Spring Boot + RabbitMQ Tutorial (Producer and Consumer)

Implementation of Producer Spring Boot App

The producer app will produce messages that comes from the Restful Web API by POST method. Since we don't have a UI for the this app, we can use Postman or SwaggerUI to POST the message. We will use SwaggerUI in this tutorial.

Messages aren't produced directly to the `Queues`. Firstly, they go to a `Exchange` with a routing key. The `Exchange`then distributes copies of messages to `Queues`. After that, consumers can consume the messages. We will use`Topic Exhange` type that can routes messages to multiple `Queues` by matching a routing key to a pattern.

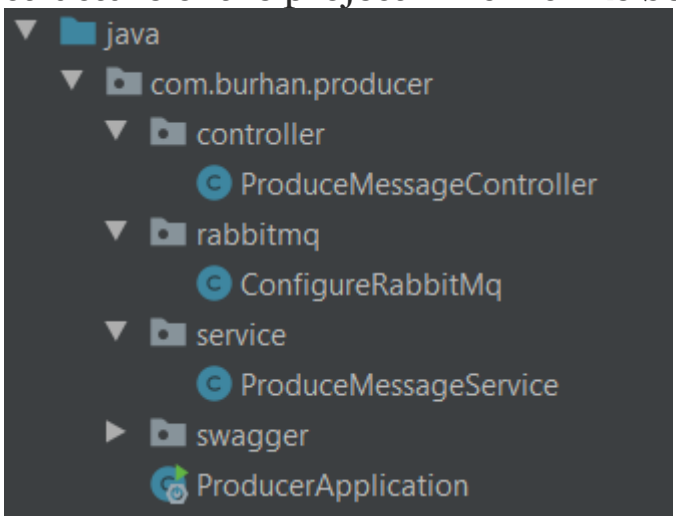
Firstly, we need to create a Spring Boot Maven app with Spring Initializr by choosing the following dependencies.

- **Spring Web**

- **Spring for RabbitMQ**
- **Lombok**

Also, we need to add following **SwaggerUI** dependencies to the`pom.xml`.

Before the implementation, I want to show you the package structure of the project which is like below.



Firstly, we will implement the `ConfigureRabbitMq` class. It will handle the configuration of the RabbitMq like below.

`ConfigureRabbitMq.java`

Then, we will implement the our service layer class which is`ProduceMessageService.java` like below.

Then, we will implement our controller layer class which is `ProduceMessageController.java` like below.

Lastly, we will implement our configuration class which is for the Swagger UI.

`ConfigureSwagger.java`

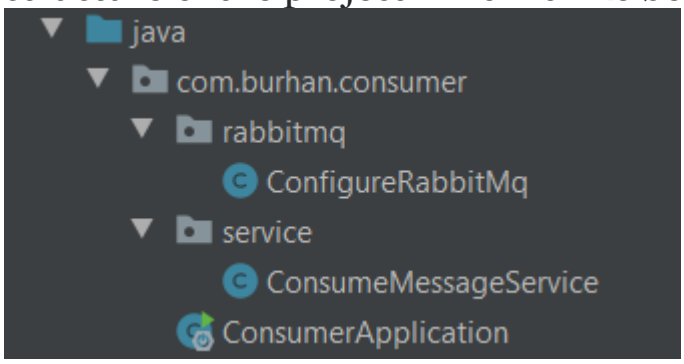
Implementation of Consumer Spring Boot App

The Consumer app will listen the queue and consume the messages from the queue. Also consumed messages will be printed to console.

Firstly, we need to create a Spring Boot Maven app with Spring Initializr by choosing the following dependencies.

- **Spring for RabbitMQ**
- **Lombok**

Before the implementation, I want to show you the package structure of the project which is like below.



Firstly, we will implement the `ConfigureRabbitMq` class again for the consumer. It will handle the configuration of the RabbitMq like below.

`ConfigureRabbitMq.java`

Then, we will implement our consumer service class which is `ConsumeMessageService.java` like below.

Running the RabbitMQ Docker Image

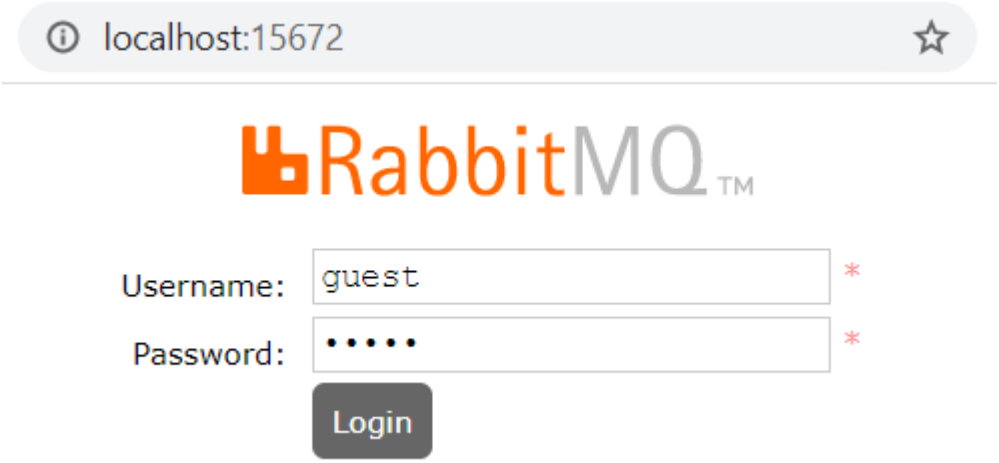
Firstly, we need to install the Docker Desktop application for running the RabbitMQ instance. After the installation, we need to pull the RabbitMQ Docker image using the following command from the terminal.

After pull process, we can run the our RabbitMQ docker image using the following command from the terminal.

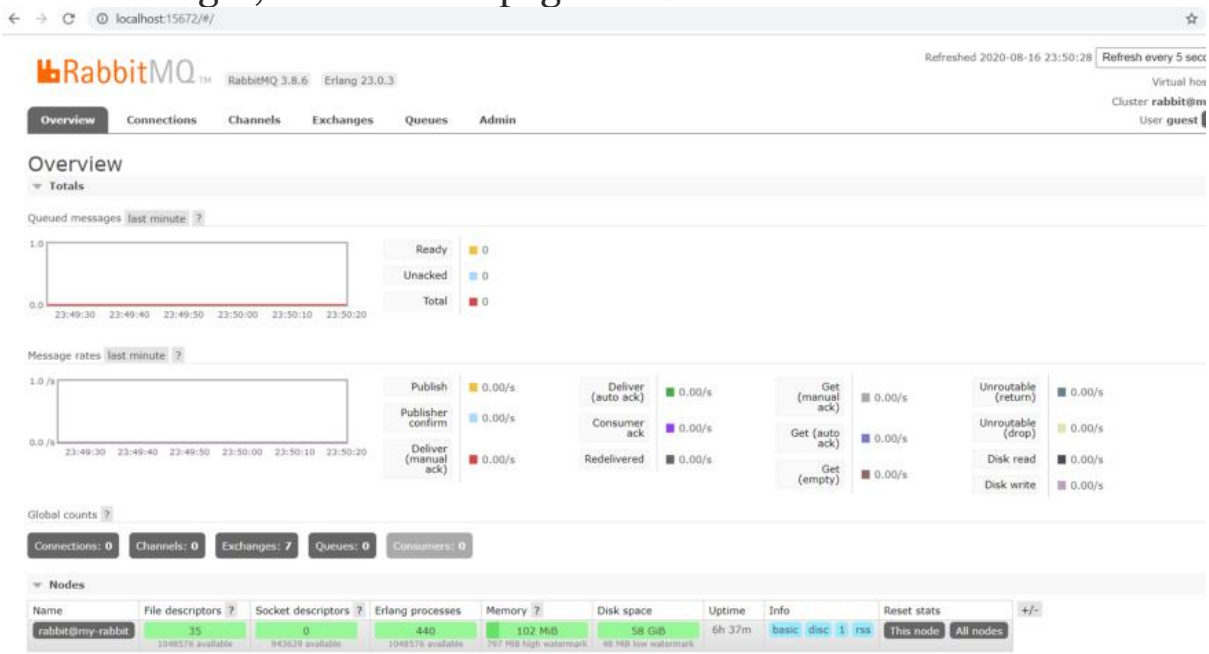
docker run --rm -it --hostname my-rabbit -p 15672:15672 -p 5672:5672 rabbitmq:3-management

After running the Docker image, we can see the RabbitMQ management UI from the ***http://localhost:15672***.

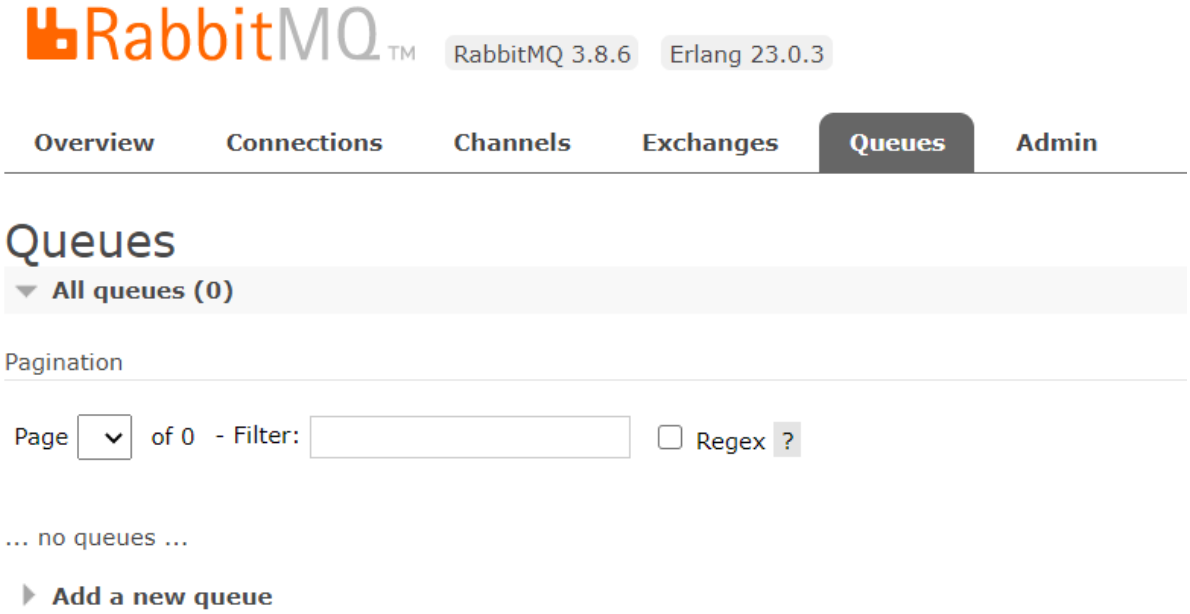
The default username and password is `guest`



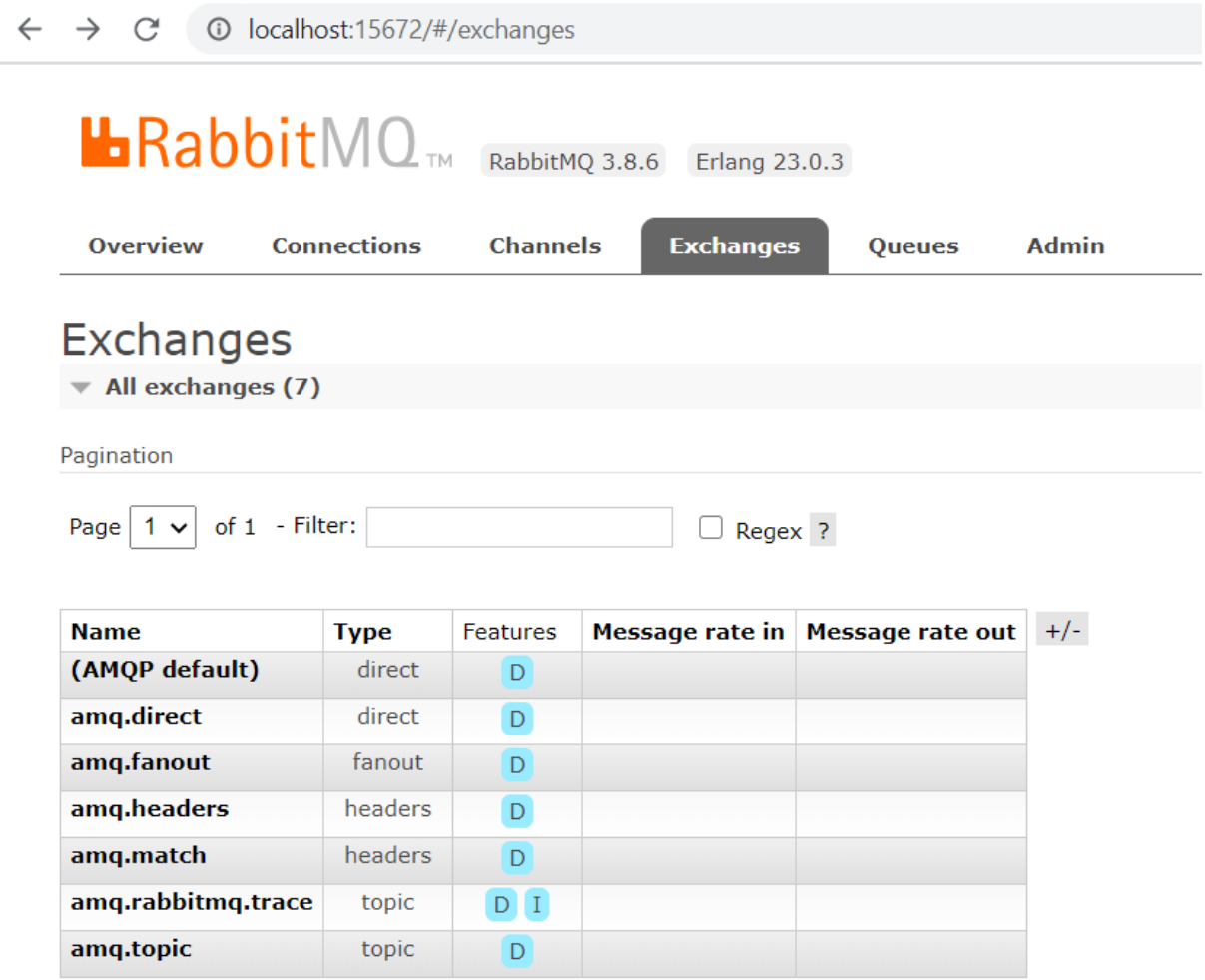
After the login, we can see a page like below.



There isn't an exchange or a queue because we will create them dynamically soon by running producer Spring Boot application.



There is no queue for now.



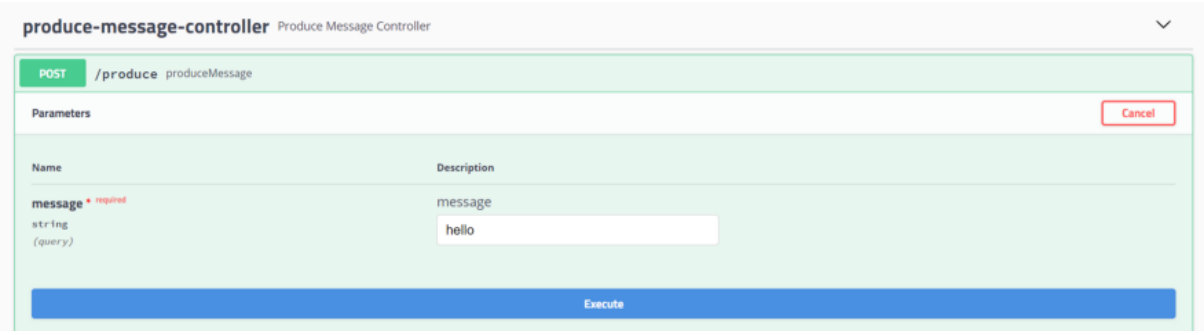
There is no created exchange for now.

Let's Try It Out

Firstly, be sure the docker image is running. After that we need to run the producer Spring Boot app. Then, we need to run the consumer Spring Boot app. With this processes, producer app will create the `myQueue` and `myExchange`. After that, consumer app will listen the queue.

To produce a message, go to localhost:8080/swagger-ui.html and click **produce-message-controller -> POST -> Try it out** buttons.

After that, enter a **message** to the message text box and click **Execute**.



We can see that the message "**hello**" is logged in the console of Consumer Spring Boot application.

```
2020-08-17 00:14:36.175  INFO 9976 --- [enerContainer-1] c.b.c.service.ConsumeMessageService      : Consumed Message: hello
```