Active MQ 中的请求 - 响应模式（Request-Response）在 Spring 与 Spring Boot 中的应用

作者：DongXiaokai0819

原文链接：https://ld246.com/article/1582810519873

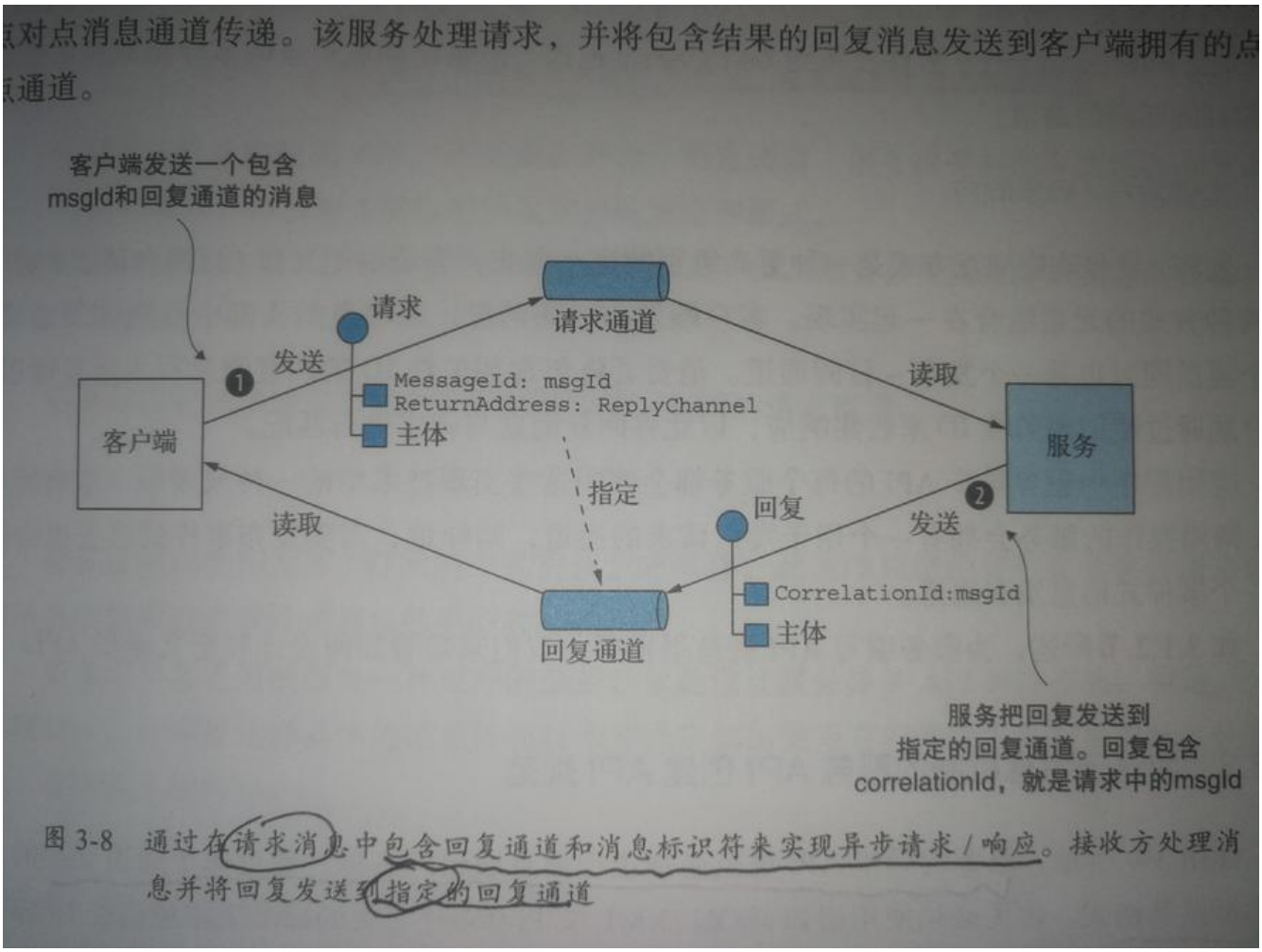
来源网站：链滴

Active MQ中的请求-响应模式（Request-Response）在Spring与Spring Boot中的应用

我在基于《异步消息模式的通信》一文中提到过这个模式，本文整理了一下该模式在Active MQ中的用，我们先来回顾一下请求-响应模式

请求/响应和异步请求/响应

客户端和服务端通过交换一对消息来实现异步请求/响应方式的交互。



    客户端必须告知服务送回复消息的位置，并且必须将回复消息与请求匹配。

    客户端发送具有**复通道头部**的命令式消息。服务端将回复消息写入回复通道，该回复消息包含与消息标识符具有相同得**相关性ID**。客户端使用相关性ID将回复消息与请求进行匹配。

注：请求-响应模式并不是JMS规范系统默认提供的一种通信方式。

在Spring中实现请求-响应模式

借助上次的Spring项目：

Spring整合Active MQ

    所谓的请求-响应模式就是在发送端（生产者）加入监听器来接收返回的消息，在接收端（消费者）加入发送消息的JmsTeplate。

发送端（生产者）

在Spring的配置文件applicationContext.xml文件中加入**对消费者返回消息所应答的监听器。**

```
<!--接收消费者应答的监听器-->
<jms:listener-container destination-type="queue" container-type="default"
```

```
                    connection-factory="connectionFactory" acknowledge="auto">
    <jms:listener destination="tempqueue" ref="getResponse"></jms:listener>
</jms:listener-container>
```

编写发送端监听器：

```
@Component
public class GetResponse implements MessageListener {
    public void onMessage(Message message) {
        String textMsg = null;
        try {
            textMsg = ((TextMessage) message).getText();
            System.out.println("GetResponse accept msg : " + textMsg);
        } catch (JMSException e) {
            e.printStackTrace();
        }
    }
}
```

然后我们需要在发送端（生产者）发送消息时，将回复消息的位置（**指定消息回复通道**）和**相关性ID**入消息。

```
@Autowired
@Qualifier("jmsQueueTemplate")
private JmsTemplate jmsTemplate;
@Autowired
private GetResponse getResponse;//注入监听回复消息的监听器

//json
public void send(String queueName, final String message) {
    jmsTemplate.send(queueName, new MessageCreator() {
        public Message createMessage(Session session) throws JMSException {
            Message msg = session.createTextMessage(message);
            //以下配置，告诉消费者如何应答
            //创建临时目的地
            Destination tempDst = session.createTemporaryQueue();
            //创建回复消息的消费者
            MessageConsumer responseConsumer = session.createConsumer(tempDst);
            //设置监听回复消息的监听器
            responseConsumer.setMessageListener(getResponse);
            //指定消息回复通道
            msg.setJMSReplyTo(tempDst);
            //生成相关性ID
            String uid = System.currentTimeMillis()+"";
            //设置相关性ID
            msg.setJMSCorrelationID(uid);
            return msg;
        }
    });
```

接收端（消费者）

在Spring的配置文件applicationContext.xml文件中加入**JmsTemplate**的bean配置。

```
<bean id="jmsConsumerQueueTemplate" class="org.springframework.jms.core.JmsTemplate
>
    <constructor-arg ref="connectionFactory"></constructor-arg>
    <!-- 队列模式-->
    <property name="pubSubDomain" value="false"></property>
</bean>
```

编写接收端（消费者）的回复消息逻辑，同发送端的代码差不多。

```
@Component
public class ReplyTo {

    @Autowired
    @Qualifier("jmsConsumerQueueTemplate")
    private JmsTemplate jmsTemplate;

    public void send(final String consumerMsg, Message producerMessage)
            throws JMSException {
        //producerMessage.getJMSReplyTo()获取回复消息通道地址
        jmsTemplate.send(producerMessage.getJMSReplyTo(),
                new MessageCreator() {
                    public Message createMessage(Session session)
                            throws JMSException {
                        Message msg
                                = session.createTextMessage("ReplyTo " + consumerMsg);
                        return msg;
                    }
                });
    }
}
```

接收端中使用ReplyTo：

```
@Component
public class QueueReceiver1 implements MessageListener {
    @Autowired
    private ReplyTo replyTo;

    public void onMessage(Message message) {
        try {
            String textMsg = ((TextMessage) message).getText();
            System.out.println("QueueReceiver1 accept msg : " + textMsg);
            // do business work;
            replyTo.send(textMsg,message);
        } catch (JMSException e) {
            e.printStackTrace();
        }
    }
}
```

在Spring Boot中实现请求-响应模式

借助上次的Spring Boot项目：

Spring Boot整合Active MQ

在Spring Boot中的用法其实和Spring中没有什么区别，毕竟都是一套东西。

发送端（生产者）

```
@Service
public class ProducerR {

    @Autowired
    private JmsMessagingTemplate jmsTemplate;

    // 发送消息，destination是发送到的队列，message是待发送的消息
    public void sendMessage(Destination destination, final String message){
        jmsTemplate.convertAndSend(destination, message);

    }

    //监听回复通道队列
    @JmsListener(destination = "out.replyTo.queue")
    public void consumerMessage(String text){
        System.out.println("从out.replyTo.queue收到报文"+text);
    }
}
```

接收端（消费者）

```
@Component
public class ConsumerR {
    // 使用JmsListener配置消费者监听的队列，其中text是接收到的消息
    @JmsListener(destination = "springboot.replyto.queue")
    @SendTo("out.replyTo.queue")//定义一个正式队列，回复
    public String receiveQueue(String text) {
        //处理业务逻辑
        System.out.println(this.getClass().getName()+" 收到的报文为:"+text);
        //返回消息内容
        return "Hello,I watch you";
    }
}
```

测试

```
@Autowired
private ProducerR producerR;

@Test
public void testReplyTo() {
    Destination destination
            = new ActiveMQQueue("springboot.replyto.queue");
    for(int i=0; i<3; i++){
        producerR.sendMessage(destination,
                "NO:"+i+";my name is Mark!!!");
```

```
    }
}
```

原文链接：Active MQ 中的请求 - 响应模式（Request-Response）在 Spring 与 Spring Boot 中的应用