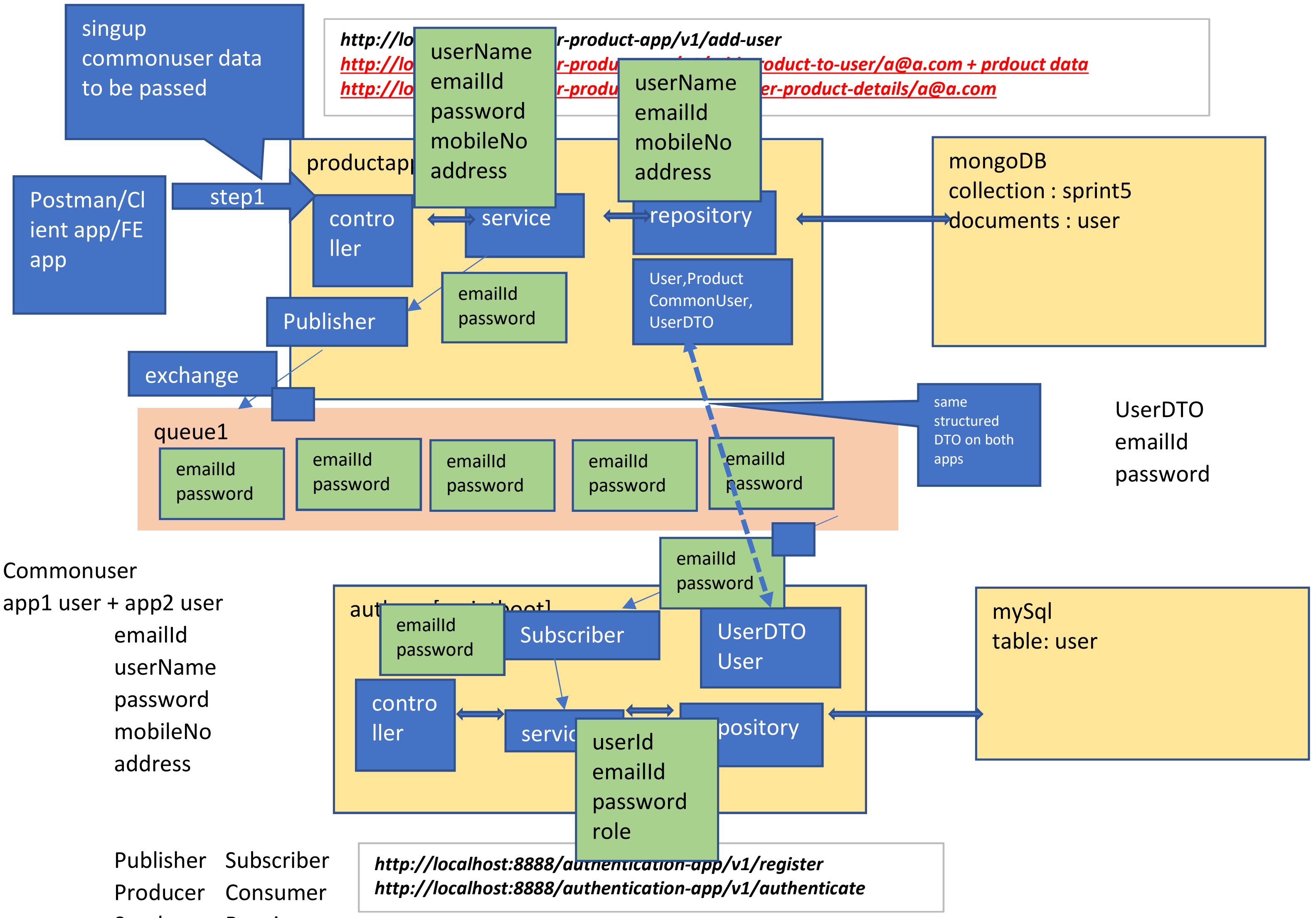


http://lo r-product-app/v1/add-user
http://lo r-produ oduct-to-user/a@a.com + prdouct data
http://lo r-produ er-product-details/a@a.com

http://localhost:8888/authentication-app/v1/register
http://localhost:8888/authentication-app/v1/authenticate

User
userName
emailId
mobileNo
address
products

UserDTO
emailId
password

User
userId
emailId
password
role

Commonuser
app1 user + app2 user
emailId
userName
password
mobileNo
address

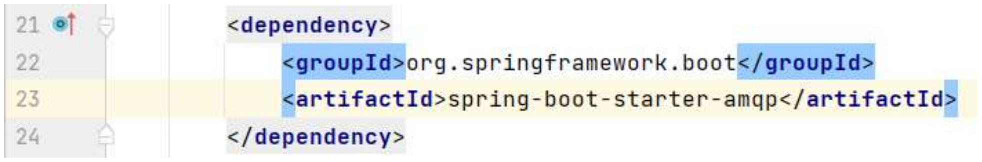
Publisher	Subscriber
Producer	Consumer
Sender	Receiver
Source	Destination

Steps to create Pub/Sub communication between product and auth applications using RabbitMQ

Step 1 Make sure both product and auth applications ready

Configure product application as producer/publisher/sender/source

Step 2 add required dependency in pom

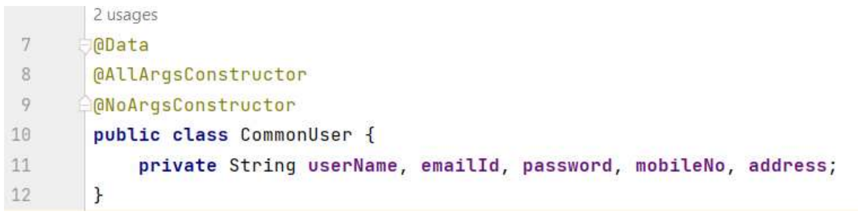


```xml
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-amqp</artifactId>
</dependency>
```

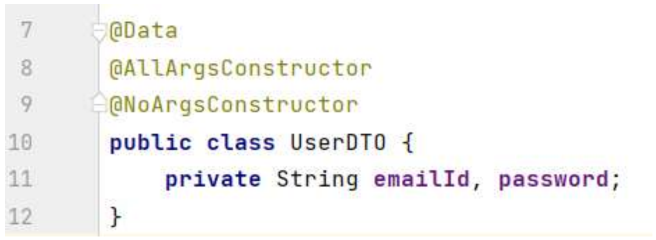
Step 3 Create required model classes

CommonUser

UserDTO



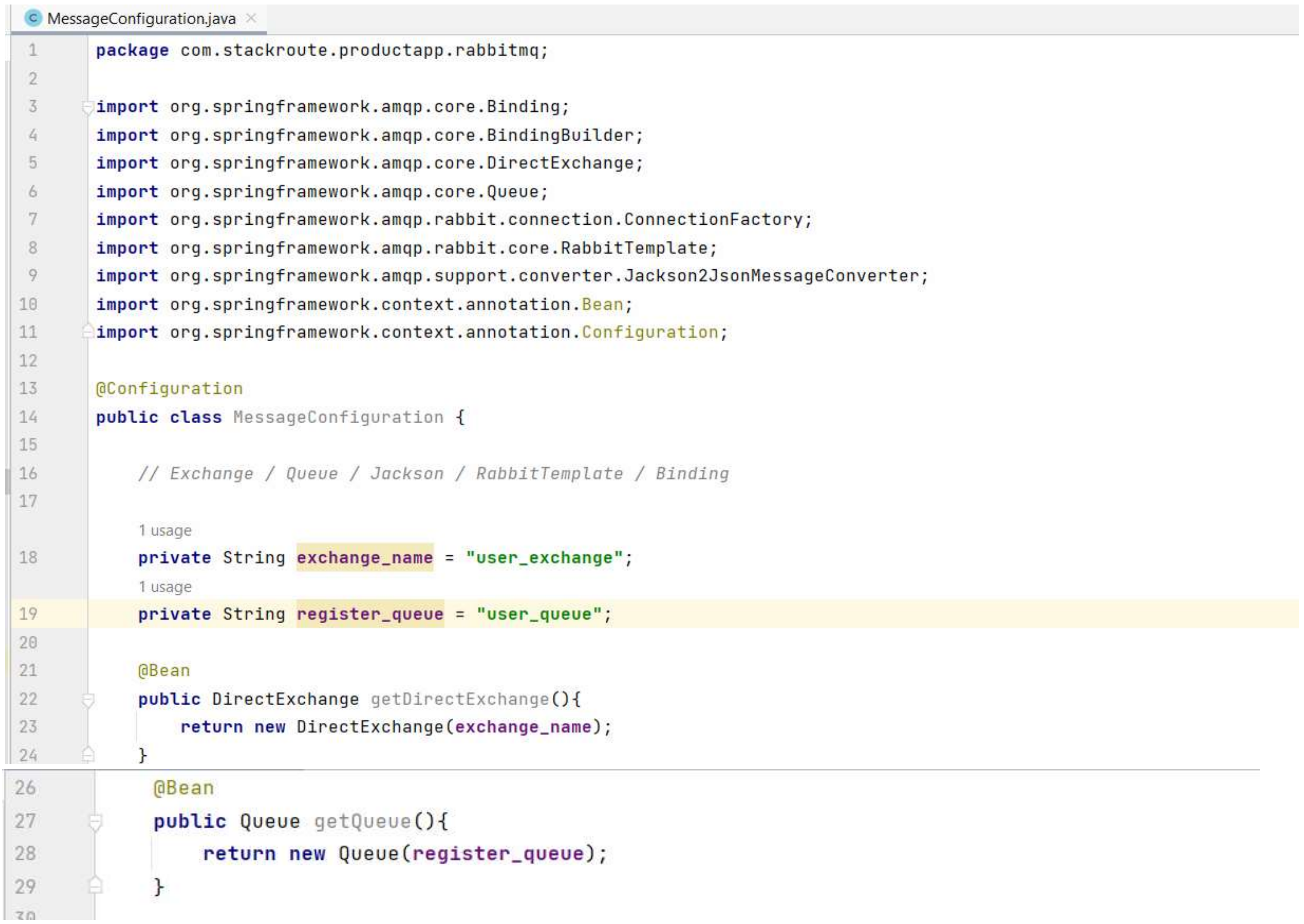
```java
2 usages
@Data
@AllArgsConstructor
@NoArgsConstructor
public class CommonUser {
    private String userName, emailId, password, mobileNo, address;
}
```



```java
@Data
@AllArgsConstructor
@NoArgsConstructor
public class UserDTO {
    private String emailId, password;
}
```

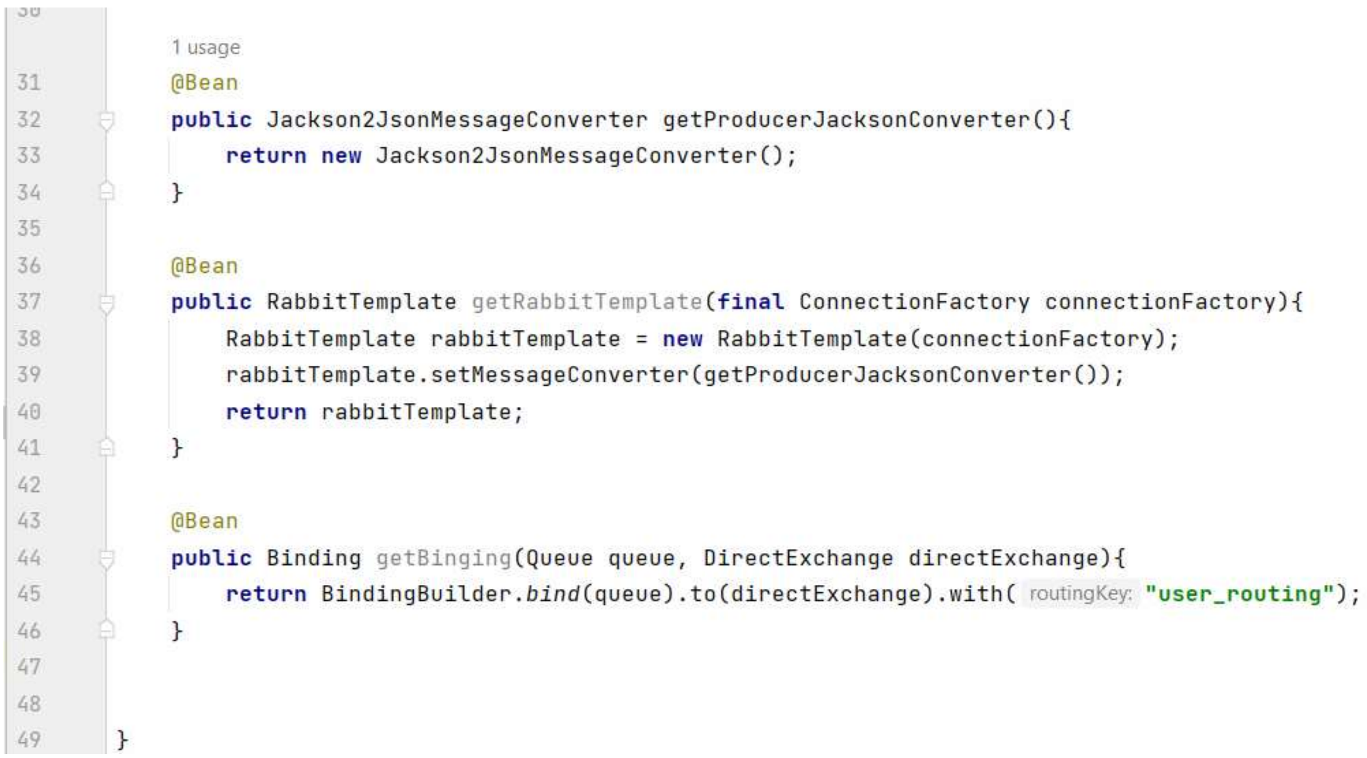
Step 4 Define message configuration with below beans

- Exchange
- Queue
- JacksonConverter
- RabbitTemplate
- Binding



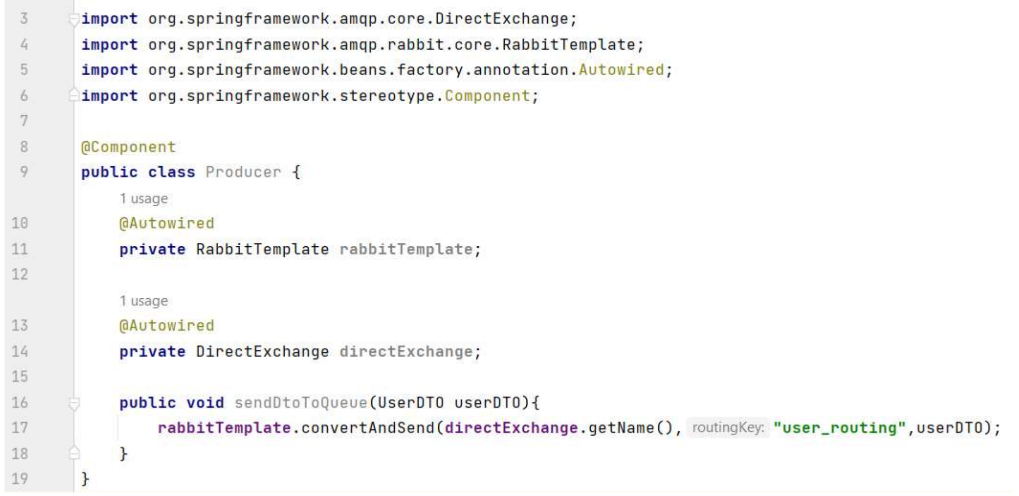
MessageConfiguration.java

```java
package com.stackroute.productapp.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 / Jackson / RabbitTemplate / Binding

    1 usage
    private String exchange_name = "user_exchange";
    1 usage
    private String register_queue = "user_queue";

    @Bean
    public DirectExchange getDirectExchange(){
        return new DirectExchange(exchange_name);
    }
    @Bean
    public Queue getQueue(){
        return new Queue(register_queue);
    }
```



```
    1 usage
    @Bean
    public Jackson2JsonMessageConverter getProducerJacksonConverter(){
        return new Jackson2JsonMessageConverter();
    }

    @Bean
    public RabbitTemplate getRabbitTemplate(final ConnectionFactory connectionFactory){
        RabbitTemplate rabbitTemplate = new RabbitTemplate(connectionFactory);
        rabbitTemplate.setMessageConverter(getProducerJacksonConverter());
        return rabbitTemplate;
    }

    @Bean
    public Binding getBinging(Queue queue, DirectExchange directExchange){
        return BindingBuilder.bind(queue).to(directExchange).with( routingKey: "user_routing");
    }


}
```

Step 5 Define producer



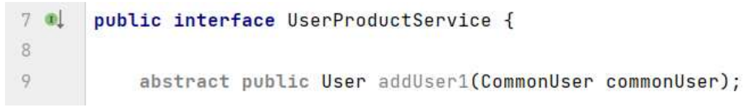
```
import org.springframework.amqp.core.DirectExchange;
import org.springframework.amqp.rabbit.core.RabbitTemplate;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Component;

@Component
public class Producer {
    1 usage
    @Autowired
    private RabbitTemplate rabbitTemplate;

    1 usage
    @Autowired
    private DirectExchange directExchange;

    public void sendDtoToQueue(UserDTO userDTO){
        rabbitTemplate.convertAndSend(directExchange.getName(), routingKey: "user_routing",userDTO);
    }
}
```

Step 6 Modify service

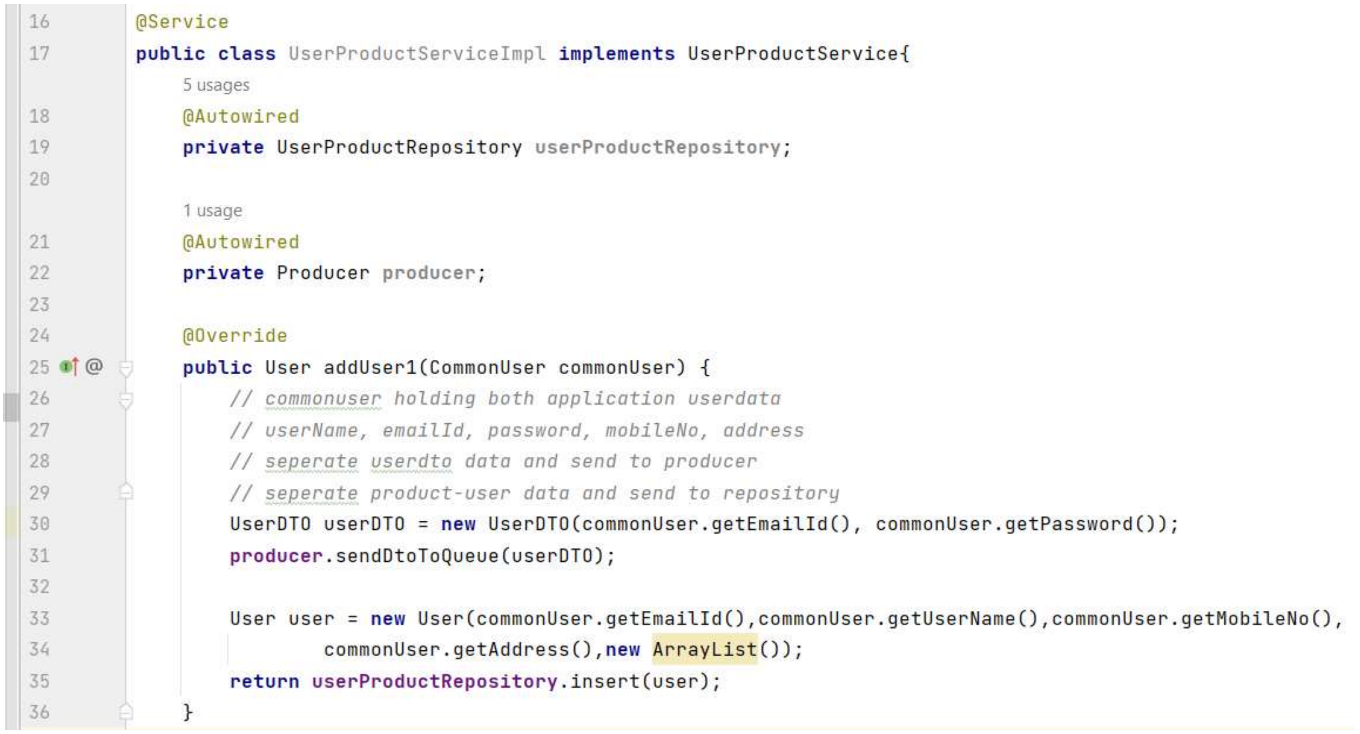
spereate product-user data and userDTO data

send userDTO to producer

send product-user data to repository



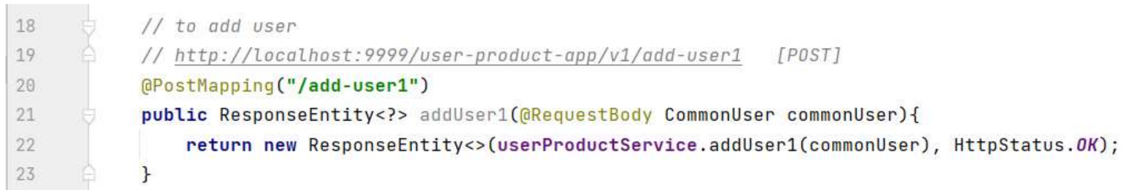
```java
public interface UserProductService {

    abstract public User addUser1(CommonUser commonUser);
```



```java
@Service
public class UserProductServiceImpl implements UserProductService{
    5 usages
    @Autowired
    private UserProductRepository userProductRepository;

    1 usage
    @Autowired
    private Producer producer;

    @Override
    public User addUser1(CommonUser commonUser) {
        // commonuser holding both application userdata
        // userName, emailId, password, mobileNo, address
        // seperate userdto data and send to producer
        // seperate product-user data and send to repository
        UserDTO userDTO = new UserDTO(commonUser.getEmailId(), commonUser.getPassword());
        producer.sendDtoToQueue(userDTO);

        User user = new User(commonUser.getEmailId(),commonUser.getUserName(),commonUser.getMobileNo(),
                commonUser.getAddress(),new ArrayList());
        return userProductRepository.insert(user);
    }
```

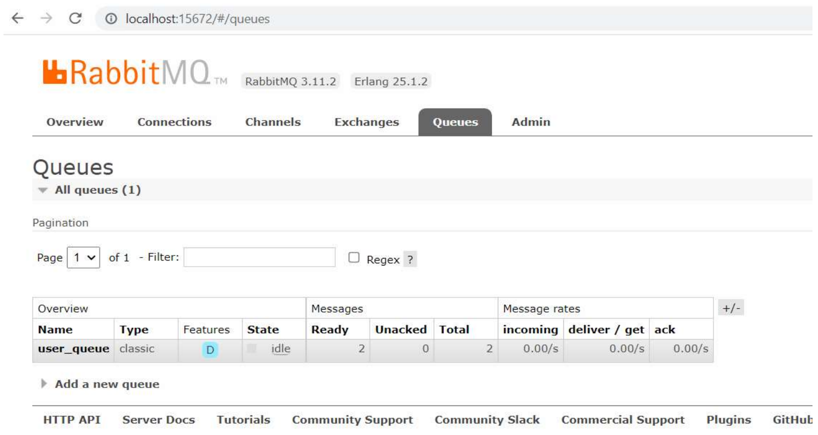
Step 7 Modify the controller

to receive common user data and send to new method of service



```java
// to add user
// http://localhost:9999/user-product-app/v1/add-user1    [POST]
@PostMapping("/add-user1")
public ResponseEntity<?> addUser1(@RequestBody CommonUser commonUser){
    return new ResponseEntity<>(userProductService.addUser1(commonUser), HttpStatus.OK);
}
```

PRODUCER IS READY
checkpoint
Start product application
Don’t start auth application
try to insert user (commonuser) record in product application
observe product-user data inserted in mongodb
observe userDTO waiting in queue



Queues

All queues (1)

Pagination

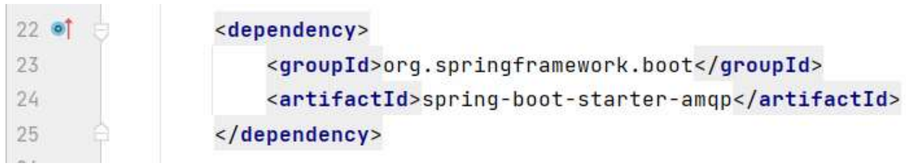
Page 1 of 1 - Filter: Regex ?

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

Add a new queue

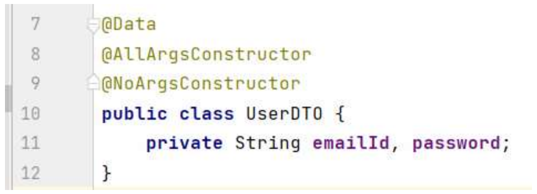
Define consumer/subscriber/receiver/destination
in auth application

Step 1 Add dependency



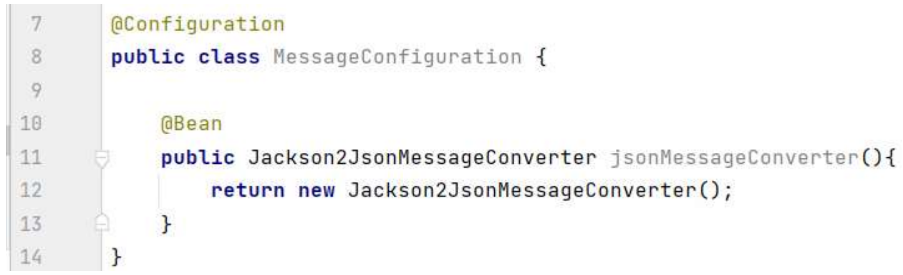
```
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-amqp</artifactId>
</dependency>
```

Step 2 Create UserDTO model



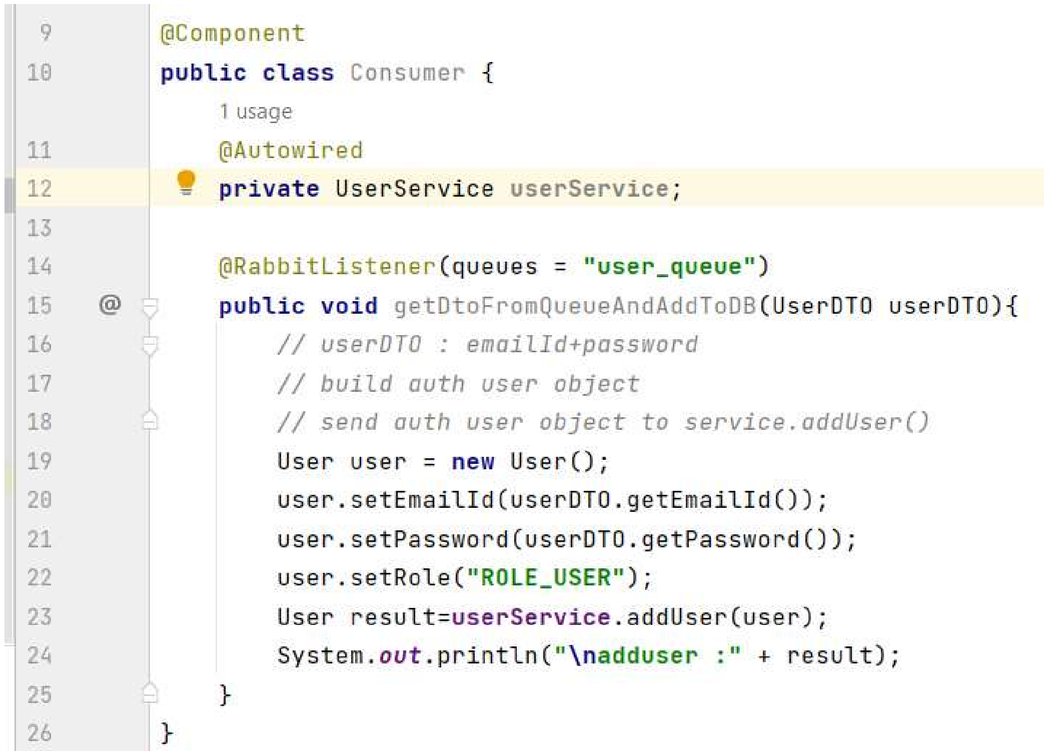
```
@Data
@AllArgsConstructor
@NoArgsConstructor
public class UserDTO {
    private String emailId, password;
}
```

Step 3 Define message configuration



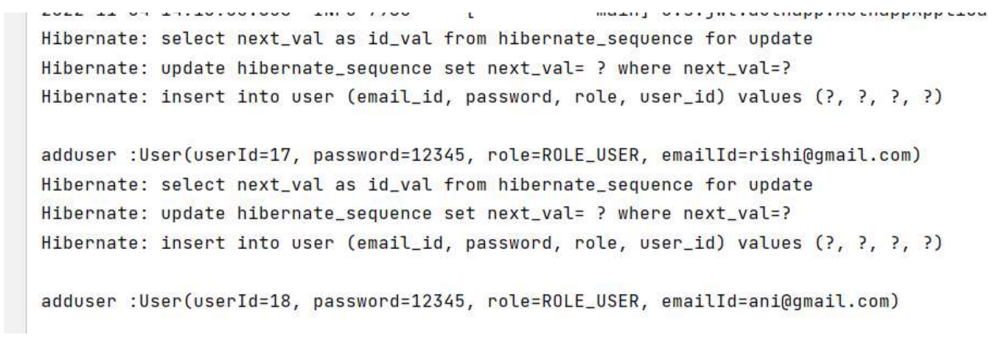
```
@Configuration
public class MessageConfiguration {

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

Step 4 Define consumer class

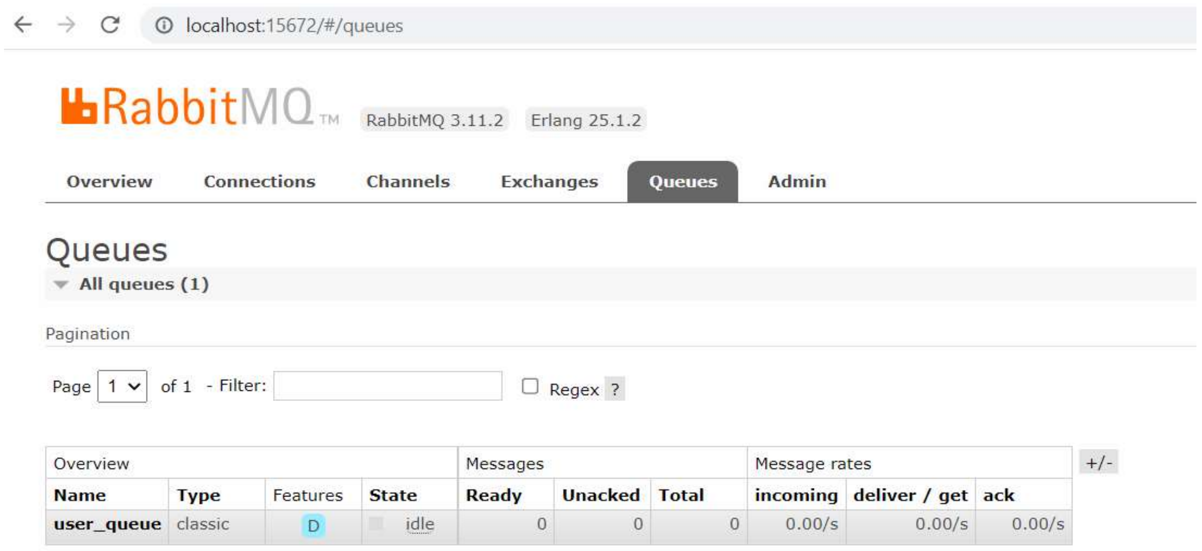


```java
@Component
public class Consumer {
    1 usage
    @Autowired
    private UserService userService;

    @RabbitListener(queues = "user_queue")
    public void getDtoFromQueueAndAddToDB(UserDTO userDTO){
        // userDTO : emailId+password
        // build auth user object
        // send auth user object to service.addUser()
        User user = new User();
        user.setEmailId(userDTO.getEmailId());
        user.setPassword(userDTO.getPassword());
        user.setRole("ROLE_USER");
        User result=userService.addUser(user);
        System.out.println("\nadduser :" + result);
    }
}
```

Consumer is ready
start consumer
observe consumer picks waiting objects from queue and process them



```
Hibernate: select next_val as id_val from hibernate_sequence for update
Hibernate: update hibernate_sequence set next_val= ? where next_val=?
Hibernate: insert into user (email_id, password, role, user_id) values (?, ?, ?, ?)

adduser :User(userId=17, password=12345, role=ROLE_USER, emailId=rishi@gmail.com)
Hibernate: select next_val as id_val from hibernate_sequence for update
Hibernate: update hibernate_sequence set next_val= ? where next_val=?
Hibernate: insert into user (email_id, password, role, user_id) values (?, ?, ?, ?)

adduser :User(userId=18, password=12345, role=ROLE_USER, emailId=ani@gmail.com)
```



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
user_queue	classic	D	idle	0	0	0	0.00/s	0.00/s	0.00/s