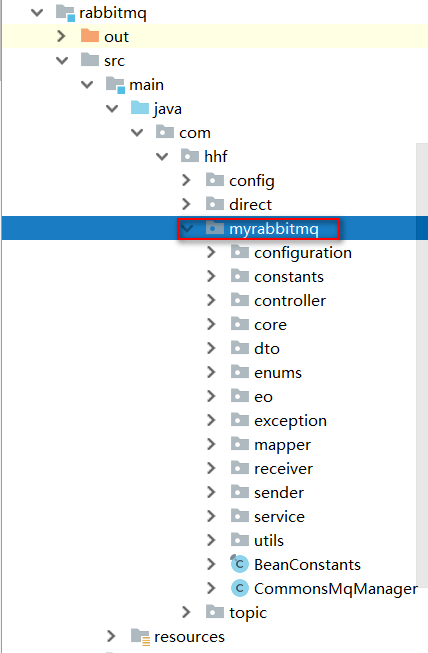
# 一、模块结构图：

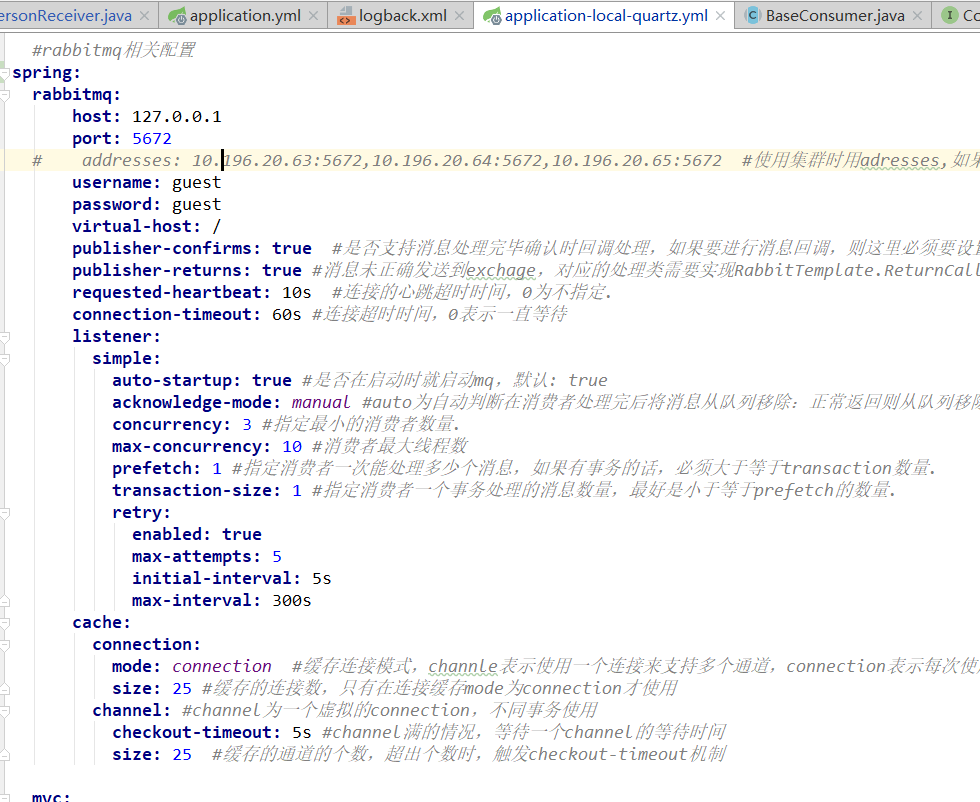


# 二、相关配置：

1、application.yml文件中配置相关配置：

**com:  
 hhf:  
 rabbitmq:  
 enabled: true** *# true为启动common中rabbitmq组件 本系统必须启用*

2、application-xxx.yml文件中配置相关配置：



|  |
| --- |
| *#rabbitmq相关配置* **spring:  rabbitmq:  host:** 127.0.0.1  **port:** 5672  *# addresses: 10.196.20.63:5672,10.196.20.64:5672,10.196.20.65:5672 #使用集群时用adresses,如果有F5或HAProxy负载均衡，把负载ip写在第一个* **username:** guest  **password:** guest  **virtual-host:** /  **publisher-confirms: true** *#是否支持消息处理完毕确认时回调处理，如果要进行消息回调，则这里必须要设置为true，消息发送方需要实现RabbitTemplate.ConfirmCallback* **publisher-returns: true** *#消息未正确发送到exchage，对应的处理类需要实现RabbitTemplate.ReturnCallback* **requested-heartbeat:** 10s *#连接的心跳超时时间，0为不指定.* **connection-timeout:** 60s *#连接超时时间，0表示一直等待* **listener:  simple:  auto-startup: true** *#是否在启动时就启动mq，默认: true* **acknowledge-mode:** *manual #auto为自动判断在消费者处理完后将消息从队列移除：正常返回则从队列移除，异常则不移除。* **concurrency:** 3 *#指定最小的消费者数量.* **max-concurrency:** 10 *#消费者最大线程数* **prefetch:** 1 *#指定消费者一次能处理多少个消息，如果有事务的话，必须大于等于transaction数量.* **transaction-size:** 1 *#指定消费者一个事务处理的消息数量，最好是小于等于prefetch的数量.* **retry:  enabled: true  max-attempts:** 5  **initial-interval:** 5s  **max-interval:** 300s  **cache:  connection:  mode:** *connection #缓存连接模式，channle表示使用一个连接来支持多个通道，connection表示每次使用,rabbitmq使用集群时，应该使用connection模式* **size:** 25 *#缓存的连接数，只有在连接缓存mode为connection才使用* **channel:** *#channel为一个虚拟的connection，不同事务使用* **checkout-timeout:** 5s *#channel满的情况，等待一个channel的等待时间* **size:** 25 *#缓存的通道的个数，超出个数时，触发checkout-timeout机制* |

3、持久化相关配置（相关数据库表）：

**本项目是将消息持久化到数据库中，相关数据库表如下：**

**b\_mq\_fail\_consumer 消费失败表**

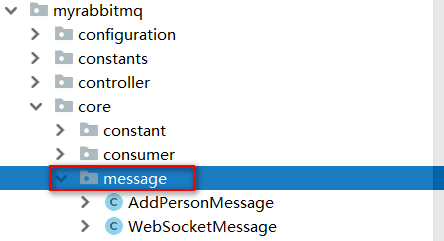
**b\_mq\_handling\_consumer 正在消费表（里面存储的是正在消费的数据，动态变化，持久化出现异常这张表中才会有数据，否则都是临时数据）**

**b\_mq\_message\_producer 生产者表**

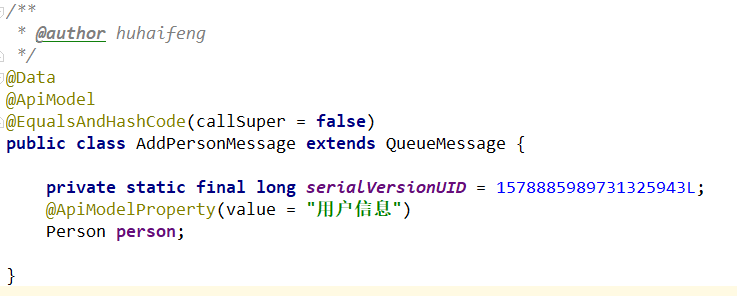
**b\_mq\_success\_consumer 消费成功表**

# 三、模块详细解析：

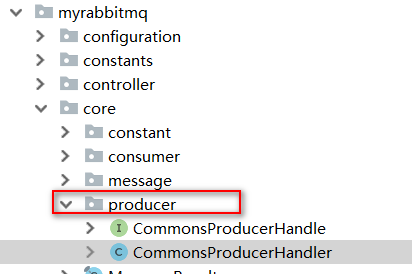
1、myrabbbitmq.core.message包：



里面存放的是发送消息的类，继承消息基类：QueueMessage



2、myrabbbitmq.core.producer包：

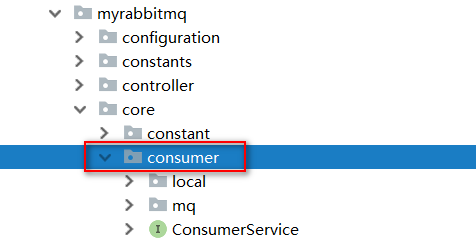


（1）封装消息的生产者发送消息和持久化数据到生产者数据库表中。（b\_mq\_message\_producer）

（2）封装消息的重发和持久化数据到数据库中。

清空b\_mq\_fail\_consumer表 🡪 往b\_mq\_success\_consumer表中插入一条记录。

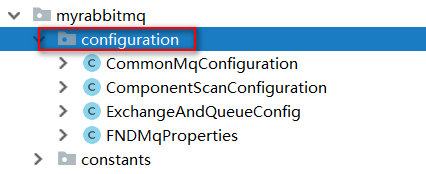
3、myrabbbitmq.core.consumer包：



**（1）ConsumerService**：消费者的逻辑接口，所有的自定义消费者都需实现此类。抽象类BaseConsumer实现了此类，所以自定义消费者类直接继承BaseConsumer抽象类即可。

**（2）BaseConsumer**：消费者的基类，所有自定义消费者必须继承此类，实现handle()方法，在方法中写具体消费逻辑。此类实现了onMessage()方法，此方法用户实现消费者数据持久化操作以及**确认消费消息机制的实现**，数据在**mq\_handling\_consumer表**与 **b\_mq\_success\_consumer** 或**b\_mq\_fail\_consumer**表中流转。

4、myrabbitmq.configuration包：

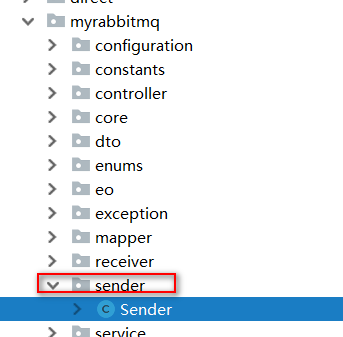


**CommonMqConfiguration**：rabbitmq封装类相关配置文件。

**ExchangeAndQueueConfig**：交换机与队列配置类，定义队列、交换机以及对应的绑定信息。

**FNDMqProperties**：yml文件中的配置注入。

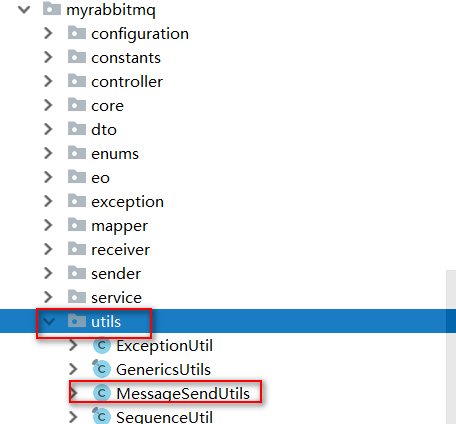
5、myrabbitmq.sender包：



**Sender**:发送消息类，新的需求在类中添加新的方法即可，在方法里面调用生产者生成消息方法，如下：



6、myrabbitmq.util包：

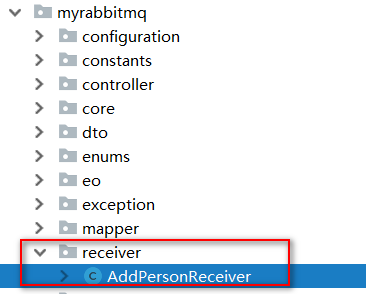


**MessageSendUtils：**消息发送工具类，在Sender中添加了新的需求方法，需在此工具类中添加对应的方法。如下：



**先获得发送对象Sender，再需求调用对应的方法。**

7、myrabbitmq.receiver包：



**AddPersonReceiver：**自定义消费者，每一个新的需求应该在此包中添加对应的一个类，继承消费者基类BaseConsumer，实现handle()方法，写消费的具体逻辑，如回写信息给关联系统，具体逻辑在此方法中编写，如下：

