

@Autowired  
private RabbitTemplate rabbitTemplate;

convertAndSend 发消息

receiveAndConvert 读消息

messageConverter 可转换消息格式

@EnableRabbit+@RabbitListener(queues = "atguigu.news")可监听消息



可创建exchange，queue和 绑定规则等

1. Linux 安装
2. 下载otp\_src\_19.3.tar.gz
3. tar -zxvf otp\_src\_19.3.tar.gz
4. cd otp\_src\_19.3
5. ./configure --enable-smp-support --enable-threads --enable-sctp --enable-kernel-poll --enable-hipe --with-ssl
6. yum install gcc glibc-devel make ncurses-devel openssl-devel autoconf
7. yum install unixODBC unixODBC-devel
8. make && make install
9. ln -s /usr/local/erlang/bin/erl /usr/local/bin/
10. vim /etc/profile

ERL\_HOME=/usr/local/erlang

PATH=$ERL\_HOME/bin:$PATH

export ERL\_HOME PATH

1. source /etc/profile
2. 下载rabbitmq-server-3.7.2-1.el7.noarch.rpm
3. rpm -ivh --nodeps rabbitmq-server-3.7.2-1.el7.noarch.rpm
4. systemctl stop firewalld
5. 执行插件rabbitmq-plugins enable rabbitmq\_management
6. rabbitmq-server
7. 创建一个用户 rabbitmqctl add\_user root root
8. 赋管理员权限 rabbitmqctl set\_user\_tags root administrator
9. 赋予访问权限 rabbitmqctl set\_permissions -p / root '.\*' '.\*' '.\*'
10. 启用WEB管理 rabbitmq-plugins enable rabbitmq\_management
11. 开机启动 chkconfig rabbitmq-server on

遇到错误执行

chown rabbitmq:rabbitmq /var/lib/rabbitmq/.erlang.cookie

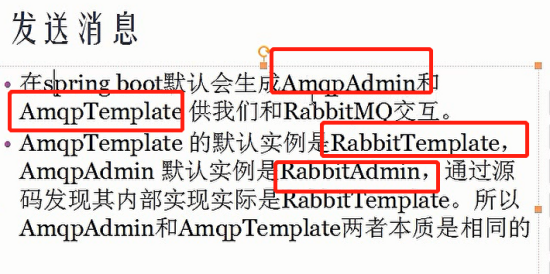
1. 与springboot整合
   1. 添加依赖

<dependency>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-starter-amqp</artifactId>  
</dependency>

* 1. rabbitmq配置在application.properties里

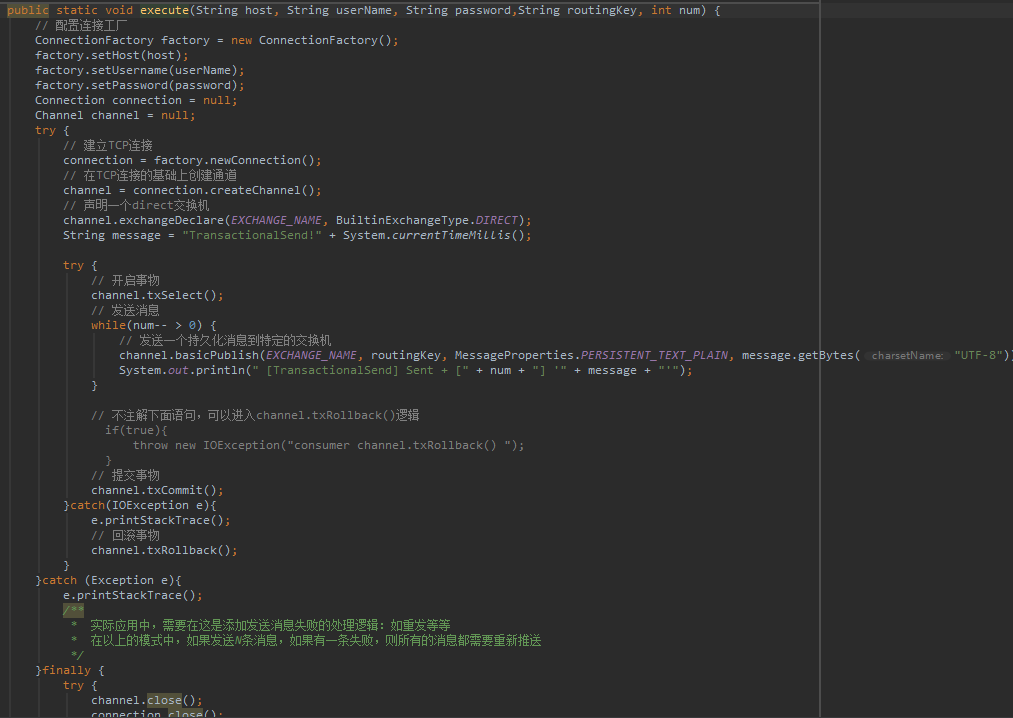
#rabbitmq连接参数  
spring.rabbitmq.host=192.168.0.113  
spring.rabbitmq.port=5672  
spring.rabbitmq.username=root  
spring.rabbitmq.password=root  
# 开启发送确认  
spring.rabbitmq.publisher-confirms=**true**# 开启发送失败退回  
spring.rabbitmq.publisher-returns=**true**# 开启ACK  
spring.rabbitmq.listener.direct.acknowledge-mode=*manual*spring.rabbitmq.listener.simple.acknowledge-mode=*manual*

* 1. rabbittemplate



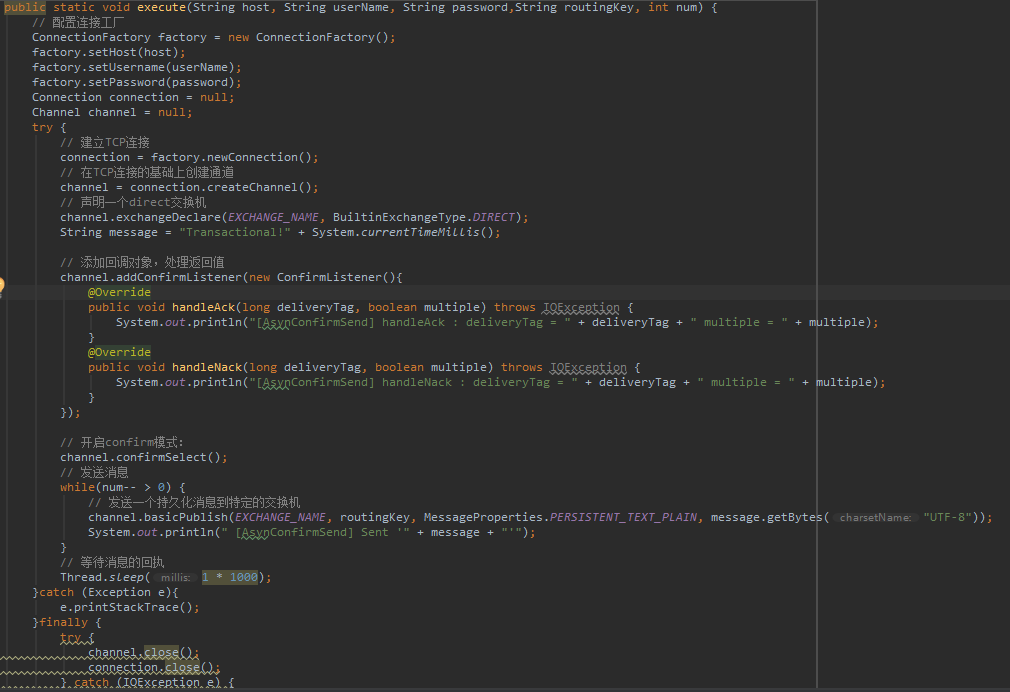
* 1. 写config,消费者和实现者 ，[简单实例](rabbitmqspringboot)
  2. [综合实例](youfanmsgqueueserv)

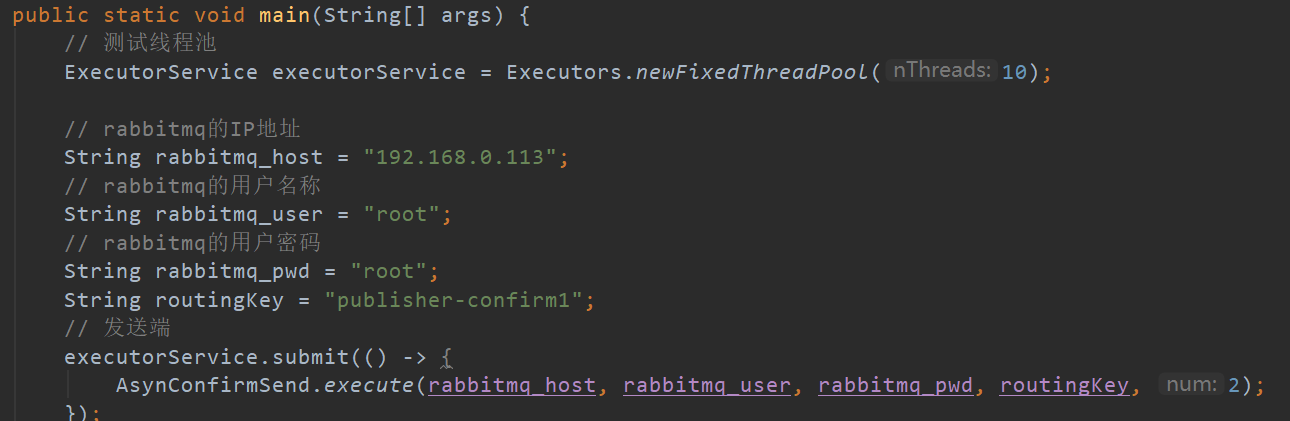
1. rabbitmq事务：提供了两种实现transaction(强一致性)和confirm。confirm分为两种：ConfirmListener和waitForConfirms
   1. 同步方式(transaction)



* 1. 异步方式(confirm)

发送端





消费端：

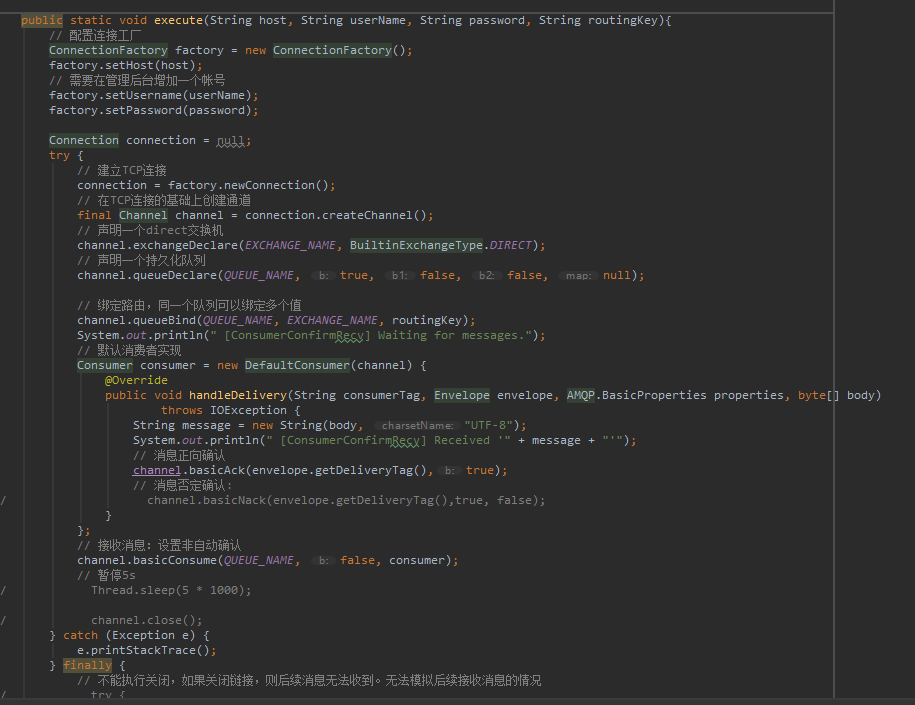
确认消息是否被有效消费。publisher端目前并没有提供监听事件，但提供了应答机制来保证消息被成功消费，应答方式

basicAck：成功消费，消息从队列中删除

basicNack：requeue=true，消息重新进入队列，false被删除

basicReject：等同于basicNack

basicRecover：消息重入队列，requeue=true，发送给新的consumer，false发送给相同的consumer





这里必须需要消费先启动，否则队列没有创建，而消息是放在这个队列里的,自然接收不到消息,并且，如果没有队列的话，队列没法和交换器绑定。