1.zuul.routes.api-a.path=/api-a/\*\*

zuul.routes.api-a.serviceId=feign-consumer

当访问地址符合/api-a/\*\*规则的时候，会被自动定位到feign-consumer服务上去，不过两行代码有点麻烦，可以用下面一行代码来代替，如下：

zuul.routes.feign-consumer=/api-a/\*\*

zuul.routes后面跟着的是服务名，服务名后面跟着的是路径规则

如果映射规则什么都不写，系统也给提供了一套默认的配置规则，默认的配置规则如下：

zuul.routes.feign-consumer.path=/feign-consumer/\*\*

zuul.routes.feign-consumer.serviceId=feign-consumer

默认情况下，Eureka上所有注册的服务都会被Zuul创建映射关系来进行路由，但是对于这里的例子来说，希望提供服务的是feign-consumer，hello-service作为服务提供者只对服务消费者提供服务，不对外提供服务，如果使用默认的路由规则，则Zuul也会自动为hello-service创建映射规则，这个时候可以采用如下方式来让Zuul跳过hello-service服务，不为其创建路由规则：

zuul.ignored-services=hello-service



有的时候我们还会遇到这样一个问题，比如我有两个服务，一个叫做feign-consumer，还有一个叫做feign-consumer-hello，此时我的路由配置规则可能这样来写：

zuul.routes.feign-consumer.path=/feign-consumer/\*\*

zuul.routes.feign-consumer.serviceId=feign-consumer

zuul.routes.feign-consumer-hello.path=/feign-consumer/hello/\*\*

zuul.routes.feign-consumer-hello.serviceId=feign-consumer-hello

此时我访问feign-consumer-hello的路径会同时被这两条规则所匹配，Zuul中的路径匹配方式是一种线性匹配方式，即按照路由匹配规则的存储顺序依次匹配，因此我们只需要确保feign-consumer-hello的匹配规则被先定义feign-consumer的匹配规则被后定义即可，但是在properties文件中我们不能保证这个先后顺序，此时我们需要用YAML来配置，这个时候我们可以删掉resources文件夹下的application.properties，然后新建一个application.yml，内容如下：

spring:

 application:

   name: api-gateway

server:

 port: 2006

zuul:

 routes:

   feign-consumer-hello:

     path: /feign-consumer/hello/\*\*

     serviceId: feign-consumer-hello

   feign-consumer:

     path: /feign-consumer/\*\*

     serviceId: feign-consumer

eureka:

 client:

   service-url:

     defaultZone: <http://localhost:1111/eureka/>

这个时候我们就可以确保先加载feign-consumer-hello的匹配规则，后加载feign-consumer的匹配规则。

上文我们说了一个zuul.ignored-services=hello-service属性可以忽略掉一个服务，不给某个服务设置映射规则，这个配置我们可以进一步细化，比如说我不想给/hello接口路由，那我们可以按如下方式配置(后面我都用yaml配置)：

zuul:

 ignored-patterns: /\*\*/hello/\*\*

此时访问/hello接口就会报404错误

此外，我们也可以统一的为路由规则增加前缀，设置方式如下：

zuul:

 prefix: /myapi

此时我们的访问路径就变成了<http://localhost:2006/myapi/feign-consumer/hello1>。

有的时候我们可能也需要在API网关上做一点业务逻辑操作，比如我现在在api-gateway项目中新建如下Controller：

@RestController

public class HelloController {

   @RequestMapping("/local")

   public String hello() {

       return "hello api gateway";

   }

}

我希望用户在访问/local时能够自动跳转到这个方法上来处理，那么此时我们需要用到Zuul的本地跳转，配置方式如下：

zuul:

 prefix: /myapi

 ignored-patterns: /\*\*/hello/\*\*

 routes:

   local:

     path: /local/\*\*

     url: forward:/local

假设我的feign-consumer中提供了一个接口/hello4，当访问/hello4接口的时候，页面重定向到/hello，默认情况下，重定向的地址是具体的服务实例的地址，而不是API网关的跳转地址，这种做法会暴露真实的服务地址，所以需要在Zuul中配置，配置方式很简单，如下：

zuul:

 add-host-header: true

表示API网关在进行请求路由转发前为请求设置Host头信息。

默认情况下，敏感的头信息无法经过API网关进行传递，我们可以通过如下配置使之可以传递：

zuul:

 routes:

   feign-consumer:

     sensitiveHeaders:

在Zuul中，Ribbon和Hystrix的配置还是和之前的配置方式一致，这里我就不赘述了，如果我们想关闭Hystrix重试机制

zuul:

 retryable: false

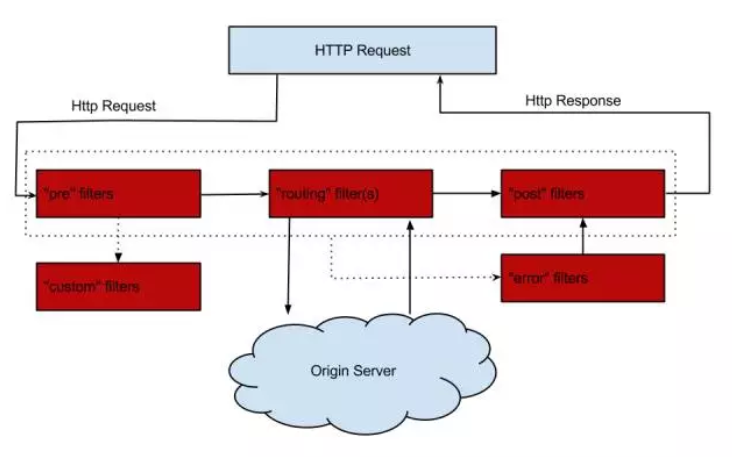
关闭某一个服务的重试机制：

zuul:

 routes:

   feign-consumer:

     retryable: false

官方给出的Zuul请求的生命周期图 

1.正常情况下所有的请求都是按照pre、route、post的顺序来执行，然后由post返回response

2.在pre阶段，如果有自定义的过滤器则执行自定义的过滤器

3.pre、routing、post的任意一个阶段如果抛异常了，则执行error过滤器，然后再执行post给出响应