**Shopping Cart Microservice with MQ**

**Completed project is uploaded in GitHub**

<https://github.com/sathishpk-hub/ShoppingCart_Microservice_SprBoot.git>

**Used Skills in this project:**

Java, SpringBoot, MicroServices, REST WebServices, JPA, Hibernate, JMS-ActiveMQ, Eureka Server, MySQL Database, Maven, JSP, Jquery, GitHub, SpringBoot Embedded Tomcat Server, STS-Eclipse IDE

**What are the task completed in this project:**

1. Created microservice-server (Eureka Server) and running at localhost with port 1111
2. Created 1st micro-service (producer-service) to display and choose shopping cart items/products.
3. Installed ActiveMQ and start in default port 8161 with credentials admin/admin.
4. Once choose product, pushing that particular product into ActiveMQ queue.
5. Created 2nd micro-service (consumer-service) to read message from ActiveMQ
6. Auto created database (productdb) and table (product) using SpringBoot JPA + Hibernate Dialect
7. Saved data into MySql table (product)
8. Written Junit test cases for above 2 micro-services and successfully passed the same.
9. Attached screenshots for all and pushed the same into GitHub
10. Registered (connected) both above micro-services with MicroService(Eureka Server port:1111)

**Brief Explanation about How completed this project:**

1. Created SpringBoot application for MicroService Eureka Server.

* New SpringBoot application created using SpringBoot-web in STS eclipse.
* @EnableEurekaServer for this application
* Mention following values in application.properties file as application name, server port, etc.
* Note: Running this Eureka Server in port 1111

1. Created 1st micro-service (producer-service) to display and choose shopping cart items/products.

* Created shopping cart application using technologies SpringBoot + MicroService +

REST WebService

* At Backend, REST Webservices will call and fetch the results of list of products for JSP UI page. (using stub repository as three static products for testing purpose)
* Once chosen product, create new Queue with name “test-queue” (using @Autowired Queue) and push the message into ActiveMQ.
* Written junit test cases and results are success.
* Note: this Eureka Client and register with Eureka Server port:1111 in application.properties file.
* Also running this producer micro-service at separate port 2222
* Register ActiveMQ broker url at same application.properties file (to pushe message from producer to ActiveMQ)

1. Created 2nd micro-service (consumer-service) to consume above products from producer micro-server(from another micro service)

* Created another application using technologies SpringBoot + MicroService +

REST WebService + JSP

* Written REST Webservice to consume products from producer-microservice
* Customer will hit URL in browser and Inventory page (list of products) will be displayed then internally (at backend) consumer micro-service will call producer micro-service and fetch the list of products and displaying in JSP UI page
* Customer choose any of the product in UI, then internally (at backend) consumer microservice will call producer microservice and push the message(here product) into ActiveMQ using one queue name “test-queue” (using @Queue annotation)
* Making this microservice project as “@EurekaDiscoveryClient”
* Mention required values in application.properties file as application.name , server.port, eureka.server.url, activeMq.url, etc.
* Note: this micro-service running on default port 8080
* MessageQueue listner will listen this messages from ActiveMQ (using JMS Listener destination/Queue name)
* Using SpringBoot JPA repository to save this received message into mySql database at ‘product’ table

1. ActiveMQ installation and running

* Installed ActiveMQ for windows 7 64bit
* Started with command goto installed bin directory > activemq.bat
* Its Running with default port 8161 by logged in default credentials admin/admin

1. Auto create database and table

* Installed MySQL DB 8.0 Command Line Client
* Set PATH & Class PATH in environmental variables
* Open cmd prompt and type mysql, then type command “mysql –u root –p”, will ask password, Set new password, then logged in.
* Its running on default port 3306
* Initially no more custom db and tables (I haven’t created any database and table)
* Note: creating database and tables automatically Using SpringBoot JPA + Hibernate Dialect and mentioned required values in consumer-service application.properties.
* spring.datasource.url = jdbc:mysql://localhost:3306/productdb?useSSL=false&createDatabaseIfNotExist=true, spring.jpa.generate-ddl=true, spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.MySQL5InnoDBDialect, spring.jpa.hibernate.ddl-auto = create

**How to run these micro-service projects:**

**Pre-setup to run this project:**

1. Download and Install ActiveMQ and start in default port 8161.

* Open cmd prompt
* Type command: D:\softwares\apache-activemq-5.15.9\bin\win64 > activemq.bat
* Go to browser with url <http://localhost:8161/admin/queues.jsp>
* It will prompt for credentials and give default admin/admin
* It will indicate how many messages in queue and producers and consumers details.

1. Download and Installed MySQL and start in default port 3306.

* Open cmd prompt
* Type command: mysql –u root –p
* Enter password
* Show databases;

1. Download microservice-server project from GitHub

* Update maven project and resolve usual setup errors if any
* Right click this project and Run As > Spring Boot App
* Once successfully started this Eureka server on console, then goto browser and hit eureka server url <http://localhost:1111/>
* Initially it will not display any other services which is 0 only.

1. Download producer-service project from GitHub

* Update maven project and resolve usual setup errors if any
* Right click this project and Run As > Spring Boot App
* Once successfully started this Eureka client on console, then goto browser and refresh same eureka server url <http://localhost:1111/>
* Now you can see 1st micro-service is running which is displaying with name “product-microservie” (I have given this name in application.properites)

1. Download consumer-service project from GitHub

* Update maven project and resolve usual setup errors if any
* Right click this project and Run As > Spring Boot App
* Once successfully started this Eureka client on console, then goto browser and refresh same eureka server url <http://localhost:1111/>
* Now you can see 2nd micro-service is running which is displaying with name “consumer-microservie” (I have given this name in application.properites)
* Note. Now running both 1st and 2nd micro-services in port 2222 & 8080, same displaying in Eureka server url
* All pre setup is done now. Will test application in below steps

**Run this project in browser UI:**

1. Goto browser and hit url <http://localhost:8080/consume/productList>

* This will show list of products(cart items) in browser UI page (actually this is ListInventory.jsp page)
* You can see 3 products namely Apple IPhone 8 plus, Samsung Note 10 & Huawei P30 Pro
* Note: At this point, no more message sending/receiving.
* Then choose any of the product by click link “Choose”, now it will show different url in browser <http://localhost:8080/consume/products/2>
* It will redirect to next page “Selection.jsp” page as indicate what are all the item you selected in previous page.
* Note:
  + At this point (once you selected product), at backend that selected particular product will be pushed into message queue ActiveMQ.
  + Then, that pushed messages will be read from ActiveMQ and printing in Eclipse “Console” page of “consumer-service” application.
  + Then, new database “productdb” & new table “product” will be created automatically in mysql database
  + Then, that received message will be saved into “product” table.

**Run by Junit Test Cases:**

1. We can run by both microservices producer & consumer by Junit test cases as well

* Goto any checkout project and Right click > Run As > Maven Test
* (OR) Right click > Run As > Junit Test
* It will run test cases and produce the results in “Console” eclipse with Test cases success. No failures.

**Please note:** for all these steps, I have attached screenshots as well for reference under Github folder “screenshots\_running\_servers\_inputOutput”

**Completed By : Sathish Packirisamy Kumar, Singapore**

--- END ----