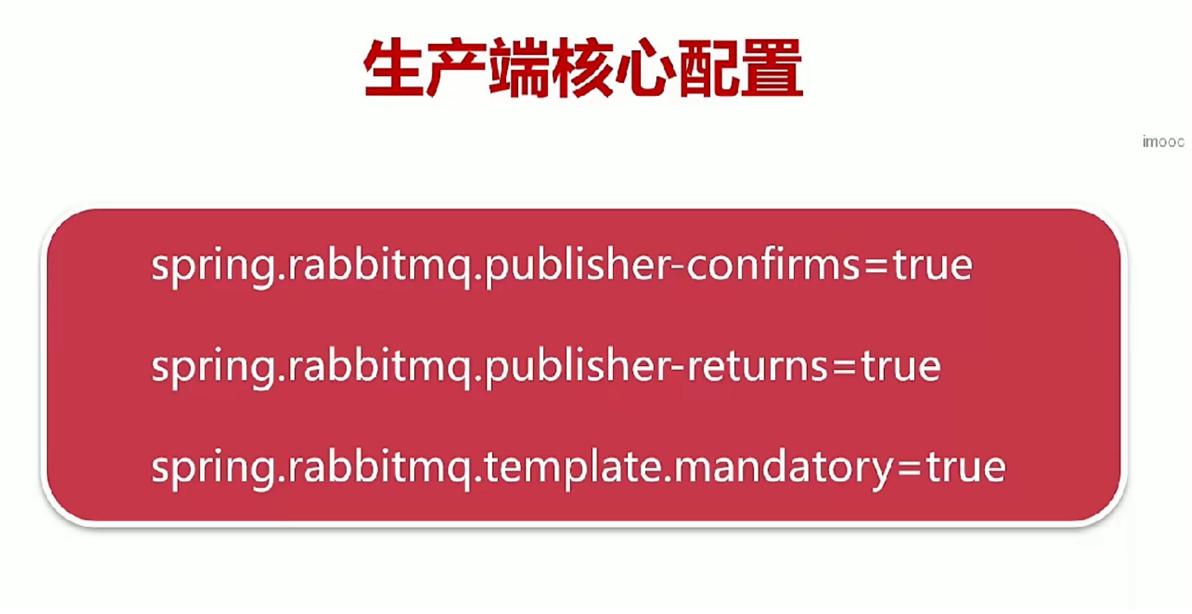
```
```

rabbitmq整合springboot

生产者



生产端核心配置

imooc

spring.rabbitmq.publisher-confirms=true

spring.rabbitmq.publisher-returns=true

spring.rabbitmq.template.mandatory=true

生产端核心配置

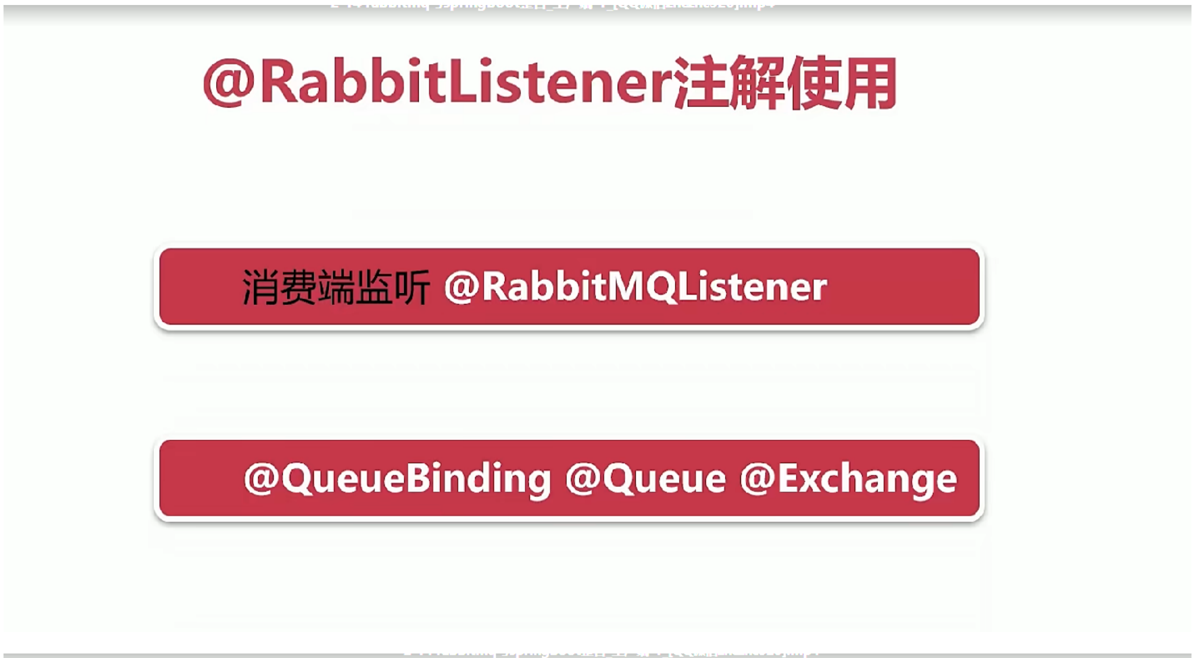
```
server.port=8001
server.servlet.context-path=/
spring.rabbitmq.addresses=192.168.1.101:5672
spring.rabbitmq.username=root
spring.rabbitmq.password=root
spring.rabbitmq.virtual-host=my_host

spring.rabbitmq.connection-timeout=15000
#是否启用消息确认模式
spring.rabbitmq.publisher-confirms=true

#设置return消息模式，注意要和mandatory一起配合使用
spring.rabbitmq.publisher-returns=true
spring.rabbitmq.template.mandatory=true
```

代码

```
package com.znc.rabbitmq.springbootrabbitproducer.component;

/**
 * @Author zhunc
 * @Date 2022/5/14 17:34
 */
```

```java
@Component
public class RabbitSender {
    @Autowired
    private RabbitTemplate rabbitTemplate;

    /**
     * 这里就是确认消息的回调监听接口，用于确认消息是否被broker所收到
     */
    final RabbitTemplate.ConfirmCallback confirmCallback = new
RabbitTemplate.ConfirmCallback() {
        /**
         *
         * @param correlationData 作为唯一的标识
         * @param ack 是否落盘成功
         * @param cause 失败的一些一次消息
         */
        @Override
        public void confirm(CorrelationData correlationData, boolean ack,
String cause) {
            System.err.println("消息结果: "+ ack + "  correlationData:" +
correlationData.getId() +" cause:"+ cause);
        }
     };

    /**
     * 对外发送消息的方法
     * @param message  具体消息内容
     * @param properties 额外的附带属性
     */
    public void send(Object message, Map<String, Object> properties) throws
 Exception{

        MessageHeaders messageHeaders = new MessageHeaders(properties);
        Message msg = MessageBuilder.createMessage(message, messageHeaders);

        rabbitTemplate.setConfirmCallback(confirmCallback);
        //指定业务唯一id
        CorrelationData correlationData = new
CorrelationData(UUID.randomUUID().toString());

        MessagePostProcessor messagePostProcessor = new MessagePostProcessor() {
            @Override
            public org.springframework.amqp.core.Message
postProcessMessage(org.springframework.amqp.core.Message message) throws
AmqpException {
                System.out.println("post to do:" + message);
                return message;
            }



        };
        rabbitTemplate.convertAndSend("exchange-1",
                 "springboot.rabbit", message, messagePostProcessor,
correlationData);

    }
}
```

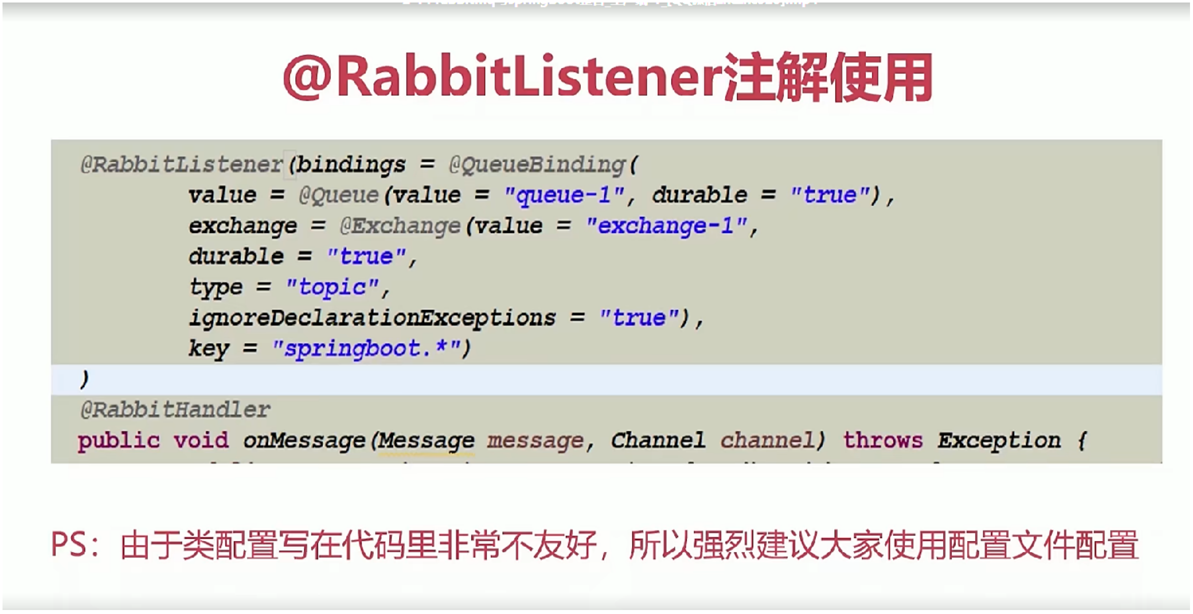
消费者



@RabbitListener注解使用

消费端监听 @RabbitMQListener

@QueueBinding @Queue @Exchange



@RabbitListener注解使用

```
@RabbitListener(bindings = @QueueBinding(
        value = @Queue(value = "queue-1", durable = "true"),
        exchange = @Exchange(value = "exchange-1",
        durable = "true",
        type = "topic",
        ignoreDeclarationExceptions = "true"),
        key = "springboot.*")
)
@RabbitHandler
public void onMessage(Message message, Channel channel) throws Exception {
```

PS：由于类配置写在代码里非常不友好，所以强烈建议大家使用配置文件配置

消费端核心配置

```
server.port=8002
server.servlet.context-path=/
spring.rabbitmq.addresses=192.168.1.101:5672
spring.rabbitmq.username=root
spring.rabbitmq.password=root
spring.rabbitmq.virtual-host=my_host
spring.rabbitmq.connection-timeout=15000

#表示消息消费成功后需要手动签收（ack）默认为auto
spring.rabbitmq.listener.simple.acknowledge-mode=manual
spring.rabbitmq.listener.simple.concurrency=5
spring.rabbitmq.listener.simple.max-concurrency=10
spring.rabbitmq.listener.simple.prefetch=1
```

代码:

```
package com.rabbit.springbootrabbitconsumer.component;

import com.rabbitmq.client.Channel;
import org.springframework.amqp.rabbit.annotation.*;
import org.springframework.amqp.support.AmqpHeaders;
import org.springframework.messaging.Message;
import org.springframework.stereotype.Component;

/**

 * @Author zhunc
 * @Date 2022/5/14 18:28
   */
   @Component
   public class RabbitReceive {
   /**
    * 组合使用监听
    * @RabbitListener  @QueueBinding @Queue @Exchange
    * @param message
    * @param channel
      */
      @RabbitListener(bindings = @QueueBinding(
          value = @Queue(value = "queue-1", durable = "true"),
          exchange = @Exchange(value = "exchange-1", durable = "true", type =
"topic"),
          ignoreDeclarationExceptions = "true",
          key = "springboot.*")
      )
      @RabbitHandler
      public void onMessage(Message message, Channel channel) throws Exception {
      //1.收到消息后进行业务端消费处理
      System.out.println("-------------------------");
      System.out.println("消费消息: "+ message.getPayload());
      //2.处理成功后 获取deliveryTag 并进行手工的ACK操作，因为我们配置文件里配置是手工签收
      Long deliveryTag = (Long)
message.getHeaders().get(AmqpHeaders.DELIVERY_TAG);
      channel.basicAck(deliveryTag, true);
      }
      }
```