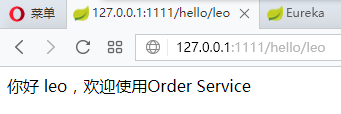
# Spring Cloud网关——Zuul&过滤器&集群方案

## 前期准备

参看之前的文章创建Eureka注册中心，以及两个服务：order-service（端口1111）、user-service（端口2222），get请求返回文本，如下：

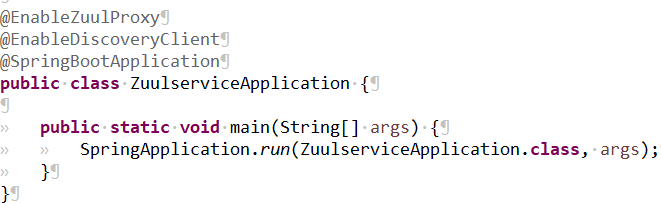




## Zuul设置

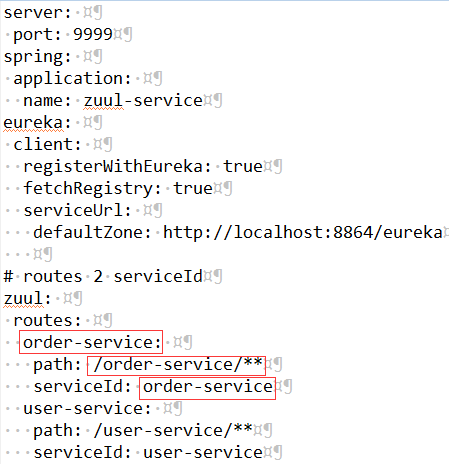
参考之前的文章创建zuul-service工程，依赖选择Web、Eureka Discovery、Zuul。pom文件就不附图了。

### 代码



如图，就多加一个@EnableZuulProxy。

### application.yml

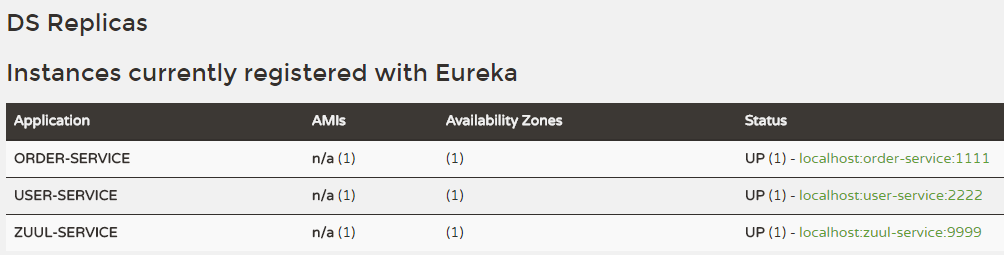


注意红框path是我们在浏览器或前端APP上访问的url，serviceId是之前两个工程的application name。

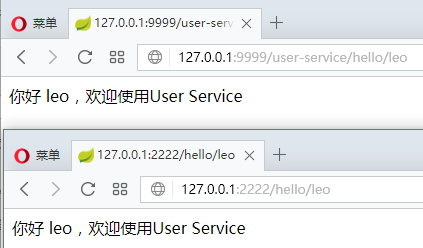
这是按照serviceId来进行路由的，还有一种是用url，这里就不写了。

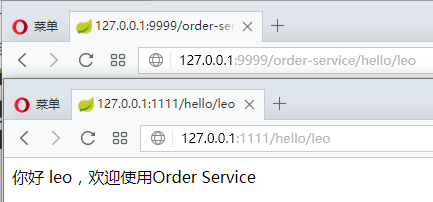
### 结果

启动所有工程，Eureka监控页面如下：



单独访问两个Service，与通过Zuul路由过去访问页面对比：





## Zuul过滤器

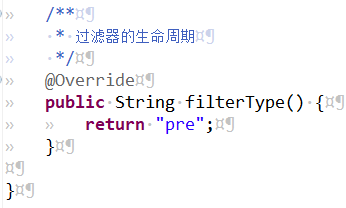
使用路由有一个很直接的好处，就是可以对访问进行某种统一的处理。比如你有N个微服务，都要求有权限的客户端访问，比如本例使用的Access Token。如果所有的微服务都要搞一个Access Token校验的话，那也太Low了。用Zuul就可以了，统一鉴权。

### 自定义过滤器代码

继承ZuulFilter。







这里主要说一下filterType，有4个值：

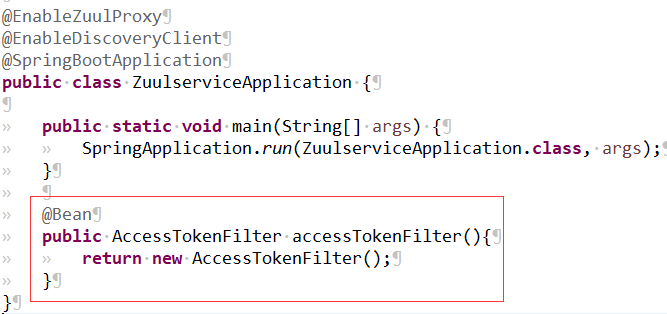
pre：进行路由前调用；

routing：进行路由的时候调用；

error：发生错误时调用；

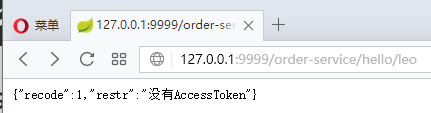
post：routing&error之后调用；

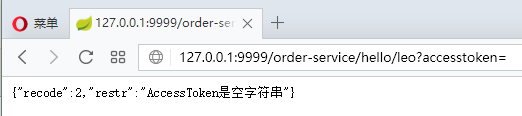
### 实例化

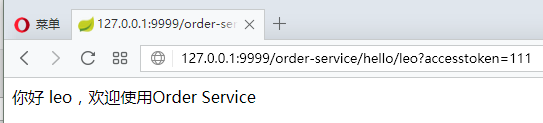


### 结果

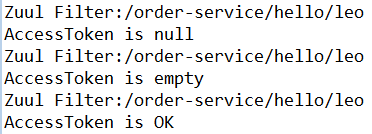
启动4个服务，分别输入如下图的URL，结果如下：







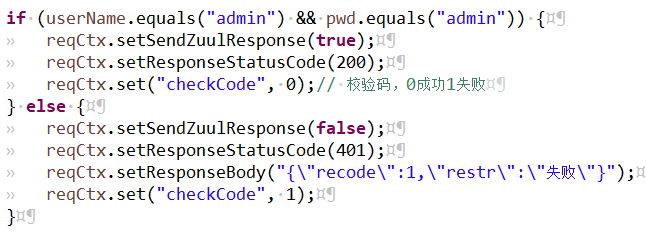
日志：



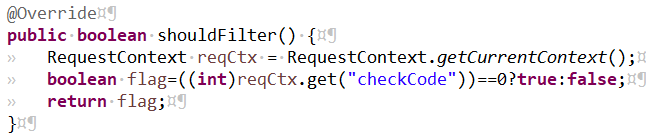
### 过滤器之间传递数据

实际开发中，我们可能需要配置多个过滤器，比如第一个过滤器是校验用户名、密码，第一个过滤器是校验此用户的是否有访问请求URL的权限。如果第一个通过了，则继续执行过滤器2，如果第一个没通过，过滤器2就不执行。

所以我们先在第一个过滤器中的校验代码里增加如下代码：



在第二个过滤器里增加如下代码：



## Zuul的高可用

因为所有的请求都要从Zuul过，生产环境中它的高可用就是必须的了。

简单的说，Zuul我们可以把他们当成普通的服务，所以我们可以部署多个，然后注册到Eureka注册中心，A微服务调用其中某个Zuul服务，再通过它转发到B服务。

这种场景适用于内部调用，对于外部调用来说（比如网页、手机客户端、第三方请求），人家总不能再去注册你的Eureka服务端吧，再说把Eureka注册中心暴露在外网也很不安全。

可以用如下办法解决，对于Zuul来说暴露在外面的也就是URL，我们可以使用Nginx对外暴露一个URL，然后转发到Zuul服务，Nginx还有负载均衡。