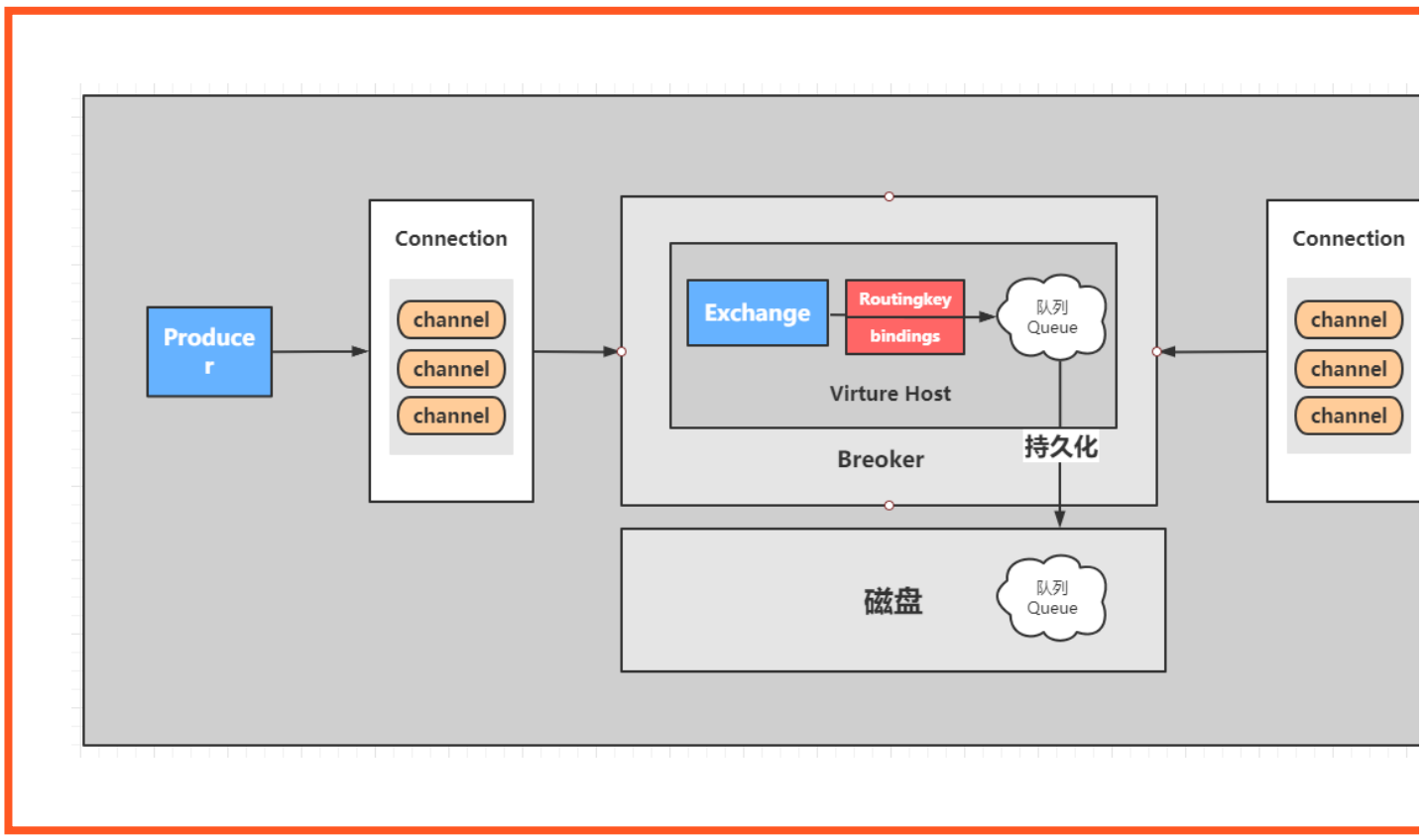
RabbitMQ-SpringBoot案例 -topic模式

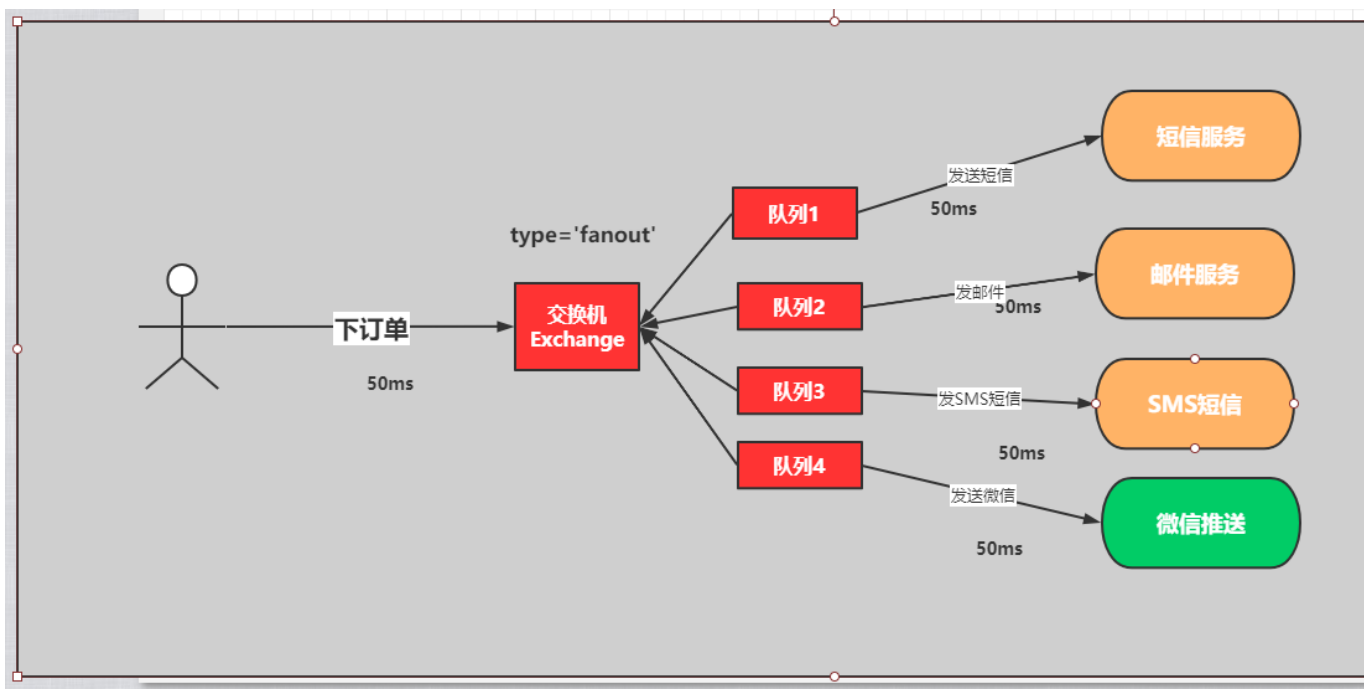
飞哥 VIP 分类：学习笔记 创建时间：2021/03/02 20:07 ☑字体 ☐皮肤

整体核心



01、目标

使用springboot完成rabbitmq的消费模式-Fanout



02、实现步骤

1：创建生产者工程：sspringboot-rabbitmq-direct-producer
2：创建消费者工程：springboot-rabbitmq-direct-consumer
3：引入spring-boot-rabbitmq的依赖
4：进行消息的分发和测试
5：查看和观察web控制台的状况

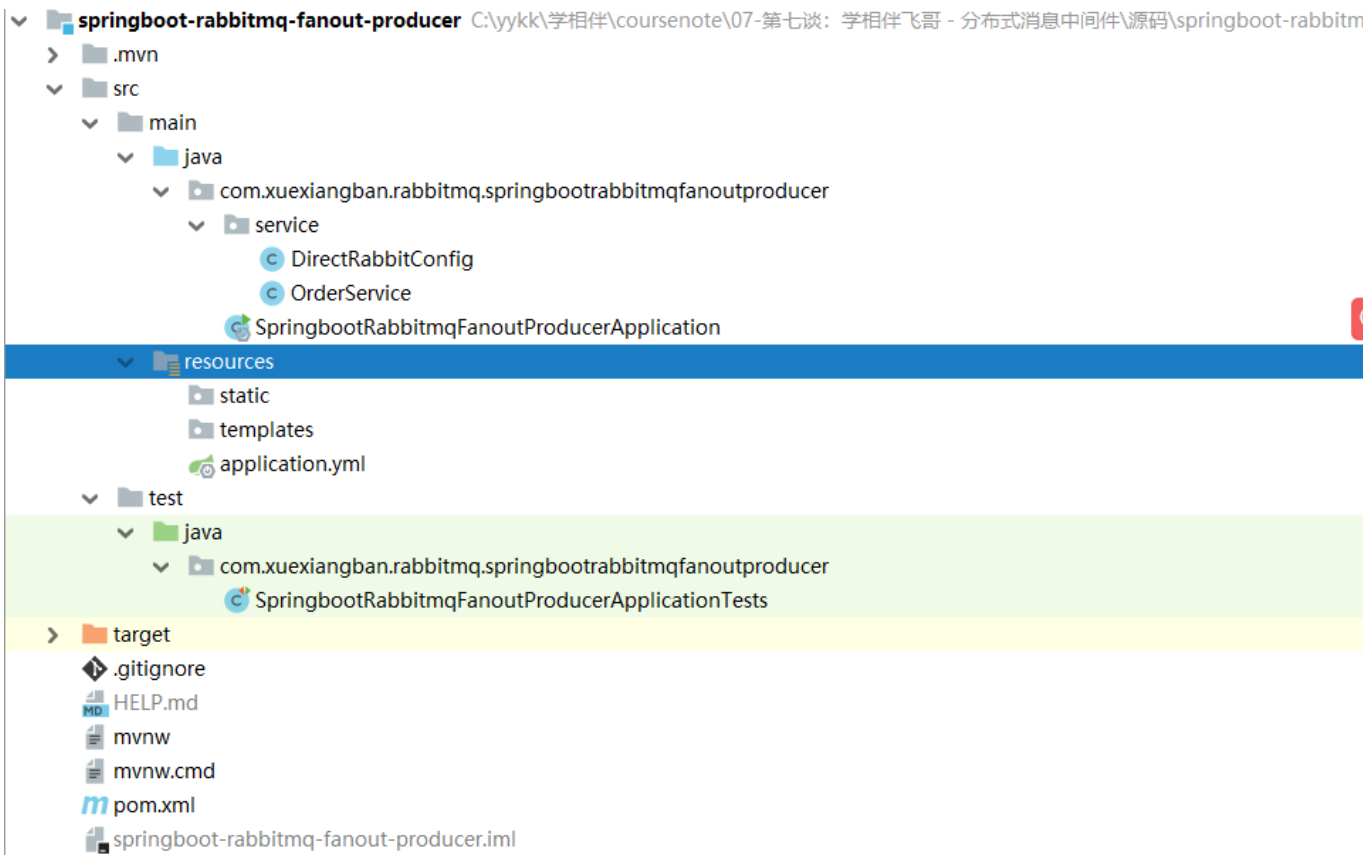
整体核心
01、目标
02、实现步骤
具体实现
03、生产者
1、创建生产者工程：sspringboot-rabbit
2、在pom.xml中引入依赖
3、在application.yml进行配置
4：定义订单的生产者
4、绑定关系
5、进行测试
04、定义消费者
1、创建消费者工程：springboot-rabbitm
2、引入依赖pom.xml
3、在application.yml进行配置
4、消费者 - 邮件服务
5、消费者 - 短信服务
6、消费者 - 微信服务
7、启动服务SpringbootRabbitmqFanou

TOP

具体实现

03、生产者

1、创建生产者工程：sspringboot-rabbitmq-direct-producer



2、在pom.xml中引入依赖

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

3、在application.yml进行配置

```yaml
# 服务端口
server:
  port: 8080

# 配置rabbitmq服务
spring:
  rabbitmq:
    username: admin
    password: admin
    virtual-host: /
    host: 47.104.141.27
    port: 5672
```

4：定义订单的生产者

整体核心
01、目标
02、实现步骤
具体实现
03、生产者
1、创建生产者工程：sspringboot-rabbit
2、在pom.xml中引入依赖
3、在application.yml进行配置
4：定义订单的生产者
4、绑定关系
5、进行测试
04、定义消费者
1、创建消费者工程：springboot-rabbitm
2、引入依赖pom.xml
3、在application.yml进行配置
3、在application.yml进行配置
4、消费者 - 邮件服务
5、消费者 - 短信服务
6、消费者 - 微信服务
7、启动服务SpringbootRabbitmqFanou

```java
package com.xuexiangban.rabbitmq.springbootrabbitmqfanoutproducer.service;

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

import java.util.UUID;

/**
 * @author: 学相伴-飞哥
 * @description: OrderService
 * @Date : 2021/3/4
 */
@Component
public class OrderService {

    @Autowired
    private RabbitTemplate rabbitTemplate;
    // 1: 定义交换机
    private String exchangeName = "direct_order_exchange";
    // 2: 路由key
    private String routeKey = "";


    public void makeOrder(Long userId, Long productId, int num) {
        // 1： 模拟用户下单
        String orderNumer = UUID.randomUUID().toString();
        // 2: 根据商品id productId 去查询商品的库存
        // int numstore = productSerivce.getProductNum(productId);
        // 3:判断库存是否充足
        // if(num >  numstore ){ return  "商品库存不足..."; }
        // 4: 下单逻辑
        // orderService.saveOrder(order);
        // 5: 下单成功要扣减库存
        // 6: 下单完成以后
        System.out.println("用户 " + userId + ",订单编号是：" + orderNumer);
        // 发送订单信息给RabbitMQ fanout
        rabbitTemplate.convertAndSend(exchangeName, routeKey, orderNumer);
    }
}
```

整体核心
01、目标
02、实现步骤
具体实现
03、生产者
1、创建生产者工程：sspringboot-rabbit
2、在pom.xml中引入依赖
3、在application.yml进行配置
4：定义订单的生产者
4、绑定关系
5、进行测试
04、定义消费者
1、创建消费者工程：springboot-rabbitn
2、引入依赖pom.xml
3、在application.yml进行配置
4、消费者 - 邮件服务
5、消费者 - 短信服务
6、消费者 - 微信服务
7、启动服务SpringbootRabbitmqFanou

4、绑定关系

```java
package com.xuexiangban.rabbitmq.springbootrabbitmqfanoutproducer.service;

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.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;

/**
 * @Author : JCccc
 * @CreateTime : 2019/9/3
 * @Description :
 **/
@Configuration
public class DirectRabbitConfig {

    //队列 起名：TestDirectQueue
    @Bean
    public Queue emailQueue() {
        // durable:是否持久化,默认是false,持久化队列：会被存储在磁盘上，当消息代理重启时仍
连接有效
        // exclusive:默认也是false，只能被当前创建的连接使用，而且当连接关闭后队列即被删除
durable
        // autoDelete:是否自动删除，当没有生产者或者消费者使用此队列，该队列会自动删除。
        //   return new Queue("TestDirectQueue",true,true,false);

        //一般设置一下队列的持久化就好,其余两个就是默认false
        return new Queue("email.fanout.queue", true);
    }

    @Bean
    public Queue smsQueue() {
        // durable:是否持久化,默认是false,持久化队列：会被存储在磁盘上，当消息代理重启时仍
连接有效
        // exclusive:默认也是false，只能被当前创建的连接使用，而且当连接关闭后队列即被删除
durable
        // autoDelete:是否自动删除，当没有生产者或者消费者使用此队列，该队列会自动删除。
        //   return new Queue("TestDirectQueue",true,true,false);

        //一般设置一下队列的持久化就好,其余两个就是默认false
        return new Queue("sms.fanout.queue", true);
    }

    @Bean
    public Queue weixinQueue() {
        // durable:是否持久化,默认是false,持久化队列：会被存储在磁盘上，当消息代理重启时仍
连接有效
        // exclusive:默认也是false，只能被当前创建的连接使用，而且当连接关闭后队列即被删除
durable
        // autoDelete:是否自动删除，当没有生产者或者消费者使用此队列，该队列会自动删除。
        //   return new Queue("TestDirectQueue",true,true,false);

        //一般设置一下队列的持久化就好,其余两个就是默认false
        return new Queue("weixin.fanout.queue", true);
    }

    //Direct交换机 起名：TestDirectExchange
    @Bean
    public DirectExchange directOrderExchange() {
        //  return new DirectExchange("TestDirectExchange",true,true);
        return new DirectExchange("direct_order_exchange", true, false);
    }

    //绑定  将队列和交换机绑定, 并设置用于匹配键：TestDirectRouting
    @Bean
    public Binding bindingDirect1() {
        return BindingBuilder.bind(weixinQueue()).to(directOrderExchange()).with
```

整体核心
01、目标
02、实现步骤
具体实现
03、生产者
1、创建生产者工程：sspringboot-rabbit
2、在pom.xml中引入依赖
3、在application.yml进行配置
4：定义订单的生产者
4、绑定关系
5、进行测试
04、定义消费者
1、创建消费者工程：springboot-rabbitm
2、引入依赖pom.xml
3、在application.yml进行配置
4、消费者 - 邮件服务
5、消费者 - 短信服务
6、消费者 - 微信服务
7、启动服务SpringbootRabbitmqFanou

```
    }

    @Bean
    public Binding bindingDirect2() {
        return BindingBuilder.bind(smsQueue()).to(directOrderExchange()).with("
    }

    @Bean
    public Binding bindingDirect3() {
        return BindingBuilder.bind(emailQueue()).to(directOrderExchange()).with(
    }

}
```

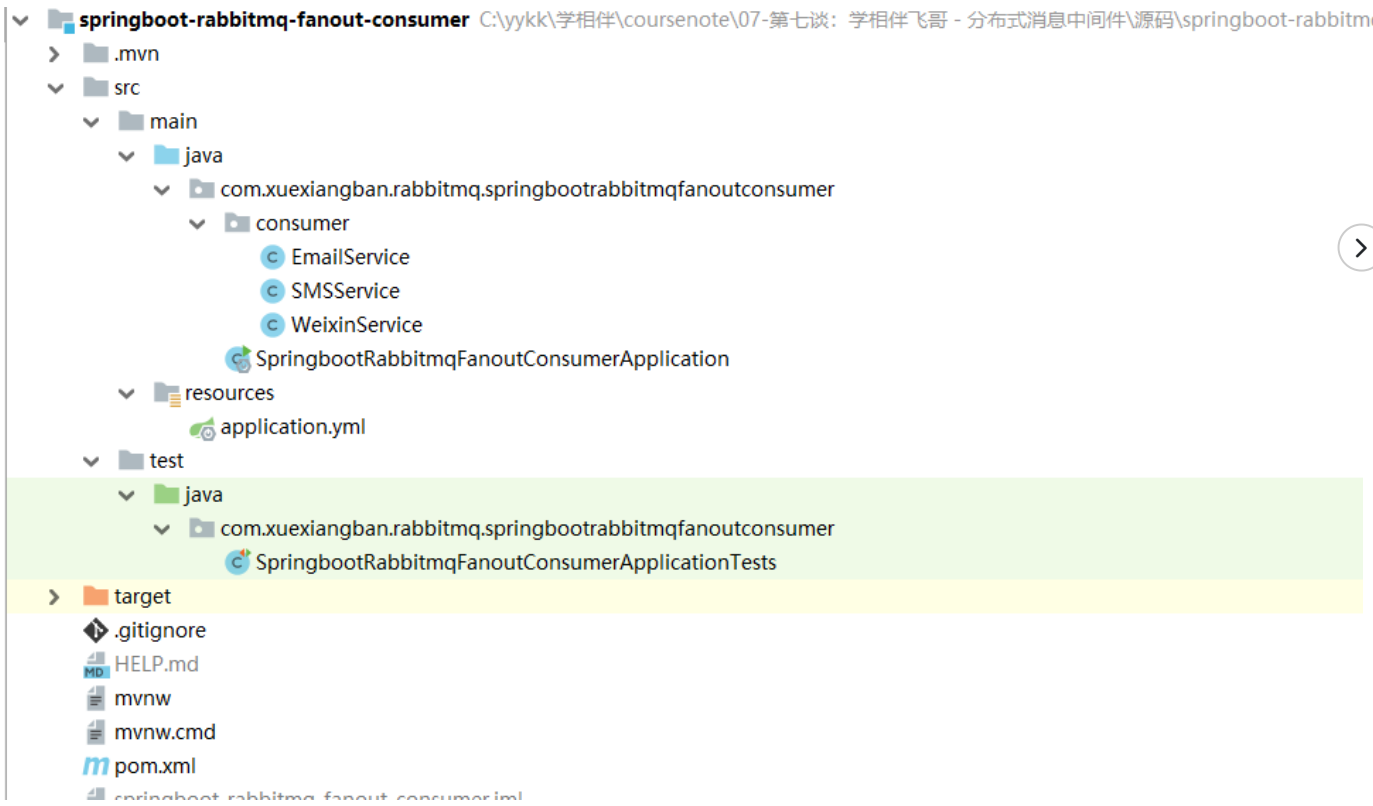
5、进行测试

```
package com.xuexiangban.rabbitmq.springbootrabbitmqfanoutproducer;

import com.xuexiangban.rabbitmq.springbootrabbitmqfanoutproducer.service.OrderSe
import org.junit.jupiter.api.Test;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.boot.test.context.SpringBootTest;

@SpringBootTest
class SpringbootRabbitmqFanoutProducerApplicationTests {

    @Autowired
    OrderService orderService;

    @Test
    public void contextLoads() throws Exception {

        for (int i = 0; i < 10; i++) {
            Thread.sleep(1000);
            Long userId = 100L + i;
            Long productId = 10001L + i;
            int num = 10;
            orderService.makeOrder(userId, productId, num);
        }
    }

}
```

04、定义消费者

1、创建消费者工程：springboot-rabbitmq-fanout-consumer

整体核心
01、目标
02、实现步骤
具体实现
03、生产者
1、创建生产者工程：sspringboot-rabbit
2、在pom.xml中引入依赖
3、在application.yml进行配置
4：定义订单的生产者
4、绑定关系
5、进行测试
04、定义消费者
1、创建消费者工程：springboot-rabbitm
2、引入依赖pom.xml
3、在application.yml进行配置
4、消费者 - 邮件服务
5、消费者 - 短信服务
6、消费者 - 微信服务
7、启动服务SpringbootRabbitmqFanou



整体核心
01、目标
02、实现步骤
具体实现
03、生产者
1、创建生产者工程：sspringboot-rabbit
2、在pom.xml中引入依赖
3、在application.yml进行配置
4：定义订单的生产者
4、绑定关系
5、进行测试
04、定义消费者
1、创建消费者工程：springboot-rabbitn
2、引入依赖pom.xml
3、在application.yml进行配置
4、消费者 - 邮件服务
5、消费者 - 短信服务
6、消费者 - 微信服务
7、启动服务SpringbootRabbitmqFanou

2、引入依赖pom.xml

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

3、在application.yml进行配置

```yaml
# 服务端口
server:
  port: 8081

# 配置rabbitmq服务
spring:
  rabbitmq:
    username: admin
    password: admin
    virtual-host: /
    host: 47.104.141.27
    port: 5672
```

4、消费者 - 邮件服务

```java
package com.xuexiangban.rabbitmq.springbootrabbitmqfanoutconsumer.consumer;

import org.springframework.amqp.core.ExchangeTypes;
import org.springframework.amqp.rabbit.annotation.*;
import org.springframework.stereotype.Component;

// bindings其实就是用来确定队列和交换机绑定关系
@RabbitListener(bindings =@QueueBinding(
        // email.fanout.queue 是队列名字，这个名字你可以自定随便定义。
        value = @Queue(value = "email.fanout.queue",autoDelete = "false"),
        // order.fanout 交换机的名字 必须和生产者保持一致
        exchange = @Exchange(value = "fanout_order_exchange",
                // 这里是确定的rabbitmq模式是：fanout 是以广播模式 、 发布订阅模式
                type = ExchangeTypes.FANOUT)
))
@Component
public class EmailService {
    // @RabbitHandler 代表此方法是一个消息接收的方法。该不要有返回值
    @RabbitHandler
    public void messagerevice(String message){
        // 此处省略发邮件的逻辑
        System.out.println("email--------------->" + message);
    }
}
```

5、消费者 - 短信服务

```java
package com.xuexiangban.rabbitmq.springbootrabbitmqfanoutconsumer.consumer;

import org.springframework.amqp.core.ExchangeTypes;
import org.springframework.amqp.rabbit.annotation.*;
import org.springframework.stereotype.Component;

// bindings其实就是用来确定队列和交换机绑定关系
@RabbitListener(bindings =@QueueBinding(
        // email.fanout.queue 是队列名字，这个名字你可以自定随便定义。
        value = @Queue(value = "sms.fanout.queue",autoDelete = "false"),
        // order.fanout 交换机的名字 必须和生产者保持一致
        exchange = @Exchange(value = "fanout_order_exchange",
                // 这里是确定的rabbitmq模式是：fanout 是以广播模式 、 发布订阅模式
                type = ExchangeTypes.FANOUT)
))
@Component
public class SMSService {

    // @RabbitHandler 代表此方法是一个消息接收的方法。该不要有返回值
    @RabbitHandler
    public void messagerevice(String message){
        // 此处省略发邮件的逻辑
        System.out.println("sms--------------->" + message);
    }
}
```

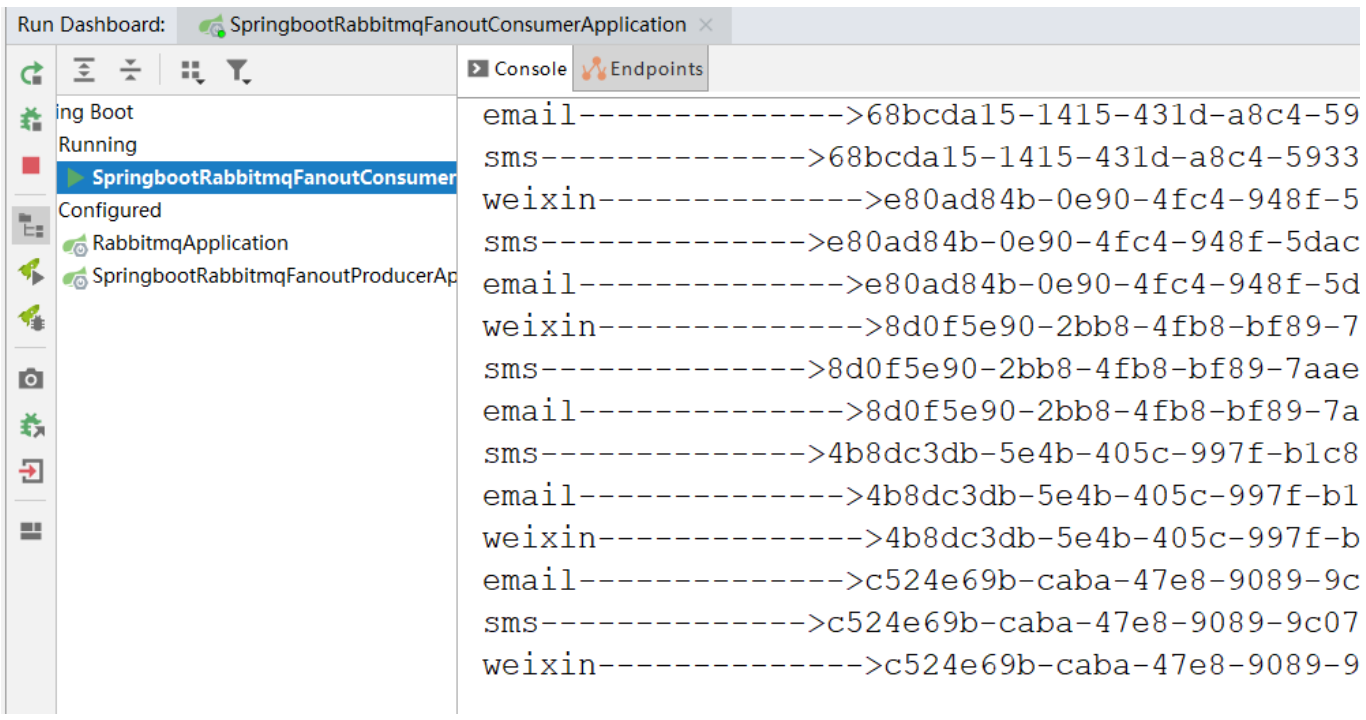
6、消费者 - 微信服务

整体核心
01、目标
02、实现步骤
具体实现
03、生产者
1、创建生产者工程：sspringboot-rabbit
2、在pom.xml中引入依赖
3、在application.yml进行配置
4：定义订单的生产者
4、绑定关系
5、进行测试
04、定义消费者
1、创建消费者工程：springboot-rabbitn
2、引入依赖pom.xml
3、在application.yml进行配置
4、消费者 - 邮件服务
5、消费者 - 短信服务
6、消费者 - 微信服务
7、启动服务SpringbootRabbitmqFanou

```java
package com.xuexiangban.rabbitmq.springbootrabbitmqfanoutconsumer.consumer;

import org.springframework.amqp.core.ExchangeTypes;
import org.springframework.amqp.rabbit.annotation.*;
import org.springframework.stereotype.Component;

// bindings其实就是用来确定队列和交换机绑定关系
@RabbitListener(bindings =@QueueBinding(
        // email.fanout.queue 是队列名字，这个名字你可以自定随便定义。
        value = @Queue(value = "weixin.fanout.queue",autoDelete = "false"),
        // order.fanout 交换机的名字 必须和生产者保持一致
        exchange = @Exchange(value = "fanout_order_exchange",
                // 这里是确定的rabbitmq模式是：fanout 是以广播模式 、 发布订阅模式
                type = ExchangeTypes.FANOUT)
))
@Component
public class WeixinService {

    // @RabbitHandler 代表此方法是一个消息接收的方法。该不要有返回值
    @RabbitHandler
    public void messagerevice(String message){
        // 此处省略发邮件的逻辑
        System.out.println("weixin-------------->" + message);
    }
}
```

整体核心
01、目标
02、实现步骤
具体实现
03、生产者
1、创建生产者工程：sspringboot-rabbit
2、在pom.xml中引入依赖
3、在application.yml进行配置
4：定义订单的生产者
4、绑定关系
5、进行测试
04、定义消费者
1、创建消费者工程：springboot-rabbitn
2、引入依赖pom.xml
3、在application.yml进行配置
4、消费者 - 邮件服务
5、消费者 - 短信服务
6、消费者 - 微信服务
7、启动服务SpringbootRabbitmqFanou

7、启动服务SpringbootRabbitmqFanoutConsumerApplication，查看效果



关于我们 | 加入我们 | 联系我们 | 帮助中心
Copyright © 广东学相伴网络科技有限公司 粤ICP备 - 2020109190 号
TOP