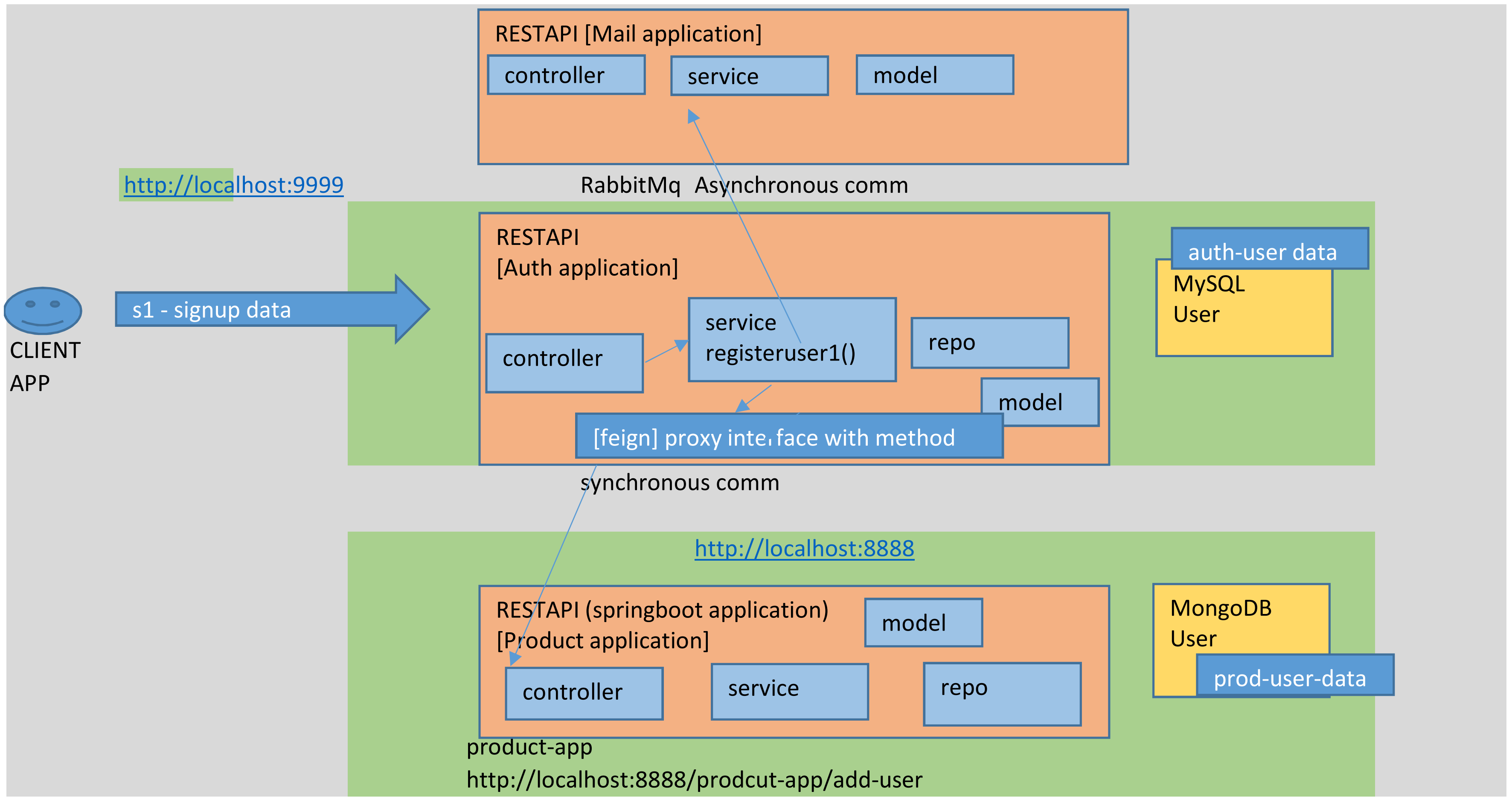
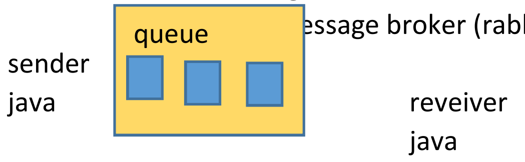


synch

step by step
sender waits till getting reply from receiver
both sender and receiver application must be alive

asynch

parallel
sender wont wait for getting any reply from receiver
Sender can send message to receiver even receiver is not live
essage broker (rabbitmq)



synch

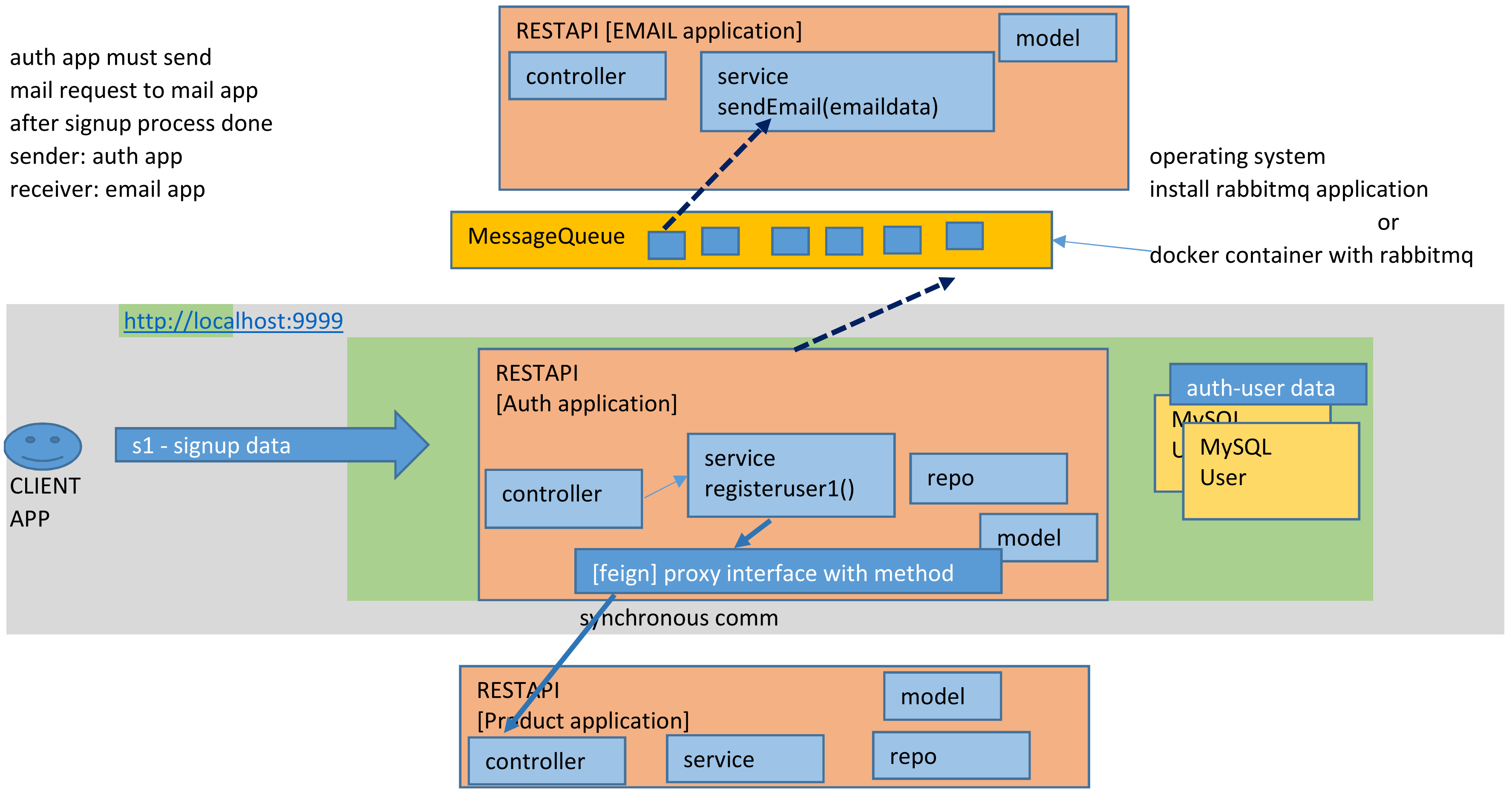
step by step
waits till get reply from receiver
sender connects directly to receiver

both sender and recevier must be active
feign client

asynch

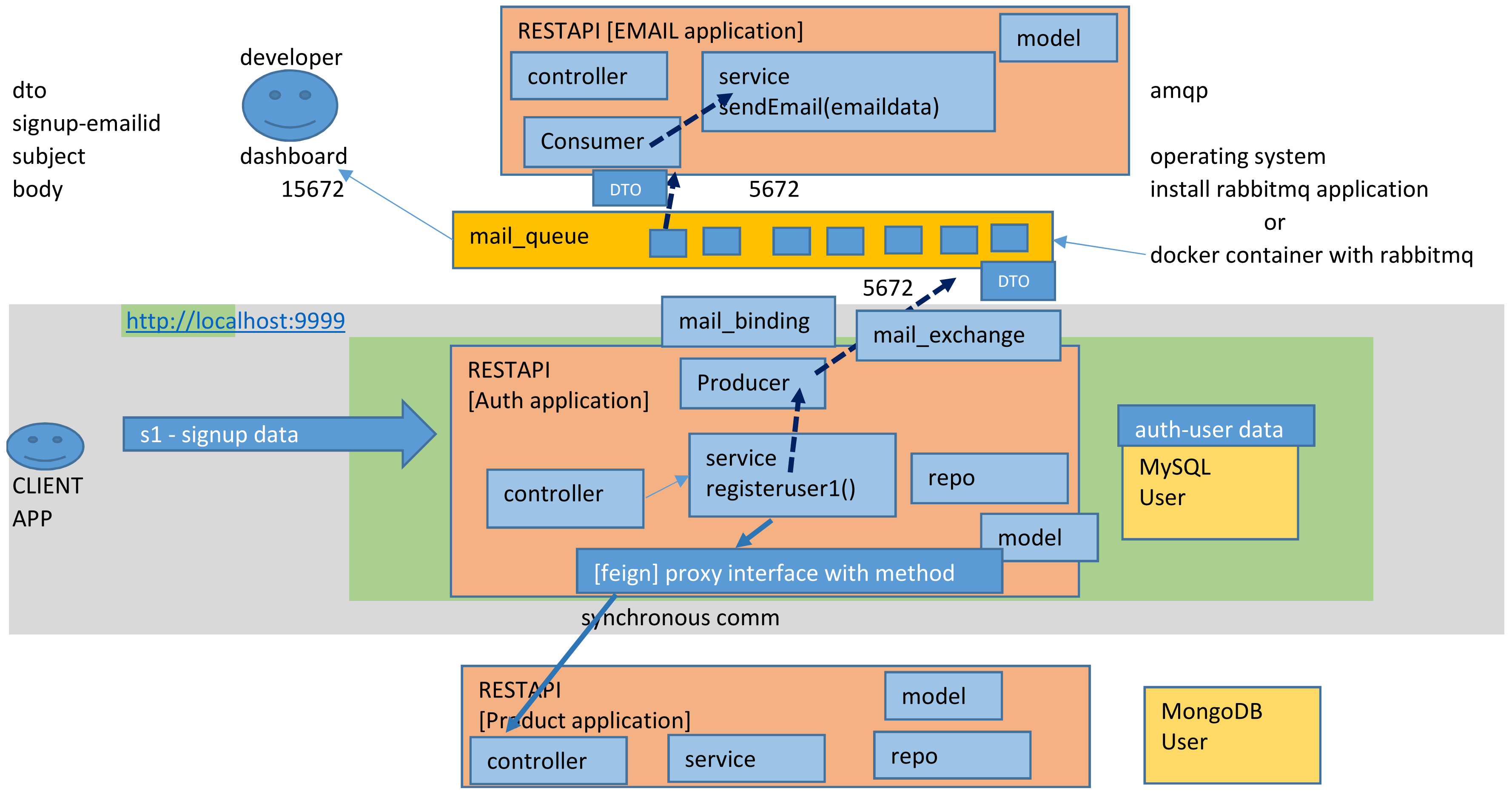
parallel
wont wait for getting reply from receiver
sender and receiver not connected directly
connect thru message broker
sender can send message even receiver is not active
rabbitmq

auth app must send
mail request to mail app
after signup process done
sender: auth app
receiver: email app



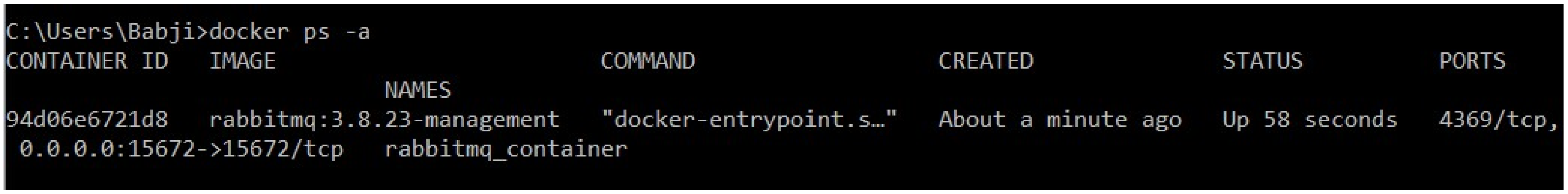
Auth app : Sender / Publisher / Producer / Source

Prod app: Receiver / Subscriber / Consumer / Destination



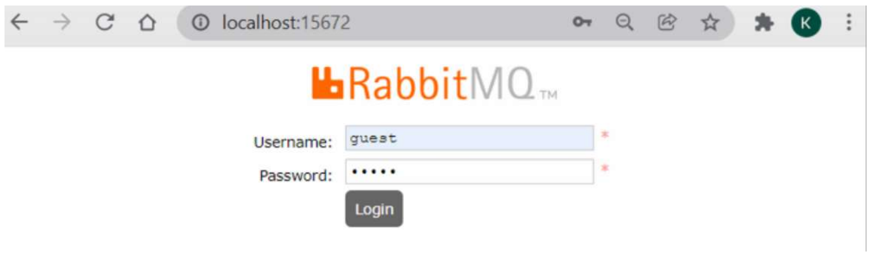
Start rabbitmq container

docker run --name rabbitmq_container -p 5672:5672 -p 15672:15672 rabbitmq:3.8.23-management



```
C:\Users\Babji>docker ps -a
CONTAINER ID   IMAGE                          COMMAND                  CREATED              STATUS          PORTS
                          NAMES
94d06e6721d8   rabbitmq:3.8.23-management   "docker-entrypoint.s…"   About a minute ago   Up 58 seconds   4369/tcp,
 0.0.0.0:15672->15672/tcp   rabbitmq_container
```

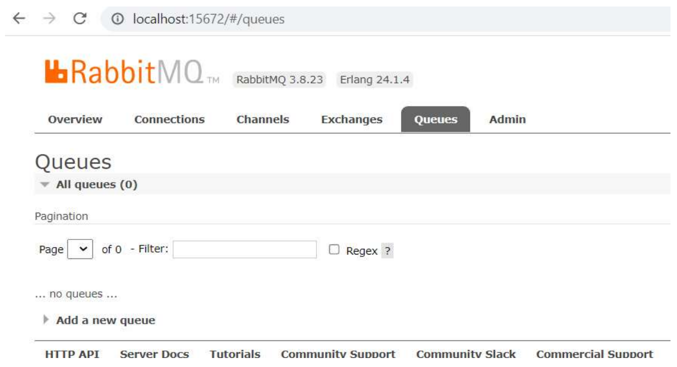
login to dashboard http://localhost:15672



localhost:15672

Username: guest

Password: •••••



localhost:15672/#/queues

RabbitMQ 3.8.23 Erlang 24.1.4

Queues

All queues (0)

Pagination

Page of 0 - Filter: Regex ?

... no queues ...

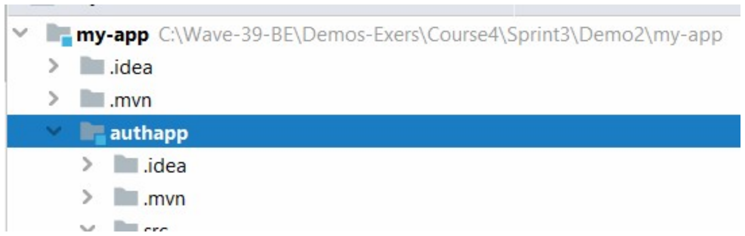
Add a new queue

Steps to create asynchronous communication between auth-app and emailap

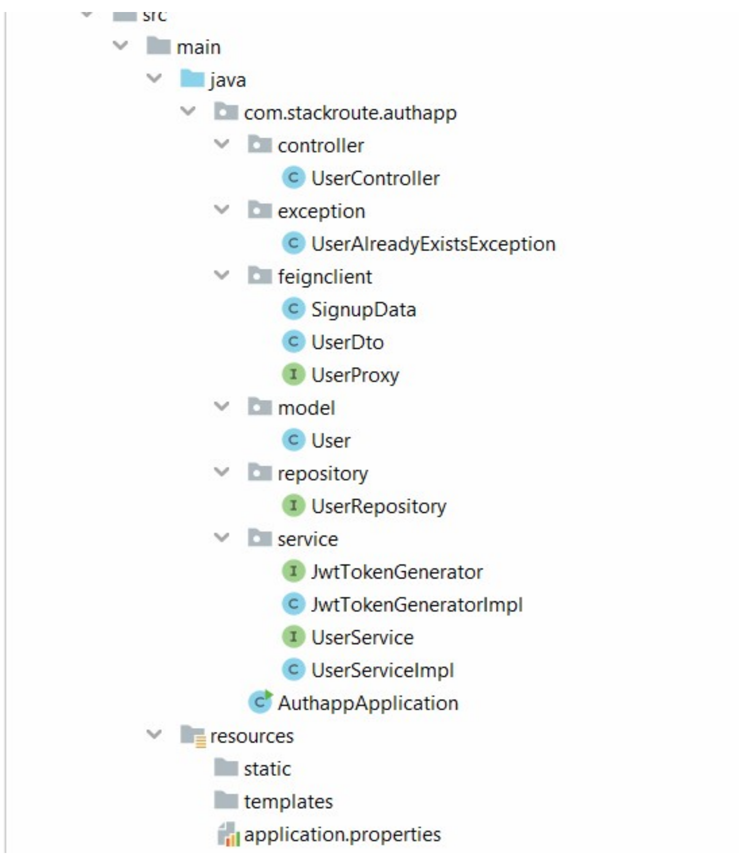
sender	auth-app
receiver	email-app

Make sure auth-app is ready with all functionalities

copy C4S2 to C4S3



```
my-app C:\Wave-39-BE\Demos-Exers\Course4\Sprint3\Demo2\my-app
  .idea
  .mvn
  authapp
    .idea
    .mvn
```

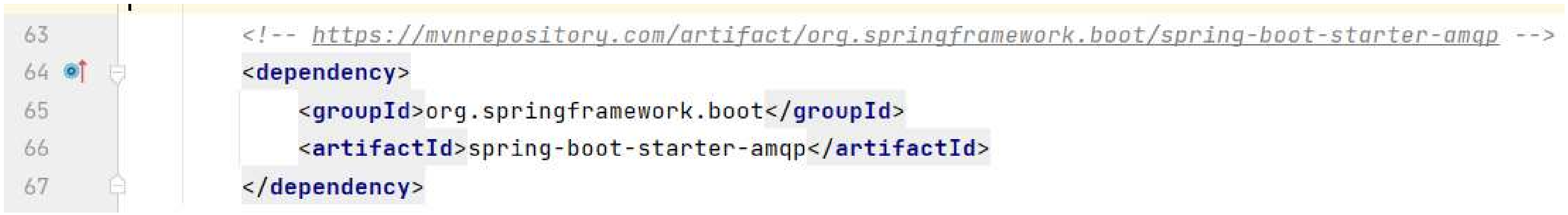


```
src
  main
    java
      com.stackroute.authapp
        controller
          UserController
        exception
          UserAlreadyExistsException
        feignclient
          SignupData
          UserDto
          UserProxy
        model
          User
        repository
          UserRepository
        service
          JwtTokenGenerator
          JwtTokenGeneratorImpl
          UserService
          UserServiceImpl
        AuthappApplication
    resources
      static
      templates
      application.properties
```

Steps to make auth-app as sender

Step 1 add required dependency in pom

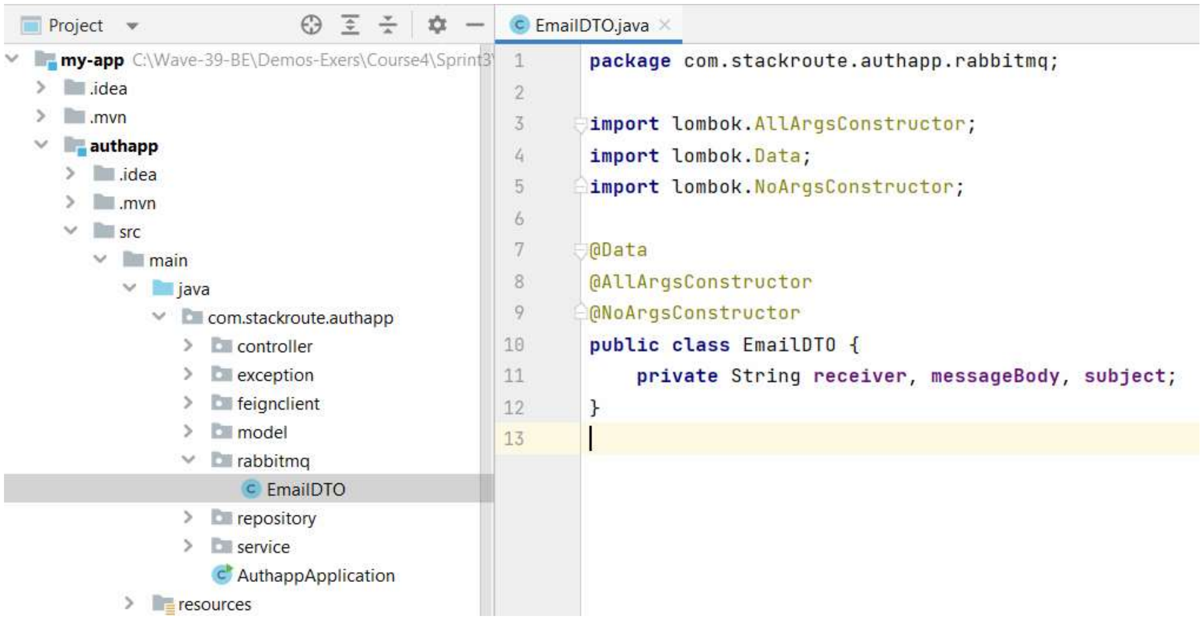
```
<dependency>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-starter-amqp</artifactId>
</dependency>
```



```
<!-- https://mvnrepository.com/artifact/org.springframework.boot/spring-boot-starter-amqp -->
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-amqp</artifactId>
</dependency>
```

Step 2 Create DTO class with required fields

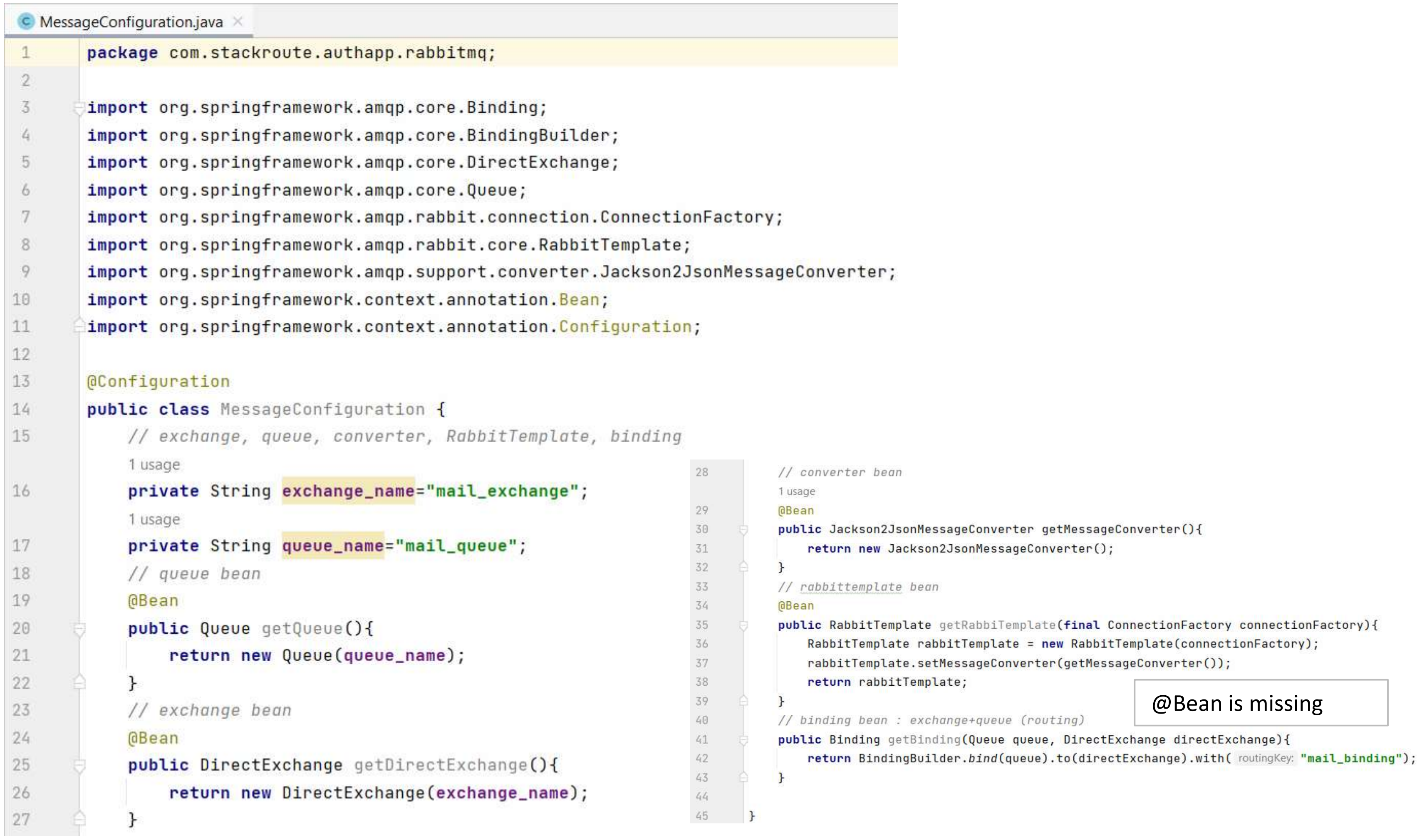
receiver, messageBody,subject



```java
package com.stackroute.authapp.rabbitmq;

import lombok.AllArgsConstructor;
import lombok.Data;
import lombok.NoArgsConstructor;

@Data
@AllArgsConstructor
@NoArgsConstructor
public class EmailDTO {
    private String receiver, messageBody, subject;
}
```

Step 3 Cretae all required beans

- Exchange
- Queue
- RabbitTemplate
- Binding
- Conerter



MessageConfiguration.java

```java
package com.stackroute.authapp.rabbitmq;

import org.springframework.amqp.core.Binding;
import org.springframework.amqp.core.BindingBuilder;
import org.springframework.amqp.core.DirectExchange;
import org.springframework.amqp.core.Queue;
import org.springframework.amqp.rabbit.connection.ConnectionFactory;
import org.springframework.amqp.rabbit.core.RabbitTemplate;
import org.springframework.amqp.support.converter.Jackson2JsonMessageConverter;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;

@Configuration
public class MessageConfiguration {
    // exchange, queue, converter, RabbitTemplate, binding
    1 usage
    private String exchange_name="mail_exchange";
    1 usage
    private String queue_name="mail_queue";
    // queue bean
    @Bean
    public Queue getQueue(){
        return new Queue(queue_name);
    }
    // exchange bean
    @Bean
    public DirectExchange getDirectExchange(){
        return new DirectExchange(exchange_name);
    }
```

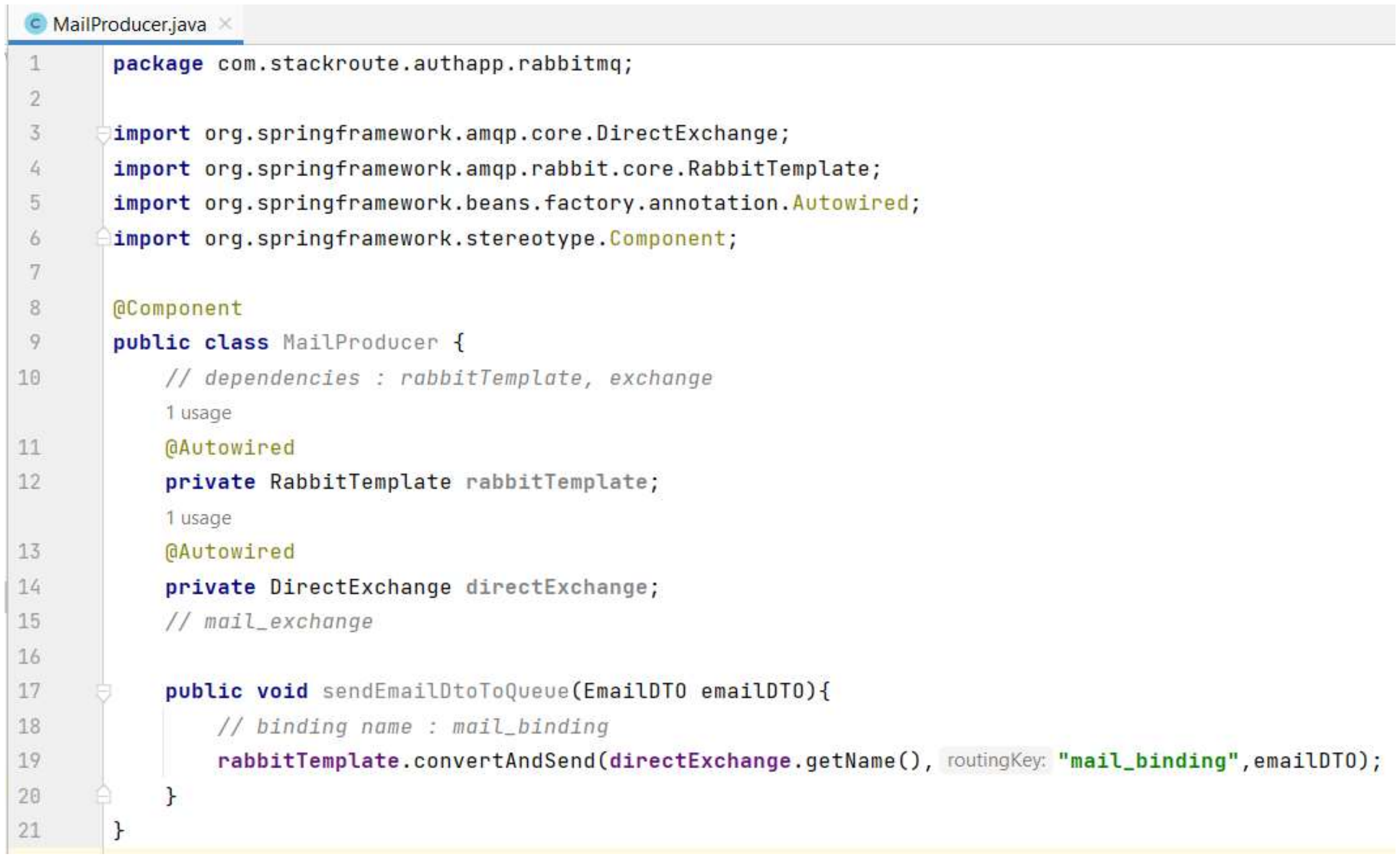
```java
    // converter bean
    1 usage
    @Bean
    public Jackson2JsonMessageConverter getMessageConverter(){
        return new Jackson2JsonMessageConverter();
    }
    // rabbittemplate bean
    @Bean
    public RabbitTemplate getRabbiTemplate(final ConnectionFactory connectionFactory){
        RabbitTemplate rabbitTemplate = new RabbitTemplate(connectionFactory);
        rabbitTemplate.setMessageConverter(getMessageConverter());
        return rabbitTemplate;
    }
    // binding bean : exchange+queue (routing)
    public Binding getBinding(Queue queue, DirectExchange directExchange){
        return BindingBuilder.bind(queue).to(directExchange).with( routingKey: "mail_binding");
    }

}
```

@Bean is missing

Step 4 Create producer

Define a method to send maildto object to queue using rabbittemple, exchange beans

```
@Component
class Producer
{
rabbitTemple, exchange
method()
}
```

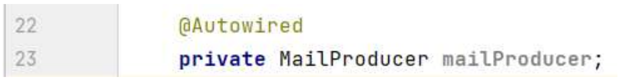


MailProducer.java

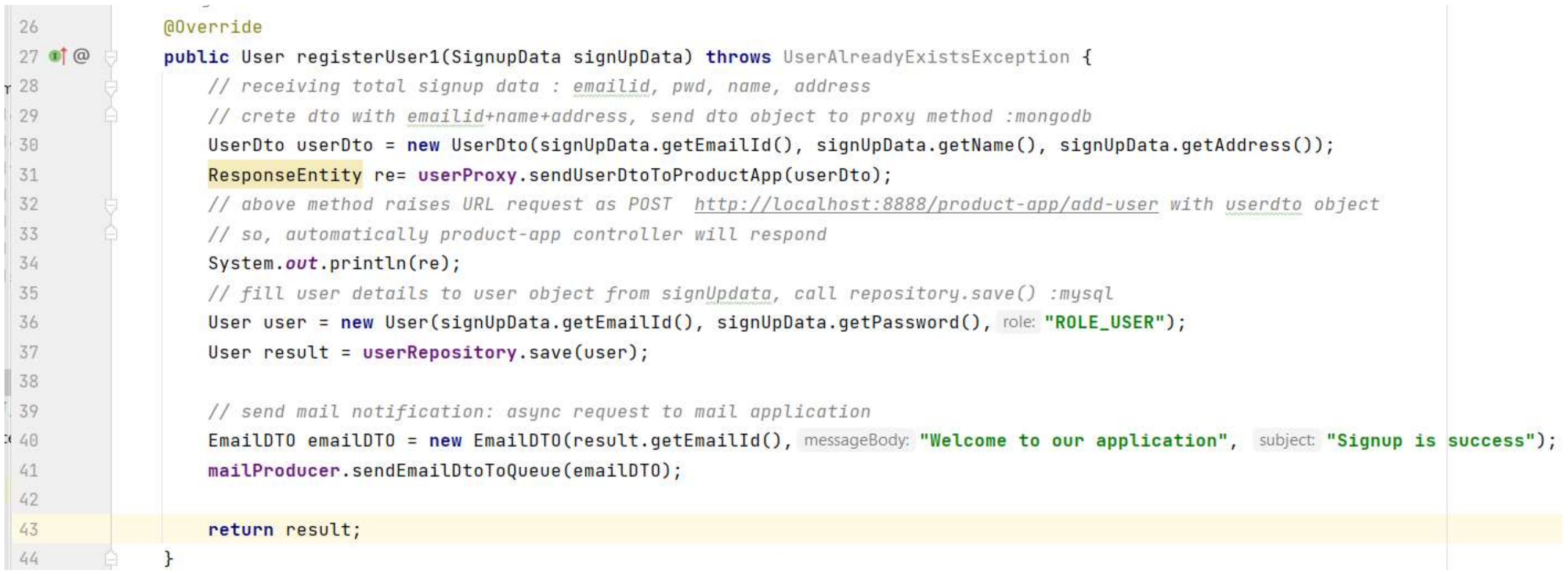
```java
package com.stackroute.authapp.rabbitmq;

import org.springframework.amqp.core.DirectExchange;
import org.springframework.amqp.rabbit.core.RabbitTemplate;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Component;

@Component
public class MailProducer {
    // dependencies : rabbitTemplate, exchange
    1 usage
    @Autowired
    private RabbitTemplate rabbitTemplate;
    1 usage
    @Autowired
    private DirectExchange directExchange;
    // mail_exchange

    public void sendEmailDtoToQueue(EmailDTO emailDTO){
        // binding name : mail_binding
        rabbitTemplate.convertAndSend(directExchange.getName(), routingKey: "mail_binding",emailDTO);
    }
}
```

Step 5 Call producer in servicelogic

Inject product as dependency into service impl class



```java
    @Autowired
    private MailProducer mailProducer;
```

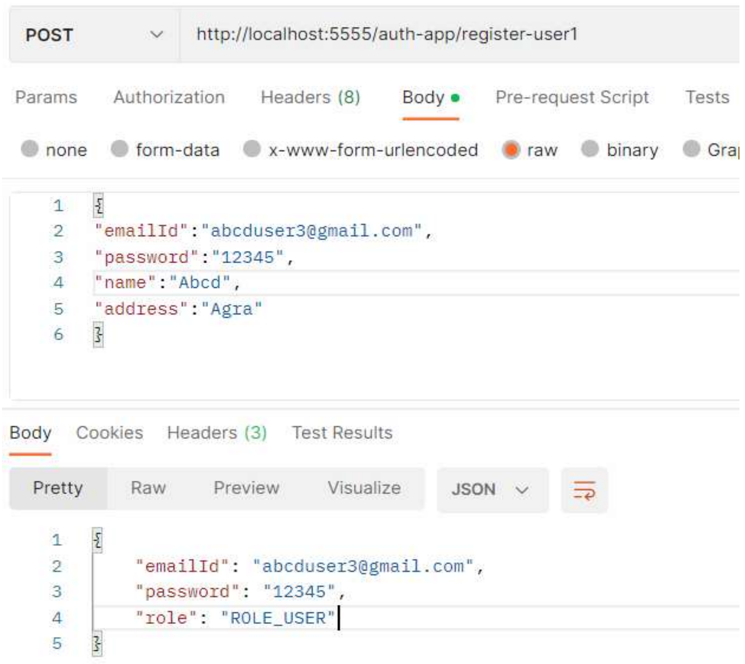


```java
    @Override
    public User registerUser1(SignupData signUpData) throws UserAlreadyExistsException {
        // receiving total signup data : emailid, pwd, name, address
        // crete dto with emailid+name+address, send dto object to proxy method :mongodb
        UserDto userDto = new UserDto(signUpData.getEmailId(), signUpData.getName(), signUpData.getAddress());
        ResponseEntity re= userProxy.sendUserDtoToProductApp(userDto);
        // above method raises URL request as POST  http://localhost:8888/product-app/add-user with userdto object
        // so, automatically product-app controller will respond
        System.out.println(re);
        // fill user details to user object from signUpdata, call repository.save() :mysql
        User user = new User(signUpData.getEmailId(), signUpData.getPassword(), role: "ROLE_USER");
        User result = userRepository.save(user);

        // send mail notification: async request to mail application
        EmailDTO emailDTO = new EmailDTO(result.getEmailId(), messageBody: "Welcome to our application",  subject: "Signup is success");
        mailProducer.sendEmailDtoToQueue(emailDTO);

        return result;
    }
```

Sender is ready
Make sure rabbitmq application/container is running

how to check whether sender sending messages into queue

run application (all apps)
perform signup request in postman



POST http://localhost:5555/auth-app/register-user1

Params | Authorization | Headers (8) | Body | Pre-request Script | Tests

none | form-data | x-www-form-urlencoded | raw | binary | Gra

```
{
"emailId":"abcduser3@gmail.com",
"password":"12345",
"name":"Abcd",
"address":"Agra"
}
```

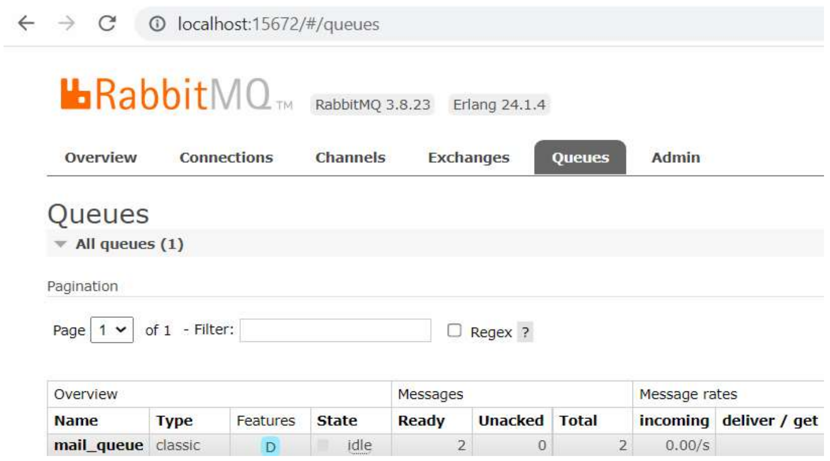
Body | Cookies | Headers (3) | Test Results

Pretty | Raw | Preview | Visualize | JSON

```
{
    "emailId": "abcduser3@gmail.com",
    "password": "12345",
    "role": "ROLE_USER"
}
```

Make sure
user a/c created in auth-app
user a/c created in product-app

objects waiting in queue



localhost:15672/#/queues

RabbitMQ 3.8.23 | Erlang 24.1.4

Overview | Connections | Channels | Exchanges | Queues | Admin

Queues

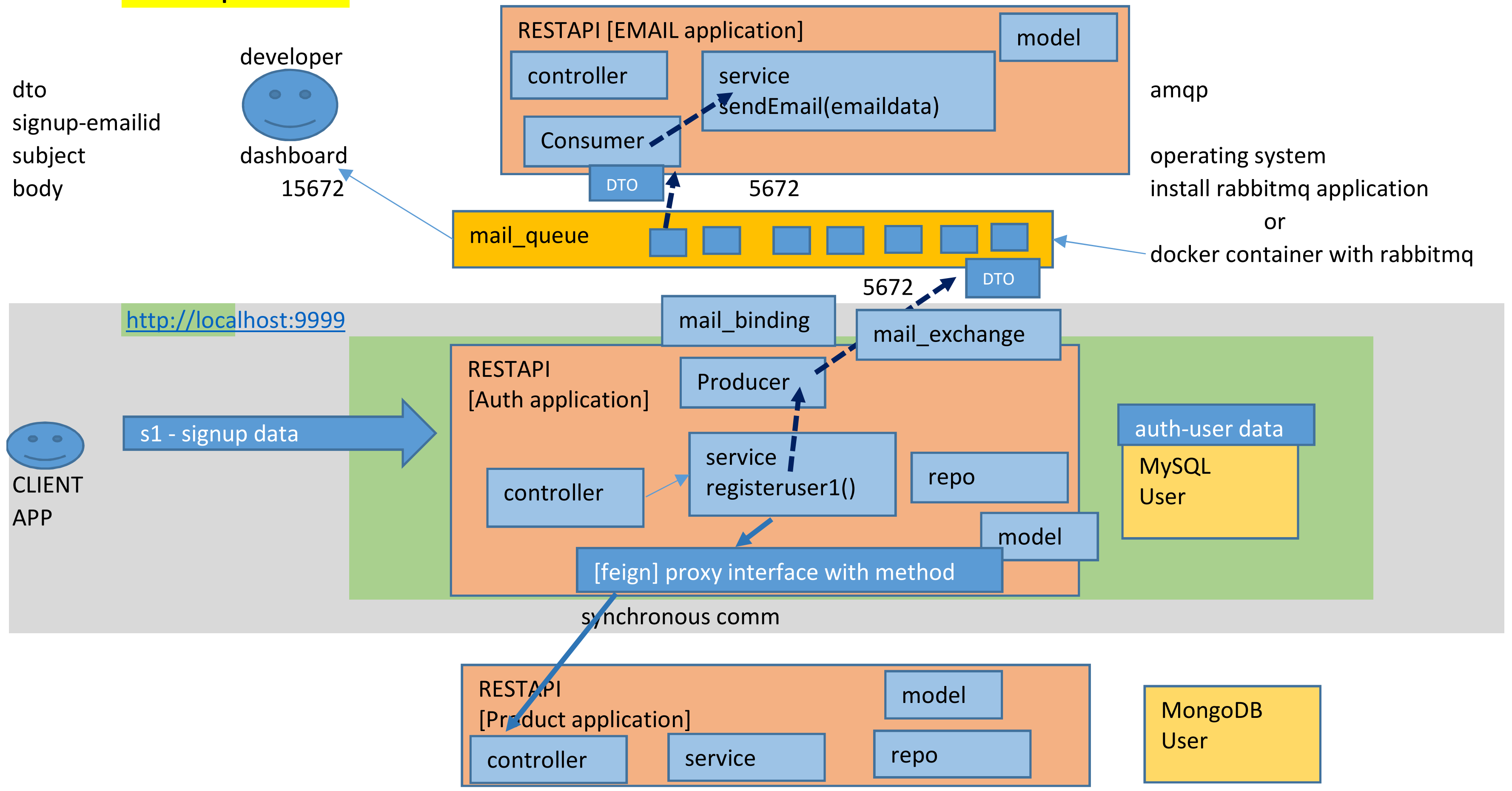
All queues (1)

Pagination

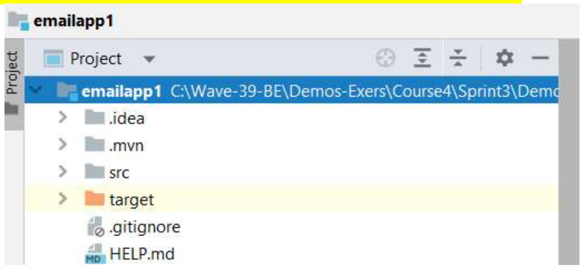
Page 1 of 1 - Filter: Regex ?

Overview				Messages			Message rates	
Name	Type	Features	State	Ready	Unacked	Total	incoming	deliver / get
mail_queue	classic	D	idle	2	0	2	0.00/s	

Receiver part



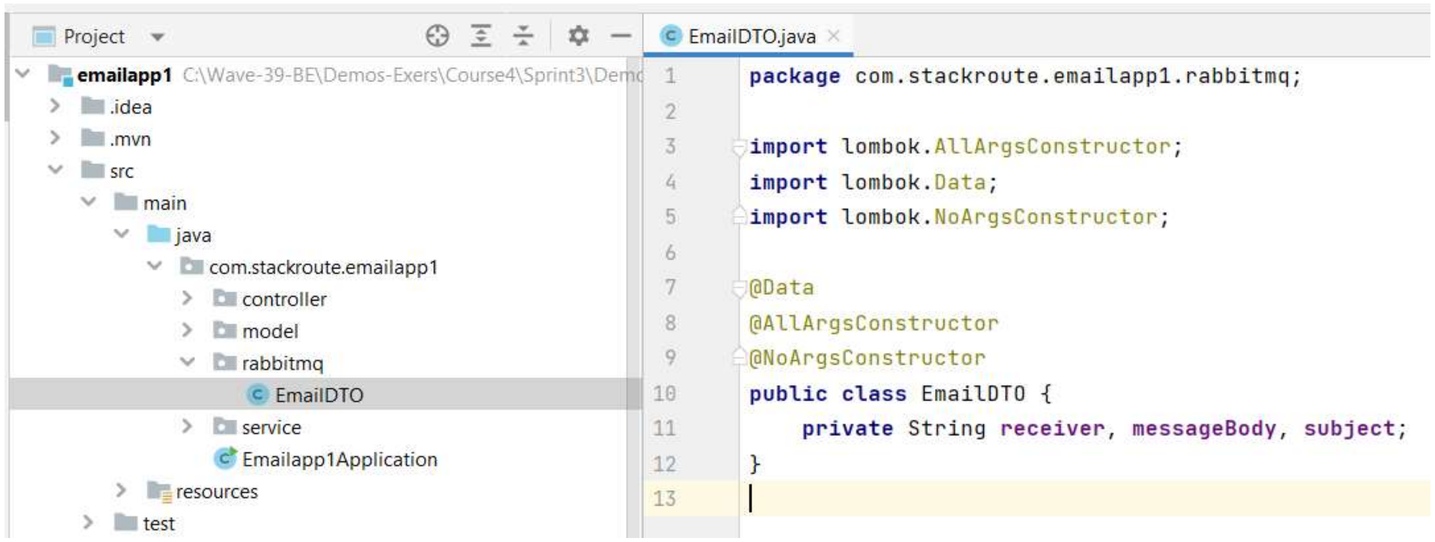
Steps to make consumer ready



Step 1 add required dependency to pom

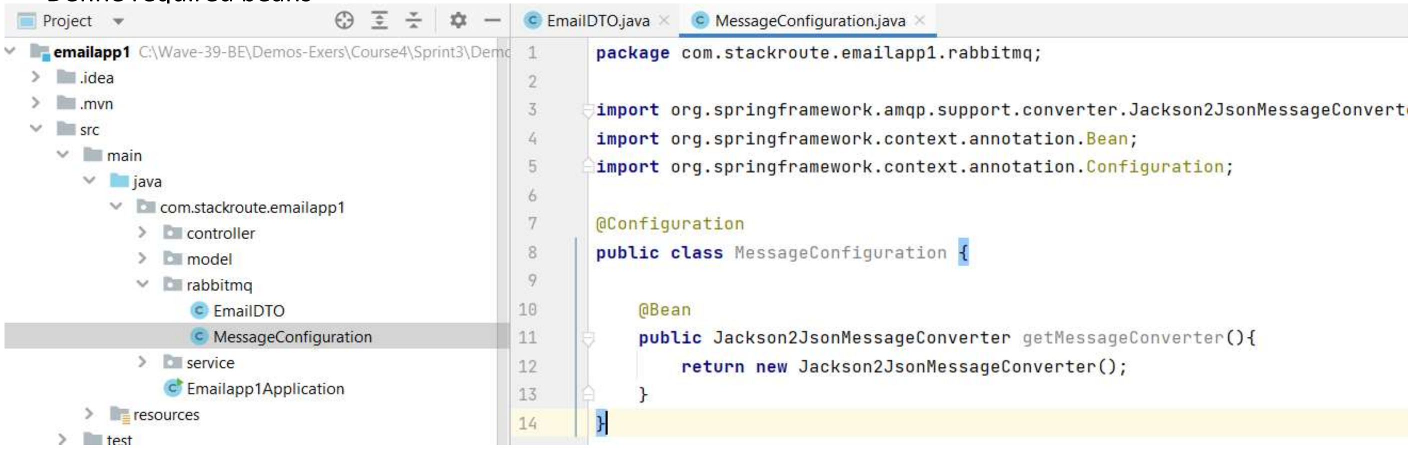
```
<dependency>
<groupId>org.springframework.boot</groupId>
<artifactId>spring-boot-starter-amqp</artifactId>
</dependency>
```

Step 2 Create DTO with same shape as sender DTO



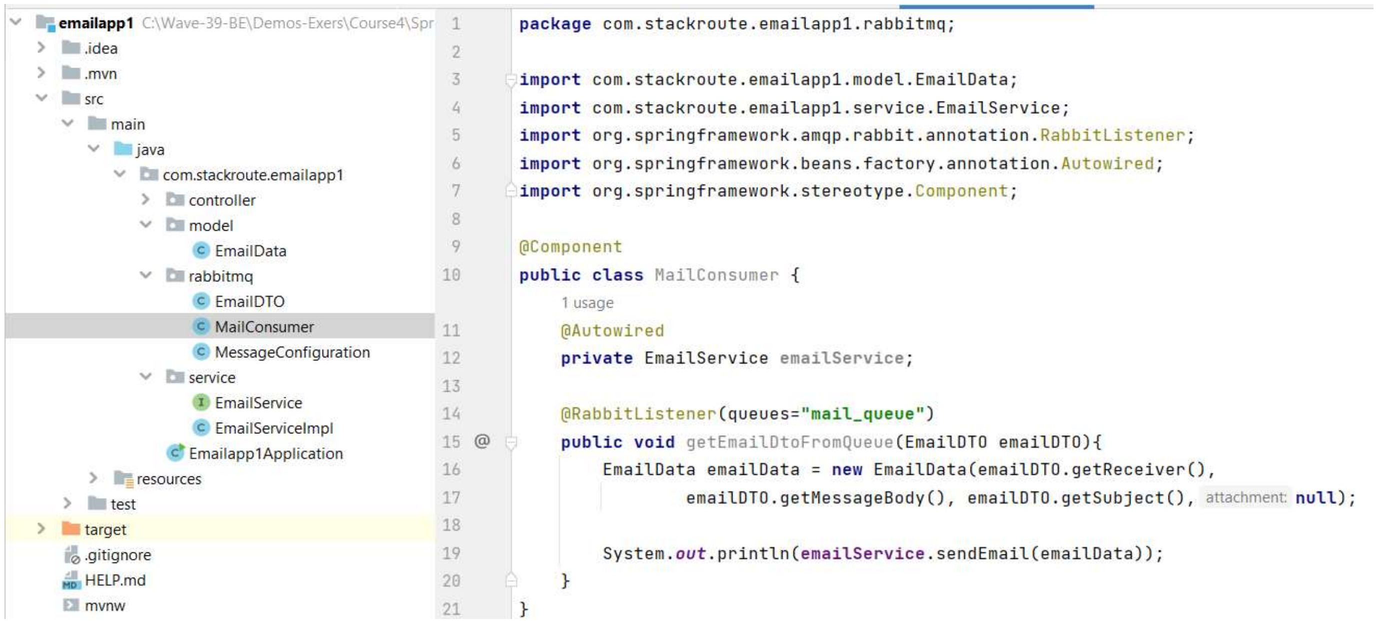
```java
package com.stackroute.emailapp1.rabbitmq;

import lombok.AllArgsConstructor;
import lombok.Data;
import lombok.NoArgsConstructor;

@Data
@AllArgsConstructor
@NoArgsConstructor
public class EmailDTO {
    private String receiver, messageBody, subject;
}
```

Step 3 Define required beans



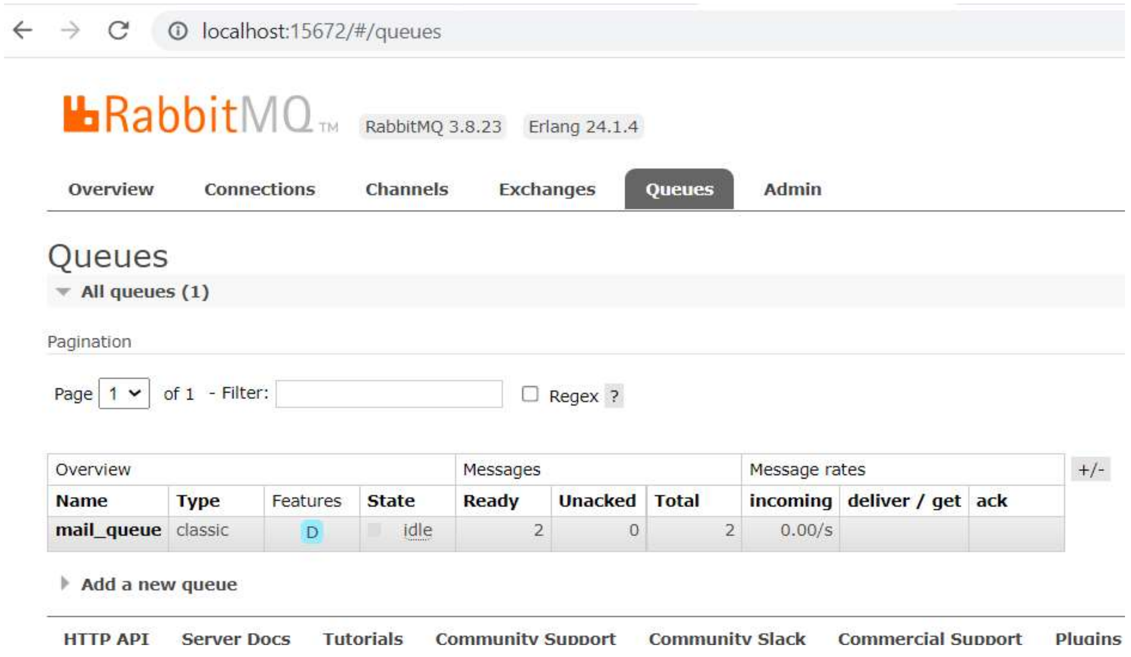
```java
package com.stackroute.emailapp1.rabbitmq;

import org.springframework.amqp.support.converter.Jackson2JsonMessageConvert
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;

@Configuration
public class MessageConfiguration {

    @Bean
    public Jackson2JsonMessageConverter getMessageConverter(){
        return new Jackson2JsonMessageConverter();
    }
}
```

Step 4 Define Consumer to receive dto object from queue and pass the same object to service method



```java
package com.stackroute.emailapp1.rabbitmq;

import com.stackroute.emailapp1.model.EmailData;
import com.stackroute.emailapp1.service.EmailService;
import org.springframework.amqp.rabbit.annotation.RabbitListener;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Component;

@Component
public class MailConsumer {
    1 usage
    @Autowired
    private EmailService emailService;

    @RabbitListener(queues="mail_queue")
    public void getEmailDtoFromQueue(EmailDTO emailDTO){
        EmailData emailData = new EmailData(emailDTO.getReceiver(),
                emailDTO.getMessageBody(), emailDTO.getSubject(), attachment: null);

        System.out.println(emailService.sendEmail(emailData));
    }
}
```

ready run
how to check



localhost:15672/#/queues

RabbitMQ 3.8.23 Erlang 24.1.4

Overview Connections Channels Exchanges Queues Admin

Queues

All queues (1)

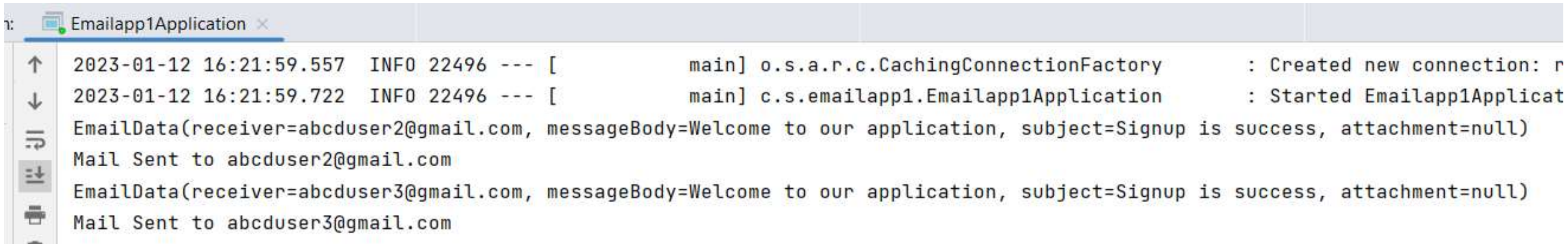
Pagination

Page 1 of 1 - Filter: Regex ?

Overview				Messages			Message rates		
Name	Type	Features	State	Ready	Unacked	Total	incoming	deliver / get	ack
mail_queue	classic	D	idle	2	0	2	0.00/s		

Add a new queue

HTTP API Server Docs Tutorials Community Support Community Slack Commercial Support Plugins



Emailapp1Application

```
2023-01-12 16:21:59.557  INFO 22496 --- [           main] o.s.a.r.c.CachingConnectionFactory       : Created new connection: r
2023-01-12 16:21:59.722  INFO 22496 --- [           main] c.s.emailapp1.Emailapp1Application       : Started Emailapp1Applicat
EmailData(receiver=abcduser2@gmail.com, messageBody=Welcome to our application, subject=Signup is success, attachment=null)
Mail Sent to abcduser2@gmail.com
EmailData(receiver=abcduser3@gmail.com, messageBody=Welcome to our application, subject=Signup is success, attachment=null)
Mail Sent to abcduser3@gmail.com
```