交接文档

# 接口模块

## 简介

接口模块是对跨部门通信接口的一个封装，主要对WS、HTTP、MQ（RocketMQ，IBMMQ）等形式接口进行了统一调用、统一记录日志。

从数据流向可以分为：接收数据接口、发送数据接口

从数据流转方式分为：WS、HTTP、MQ等

从数据格式分为：Object、XML、JSON等

对于一个接口使用者来说，调用一个接口就是一个将一个输入参数输入后得到一个输出即刻（有时输入输出可能是空，如null,void）。所以我们设计了public interface Interface<D,R> 接口，这样可以保证了我们的**统一调用**。

根据数据流向我们将接口分为接收数据型和发送数据型。分别对应public abstract class DefaultReceiveService<D,R,IT extends IData,BDT,BRT>和public abstract class DefaultSendService<D,R,I extends IData,IT,IRT,IRO> ，这这两个类中，各自实现了一个模板方法，一个是针对发送的，一个是针对接收的。通过模板方法可以进行统一的日志记录和异常处理。

根据数据流转方式我们在继承数据流向上进行了功能增加，具体体现在以下几个类：public abstract class CxfWsSendService<D,R,I extends IData,IT,IRT,IRO>

public abstract class IbmmqSendService<D,R,I extends IData>

public abstract class RocketMQSendService<D,R,I extends IData>

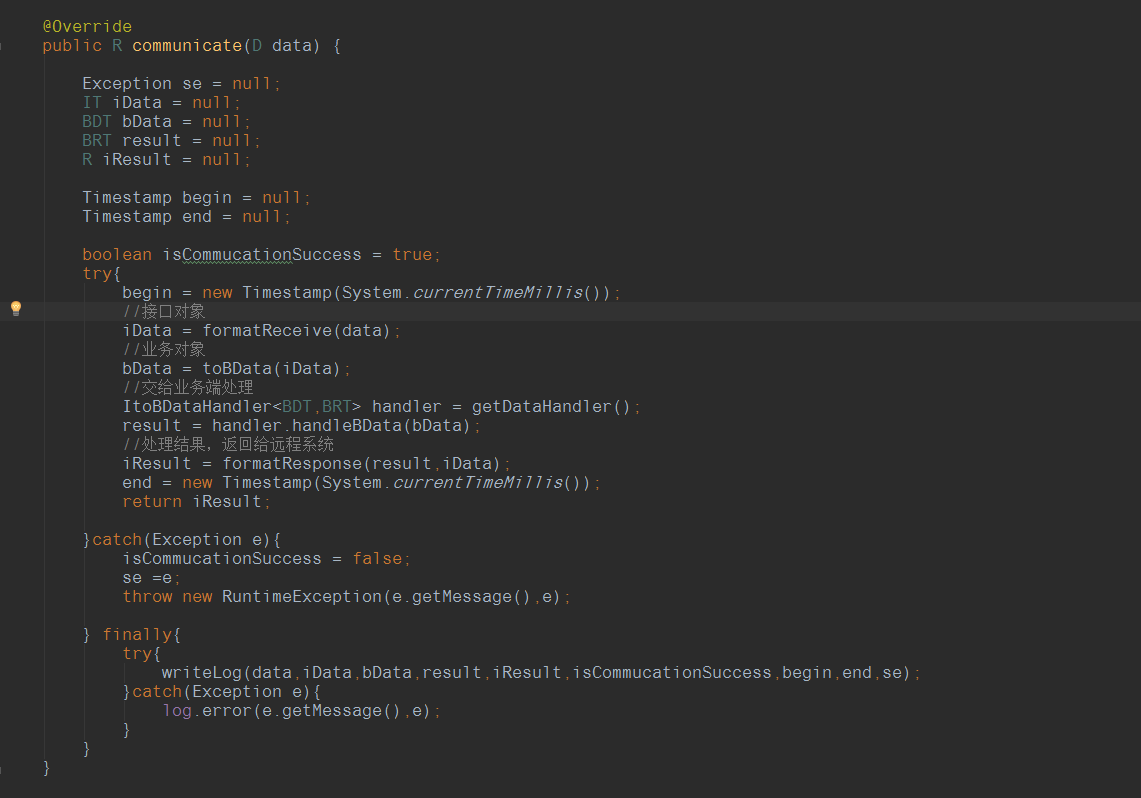
public abstract class HttpClientSendService<D,R,I extends IData,IT,IRT,IRO>等

## 设计模式

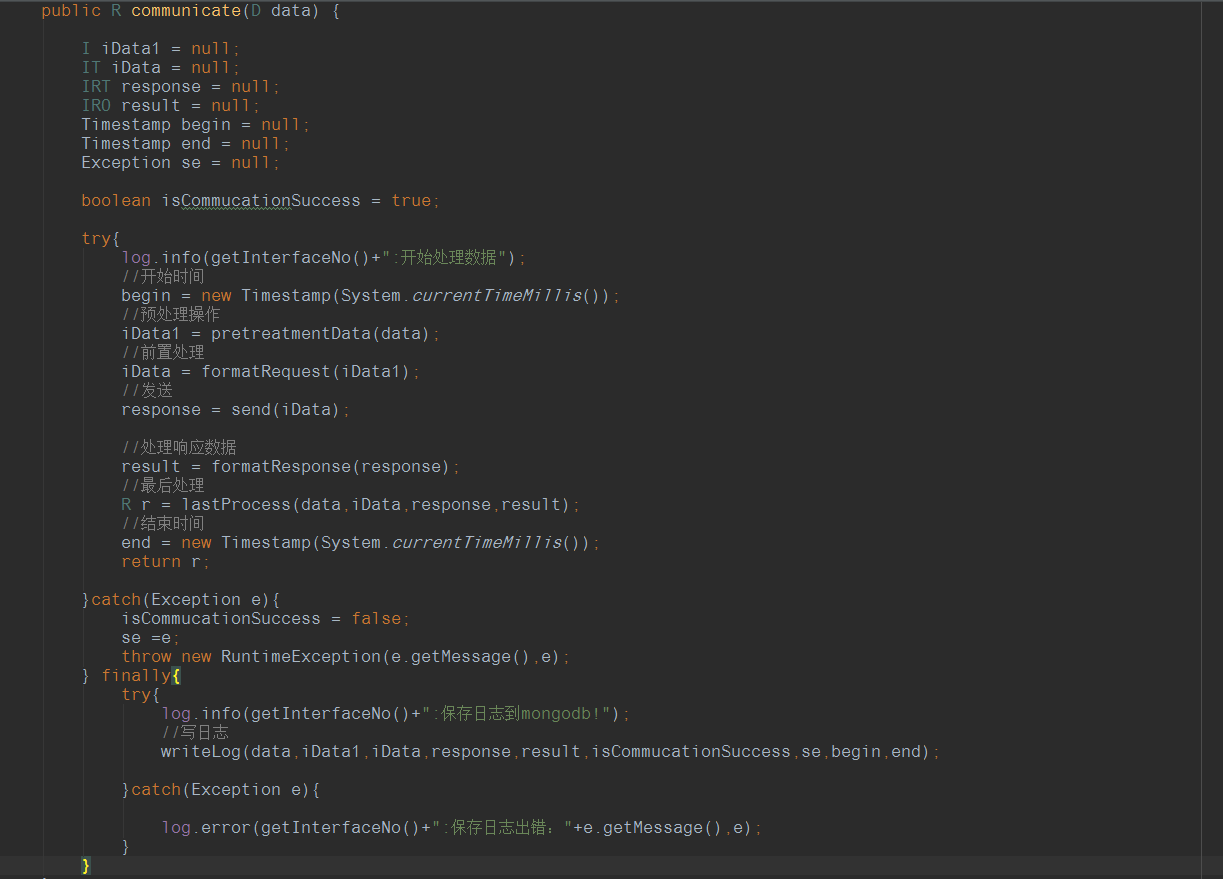
### 模板方法模式

框架中主要使用了模板方法模式。

接收模板方法



发送模板方法



# 库存模块

## 简介

库存模块主要是针对设置库存和设置虚拟仓库。设置库存可以通过API、管理页面等途径，设置虚拟库区是通过管理页面设置。

## 核心接口

B3.1(推送给ATP虚拟库区设置接口)

B3.1\_old(推送给DRG虚拟库区设置接口)

B3.2(POP推送库存给ATP)

D8.31(ATP推送库存给POP)

## 核心定时器

npop-大库存量商品给DRG推送商品可卖数 B3.2

com.founder.bbc.task.SendGoodsStockTask

npop-小库存量商品给DRG推送商品可卖数 B3.2

com.founder.bbc.task.SendGoodsStockTask

npop-将新建商品库存写入可卖数任务表

com.founder.bbc.task.SendGoodsStockTask

npop-统计更新商品总库存和最小仓库库存

com.founder.bbc.task.GoodsStockTask

pop\_stats-task商品首页预警数

com.founder.bbc.task.IndexShopJob

npop-初始化虚拟库区

StorageDispatchingAction

npop-已归档库区库存数据定时器启动

DeleteRepertoryTask

## 相关负责人

库存：任飞龙

库区：申玉柱

## 常遇问题

### 商家设置库存后不生效

以上问题的可能有几种原因导致：

1：B3.2接口没有推送成功

2：ATP没有消费成功B3.2接口

3：缺少店铺主数据

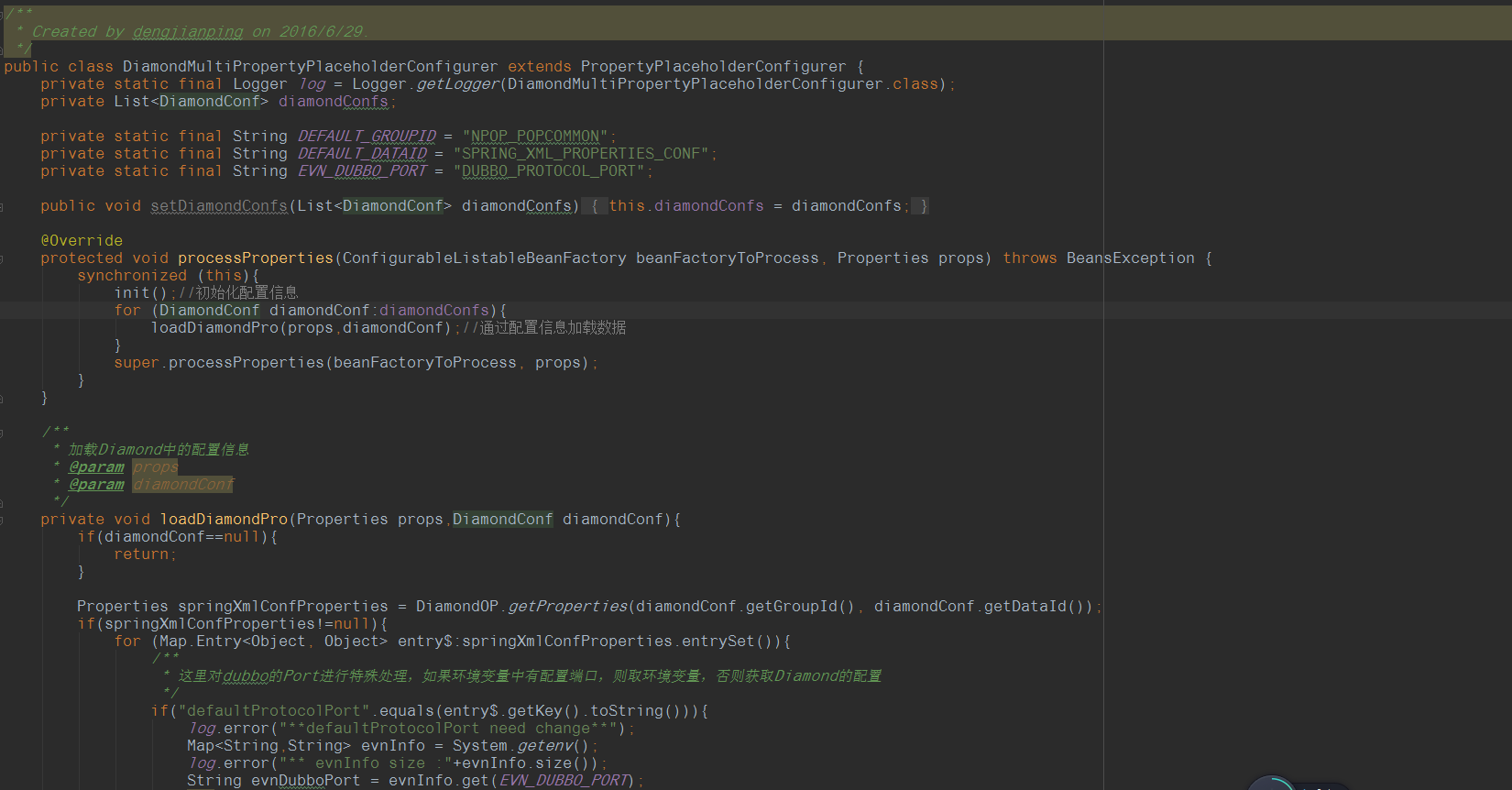
4：缺少商品主数据

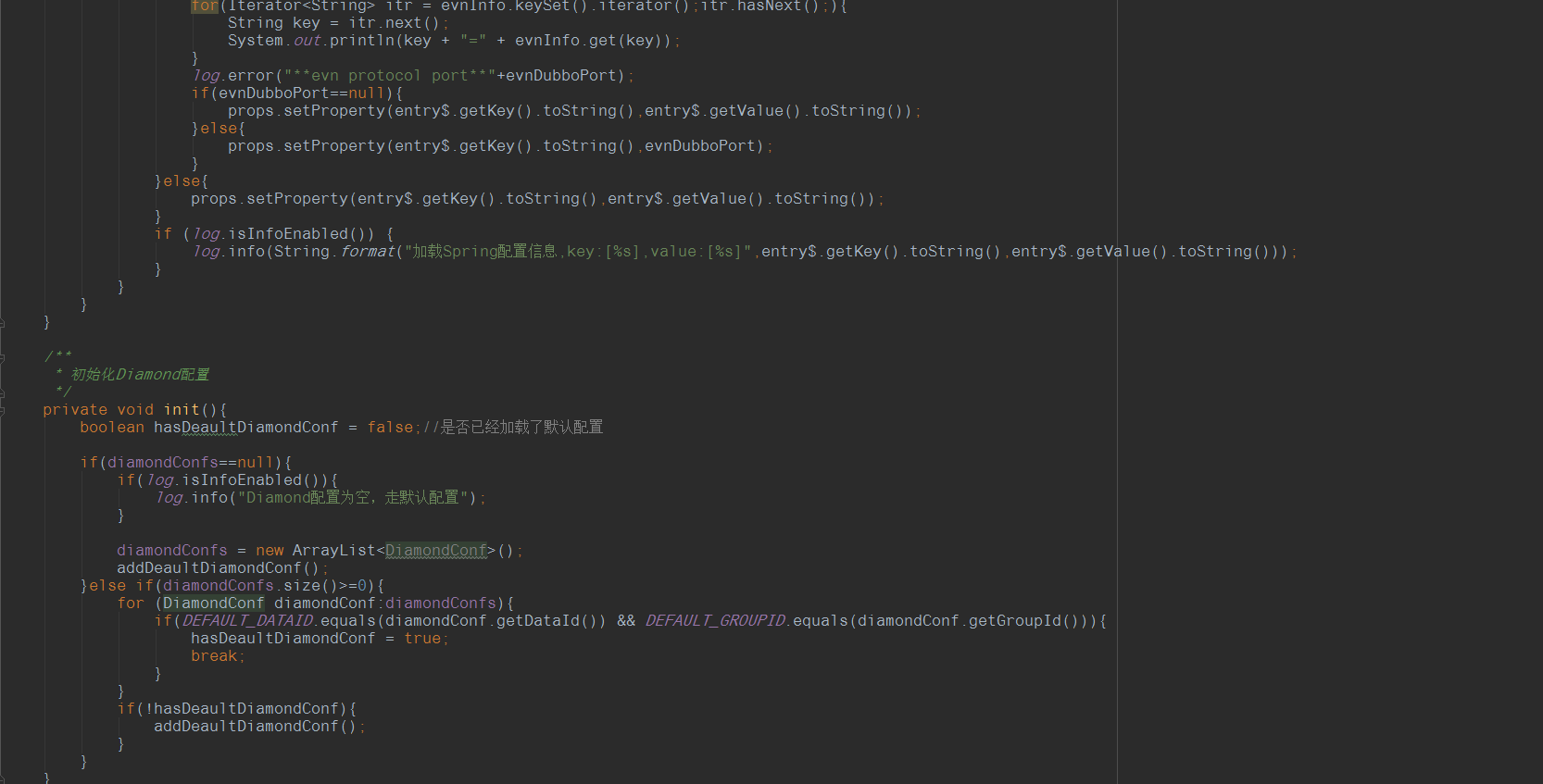
5：缺少库区主数据

# 公共模块

## Dubbo配置

通过Diamond配置dubbo的核心类是继承了Spring的PropertyPlaceholderConfigurer，来通过占位符在Spring的配置文件中动态载入。默认加载Diamond里面的NPOP\_POPCOMMON/SPRING\_XML\_PROPERTIES\_CONF节点数据。核心实现为：DiamondMultiPropertyPlaceholderConfigurer



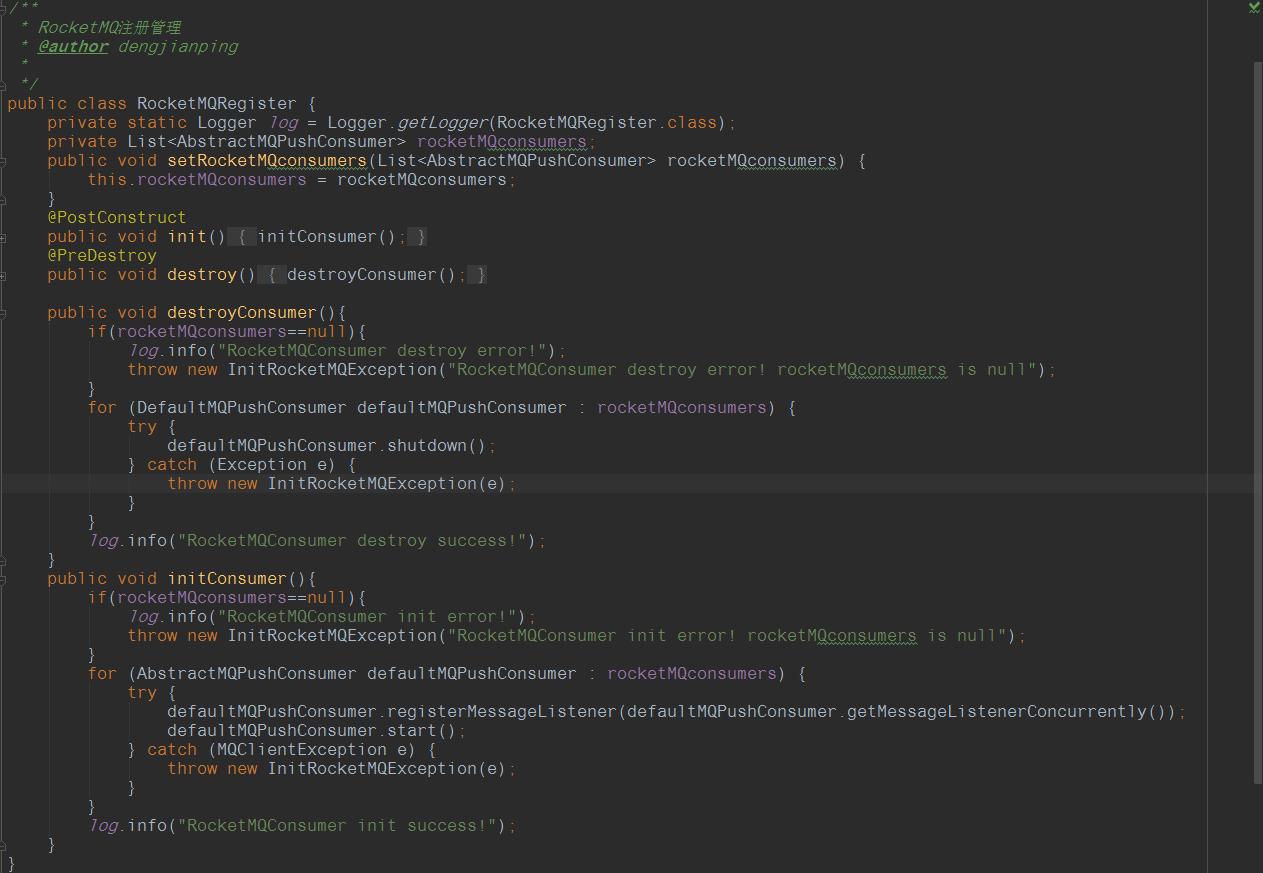




## RocketMQ消费者

### 注册

RocketMQRegister类中有两个注解比较关键和一个List<AbstractMQPushConsumer> rocketMQconsumers，@PostConstruct是表示在构建对象前需要做的事情，@PreDestroy在结束对象时需要做的事情。rocketMQconsumers存放着注册的消费者。核心代码：



### 消费者

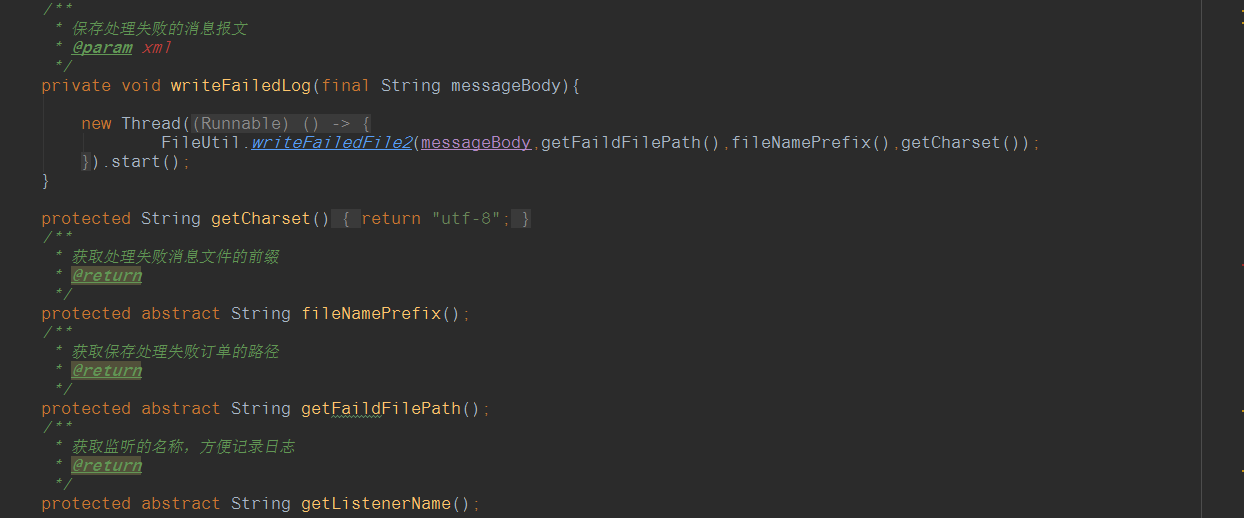
AbstractMQPushConsumer是继承了DefaultMQPushConsumer，在类中扩展了以下方法



通过扩展方法来进行不同消费者的配置，配置的时候可以通过读取Diamond中的数据进行动态配置。

### 监听

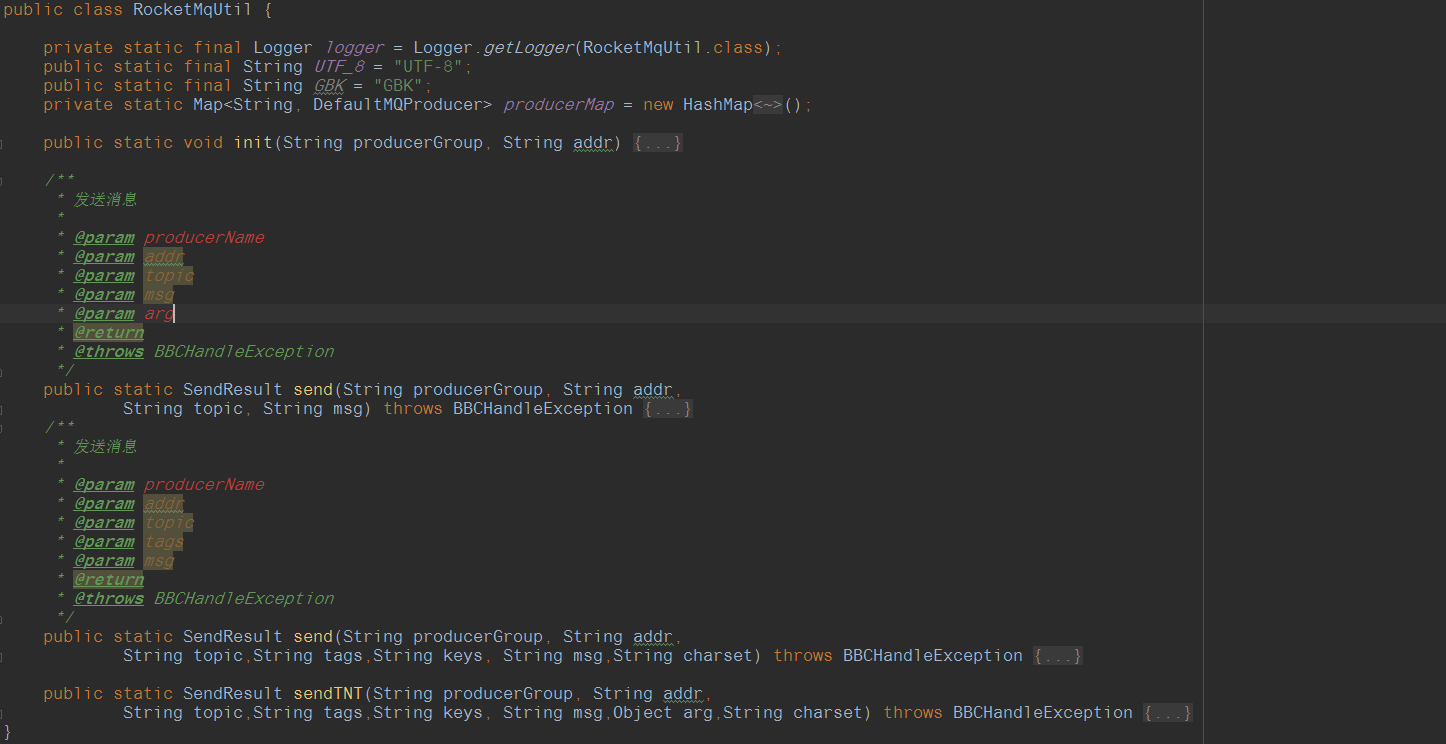
AbstractRocketMQListener继承了MessageListenerConcurrently，在此基础上进行了扩展，主要扩展了一下方法：



通过这些扩展方法来记录报文，以便之后的重处理。

## RocketMQ生产者

RocketMqUtil是核心的工具类，并且是线程安全的。默认会加载producer到producerMap中。sendTNT方法是支持顺序消费的方式。

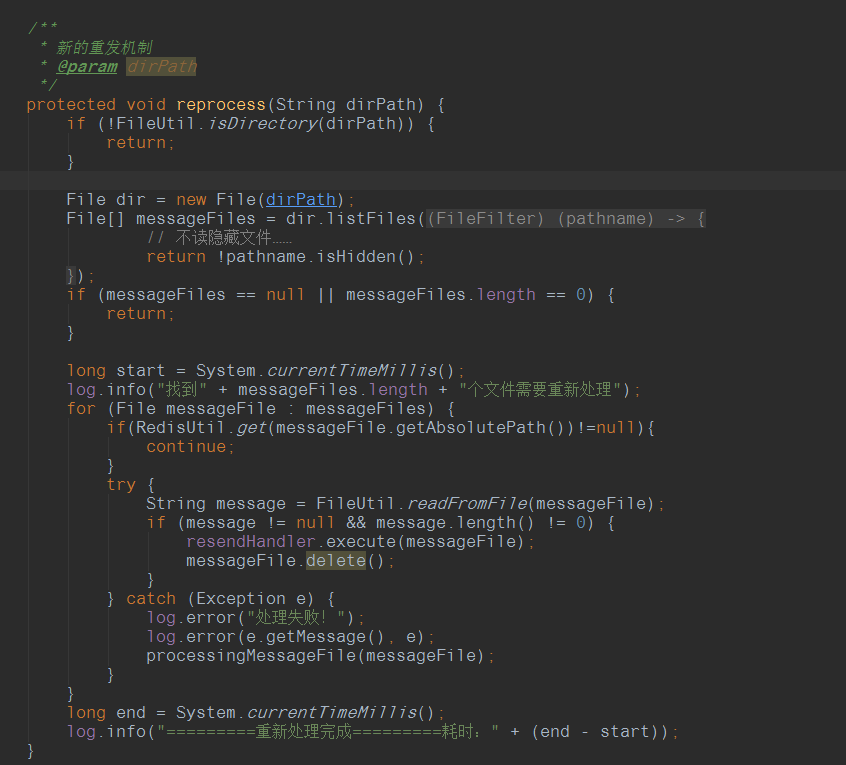


## RocketMQ消息重处理

消息冲处理，主要是通过定时器来定时的扫描指定路径下的报文，然后调用对应的接口来处理消息。

### 定时器

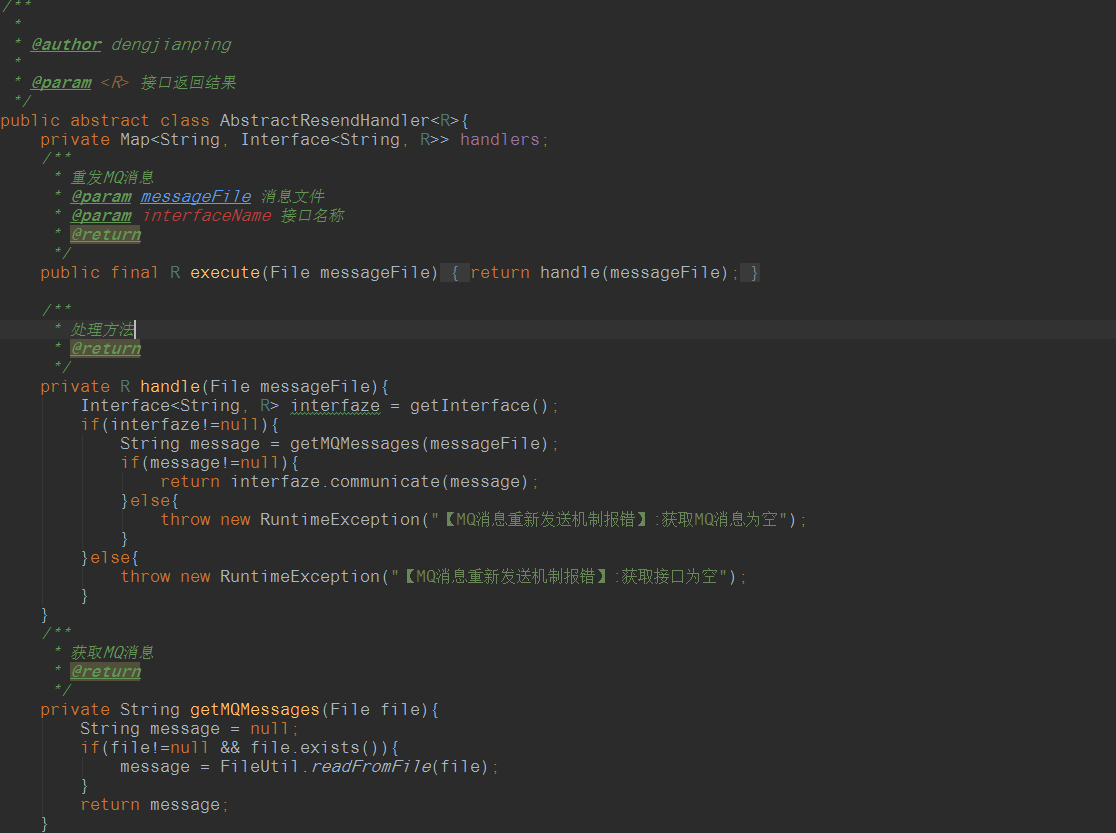
AbstractReprocessingJob是继承了TimerTask。构建了一个模板方法reprocess，不同的定时器传入不同的文件路径即可。

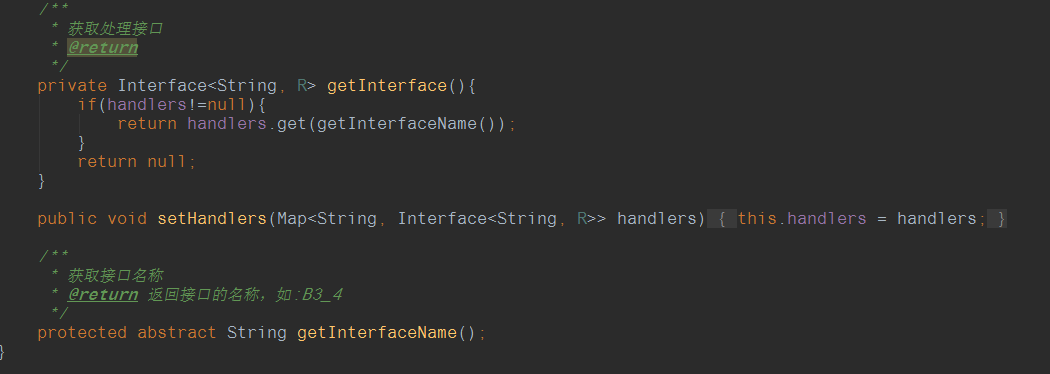




### 处理handler

AbstractResendHandler是定时器重新处理的具体handler。在该类中有一个handlers，是一个map结构，value是对应的真实处理接口Interface<String, R>





# 返利项目

## 动态代理（实现dubbo接口记录日志）

核心类RecordLoggerInvocationHandler

