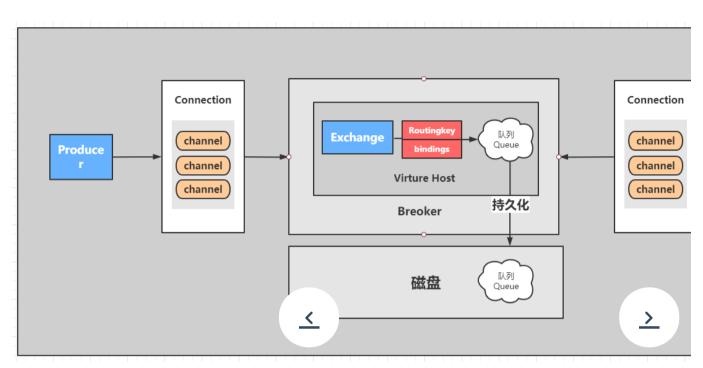
RabbitMQ-SpringBoot案例 -fanout模式

___<u>飞哥_</u> ∪IP 分类: <u>学习笔记</u> 创建时间: 2021/03/02 20:06 <u>▼字体</u> □皮肤

整体核心



整体核心

01、目标

02、实现步骤

具体实现

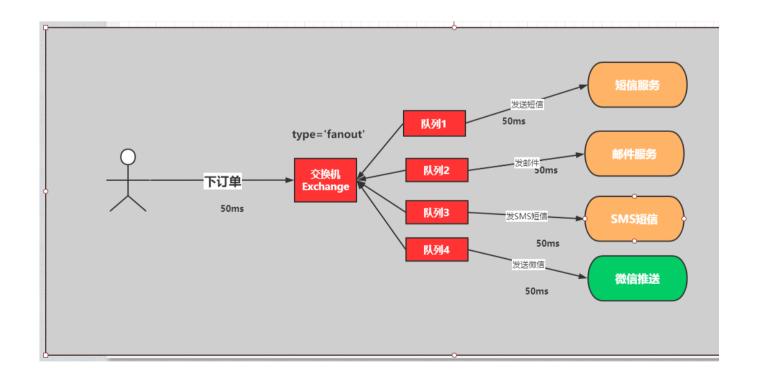
03、生产者 🗸

1、创建生产者工程:sspringboot-rabbit

- 2、在pom.xml中引入依赖
- 3、在application.yml讲行配置
- 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
 - <u>2、引入依赖pom.xml</u>
 - 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

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-fanout-producer
- 2: 创建消费者工程: springboot-rabbitmq-fanout-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

具体实现

02、实现步骤

03、生产者 🗸

具体实现

03、生产者

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

```
1、创建生产者工程: sspringboot-rabbit
📭 springboot-rabbitmq-fanout-producer C:\yykk\学相件\coursenote\07-第七谈: 学相伴飞哥 - 分布式消息中间件\源码\springboot-rabbitm
> mvn
                                                                                                       2、在pom.xml中引入依赖

✓ Image: Src

                                                                                                       3、在application.yml讲行配置
  main
                                                                                                       4: 定义订单的生产者

    com.xuexiangban.rabbitmq.springbootrabbitmqfanoutproducer

          service
                                                                                                       4、绑定关系

    DirectRabbitConfig

                OrderService
                                                                                                       5、进行测试
             SpringbootRabbitmqFanoutProducerApplication
                                                                                                     04、定义消费者 ~
          static
                                                                                                       1、创建消费者工程: springboot-rabbitn
          templates
          d application.yml
                                                                                                       2、引入依赖pom.xml

✓ limitest
                                                                                                       3、在application.yml进行配置
        Com.xuexiangban.rabbitmq.springbootrabbitmqfanoutproducer
                                                                                                       4、消费者 - 邮件服务
             SpringbootRabbitmqFanoutProducerApplicationTests
> iii target
                                                                                                       5、消费者 - 短信服务
   .gitignore
  HELP.md
                                                                                                       6、消费者 - 微信服务
   mvnw
                                                                                                       7、启动服务SpringbootRabbitmqFanou
   mvnw.cmd
  lmx.moq m
   springboot-rabbitmq-fanout-producer.iml
                                        <u>></u>
```

2、在pom.xml中引入依赖

3、在application.yml进行配置

```
1. # 服务端口
2. server:
3.
     port: 8080
4.
5. # 配置rabbitmg服务
6. spring:
7.
     rabbitmq:
        username: admin
8.
        password: admin
9.
        virtual-host: /
10.
       host: 47.104.141.27
11.
12.
        port: 5672
```

4: 定义订单的生产者



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

2.
3. import org.springframework.amqp.rabbit.core.RabbitTemplate;
                                                                                   整体核心
4. import org.springframework.beans.factory.annotation.Autowired;
                                                                                   01、目标
5. import org.springframework.stereotype.Component;
6.
                                                                                   02、实现步骤
7. import java.util.UUID;
                                                                                   具体实现
8.
                                                                                   03、生产者 🗸
9. /**
10. * @author: 学相伴-飞哥
                                                                                     1、创建生产者工程: sspringboot-rabbit
11. * @description: OrderService
                                                                                     2、在pom.xml中引入依赖
12. * @Date : 2021/3/4
                                                                                     3、在application.yml讲行配置
13. */
14. @Component
                                                                                     4: 定义订单的生产者
15. public class OrderService {
                                                                                     4、绑定关系
16.
                                                                                     5、进行测试
17.
       @Autowired
18.
       private RabbitTemplate rabbitTemplate;
                                                                                   04、定义消费者 🔻
19.
       // 1: 定义交换机
                                                                                     1、创建消费者工程: springboot-rabbitn
       private String exchangeName = "fanout_order_exchange";
20.
                                                                                     2、引入依赖pom.xml
       // 2: 路由key
21.
22.
       private String routeKey = "";
                                                                                     3、在application.yml进行配置
23.
                                                                                     4、消费者 - 邮件服务
24.
                                                                                     5、消费者 - 短信服务
       public void makeOrder(Long userId, Long productId, int num) {
25.
           // 1: 模拟用户下单
26.
                                                                                     6、消费者 - 微信服务
           String orderNumer = UUID.randomUUID().toString();
27.
                                                                                     7、启动服务SpringbootRabbitmqFanou
           // 2: 根据商品id productId 去查询商品的库存
28.
29.
           // int numstore = product ivce.getProductNum(productId);
           // 3:判断库存是否充足
30.
                                                                           <u>></u>
           // if(num > numstore ) ____ urn "商品库存不足..."; }
31.
           // 4: 下单逻辑
32.
           // orderService.saveOrder(order);
33.
34.
           // 5: 下单成功要扣减库存
           // 6: 下单完成以后
35.
           System.out.println("用户 " + userId + ",订单编号是:" + orderNumer);
36.
           // 发送订单信息给RabbitMQ fanout
37.
           rabbitTemplate.convertAndSend(exchangeName, routeKey, orderNumer);
38.
39.
       }
40. }
```

4、绑定关系



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

2.
3. import org.springframework.amqp.core.Binding;
                                                                             整体核心

    import org.springframework.amqp.core.BindingBuilder;

                                                                             01、目标
import org.springframework.amqp.core.DirectExchange;
import org.springframework.amqp.core.Queue;
                                                                             02、实现步骤
7. import org.springframework.context.annotation.Bean;
                                                                             具体实现
8. import org.springframework.context.annotation.Configuration;
                                                                             03、生产者 ~
9.
10. /**
                                                                               1、创建生产者工程: sspringboot-rabbit
   * @Author : JCccc
11.
                                                                              2、在pom.xml中引入依赖
    * @CreateTime : 2019/9/3
                                                                               3、在application.yml讲行配置
    * @Description :
13.
14.
   **/
                                                                               4: 定义订单的生产者
15. @Configuration
                                                                              4、绑定关系
16. public class DirectRabbitConfig {
                                                                              5、进行测试
17.
18.
      //队列 起名:TestDirectQueue
                                                                             19.
       @Bean
                                                                               1、创建消费者工程: springboot-rabbitm
       public Queue emailQueue() {
20.
                                                                               2、引入依赖pom.xml
          // durable:是否持久化,默认是false,持久化队列:会被存储在磁盘上,当消息代理重启时作
   连接有效
                                                                               3、在application.yml进行配置
22.
          // 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[
                                                                               4、消费者 - 邮件服务
   durable
                                                                               5、消费者 - 短信服务
          // 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。
23.
          // return new Queue("TestDirectQueue", true, true, false);
24.
                                                                              6、消费者 - 微信服务
25.
                                                                               7、启动服务SpringbootRabbitmqFanou
          //一般设置一下队列的持久化就好,其余两个就是默认false
26.
27.
          28.
       }
                                 <u>></u>
29.
       @Bean
30.
       public Queue smsQueue() {
31.
          // durable:是否持久化,默认是false,持久化队列:会被存储在磁盘上,当消息代理重启时位
32.
   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。
33.
   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。
34.
               return new Queue("TestDirectQueue", true, true, false);
35.
36.
37.
          //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);
38.
39.
       }
40.
41.
       @Bean
42.
       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,当消息代理重启时位
43.
   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算。
44.
   durable
45.
          // 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);
46.
47.
          //一般设置一下队列的持久化就好,其余两个就是默认false
48.
49.
          return new Queue("weixin.fanout.queue", true);
       }
50.
51.
       //Direct交换机 起名:TestDirectExchange
52.
       @Bean
53.
       public DirectExchange fanoutOrderExchange() {
54.
55.
          // return new DirectExchange("TestDirectExchange", true, true);
          return new DirectExchange("fanout order exchange", true, false);
56.
57.
       }
58.
       //绑定 将队列和交换机绑定,并设置用于匹配键:TestDirectRouting
59.
60.
       @Bean
       public Binding bindingDirect1() {
61.
          return BindingBuilder.bind(weixinQueue()).to(fanoutOrderExchange()).with
62.
```



```
63.
        }
64.
65.
        @Bean
        public Binding bindingDirect2() {
66.
                                                                                        整体核心
67.
            return BindingBuilder.bind(smsQueue()).to(fanoutOrderExchange()).with(""
                                                                                       01、目标
68.
        }
                                                                                       02、实现步骤
69.
70.
        @Bean
                                                                                        具体实现
        public Binding bindingDirect3() {
71.
                                                                                       03、生产者 🔻
72.
            return BindingBuilder.bind(emailQueue()).to(fanoutOrderExchange()).with(
73.
        }
74.
75. }
```

5、进行测试

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

                                                                                        04、定义消费者 🔻
2.
3. import com.xuexiangban.rabbitmq.springbootrabbitmqfanoutproducer.service.OrderSe
4. import org.junit.jupiter.api.Test;
5. import org.springframework.beans.factory.annotation.Autowired;
6. import org.springframework.boot.test.context.SpringBootTest;
7.
@SpringBootTest
9. class SpringbootRabbitmqFanoutProducerApplicationTests {
10.
11.
        @Autowired
12.
        OrderService orderService;
13.
                                      <u>></u>
14.
        @Test
15.
        public void contextLoads() throws Exception {
16.
17.
            for (int i = 0; i < 10; i++) {
18.
                Thread.sleep(1000);
19.
                Long userId = 100L + i;
20.
                Long productId = 10001L + i;
21.
                int num = 10;
                orderService.makeOrder(userId, productId, num);
22.
23.
            }
24.
        }
25.
26. }
```

04、定义消费者

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



1、创建生产者工程: sspringboot-rabbit

1、创建消费者工程: springboot-rabbitn

7、启动服务SpringbootRabbitmqFanou

2、在pom.xml中引入依赖

4: 定义订单的生产者

2、引入依赖pom.xml

4、消费者 - 邮件服务

5、消费者 - 短信服务

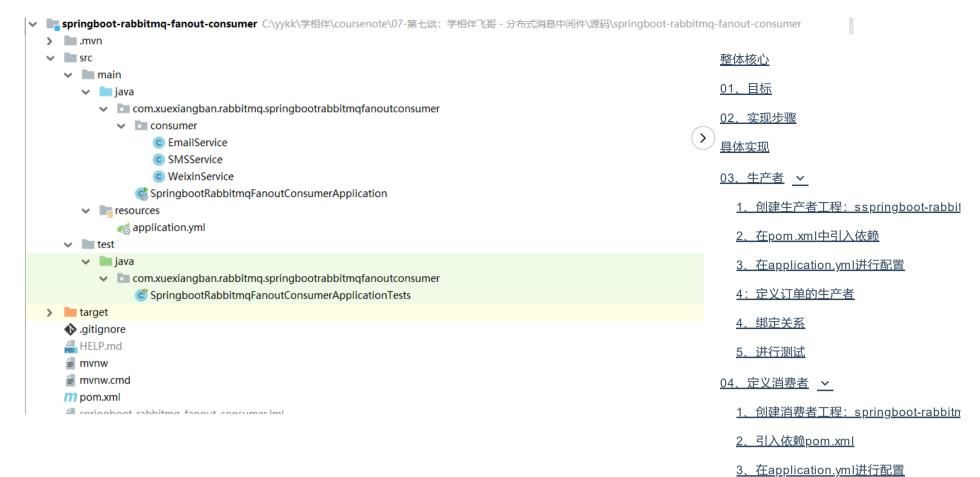
6、消费者 - 微信服务

3、在application.yml进行配置

4、绑定关系

5、进行测试

3、在application.yml讲行配置



2、引入依赖pom.xml

3、在application.yml进行配置

```
1. # 服务端口
2. server:
3.
     port: 8081
4.
5. # 配置rabbitmq服务
6. spring:
7.
     rabbitmq:
       username: admin
8.
9.
       password: admin
10.
       virtual-host: /
11.
       host: 47.104.141.27
12.
       port: 5672
```

4、消费者 - 邮件服务



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

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

2.
3. import org.springframework.amqp.core.ExchangeTypes;
                                                                                整体核心
4. import org.springframework.amqp.rabbit.annotation.*;
                                                                                01、目标
5. import org.springframework.stereotype.Component;
6.
                                                                                02、实现步骤
7. // bindings其实就是用来确定队列和交换机绑定关系
                                                                                具体实现
   @RabbitListener(bindings =@QueueBinding(
                                                                                03、生产者 ~
9.
           // 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"),
10.
                                                                                  1、创建生产者工程: sspringboot-rabbit
           // order.fanout 交换机的名字 必须和生产者保持一致
11.
                                                                                  2、在pom.xml中引入依赖
           exchange = @Exchange(value = "fanout_order_exchange",
12.
                  // 这里是确定的rabbitmq模式是:fanout 是以广播模式 、 发布订阅模式
                                                                                  3、在application.yml讲行配置
13.
14.
                  type = ExchangeTypes.FANOUT)
                                                                                  4: 定义订单的生产者
15. ))
                                                                                  4、绑定关系
16. @Component
                                                                                  5、进行测试
17. 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
18.
                                                                                04、定义消费者 💙
19.
       @RabbitHandler
                                                                                  1、创建消费者工程: springboot-rabbitm
20.
       public void messagerevice(String message){
                                                                                  2、引入依赖pom.xml
           // 此处省略发邮件的逻辑
21.
           System.out.println("email----->" + message);
22.
                                                                                  3、在application.yml进行配置
23.
       }
                                                                                  4、消费者 - 邮件服务
24. }
                                                                                  5、消费者 - 短信服务
                                                                                  6、消费者 - 微信服务
```

5、消费者 - 短信服务

```
    package com.xuexiangban.rabbitm

                                      ngbootrabbitmqfanoutconsumer.consum
                                   <
                                                                         <u>></u>
2.
3. import org.springframework.amqp.co.e.ExchangeTypes;
4. import org.springframework.amqp.rabbit.annotation.*;
import org.springframework.stereotype.Component;
6.
7. // 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。
9.
           value = @Queue(value = "sms.fanout.queue",autoDelete = "false"),
10.
           // order.fanout 交换机的名字 必须和生产者保持一致
11.
           exchange = @Exchange(value = "fanout_order_exchange",
12.
                  // 这里是确定的rabbitmq模式是:fanout 是以广播模式 、 发布订阅模式
13.
14.
                  type = ExchangeTypes.FANOUT)
15. ))
16. @Component
17. public class SMSService {
18.
19.
       // @RabbitHandler 代表此方法是一个消息接收的方法。该不要有返回值
       @RabbitHandler
20.
       public void messagerevice(String message){
21.
           // 此处省略发邮件的逻辑
22.
           System.out.println("sms----->" + message);
23.
24.
25. }
```

6、消费者 - 微信服务

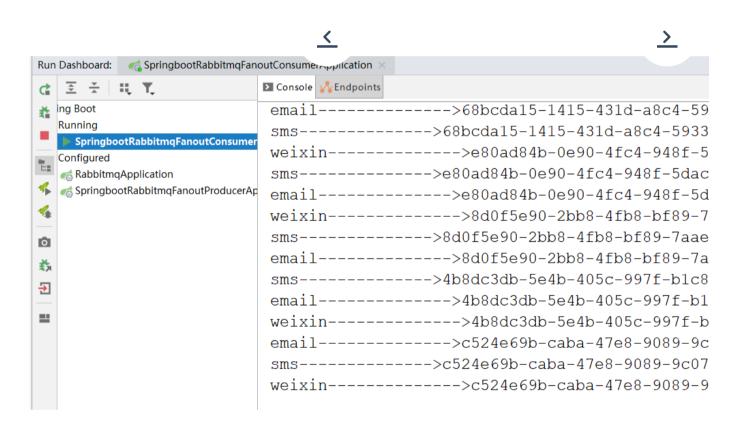


7、启动服务SpringbootRabbitmqFanou

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

2.
3. import org.springframework.amqp.core.ExchangeTypes;
                                                                                整体核心
4. import org.springframework.amqp.rabbit.annotation.*;
                                                                                01、目标
5. import org.springframework.stereotype.Component;
6.
                                                                                02、实现步骤
7. // bindings其实就是用来确定队列和交换机绑定关系
                                                                                具体实现
   @RabbitListener(bindings =@QueueBinding(
                                                                                03、生产者 ~
9.
          // 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"),
10.
                                                                                 1、创建生产者工程: sspringboot-rabbit
          // order.fanout 交换机的名字 必须和生产者保持一致
11.
                                                                                 2、在pom.xml中引入依赖
           exchange = @Exchange(value = "fanout_order_exchange",
12.
                                                                                 3、在application.yml讲行配置
                  // 这里是确定的rabbitmq模式是:fanout 是以广播模式 、 发布订阅模式
13.
14.
                  type = ExchangeTypes.FANOUT)
                                                                                 4: 定义订单的生产者
15. ))
                                                                                 4、绑定关系
16. @Component
                                                                                 5、进行测试
17. public class WeixinService {
18.
                                                                                19.
       // @RabbitHandler 代表此方法是一个消息接收的方法。该不要有返回值
                                                                                 1、创建消费者工程: springboot-rabbitm
       @RabbitHandler
20.
                                                                                 2、引入依赖pom.xml
       public void messagerevice(String message){
21.
          // 此处省略发邮件的逻辑
22.
                                                                                 3、在application.yml进行配置
           System.out.println("weixin----->" + message);
23.
                                                                                 4、消费者 - 邮件服务
24.
       }
                                                                                 5、消费者 - 短信服务
25. }
                                                                                 6、消费者 - 微信服务
```

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 <u>粤ICP备 - 2020109190</u>



7、启动服务SpringbootRabbitmqFanou