**十次方**

**一．项目介绍：**

是一个专供程序员交友社交平台，功能模块包括了文章，问答，招聘，活动，吐槽，交友，用户中心，搜索中心以及第三方登陆。采用的是前后端分离的模块化设计，前后端可以单独开发互不影响，后端使用的是微服务系统架构，

**二．技术架构：**

**前端：***Node.js+Vue.js+ElementUI+NUXT*

**后端：***SpringBoot+Springcloud+springData+SpringMVC*(spring全家桶)

**前端技术栈：**

1.Vue.js

2.Node.js

3.npm

4.element UI库

5……

**微服务架构技术栈：**

1.SpringCloudConfig(集中配置中心)

2.Ribbon(负载均衡)

3.Hystrix(熔断器)

4.Zuul(服务网关)

5.Bus(消息总线)

**持久化技术栈：**

1.Spring Data JPA

2.Spring Data Redis

3.Spring Data MongoDB

4.Spring Data Elasticsearch

**数据库&索引库&消息队列技术栈：**

1.MySQL读写分离

2.Redis集群

3.mongoDB集群

4.RabbitMQ

5.ElasticSearch集群

**外部接口：**

1.阿里大于短信接口

2.腾讯第三方登陆

3.阿里云oss

**三．模块划分：**

分为18个子模块，（其中17个是微服务），一个公共子模块

**四．业务讲解：**

**招聘：分为企业信息和招聘信息 （SpringDataJPA,SpringDataRedis）**

企业在搜索的时候搜索ishot为1（热门）的记录

（招聘表状态 0：关闭，1：开启，2：推荐）

在查找推荐职位时，查找state（状态）为2（推荐），并且根据日

期进行降序排序，查询前4条查找最新职位时，查找状态不为0，并且一日期进行排序，查找前12条

**文章：**为了提高查询的性能，我们在查询使用了进行缓存处理，并且在修改或删除

后清楚缓存，使用Redis和SpringCache

**吐槽和评论：**由于评论的数据量大,价值相对来说较低，对于这个模块的数据存储采

用了MongoDB 。需求每个用户只能点赞一次，不能重复点赞。

**搜索：** 文章索搜，使用ElasticSearch ，ik分词器，ElasticSearch域MySQL数据同步

**用户注册：（消息中间件RabbitMQ,阿里大于，redis）**

生成手机验证码，存入Redis并发送到RabbitMQ，从RabbitMQ中提取消息，

调用阿里大于短信接口。

密码使用BCrypt密码加密

**交友：（Eureka,Feign,Zuul,SpringCloudConfig,SpringCloudBus）**

（1）当用户登陆后在推荐好友列表中点击“心”，表示喜欢此人 ，在数据库tb\_friend表中 插入一条数据，islike 为0

（2）当你点击了喜欢过的人，也喜欢了你 , 表示互粉成功！也向tb\_friend表中插入一条 数据，islike为1 ，并且将你喜欢她的数据islike也修改为1

（3）当你点击了不喜欢某人（点击了叉），向tb\_nofriend添加记录.

（4）当两个人互粉后，其中一人不喜欢对方了，删除好友表中的记录 ，向非好友表中添 加记录，

在交友微服务中使用Feign调用用户微服务的更改粉丝数与关注数

**登陆：**当用户点击注册登录按钮时会触发注册登录流程或者用户在未登录情况下在头

条板块进行发布、点赞、收藏操作；在问答板块进行回答、点赞、提问、评论、收藏、添加

标签操作；在活动版块点击我想参加、立即报名、评论操作；在交友板块发布邀约、点击喜

欢、不喜欢、私信按钮、完善信息操作；在吐槽板块评论、点赞、收藏操作；在招聘页面收

藏、投简历操作 以上几个操作会触发注册登录条件。

**五．负责模块：**

**吐槽和评论：**

实现了基本的增删改查，根据上级ID查询吐槽列表，吐槽点赞，通过Redis控制哦用户不能重复点赞，发布吐槽，由于数据量较大，操作频繁，数据价值相对较低，对这个模块的数据存储采用了MongoDB，MongoDb的访问速度也比MySQL高采用了SpringDataMongoDB,使用了MongoRespository完成CURD（创建（Create）、更新（Update）、读取（Retrieve）和删除（Delete））和复杂的查询

***框架：***

1. ***SpringBoot***

* **什么是SpringBoot？** 是Spring开源组织下的子项目，主要简化了Spring的难度。遵循约定大于配置的原则，简省了繁重的配置，提供了各种启动类，让文件的配置，应用部署变得简单。
* **为什么要用SpringBoot？**  优点很多，独立运行，简化配置，自动配置，无代码生成和XML配置，应用监控，上手容易。
* **SpringBoot的核心配置文件有哪几个？区别是什么？**核心配置文件是application和bootstrap配置文件。application主要用于SpringBoot项目自动化配置。
* **SpringBoot配置文件有哪几种格式？有什么区别？** .properties和.yml，主要区别是书写格式不同，另外，[.yml格式不支持@PropertySource](mailto:.yml格式不支持@PropertySource)注解导入配置
* **SpringBoot的核心注解？** 启动类上的注解是@SpringBootApplication，也是SpringBoot的核心注解，主要包含以下三个注解：

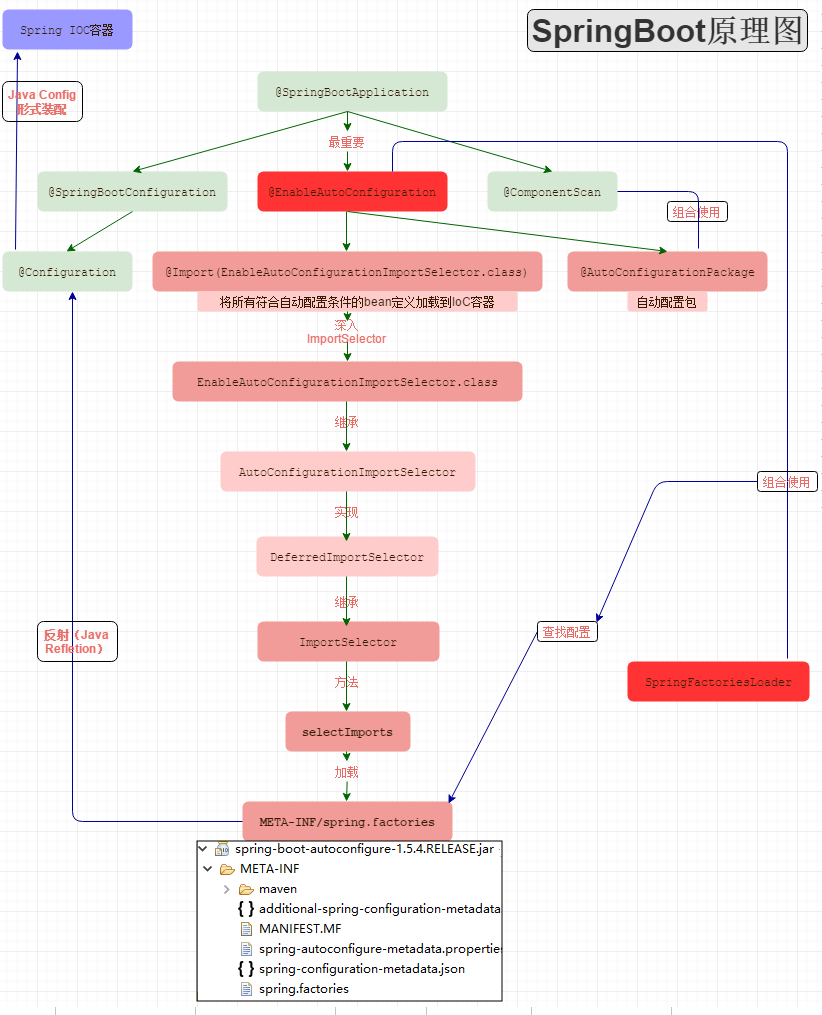
**a).**@SpringBootConfiguration： 继承了Configuration，二者表示的都是当前是注解类。任何一个标注了@Configuration的Java类定义都是一个JavaConfig配置类，并会将当前类内声明的一个或多个以@Bean注解标记的方法的实例纳入到spring容器中，并且实例名就是方名。

**b).** @ComponentScan：Spring组件扫描。功能其实就是自动扫描并加载符合条件的组件（比如@Component和@Repository等）或者bean定义，最终将这些bean定义加载到IoC容器中。//扫描路径设置

**c).**@EnableAutoConfiguration：开启SpringBoot注解功能，SpringBoot四大神器之一（auto-configuration、starters、cli、actuator）,是借助@Import的帮助，将所有符合自动配置条件的bean定义加载到IOC容器。两个比较重要的注解:

@AutoConfigurationPackage:自动配置包，返回了当前主程序同类以及子级的包组件

@Import:导入自动配置



* **SpringBoot自动配置原理？**通过注解@EnableAutoConfiguration，@Configuration, @ConditionalOnClass 就是自动配置的核心，首先它得是一个配置文件，其次根据类路径下是否有这个类去自动配置

* **SpringBoot可以兼容老SpringBoot项目吗？如何做？**可以兼容，使用@ImportResource注解导入老Spring项目配置文件

1. **SpringCloud**
2. **什么是SprigCloud?**

是一系列框架的有序集合，SpringCould离不开SpringBoot，属于依赖关系

1. **主要框架：** 服务发现——Netflix Eureka

服务调用——Netflix Feign

熔断器——Netflix Hystrix

服务网关——Netflix Zuul

分布式配置——Spring Cloud Config

消息总线 —— Spring Cloud Bus

1. **SpringCloud和dubbo的区别?**dubbo由于是二进制的传输，占用带宽会更少

* Eureka是Netfilx的一个子模块，基于REST的服务，它采用了C-S设计架构，Eureka包含两个组件：Eureka Server(服务的提供者)和Eureka Client(服务的消费者)。

1. ***SpringData***

* SpringData的主要模块：(十次方中所用到的)

Spring Data JPA

Spring Data Redis：提供了在Spring应用中通过简单的配置访问redis服务，对reids底层开发包(Jedis,  JRedis, and RJC)进行了高度封装，RedisTemplate 提供了redis各种操作。

SpringData提供了很多模块去支持各种数据库的操作

* JpaRepository是最基础的接口，提供了基本的增删改查,
* JpaSpecificationExecutor，提供了对动态查询的支持，用于做复杂的条件查询

***技术：***

***一．缓存的使用：***

（为了让我们的代码更容易维护，我们创建了一个类集中处理异常）

（不同的微服务里面有很多容器，如果都对一张表进行操作，那么他们的ID就不能一样，所以我们使用推特的雪花算法，用的时候放在utils工具包下即可。其实就是一个类，用的时候把他加入spring容器中即可。有的地方可能不需要用这个工具类，则什么时候使用就什么时候放入spring中。）

* 招聘微服务开发：包括了企业信息和招聘信息
* 在进行查询的时候，为了提高查询性能，通常采用Redis缓存解决。需要在修改和删除后清除缓存
* **什么样的数据需要放进缓存？**1.数据量不大 2.访问频率高 3.数据更改频率低
* **SpringCache常用的注解**：1. @Cacheable （在执行后会缓存其返回结果，用于查询）2. CacheEvict （其执行前或执行后移除Spring Cache中的某些 元素）

@EnableCaching开启缓存支持

**1.Spring Cache和Redis的区别；**

* Spring Cache是代码级的缓存，实际上还是使用JVM的内存来缓存对象，会造成的两的内存消耗，好处是使用方便。
* Redis是一个缓存服务器，是单纯的使用内存进行的缓存，
* 在集群的环境下，每台服务器的spring cache是不同步的，这样会出问题的**，spring cache只适合单机环境。Redis是设置单独的缓存服务器，所有集群服务器统一访问redis，不会出现缓存不同步的情况**
* **Redis可以设置过期时间，SpringCache不可以**

***二．MongoDB的使用：***

* 吐槽和评论两个功能因数据量大，写入操作频繁，价值比较低，所以我们采用MongoDB实现数据库的存储，（如果数据格式不确定，却少数据库管理人员，不怎么使用关联查询和事务，不是特别追求服务的稳定性可以使用MongoDB）

1. **什么是MongoDB？**

* **MongoDB 是一个跨平台的，面向文档的数据库，它介于关系数据库和非关系数据库之间，是非关系数据库当中功能最丰富，最像关系数据库的产品，他支持的数据结构是类似JSON的BSON 格式可以存储比较复杂的数据类型**
* 优点：不存在SQL注入，不需要提前创建表，可以任意添加减少字段，字段数据格式自由，可以处理JSON结构，动态查询，高效存贮二进制大对象（照片视频等）
* 缺点：不支持事物，没有MySQL那样成熟的维护工具
* 不是很重要的数据可以使用MongoDB，比较重要的使用可以用使用MySQL
* MongoDB 的逻辑结构是一种层次结构主要由，文档(document)、集合(collection)、数据库(database)这三部分组成的

1. **SpringDataMongoDB**

* SpringDataMongoDB是SpringData家族成员之一，用于操作MongoDb的持久层框架，封装了底层的mongodbdriver。吐槽服务采用的是SpringDataMongoDB框架
* 通过Redis控制用户不能重复点赞

1. **你的工程是如何操作MongoDB的？**

采用了SpringDataMongoDB,使用了MongoRespository完成CURD（创建（Create）、更新（Update）、读取（Retrieve）和删除（Delete））和复杂的查询

***三．分布式搜索引擎ElasticSearch***

* **ElasticSearch提供了强大的索引功能，倒叙索引，而且是开箱即用的，部署简单**
* 分布式的实时文件存储，每个字段都被索引并可被搜索
* **（搜索引擎和站内搜索的区别）**

**1.Soler和ElasticSearch的区别？**

**二者都是以lucene（**Lucene是一套用于全文检索和搜寻的开源程序库，不是现成的搜索引擎产品，但可以用来制作搜索引擎产品**）为底层**

**Elasticsearch适合于实时搜索应用，solr不适合实施索搜**

**（**Logstash是一个开源数据收集引擎，具有实时管道功能。Logstash可以动态地将来自不同数据源的数据统一起来，并将数据标准化到你所选择的目的地。**）**

Soler会造成io堵塞，elasticsearch不会

1. **什么是RabbitMQ？**

是以管道的方式做消息传递

1. **RabbitMq的发送模式**

直接模式 分列模式 主题模式 headers

1. **常用的消息队列：**

ActiveMq(最慢，安全),

RabbitMq(安全性足以用金融，),

Kafka(高吞吐量，专供大数据使用，最快，不安全)

1. **RabbitMQ和Kafka简单的对比**

* RabbitMQ是遵循**AMQP**（高级消息队列协议）协议，以broker为中心，有消息队列机制，吞吐量不如Kafka,支持miror的queue,主queue失效，miror queue接管,用在**可靠性要求比较高**的消息传递上
* Kafka是开源的**消息发布订阅**系统，遵循一般的MQ结构，没有消息确认机制，有高的吞吐两=量，broker支持主备模式，用于处理活跃的流式数据，大数据量的数据处理

1. **Spring Security 密码加密**

提供了BCryptPasswordEncoder类,实现Spring的PasswordEncoder接口使用BCrypt强 哈希方法来加密密码。BCrypt强哈希方法 每次加密的结果都不一样

1. **什么是JWT**

JSON Web Token（JWT）是一个非常轻巧的规范。这个规范允许我们使用JWT在用 户和服务器之间传递安全可靠的信息, 一个JWT实际上就是一个字符串，它由三部分组成，头部、载荷与签名

项目中用于注册登录JET令牌验证，用户删除，吐槽验证

**七.为什么要使用熔断器？**

在微服务架构中通常会有多个服务层调用，基础服务的故障可能会导致级联故障， 进而造成整个系统不可用的情况，这种现象被称为服务雪崩效应。服务雪崩效应是一种 因“服务提供者”的不可用导致“服务消费者”的不可用,并将不可用逐渐放大的过程。

**八．为什么需要微服务网关？**

微服务工程统一入口，方便前端调用，集中处理权限问题

不同的微服务一般有不同的网络地址，微服务网关是介于客户端和服务器端之间的中 间层，所有的外部请求都会先经过微服务网关。

Zuul组件的核心是一系列的过滤器, 可以完成以下功能：(身份认证和安全, 审查与监控,动态路由，压力测试，负载分配…)

**九.集中配置组件SprinfCloudConfig**

为了方便服务配置文件统一管理，实时更新

**十．消息总线SpringCloudBus**

不重启微服务的情况 下更新配置, 让其立刻生效

**十一. Git(**分布式的版本控制系统**)**

作用：是版本控制的分布式工具，另一个就是作为开源代码库