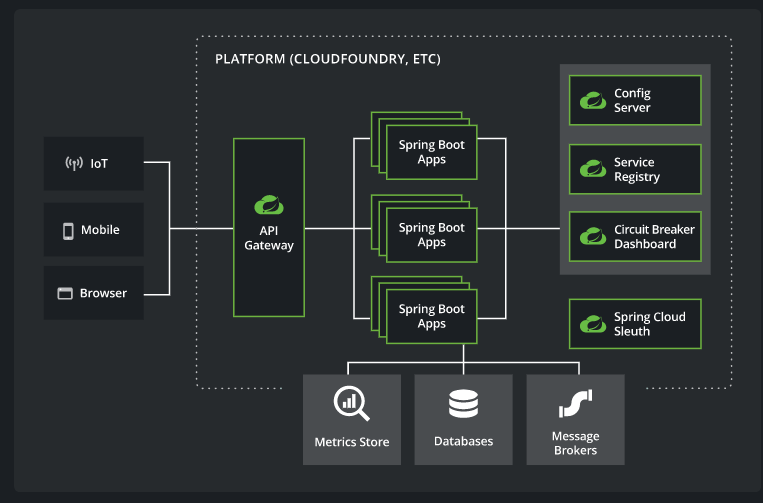
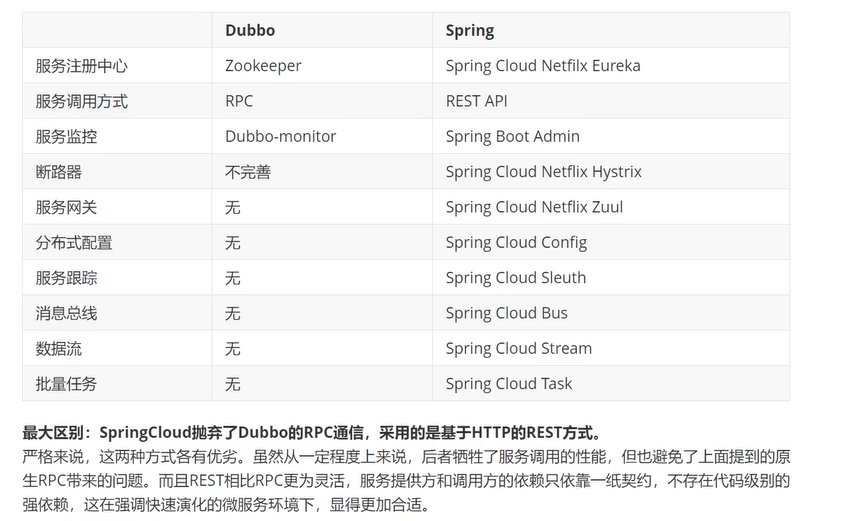
Eureka，ribbon，hystrix（服务熔断），zuul（路由）





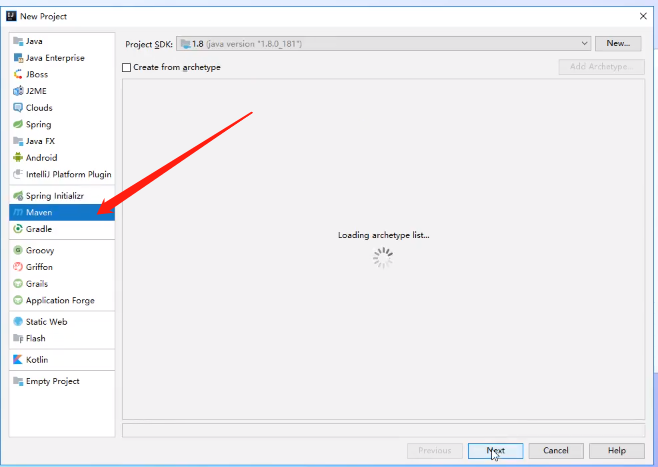
基本套路



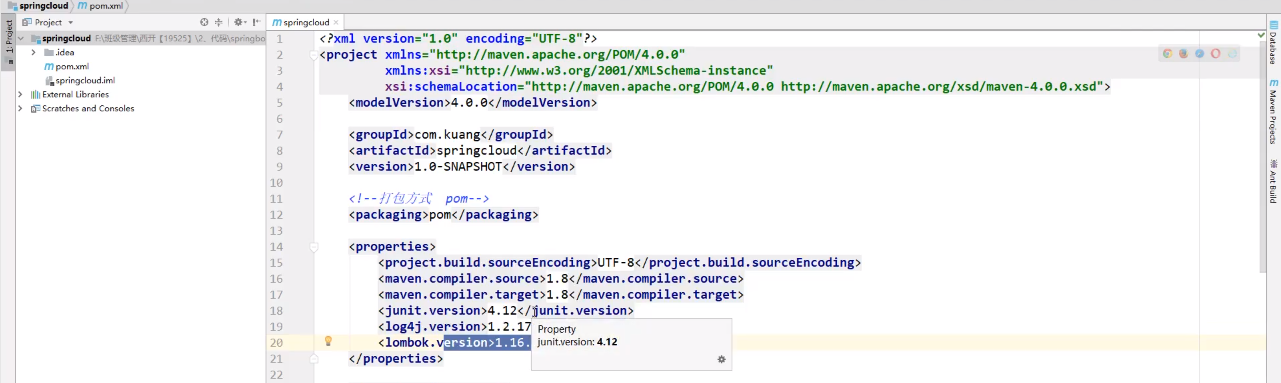




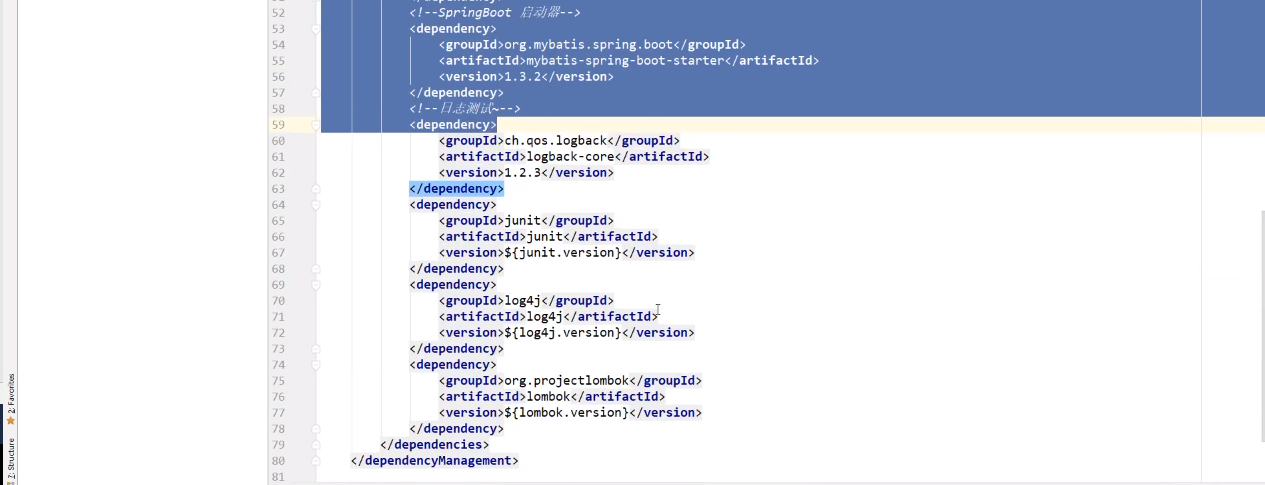
步骤：



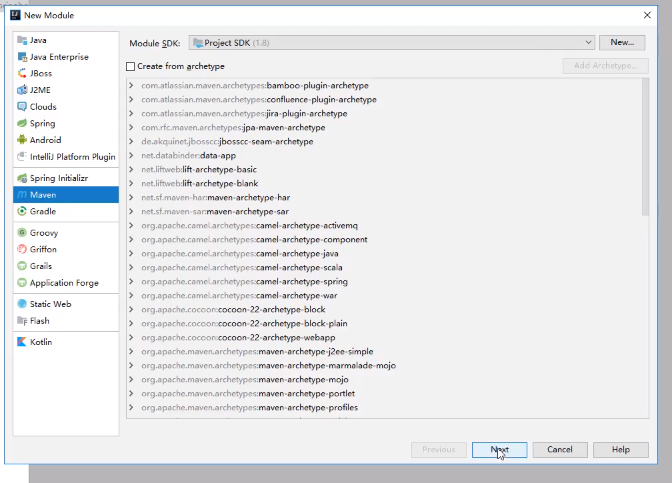
删除src





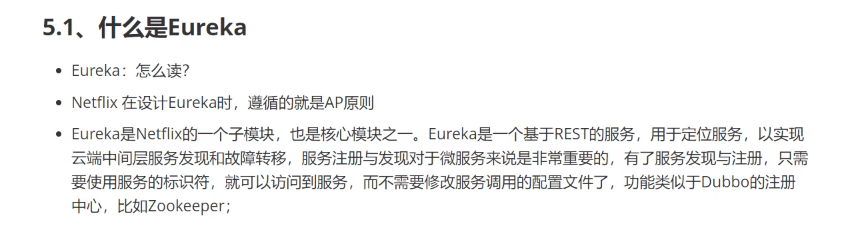
n

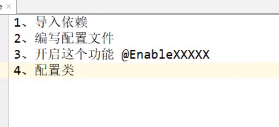
new module



# 服务注册与发现

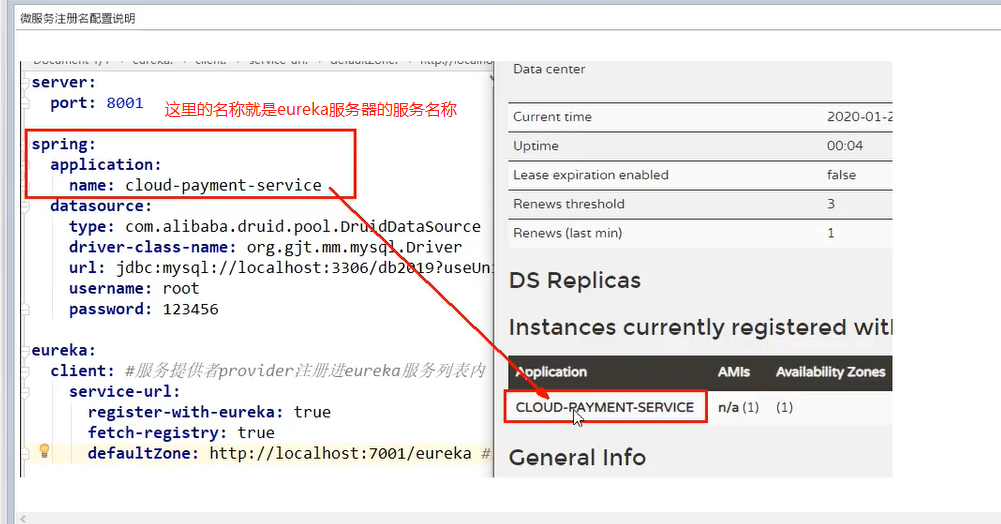
## Eureka





创建Eureka服务器，然后创建Eureka客户端（也就是个个Spring boot编写的service）。





eureka自我保护机制



Eureka已经停止更新。

## Zookeeper

Zookeeper可以代替eureka实现服务注册与发现（用的不多）

Zookeeper服务注册中心需要安装软件，不是像eureka那样写个程序当服务器。

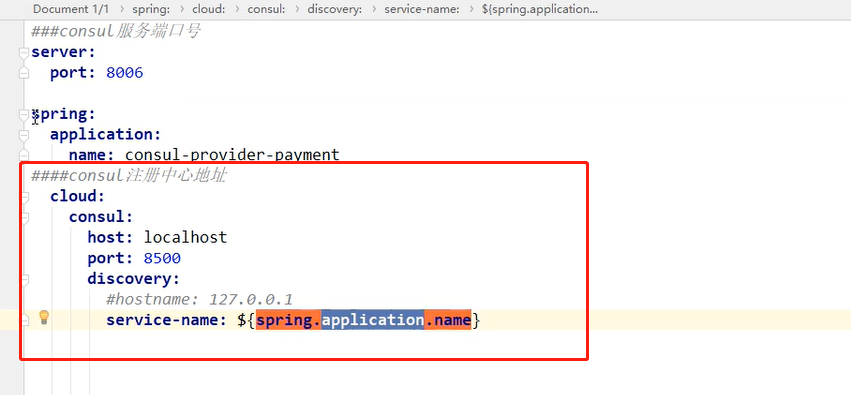


Zookeeper的服务节点是临时性的。服务奔了后，过一段时间就会清理。

## Consul

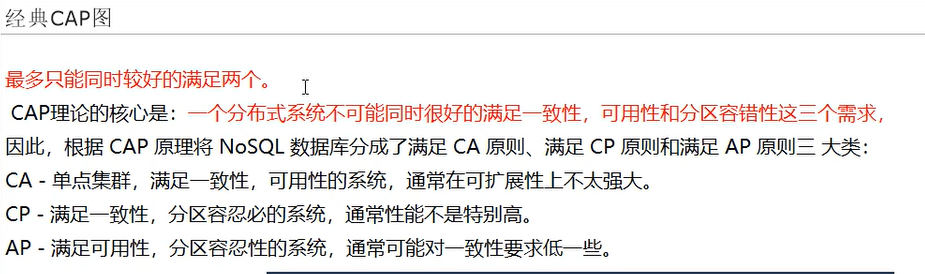
也是需要安装软件。

微服务的注册，如下图所示



## 三个注册中心的异同点



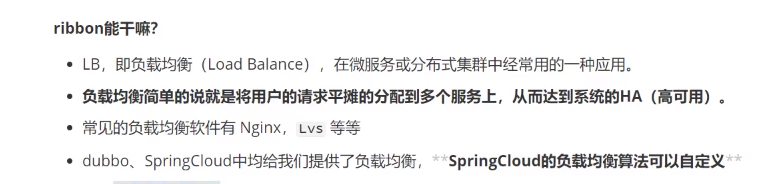




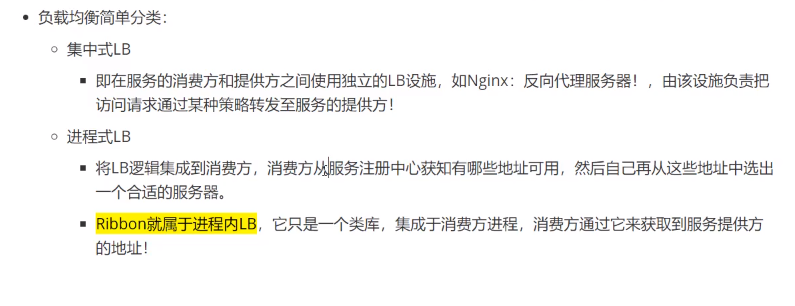


# 负载均衡

基于客户端配置

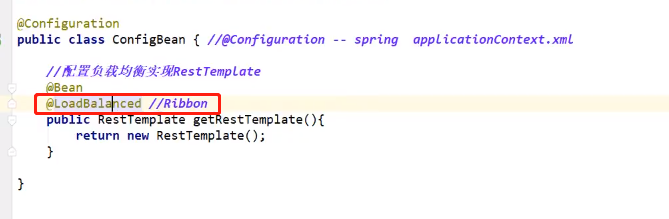


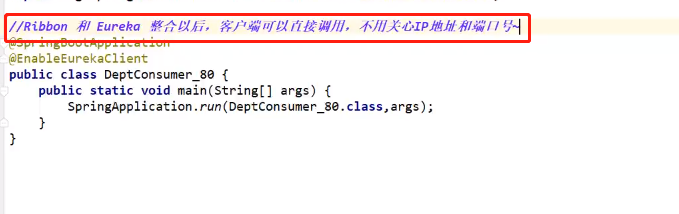
负载均衡简单分类

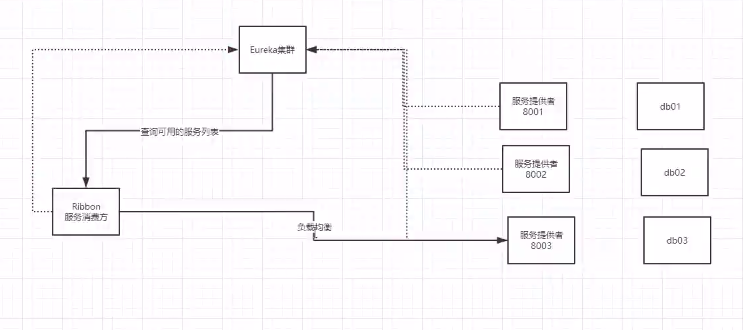


## Ribbon

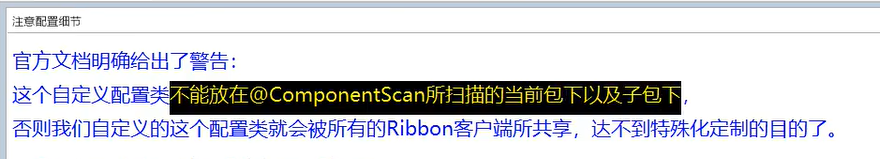
在eureka maven中已经包含了ribbon的jar包了。

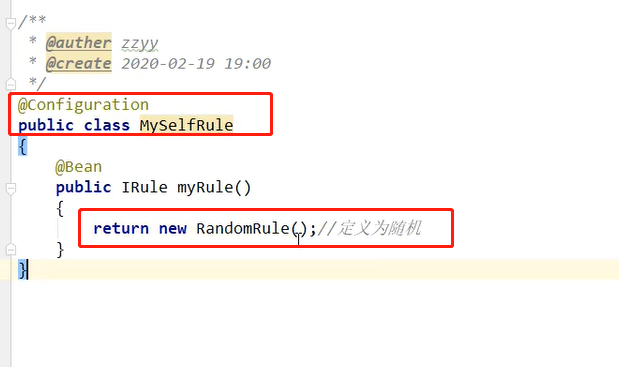




**修改负载均衡的策略**

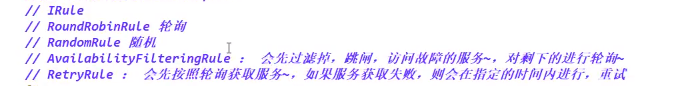




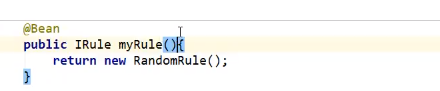


**自定义负载均衡算法**

Ribbon一般负载均衡有轮询，权重，随机…几种算法。可以通过**IRule**接口查看



在@Configuration描述的方法中添加下图代码（使用的是随机的负载均衡策略），即可改变负载均衡策略。



自定义负载均衡的策略可以查看Ribbon的源码，照猫画虎。

使用Ribbon的步骤：

1. 导入依赖，



1. 编写客户端需要到哪里获取服务



1. 加入注解

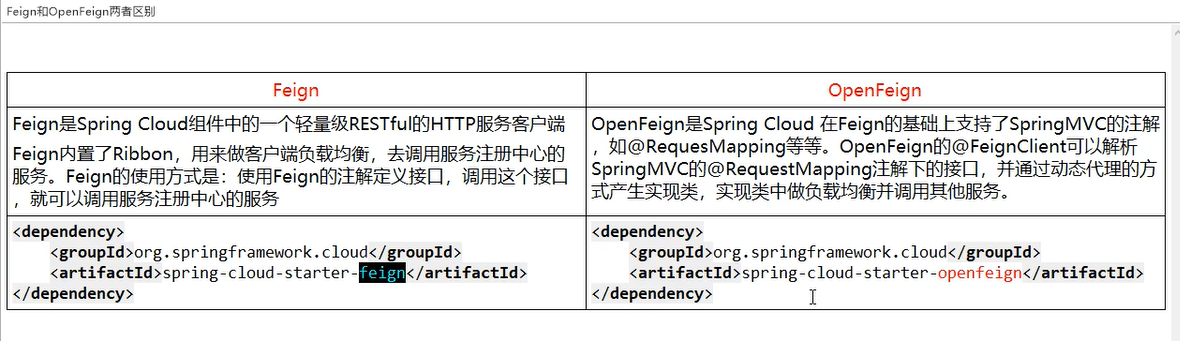


## openFeign服务接口调用

更加符合面向接口编程的习惯

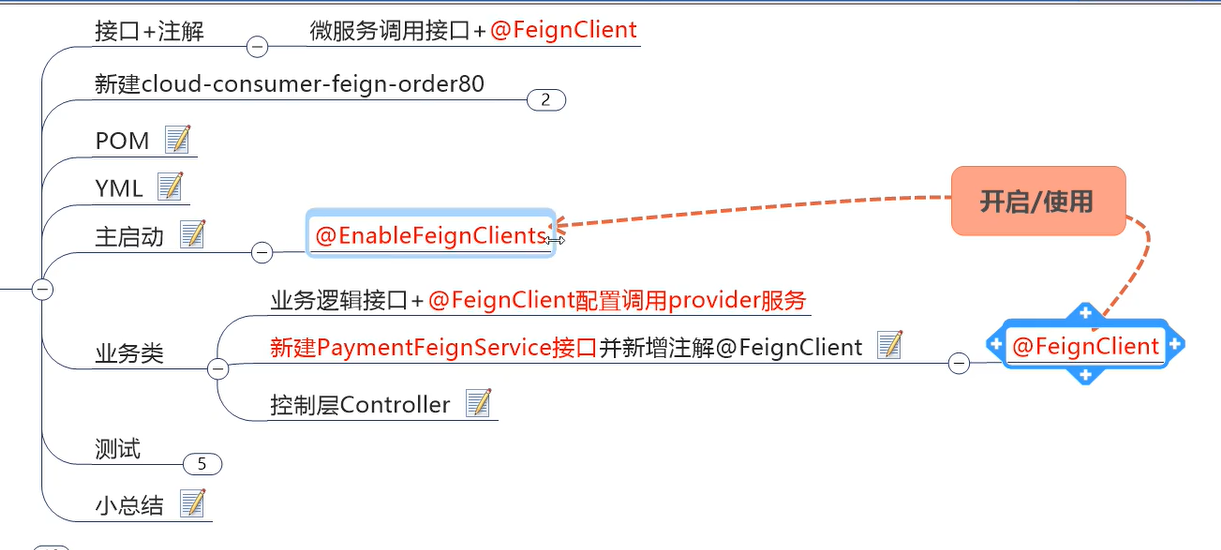


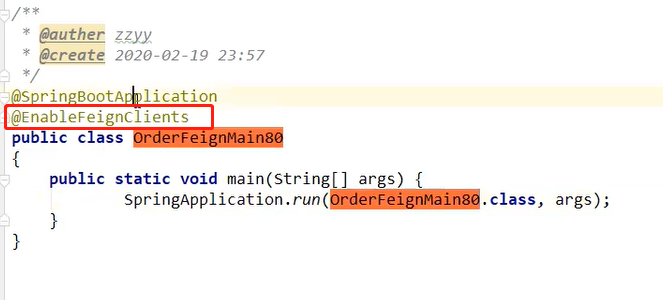
**Feign跟openfeign的区别**



**实现代码**

使用步骤





Service接口+@FeignClient（在消费端利用这种模式，这里的service对应服务提供方的service）



**设置超时控制**

默认超时的时长是1秒，当请求需要较长的时间时，可以手动设置超时的控制，在yml文件中添加下图配置



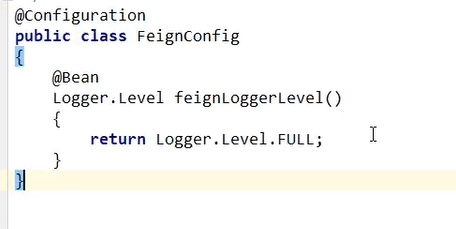
**日志**

日志级别



**步骤：**

*写一个类如下图：*



*在yml文件中添加以下语句*



# Hystrix

## 服务熔断机制，防止出现雪崩效应。

发生在服务端

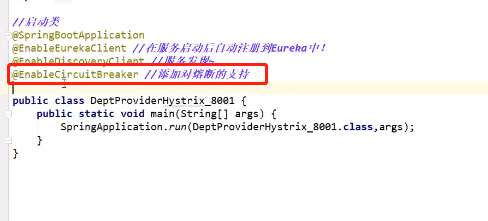
1. 导入依赖



1. 在需要熔断的方法上添加熔断的解决方案

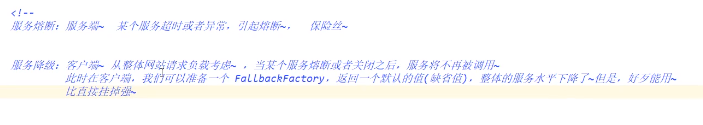


1. 开启支持



## 服务降级

发生在客户端

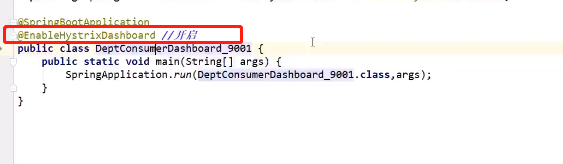


## 服务监控

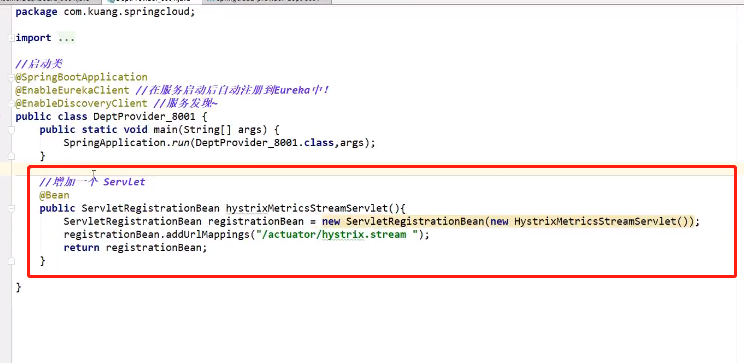
Hystrix Dashboard



写一个监控程序只需要添加上图依赖，添加下图注解即可



在需要监控的地方添加



# Zuul路由网关

