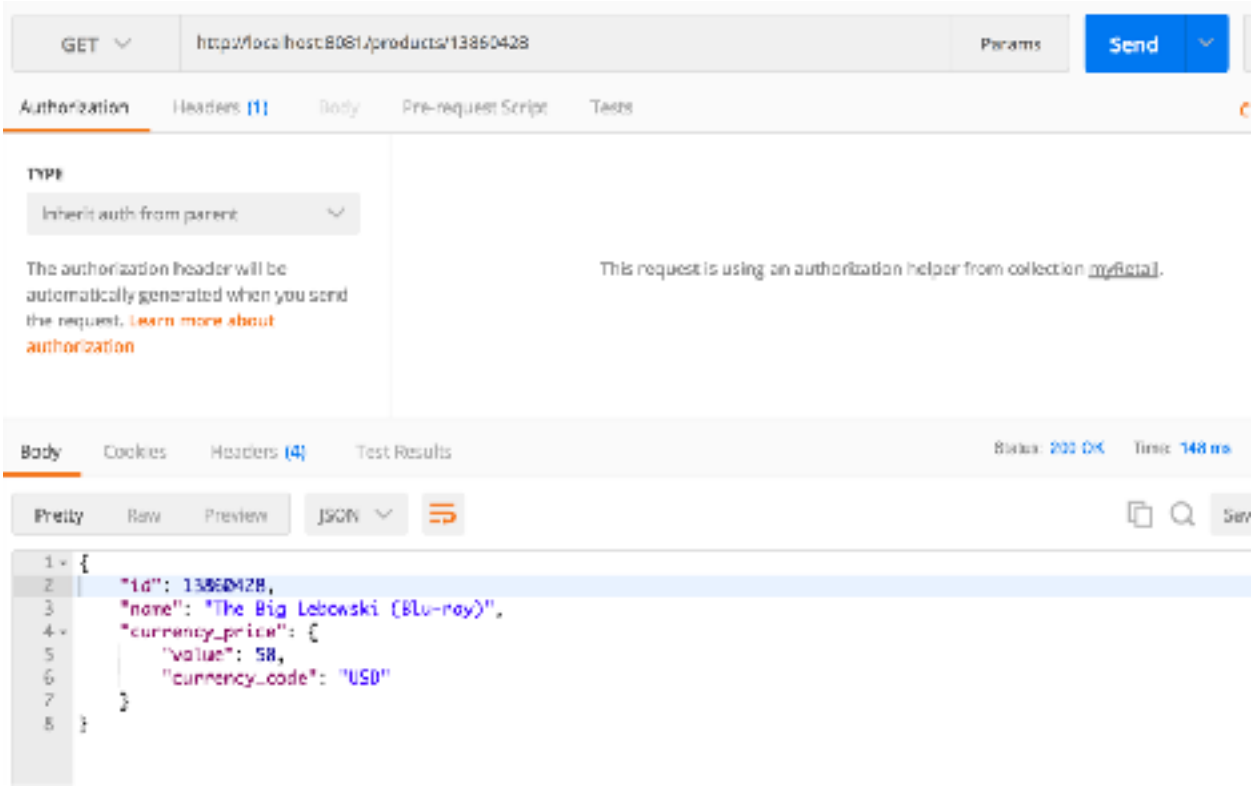
5 Testing Results

The testcases are implemented using 'mockito' framework under the folder 'src\test\java\ '.
The test cases can be executed by running the command 'mvn test'

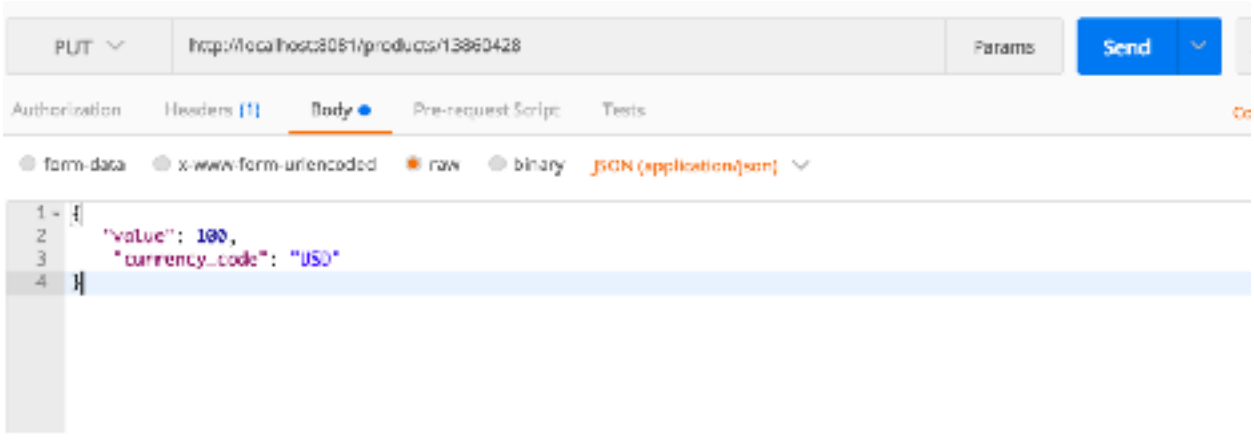
^^^PostMan UI:

Attached the Test results using Postman UI

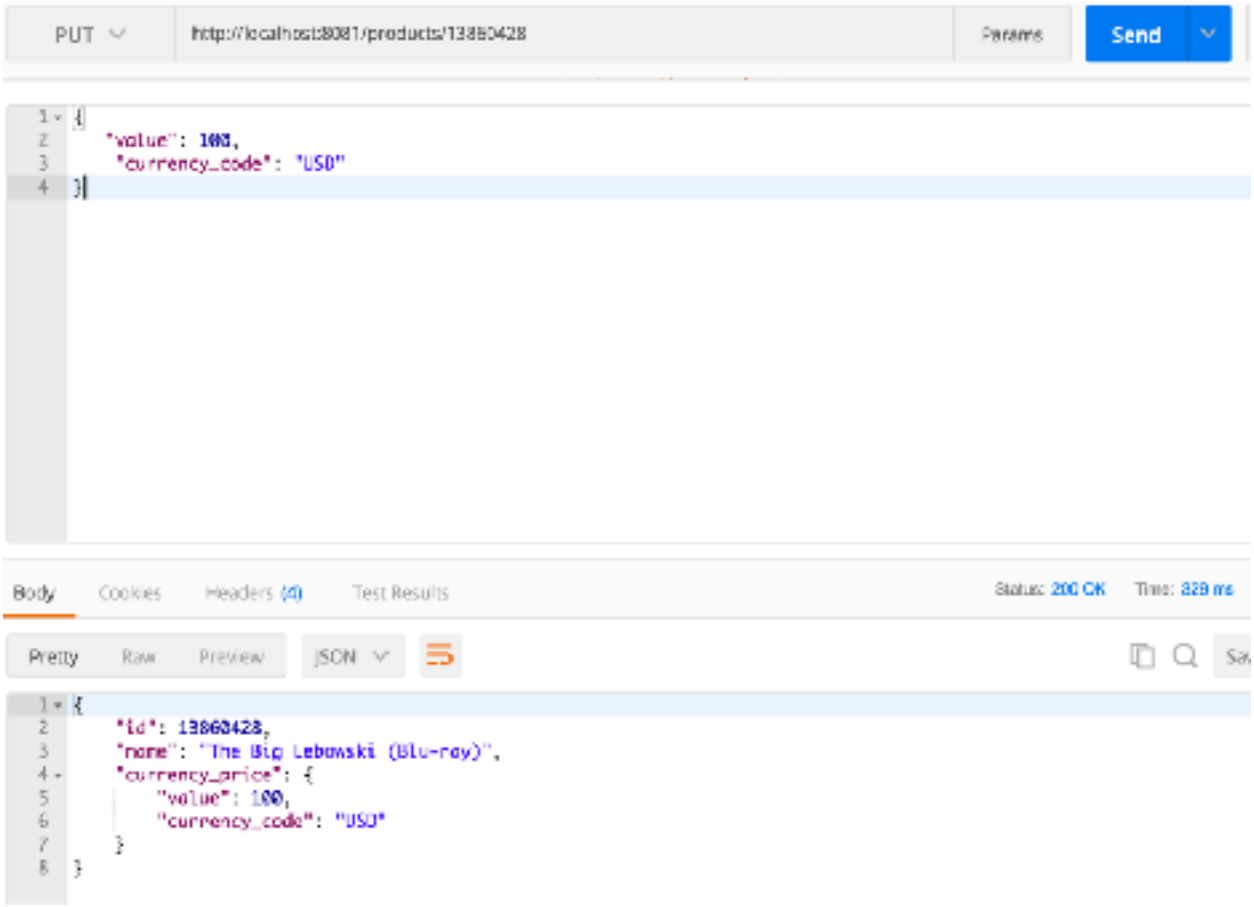
1. GET Product Request



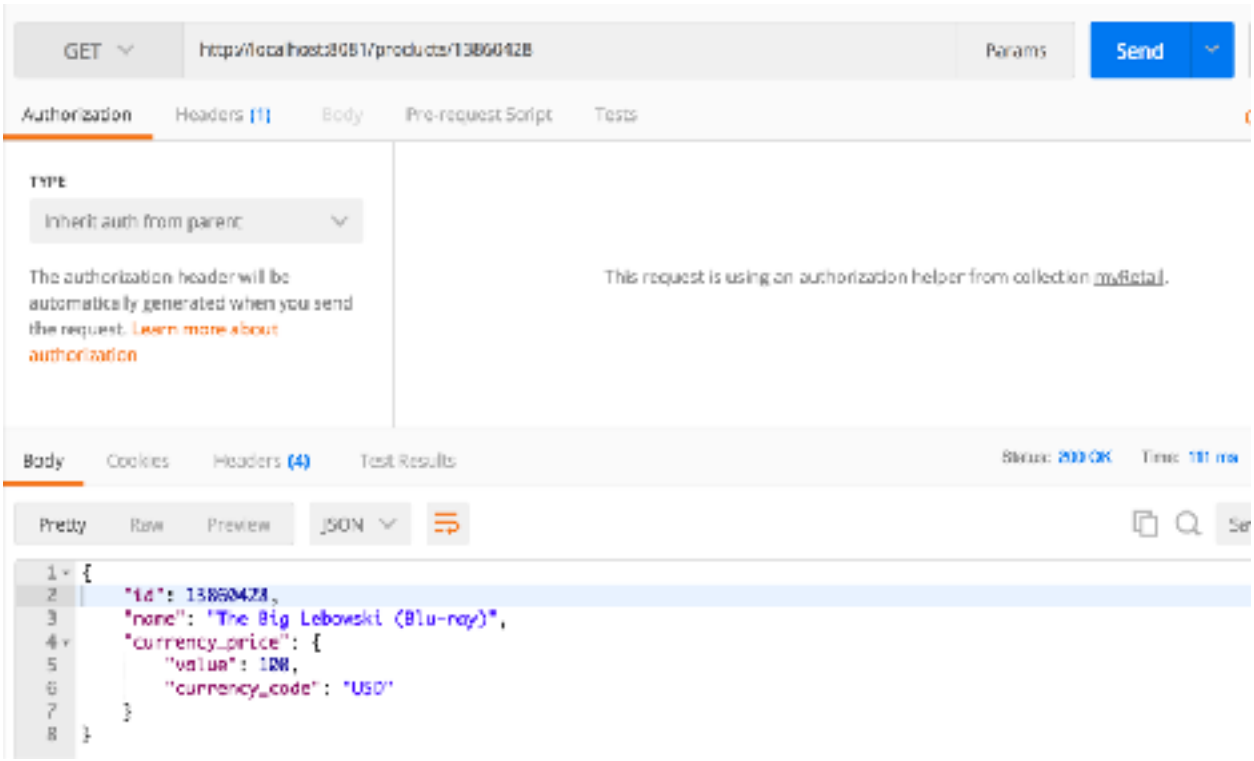
2. PUT: to update the price for a given product id



2.1 Updated Price: Response of the PUT API



2.2 Updated Data from the GET product API



_____Current Automated Test Coverage: