RabbitMQ Study Notes:

Working Example:

https://springframework.guru/spring-boot-messaging-with-rabbitmq/

1. use docker-compose up to start RabbitMQ server
2. run producer service from rabbitmq-solution-source-code
3. run consumer service.
4. Send a post request from Postman.

Let’s go back to the microservices and run the Producer Service and Consumer Service respectively.

To test the flow of messaging with RabbitMQ, you can use [Postman](https://www.postman.com/).

1. In Postman, select **POST** from the drop-down list.
2. Then type the

localhost:9091/api/v1/user

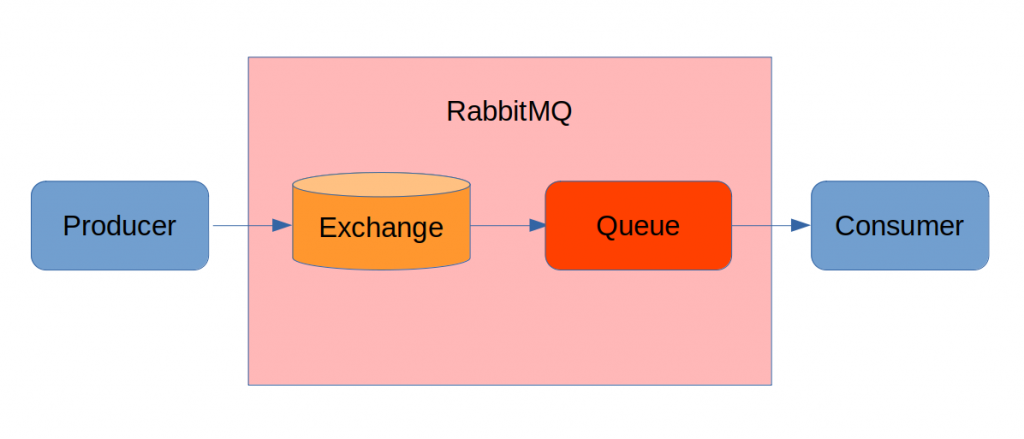
 URL.

1. Click the **Body** tab.
2. Select the **raw** radio button and then select **JSON** from the drop-down list.
3. Type

{ "userId": "1","userName": "Mark" }

 to

Message Broker: support several message patterns.



Spring Side Note: Autowired on property; on setter and on constructor;

Key Components for RabbitMQ: AMQP

Create container network to support cluster.

Use yml or property file to get values with spring @Value annotation

Basic Structure: Producer II Exchange|Queue| Consumer

RabbitTemplate; RabbitListenerConfigurer; ConnectionFactory;

BindingBuilder: queue, exchange, routingKey

Sender or Producer only need Exchange and RoutingKey

Consumer need queue, exchange, routingKey to listen

<https://www.baeldung.com/rabbitmq>

MQ console: guest/guest

<http://localhost:15672/#/>

used docker-compose to get RabbitMQ up with Zipkin.

The docker-compose file is under rabbitmq-tutorials project.

And one under apps2022:

<https://spring.io/guides/gs/messaging-rabbitmq/#scratch>

Will use this to send the message in rabbitMQ

CommandLineRunner is a simple Spring Boot interface with a run method. Spring Boot will automatically call the run method of all beans implementing this interface after the application context has been loaded.

Most console applications will only have a single class that implements CommandLineRunner. If your application has multiple classes that implement CommandLineRunner, the order of execution can be specified using [Spring's @Order annotation](https://www.baeldung.com/spring-order).

RabbitMQ Commands:

# run a standalone instance

docker network create rabbits

docker run -d --rm --net rabbits --hostname rabbit-1 --name rabbit-1 rabbitmq:3.8

docker logs rabbit-1

docker exec -it rabbit-1 bash ( get inside the container bash )

###rabbitmqctl ( to manage RabbitMQ cli way) or from management console ( localhost:15672)

### rabbitmq-plugins ( to manage plugins )

Docker run -d –-rm -–net rabbits -p 8080:15672 -–hostname rabbit-1 -–name rabbit-1 rabbitmq:3.8

# how to grab existing erlang cookie

docker exec -it rabbit-1 cat /var/lib/rabbitmq/.erlang.cookie

# clean up

docker rm -f rabbit-1

Diagram

Description automatically generated

Code Example:

<https://blog.devgenius.io/part-4-how-to-configure-messaging-with-rabbitmq-in-a-spring-boot-application-a73e2453da95>

start RabbitMQ with this file: docker-compose.yml to start RabbitMQ.

SideNotes: springboot profile

**application-dev.properties**

**application-test.properties**

**application-prod.properties**

**application.properties:** spring.profiles.active=dev

Next Topic to study:

RabbitMQ with network and clusters.