**Shopping Product Microservice**

**(Spring Boot, Active MQ and Database)**

**How to run:**

**No setup required to run this project**. Maven should be configured.

Should have access to Mvn repository. No target is shipped.

1. Download/Clone from https://github.com/bairagis/mqdemo.git
2. Unzip mqdemo.zip in local directory
3. Go to that directory and run: **mvn spring-boot:run**
4. The project will start on 8092 on localhost
5. Postman is used to test

GET <http://localhost:8092/api/v1/products>

POST <http://localhost:8092/api/v1/product/publish>

Sample :

{

"productName":"iPad4",

"quantity":"5",

"productCode":"IPAD",

"customerId":"S456545"

}

**Sample Screenshots:**

1. Get No records found

![A screenshot of a cell phone

Description automatically generated]()

1. Post product message

![A screenshot of a cell phone

Description automatically generated]()

1. Error Scenario 2: Product Name Null

![A screenshot of a cell phone

Description automatically generated]()

1. Get All Products

![A screenshot of a cell phone

Description automatically generated]()

1. Code Structure, Unit Testing, Code Coverage

![A screenshot of a social media post

Description automatically generated]()

**Components and Flow**

1. The microservice controller ProductMessageController exposes two end points to get products and another to publish product to DB via Active MQ
2. Embedded Active MQ is used from Spring Starter
3. Embedded H2 database is used
4. Database Table PRODUCTS is auto generated during start-up
5. When user POST the Product message it is sent to embedded queue destination
6. Saved data into H2 database table (products).
7. Written Junit test cases for to have more than 85% code coverage.
8. Negative Test cases checks Business exceptions like product name mandatory etc. as well
9. All DB and JMS are Transactional to handle Rollback for fault tolerance
10. Global Error handler and Payload validation is developed

**Flow**

1. GET endpoint displays all products that are present in DB
2. If there is no product in DB, NOT\_FOUND error is sent back
3. All DB calls are Transactional and rollback keep the message at Queue for redelivery
4. This project exposes two micro services rest APIs

Get all product list <http://localhost:8092/api/v1/products> http method : GET

it returns all product list from embedded DB (only for demo).

POST product message payload: http://localhost:8092/api/v1/product/publish http method: POST

When this request come to rest controller, controller method push this product entity in JMS Active MQ(queue named “cart queue”) in String format ( product json) and return “Submitted” if success.

1. Subsequent GET call fetches the records from DB
2. Junit Spring Boot test class is written with success and failure test cases.

Sudip Bairagi, IBM Singapore

16/06/2020