目录

[一、架构设计 1](#_Toc503863237)

[产品愿景 1](#_Toc503863238)

[需求分析 1](#_Toc503863239)

[设计描述 1](#_Toc503863240)

[设计图 1](#_Toc503863241)

[二、网络请求框架（retrofit与拦截器与腾讯IM） 1](#_Toc503863242)

[主动请求 1](#_Toc503863243)

[接口支持 1](#_Toc503863244)

[拦截器 1](#_Toc503863245)

[三、多线程与线程池框架 1](#_Toc503863246)

[RxJava 1](#_Toc503863247)

[线程池管理 1](#_Toc503863248)

[四、缓存 1](#_Toc503863249)

[五、数据库管理 1](#_Toc503863250)

[JPA+注解 1](#_Toc503863251)

[正常查询 1](#_Toc503863252)

[写入 1](#_Toc503863253)

[分包图 1](#_Toc503863254)

[六、消息管理 1](#_Toc503863255)

[七、数据异步写入 1](#_Toc503863256)

[八、Config配置 1](#_Toc503863257)

[九、轮训实现 1](#_Toc503863258)

[十、项目版本发布 1](#_Toc503863259)

# 一、架构设计

## 产品愿景

流畅稳定运行弹幕Server，保证server与客户端的链接稳定性。

## 需求分析

弹幕Server需要独立于NodeJs后台服务与CMS，给弹幕客户端提供消息服务以及寻常接口。现客户端部分调用CMS接口，部分调用NodeJs接口，以及通过webview连接NodeJs的webscocket接收节目消息。

## 设计描述

#### 1、消息通信方式修改

取消原通过webview连接NodeJs的websocket的方式，通过腾讯IM进行消息通信，增强稳定性。

#### 2、弹幕Server立场

不仅给弹幕客户端提供接口，也会给NodeJs与CMS提供

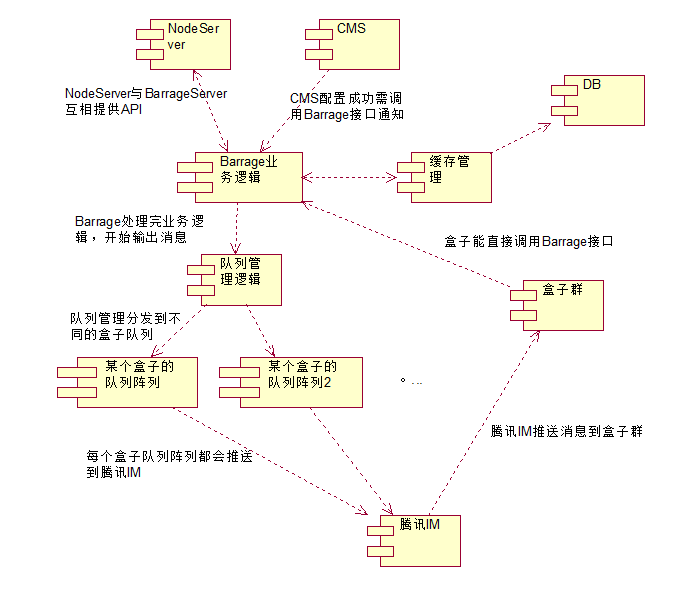
#### 3、弹幕Server消息来源

提供NodeJs消息写入接口，在NodeJs 发送IM消息给手机端的同时调用弹幕Server的此接口。即弹幕Server与手机端接收到的消息保持一致。

#### 4、消息处理

弹幕Server接收到消息之后进行一些逻辑处理然后广播到所有消息队列，然后不同的消息队列可能会再次按照逻辑进行处理

## 设计图



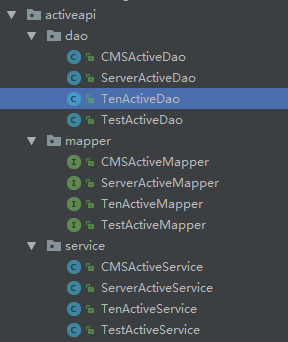
# 二、网络请求框架（retrofit与拦截器与腾讯IM）

## 主动请求

#### 1、请求设计

采用成熟Retrofit+RxJava来完成API请求，虽然Retrofit也支持异步请求，但是并没有多线程支持，所以使用RxJava能够让各种请求适用请求环境，例如IM消息发送，有些消息需要连续性，则需要使用单线程池，有些消息不需要连续性，但是还是要限制并发，还是用线程池。

#### 2、请求分包图



网络请求先按照类型进行拆分，现类型有CMS、NodeJsServer、腾讯IM，然后在使用DAO模式拆分，mapper只负责接口请求的映射，dao提供Retrofit框架的封装以及redis的集成，部分接口不需要连续请求，保存redis即可，例如：获取店铺的IM群组ID

## 接口支持

接口支持部分按照Springboot的方式进行处理，先按照调用类型进行拆分，现有类型Client（弹幕客户端）、CMS、Server、ten（腾讯IM为特殊存在，特别挑出来）。

Controller中不存在任何逻辑，仅提供接口参数获取与service跳转。

## 拦截器

#### 1、拦截器说明

Springboot中拦截器有多种，本项目使用的有异常处理、加解密、返回字段限制。

#### 2、拦截器BarrageResponseBodyAdvice

拦截器在common/interceptormanager/中，现在生效的仅支持的接口的返回值，即BarrageResponseBodyAdvice，每个支持的接口都可以使用注解SerializedField，此为自定义注解，包含encode，includes，excludes，enable。

Encode：true，使用加密，会将返回值所有字段加密，现加密方式为Base64,可修改DesUtil类来达到修改加密方式

Enable：true，是否使用使用此拦截

Includes：设置包含的字段，返回的body中就只有这几个字段

Excludes：设置排除的字段，返回的body中就只有这几个字段，与includes互斥

#### 3、拦截器BarrageRequestBodyAdvice

与BarrageResponseBodyAdvice一样，不过处理的是请求的拦截，只拦截json格式的请求，此拦截器暂时未使用，代码会有一点问题，如果要使用，参考BarrageResponseBodyAdvice写法

#### 4、拦截器BarrageRequestParamWrapper

与BarrageRequestBodyAdvice一样，不过处理的是表单格式的请求，具体使用并未写逻辑，不过确认是可行的

#### 5、拦截器AnyExceptionHandler

拦截项目中所有异常，返回异常的message与errorcode

# 三、多线程与线程池框架

## RxJava

提供链式处理与各种线程池，但是设置有局限性，所以线程池的创建交给外部，RxJava仅有使用权

## 线程池管理

各种根据key创建单线程池，与多核心线程池，根据具体情况选择使用。现提供单线程池HashMap，5核心线程池，10核心线程池。

#### 1、单线程池

礼物赞赏等需要保持count的消息需要使用此线程池保持一致性；数据库异步写入，需要保持一致性的写入操作；消息生产者，使用单线程池防止出现异常

#### 2、5核心线程池

无序消息的发送

#### 3、10核心线程池

无序DB的异步写入

# 四、缓存

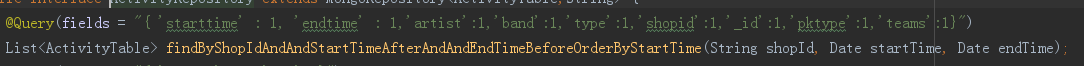
此项目使用Redis进行数据缓存，缓存都处于DAO层，发现缓存则直接返回，若无缓存再回去请求接口或者查询数据库。

Redis缓存使用分包模式，相同类型的会归类，结构清晰

# 五、数据库管理

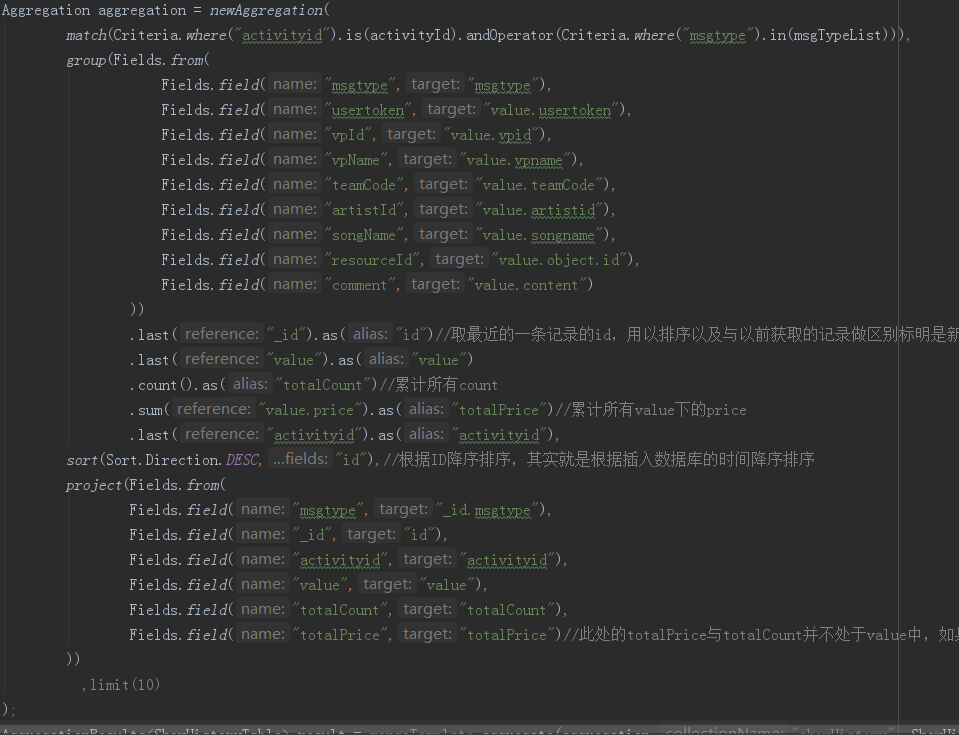
## JPA+注解

数据库查询大量使用JPA与注解，使用JPA，简洁易懂，注解能标注返回的具体字段提供查询速度。例如：



## 正常查询

除了JPA+注解，也支持正常查询，有些复杂模式只能如此，例如聚合查询：



## 写入

数据库写入分两种方式，1、直接写入；2、队列异步写入，使用队列异步写入需要注意，此写入必须是非一致性

## 分包图



# 六、消息管理

#### 1、设计模式

消息队列使用观察者模式，或者生产者与消费者

IQueue：队列行为接口，定义一个二维队列该有的行为，此接口可通用

IQueueObserver:观察者接口定义，用于生产者广播

BaseQueue：队列实现基础类，实现IQueue接口

IMsgQueue：消息队列专属接口，定义各种消息发送的方法

MessageQueue：继承BaseQueue并实现IMsgQueue接口，所有的消息队列都需要集成此类

MessageProducer：生产者，使用单线程池接收所有消息并广播给所有注册者

#### 2、二维消息队列

因消息存在优先级，所以采用空间换时间的方法为每个队列类开辟二维空间，即对外开放的队列对内其实可以有多个队列，每个内队列可以设置优先级，队列poll的时候会优先从高优先级内队列优先poll。

#### 3、线程规划

生产者使用RxJava+单线程池接收消息。

消费者每次启动poll链均从RxJavaIO线程池中随机获取一个线程执行，具体队列逻辑如何处理可以在队列中再次使用其他线程池方案，如下图所示，startPollChain只要消息没有结束，那么只会执行一次

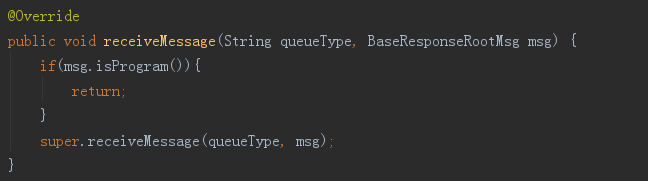


#### 4、使用注意

遵循对修改关闭原则，如果有新消息新增，最好直接创建一个新的消息队列。

腾讯IM的消息发送接口并发为一秒100条，所以消息队列消息发送尽量添加延迟，防止超过并发导致消息发送失败。

消息队列按照消息类型进行接收，但是偶尔也会有除消息类型之外的新类型，例如：从NodeJs服务发送过来的消息与客户端请求弹幕Server，弹幕Server发送的消息，此情况现并未添加类型，而是使用bool值来判断，如果有新的类型添加，最好使用type进行区分，如下图：

节目消息，表示是客户端请求，弹幕Server需要发送的消息，这种消息有特殊的处理队列，别的队列不需要处理。

# 七、数据异步写入

与消息队列类似，使用二维队列+redis实现数据库异步写入。

# 八、Config配置

#### 1、WebSecurityConfig

Springboot安全配置，可配置端口，可访问的路由，等等，现只配置了端口与所有路由可访问

#### 2、RedisConfig

配置了Redis的序列化方式，默认此配置即可

#### 3、SslConfig

可以通过此配置支持https，现如今没有用上

# 九、轮训实现

项目中大量逻辑用到delay执行与循环执行，本项目采用ScheduledThreadPoolExecutor来实现，创建TimerManager来管理，好处是所有轮训只占用一个线程，大大降低系统消耗。

# 十、项目版本发布

版本发布需要修改一个文件，application.yml,最好修改为环境变量配置，修改Redis，MongoDB与app下的buildType即可

# 十一、部署

编译，编译命令：mvn clean package，生成release版本的jar包

上传至服务端弹幕目录覆盖原jar包，分别执行stop和start shell文件

线上环境与测试环境已经部署了定时手机nohup.out日志收集的服务，所以不用担心nohup文件过大问题，收集的日志在logs/BarrageServer/consol中，与log4j产生的日志不一样分开存储，如果不需要nohup的控制台日志，可以考虑修改start shell文件