SpringBoot集成RabbitMQ

## 1:pom.xml

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-amqp</artifactId>

</dependency>

2:application.properties

server.port=8000

#服务器配置

spring.application.name=rabbitmq-hello-sending

#rabbitmq连接参数

spring.rabbitmq.host=localhost

spring.rabbitmq.port=5672

spring.rabbitmq.username=root

spring.rabbitmq.password=ilovelaohan

# 开启发送确认

spring.rabbitmq.publisher-confirms=true

# 开启发送失败退回

spring.rabbitmq.publisher-returns=true

# 开启ACK

spring.rabbitmq.listener.direct.acknowledge-mode=manual

spring.rabbitmq.listener.simple.acknowledge-mode=manual

3:配置消息队列

@Configuration

public class RabbitConfig {

@Bean

public Queue QueueA() {

return new Queue("hello");

}

@Bean

public Queue QueueB() {

return new Queue("helloObj");

}

/\*\*

\* Fanout 就是我们熟悉的广播模式或者订阅模式，给Fanout交换机发送消息，绑定了这个交换机的所有队列都收到这个消息。

\* @return

\*/

@Bean

FanoutExchange fanoutExchange() {

return new FanoutExchange("ABExchange");

}

@Bean

Binding bindingExchangeA(Queue QueueA, FanoutExchange fanoutExchange) {

return BindingBuilder.bind(QueueA).to(fanoutExchange);

}

@Bean

Binding bindingExchangeB(Queue QueueB, FanoutExchange fanoutExchange) {

return BindingBuilder.bind(QueueB).to(fanoutExchange);

}

}

4：发送消息

Spring AMQP提供了一个发送和接收消息的操作模板类AmqpTemplate。 AmqpTemplate它定义包含了发送和接收消息等的一些基本的操作功能。RabbitTemplate是AmqpTemplate的一个实现。

RabbitTemplate支持消息的确认与返回，为了返回消息，RabbitTemplate 需要设置mandatory 属性为true,并且CachingConnectionFactory 的publisherReturns属性也需要设置为true。返回的消息会根据它注册的RabbitTemplate.ReturnCallback setReturnCallback 回调发送到给客户端，

一个RabbitTemplate仅能支持一个ReturnCallback 。

为了确认Confirms消息, CachingConnectionFactory 的publisherConfirms 属性也需要设置为true，确认的消息会根据它注册的RabbitTemplate.ConfirmCallback setConfirmCallback回调发送到给客户端。一个RabbitTemplate也仅能支持一个ConfirmCallback.

@Service

public class SendService implements RabbitTemplate.ReturnCallback {

@Resource

private RabbitTemplate rabbitTemplate;

public void send() {

String context = "你好现在是 " + new Date() +"";

System.out.println("HelloSender发送内容 : " + context);

// this.rabbitTemplate.setConfirmCallback(this);

this.rabbitTemplate.setReturnCallback(this);

this.rabbitTemplate.setConfirmCallback((correlationData, ack, cause) -> {

if (!ack) {

System.out.println("HelloSender消息发送失败" + cause + correlationData.toString());

} else {

System.out.println("HelloSender 消息发送成功 ");

}

});

this.rabbitTemplate.convertAndSend("hello", context);

}

// public void sendObj() {

// MessageObj obj = new MessageObj();

// obj.setACK(false);

// obj.setId(123);

// obj.setName("zhangsan");

// obj.setValue("data");

// System.out.println("发送 : " + obj);

// this.rabbitTemplate.convertAndSend("helloObj", obj);

// }

@Override

public void returnedMessage(Message message, int replyCode, String replyText, String exchange, String routingKey) {

// TODO Auto-generated method stub

System.out.println("sender return success" + message.toString()+"==="+replyCode+"==="+replyText+"==="+exchange);

}

}

5：接受消息

@Component

@RabbitListener(queues = "hello")

public class ReceiveService {

@RabbitHandler

public void process(String hello,Channel channel, Message message) throws IOException {

System.out.println("HelloReceiver收到 : " + hello +"收到时间"+new Date());

try {

//告诉服务器收到这条消息 已经被我消费了 可以在队列删掉 这样以后就不会再发了 否则消息服务器以为这条消息没处理掉 后续还会在发

channel.basicAck(message.getMessageProperties().getDeliveryTag(),false);

System.out.println("receiver success");

} catch (Exception e) {

e.printStackTrace();

//丢弃这条消息

//channel.basicNack(message.getMessageProperties().getDeliveryTag(), false,false);

System.out.println("receiver fail");

}

}

}

消息确认  因为我在属性配置文件里面开启了ACK确认 所以如果代码没有执行ACK确认 你在RabbitMQ的后台会看到消息会一直留在队列里面未消费掉 只要程序一启动开始接受该队列消息的时候 又会收到

channel.basicAck(message.getMessageProperties().getDeliveryTag(),false);

//消息的标识，false只确认当前一个消息收到，true确认所有consumer获得的消息

channel.basicAck(message.getMessageProperties().getDeliveryTag(), false);

//ack返回false，并重新回到队列，api里面解释得很清楚

channel.basicNack(message.getMessageProperties().getDeliveryTag(), false, true);

//拒绝消息

channel.basicReject(message.getMessageProperties().getDeliveryTag(), true);