高可用：LVS+keepalive

熔断技术：hystrix

Zuul：负载均衡与路由网关

1.网关：路由转发+过滤器

2.服务注册发现：调用和被调用方的信息维护

注册中心的作用：暴露服务,注册服务,负载均衡

3.配置中心：管理配置,动态更新 diamond

4.链路追踪：分析调用链路耗时

5.负载均衡器：分发负载

6.熔断：保护自己和被调用方

分布式系统的不足：分布式事务，服务治理

注册中心

服务管理，核心是有个服务注册表,心跳机制动态维护

Zookeep,Eureka,consul

CAP理论 一致性-可用性-分区容错性

Zk：cp设计，一致性与容错性保证,会有主从,leader选举 如金融行业

Eureka：ap设计，无主从，某一个节点挂了，自动切换其他节点，去中心化 如电商行业

**server**:  
 **enable-self-preservation**: **true**/(false)

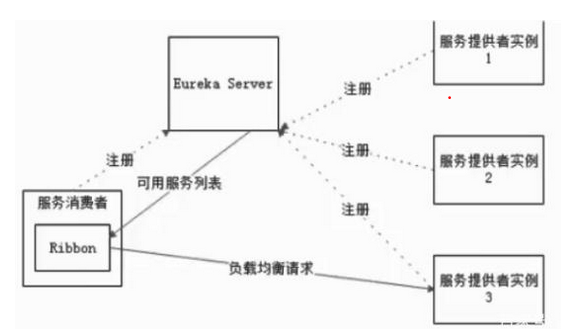
自我保护模式：

默认开启：当eureka client挂掉之后,eureka server不会将服务从注册列表剔除,会继续返回调用者关于被调用方的信息

关闭之后：eureka server将服务从注册列表剔除，调用者无法获取被调用方挂掉的信息

Ribbon,Feign,以及注册中心高可用

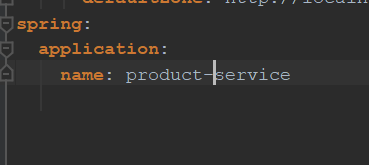
Ribbon示意图：负载均衡，分发请求



注意点1：

java.lang.IllegalStateException: Request URI does not contain a valid hostname

使用ribbon实现负载均衡的时候，服务名称不能用下划线，换成中划线



注意点2：

当eureka-client挂掉之后，eureka-server依然会保持client在控制台一段时间



Ribbon调用源码分析：

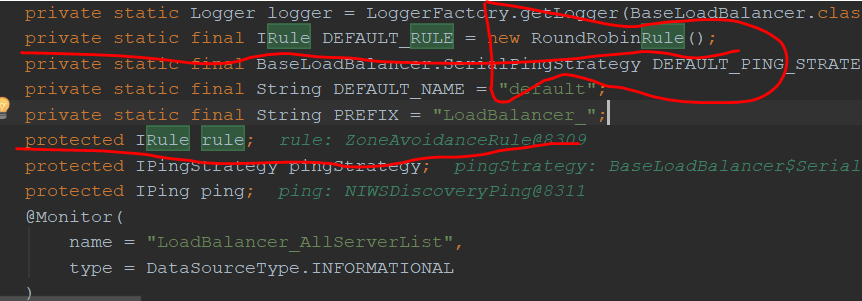
分析@Loadbalanced

首先从注册中心获取provider的liebiao

通过一定的策略选择其中一个节点

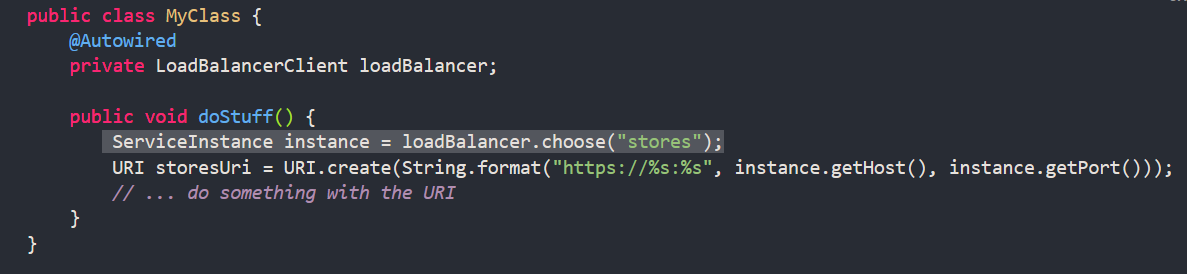
再返回给resttemplate调用

默认是轮询策略



源码调用链：

1. 根据官网提示查找org.springframework.cloud.client.loadbalancer.LoadBalancerClient以及实现类org.springframework.cloud.netflix.ribbon.RibbonLoadBalancerClient

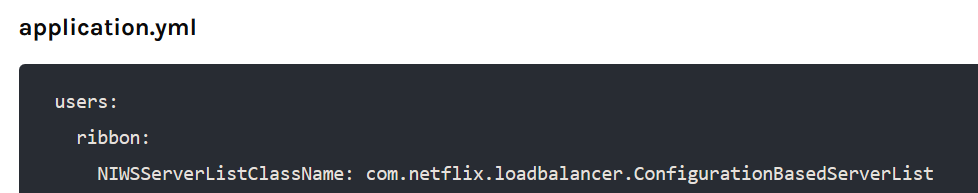


1. org.springframework.cloud.netflix.ribbon.RibbonLoadBalancerClient#choose(java.lang.String, java.lang.Object) 根据此方法逐步进行分析
2. org.springframework.cloud.netflix.ribbon.RibbonLoadBalancerClient#getServer(com.netflix.loadbalancer.ILoadBalancer, java.lang.Object) 得到provider端信息
3. com.netflix.loadbalancer.BaseLoadBalancer#chooseServer在此处进行默认轮询规则的选定

修改策略：

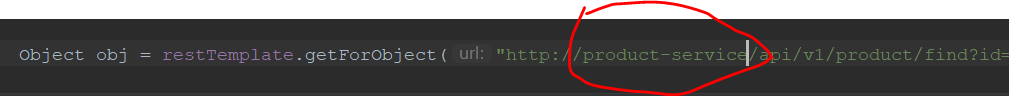
#### https://cloud.spring.io/spring-cloud-static/Hoxton.SR1/reference/htmlsingle/

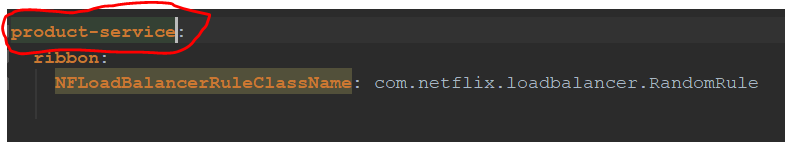
#### 搜索关键字[Customizing the Ribbon Client by Setting Properties](https://cloud.spring.io/spring-cloud-static/Hoxton.SR1/reference/htmlsingle/#customizing-the-ribbon-client-by-setting-properties)



Users：为provider方的应用名称即 product-service

注意：users在yml中的书写应与程序中大小写保持一致，否则规则修改不生效





Feign使用以及源码流程

搜索：feign调用传参

https://cloud.spring.io/spring-cloud-static/Hoxton.SR1/reference/htmlsingle/

#### 搜素关键字[6.1.7. Feign Inheritance Support](https://cloud.spring.io/spring-cloud-static/Hoxton.SR1/reference/htmlsingle/#spring-cloud-feign-inheritance)

Feign包含ribbon



Ribbon的负载均衡是通过哪种方式实现的：拦截器

Spring默认的json处理工具是：Jackson

Ribbon的负载均衡,用于决定调用哪一个服务的接口是：IRule

Feign在不配置的情况下,最少几秒会超时: 1秒

Hystrix 熔断,降级

熔断和降级的互相交集

相同点：

1. 为了防止系统崩溃
2. 体验到某些功能不可用

不同点：

1.服务熔断一般是下游服务（被调用方）故障导致，而服务降级是从整体负荷去考虑，由调用方控制

调用服务出错开始熔断，采用降级的方法返回数据

降级针对接口方法调用处，失败之后需要实现接口的具体降级处理类

源码剖析hystrix的超时策略：

Dashboard：

网页地址：<http://localhost:8763/hystrix/>

监控地址：http://localhost:8763/actuator/hystrix.stream



Zuul网关

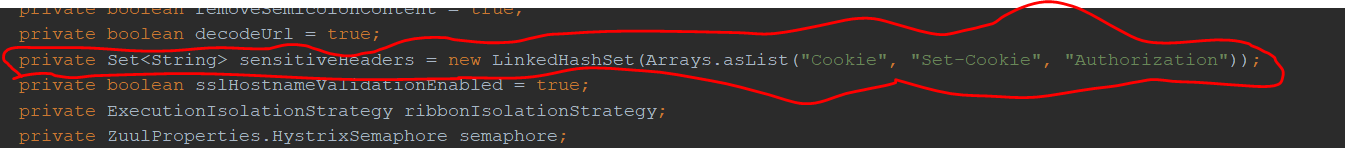
网关访问默认规则：[http://gateway:port（端口号）/service-id（应用名称，可以是商品，订单，积分等应用的名称）/\*\*](http://gateway:port（端口号）/service-id（应用名称，可以是商品，订单，积分等应用的名称）/**)（服务路径）

内外网的隔离

1.路由名称自定义问题

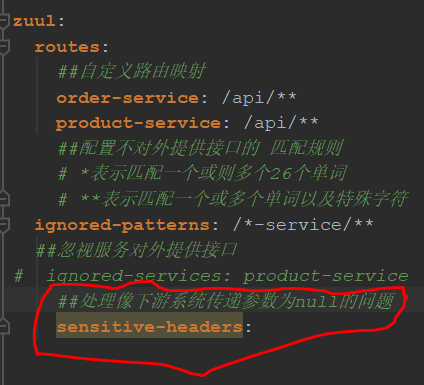
不同服务的路由映射不能重复，负责最后面的会覆盖前面的

2.http请求头过滤问题

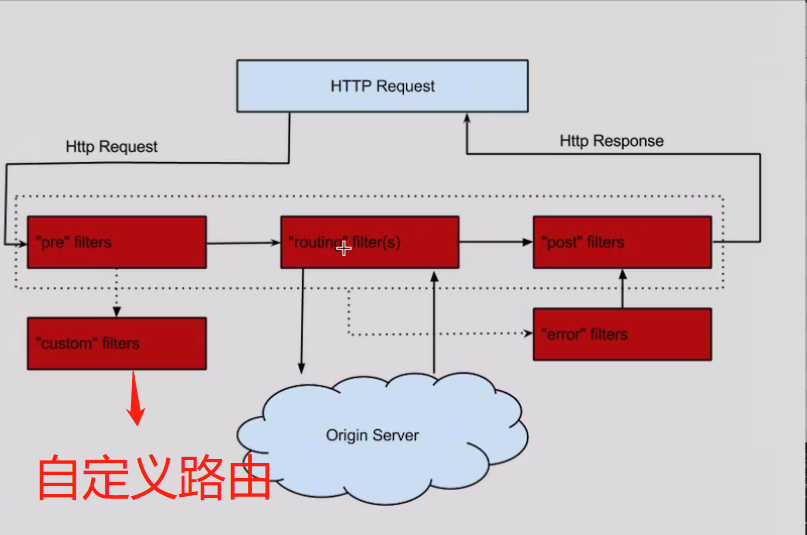


像下游系统传递信息时候，会将这三个值给过滤掉

处理方法



3.过滤器执行顺序问题，order值越小，越先执行



4.有一个共享上下文对象：RequestContext

接口限流：结合guava做限流

Mysql最大连接数：3000

Guava：tryAuquire不会阻塞

Acquire会进行阻塞

链路追踪sleuth与zipkin

Sleuth：日志埋点