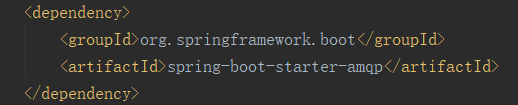
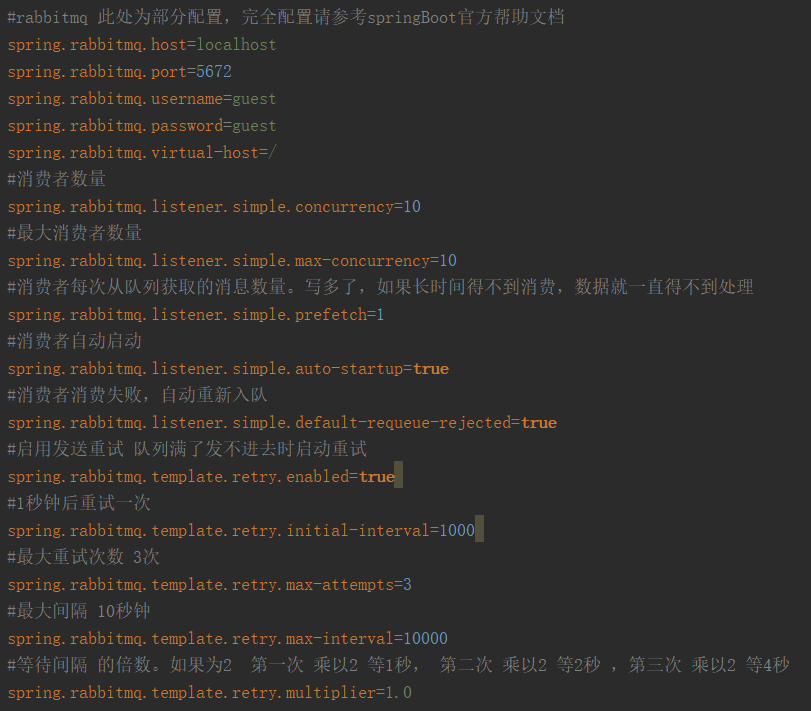
1. 添加依赖：



Spring AMQP 是基于 Spring 框架的AMQP消息解决方案，提供模板化的发送和接收消息的抽象层，提供基于消息驱动的 POJO的消息监听等，很大方便我们使用RabbitMQ程序的相关开发。

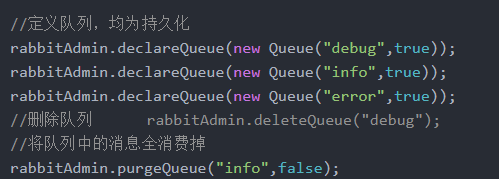
1. 配置Rabbit相关参数：



1. 消息队列的操作：RabbitAdmin与rabbitTemplate
2. rabbitAdmin: 该类封装了对 RabbitMQ 的管理操作
3. Exchange操作：



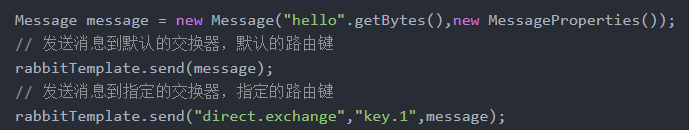
1. Queue操作：



1. Binding操作：

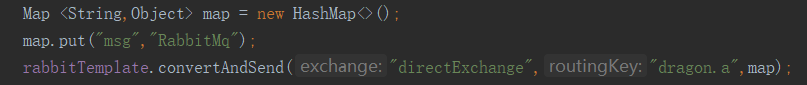


1. rabbitTemplate: Spring AMQP 提供了 RabbitTemplate 来简化 RabbitMQ 发送和 接收消息操作
2. 发送消息：
   * 1. send():



* + 1. convertAndSend():

（自动 Java 对象包装成 Message 对象）

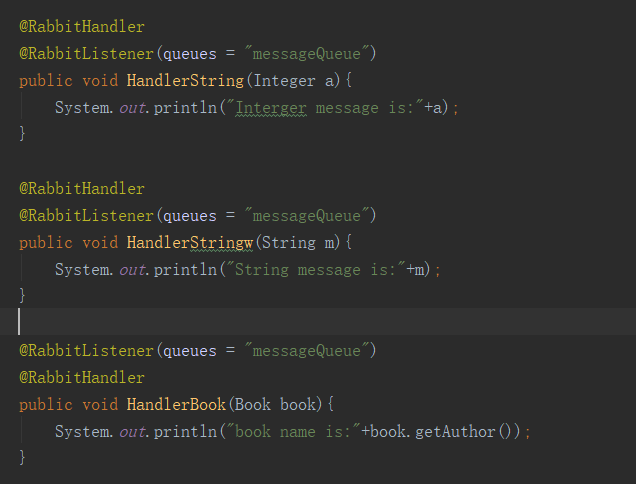


1. 接收消息：

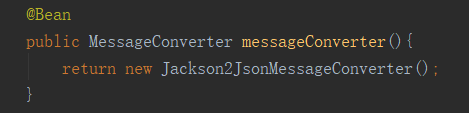




1. Rabbitmq注解：
2. @EnableRabbit:启动类使用@EnableRabbit启用@RabbitListener
3. @RabbitListener: 用于注册Listener时使用的信息; 使用 @RabbitListener 注解标记方法，当监听到队列中有消息时则会进行接收并处理
4. @RabbitHandler: @RabbitListener 和 @RabbitHandler结合使用，@RabbitListener标注在类上面表示当有收到消息的时候，就交给 @RabbitHandler 的方法处理，具体使用哪个方法处理，根据 MessageConverter 转换后的参数类型不同类型的消息使用不同的方法来处理。



1. 将数据转换成json发送：



1. 消息确认机制：

（1）消费者在处理消息的时候偶尔会失败或者有时会直接崩溃掉。而且网络原因也有可能引起各种问题，对于此AMQP有两种处理方式：

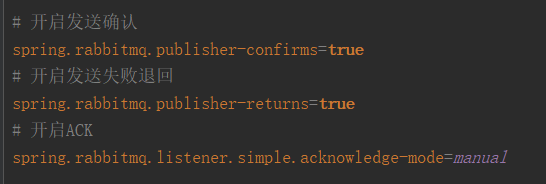
a.自动确认模式（automatic acknowledgement model）：当RabbbitMQ将消息发送给应用后，消费者端自动回送一个确认消息，此时RabbitMQ删除此消息。

b.显式确认模式（explicit acknowledgement model）：消费者收到消息后，可以在执行 一些逻辑后，消费者自己决定什么时候发送确认回执（acknowledgement），RabbitMQ收到回执后才删除消息，这样就保证消费端不会丢失消息

如果一个消费者在尚未发送确认回执的情况下挂掉了，那么消息会被重新放入队列，并且在还有其他消费者存在于此队列的前提下，立即投递给另外一个消费者。如果当时没有可用的消费者了，消息代理会死等下一个注册到此队列的消费者，然后再次尝试投递。(RabbitMQ里的消息是不会过期。当消费者挂掉后，RabbitMQ会不断尝试重推。所有单个消息的推送可能花费很长的时间)

（2）实现：

a.手动应答配置：



b.实现代码：

void basicAck(long deliveryTag, boolean multiple) throws IOException;

第一个参数deliveryTag：发布的每一条消息都会获得一个唯一的deliveryTag，(任何channel上发布的第一条消息的deliveryTag为1，此后的每一条消息都会加1)，deliveryTag在channel范围内是唯一的(deliveryTag获取方法：message.getMessageProperties().getDeliveryTag())

第二个参数multiple：批量确认标志。如果值为true，则执行批量确认，此deliveryTag之前收到的消息全部进行确认; 如果值为false，则只对当前收到的消息进行确认



1. 拒绝消息：

void basicReject(long deliveryTag, boolean requeue) throws IOException;

deliveryTag: 发布的每一条消息都会获得一个唯一的deliveryTag，deliveryTag在channel范围内是唯一的

requeue: 表示如何处理这条消息，如果值为true，则重新放入RabbitMQ的发送队列，如果值为false，则通知RabbitMQ销毁这条消息



b. void basicNack(long deliveryTag, boolean multiple, boolean requeue) throws IOException;

对多条消息拒绝

第一个参数deliveryTag：发布的每一条消息都会获得一个唯一的deliveryTag，deliveryTag在channel范围内是唯一的

第二个参数multiple：批量确认标志。如果值为true，包含本条消息在内的、所有比该消息deliveryTag值小的 消息都被拒绝了（除了已经被 ack 的以外）;如果值为false，只拒绝三本条消息

第三个参数requeue：表示如何处理这条消息，如果值为true，则重新放入RabbitMQ的发送队列，如果值为false，则通知RabbitMQ销毁这条消息

1. 设置预取消息数量：

RabbitMQ收到消息后，就向消费者送。但是如果消息过多，且消息的数量超过了消息者处理能力从而导致其崩溃。此时我们可以通过prefetchCount 限制每个消费者在收到下一个确认回执前一次可以最大接受多少条消息。即如果设置prefetchCount =1，RabbitMQ向这个消费者发送一个消息后，再这个消息的消费者对这个消息进行ack之前，RabbitMQ不会向这个消费者发送新的消息

channel.basicQos(1);

1. 消息分配策略：

如果同一个队列，有多个消费者消费这个队列。RabbitMQ默认是按照轮询的策略发送消息，即发送的顺序是消费者1，消费者2，消费者1，消费者2…。所以平均下来，每个消费者消费的消息数量几乎相同。

RabbitMQ之消息持久性、确认机制、拒绝、预取数量、分配策略：

https://blog.csdn.net/hry2015/article/details/79078312

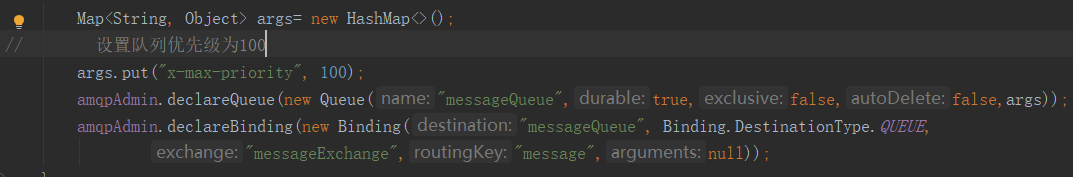
1. 设置优先级：

重要：只有当消费者不足，不能及时进行消费的情况下，优先级队列才会生效；

同时队列、消息上均要设置优先级才可生效

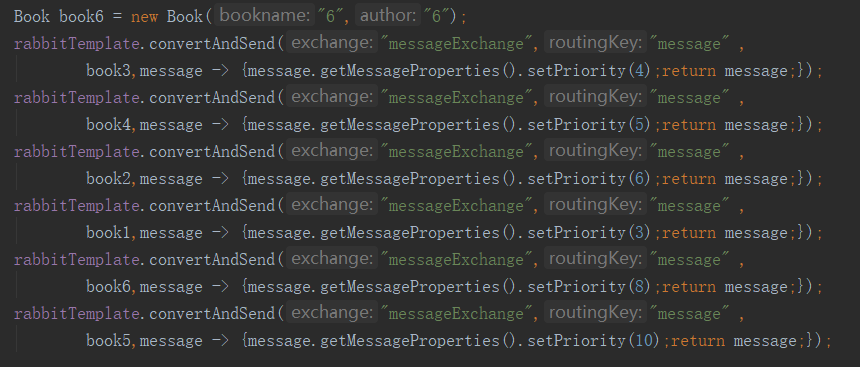
1. 队列：

队列优先级只能声明一次，不可改变



1. 消息：

优先级高的消息优先被消费



设置成功后消息按照优先级顺序排列：

