一、整体说明

1、目标

对融媒体系统框架进行2.0重构，系统能够云端分布式部署，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）Stream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进行数据缓存和session同步

消息队列：使用rabbitmq或者kafka，进行同级服务任务的交互

数据库：使用Mybatis框架与数据库通讯，并兼容多种数据库

3、概念

1）微服务：

a)基本思想在于考虑围绕着业务领域组件来创建应用，这些应用可独立地进行开发、管理和加速。在分散的组件中使用微服务云架构和平台，使部署、管理和服务功能交付变得更加简单。

b)利用组织的服务投资组合，然后基于业务领域功能分解它们，在看到服务投资组合之前，它还是一个业务领域。

c)通用微服务架构：dubbo和spring cloud （shineon2.0选用spring cloud）

2）spring cloud：

Spring Cloud是一系列框架的有序集合。它利用[Spring Boot](https://baike.baidu.com/item/Spring%20Boot/20249767)的开发便利性巧妙地简化了分布式系统基础设施的开发，如服务发现注册、配置中心、消息总线、负载均衡、断路器、数据监控等，都可以用Spring Boot的开发风格做到一键启动和部署。

3）spring boot与传统spring框架区别

（1）可以创建独立的[Spring](https://baike.baidu.com/item/Spring/85061)应用程序，并且基于其Maven或Grandel插件，可以创建可执行的JARs和WARs；

（2）内嵌Tomcat或Jetty等Servlet容器；

（3）提供自动配置的“starter”项目对象模型（POMS）以简化[Maven](https://baike.baidu.com/item/Maven/6094909)配置；

（4）尽可能自动配置Spring容器；

（5）提供准备好的特性，如指标、健康检查和外部化配置；

（6）绝对没有代码生成，不需要XML配置。

4）消息队列：

一个中间软件模块，把消息从一个软件服务传递到另外一个软件服务上去

作用：实现软件解耦，最大限度的减少程序之间的相互依赖，提高系统可用性以及可扩展性，同时还增加了消息的可靠传输和事务管理功能。

5）Nginx

Nginx是一款自由的、开源的、高性能的HTTP服务器和反向代理服务器；同时也是一个IMAP、POP3、SMTP代理服务器；Nginx可以作为一个HTTP服务器进行网站的发布处理，另外Nginx可以作为反向代理进行负载均衡的实现。

6)Docker

Docker 是一个[开源](https://baike.baidu.com/item/%E5%BC%80%E6%BA%90/246339)的应用容器引擎，让开发者可以打包他们的应用以及依赖包到一个可移植的镜像中，然后发布到任何流行的 [Linux](https://baike.baidu.com/item/Linux)或Windows 机器上，也可以实现[虚拟化](https://baike.baidu.com/item/%E8%99%9A%E6%8B%9F%E5%8C%96/547949)。容器是完全使用[沙箱](https://baike.baidu.com/item/%E6%B2%99%E7%AE%B1/393318)机制，相互之间不会有任何接口。

7)solr

solr是java搜索引擎Lucene的更高一层封装，通过webapp服务器实现可视化界面，方便使用人员配置、访问和调用

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；需要做到基本的高内聚低耦合，下一层不会依托于上一层系统，同层系统之间不存在直接交互。

2）非上下层直接交互的情况，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）服务层间的内部交互使用RPC的方式，禁止接口交互直接传递json，然后进行解析的方式

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

4）禁止使用基本类型，使用基本类型的包装类

5）时间类型，后台周转和使用，尽量使用long型

6）String除了日志输出以外，禁止再代码中出现常量，出现两次以上的常量，需要放到常量对象中

3、系统命名规范

注册中心： shineon-eurake

接口层系统： shineon-api-系统名称

综合服务层系统： shineon-server-系统名称

基础服务系统：shineon-base-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

3、ConverTool（待完善）用来生成pojo对象和通用对象的映射和转换工具

第一次运行，会有映射误差，具体的映射关系会生成对应的mapper.xml文件，可以修改xml文件中的映射关系，重新生成（每次生成会删除已经无用的映射关系）