一、整体说明

1、目标

对融媒体系统框架进行2.0重构，系统能够分布式部署，横向扩展，并继续支持window和linux的快速部署。通过服务分层解耦，支持多种数据库。

2、基本组成

前端包括：app、html、c/s客户端

系统服务：整体使用Spring Clouds框架，其中组件包括：

1）Eurake注册中心，进行微服务的管理；

2）Hystrix进行服务的降级和熔断保护

3）Ribbon实现内部服务的负载均衡

4）Bus进行mq消息通讯的管理

5）Slecth对系统交互进行跟踪

6）Zuul作为外部交互的网关

缓存：使用redis进行缓存和session同步

数据库：使用Mybatis框架与数据库通讯，并兼容多种数据库

二、系统设计

系统整体分为四层：web层，接口层，服务层，数据库层，如下图：



其中服务层可以分为：基础服务层和综合服务层

1、系统目录层次

核心代码存储于main/java/com/shineon/coder下

Coder下的目录

db 存储数据库映射关系及POJO

server 用来存放逻辑处理和基本数据处理

controller 用来存放对外的接口

tool.common 用来存放公共工具

tool.convert 用来存放对象转换（object、json、xml）

version 用来存储版本控制的内容

2、redis缓存

Redis使用集群部署，数据存储分片，持久化使用AOF方式

三、系统概要设计

1、uauth统一用户认证

基本用例图，如下：



四、系统约束

1、整体约束

1）系统间的交互只能上下层之间进行通讯，禁止同层服务进行通讯，禁止跨层交互

2）如果必须同层通讯，通过消息队列（mq）进行

3）系统缓存存储到redis中，object类型的存储，需要存储json结构，并提供json的对象之间相互转换的工具类

4）服务层间的交互使用RPC的方式，禁止使用json交互

2、代码规范

1）类的命名，首字符大写，使用驼峰式命名

2）常量需要全大写，单词间用 \_ 链接

3）变量首字母小写，采用驼峰式命名

3、系统命名规范

注册中心： shineon\_eurake

接口层系统： shineon\_api\_系统名称

服务层系统： shineon\_server\_系统名称

数据库层系统： shineon\_db\_系统名称

4、DAO对象构建规范

1） 获取单个对象的方法用 get 做前缀。

2） 获取多个对象的方法用 list 做前缀。

3） 获取统计值的方法用 count 做前缀。

4） 插入的方法用 insert 做前缀。

5） 删除的方法用 delete 做前缀。

6） 修改的方法用 update 做前缀。

5、线程规范

1）定义任务使用runnable接口

2）加锁禁用synchronized，使用Lock

3）需要保证原子性和可见性的操作，使用volite和原子类

4）多线程中键值容器，使用CurrentHashMap

6、端口范围

注册中心 端范围口 81- 89

Config 端口范围 90-99

服务 端口范围 121-150

接口 端口范围 160-180

Zuul 端口范围 181- 200

Nigix 端口范围 210 -230

五、工具

1、db层mapper生成工具dbBuildTool（优化中）

初始的mapper文件及pojo文件的生成，可以使用dbBuildTool工具

使用说明：

1、配置DBConfig.xml的数据库信息，和要生成的表

2、运行start.cmd ,生成的文件存储到db文件夹中

2、项目基本目录生成工具类ProjectInit

配置classPaths中的目录结构，运行main函数，生成要使用的目录结构