11.1 服务网关

Zuul 和 Gateway

由于每一个微服务的地址都有可能发生变化,无法直接对外公布这些服务地址,基于安全以及高内聚低 耦合等设计,我们有必要将内部系统和外部系统做一个切割。

一个专门用来处理外部请求的组件, 就是服务网关。

- 权限问题统一处理
- 数据剪裁和聚合
- 简化客户端的调用
- 可以针对不同的客户端提供不同的网关支持

Spring Cloud 中, 网关主要有两种实现方案:

- Zuul
- Spring Cloud Gateway

11.2 Zuul

Zuul 是 Netflix 公司提供的网关服务。

Zuul 的功能:

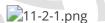
- 权限控制,可以做认证和授权
- 监控
- 动态路由
- 负载均衡
- 静态资源处理

Zuul 中的功能基本上都是基于过滤器来实现,它的过滤器有几种不同的类型:

- PRE
- ROUTING
- POST
- ERROR

11.2.1 HelloWorld

首先创建项目,添加 Zuul 依赖。



项目创建成功后,将 zuul 注册到 eureka上:

```
spring.application.name=zuul
server.port=2020
eureka.client.service-url.defaultZone=http://localhost:1111/eureka
```

然后在启动类上开启网关代理:

```
@SpringBootApplication
@EnableZuulProxy//开启网关代理
public class ZuulApplication {

public static void main(String[] args) {
    SpringApplication.run(ZuulApplication.class, args);
}
```

配置完成后,重启 Zuul,接下来,在浏览器中,通过 Zuul 的代理就可以访问到 provider 了。

http://localhost:2020/provider/hello

在这个访问地址中,provider 就是要访问的服务名称,/hello 则是要访问的服务接口。

这是一个简单例子, Zuul 中的路由规则也可以自己配置。

```
zuul.routes.javaboy-a.path=/javaboy-a/**
zuul.routes.javaboy-a.service-id=provider
```

上面这个配置,表示 /javaboy-a/**,满足这个匹配规则的请求,将被转发到 provider 实例上。 上面两行配置,也可以进行简化:

```
zuul.routes.provider=/javaboy-a/**
```

11.2.2 请求过滤

对于来自客户端的请求,可以在 Zuul 中进行预处理,例如权限判断等。

定义一个简单的权限过滤器:

```
@Component
public class PermissFilter extends ZuulFilter {
   /**
    * 过滤器类型,权限判断一般是 pre
    * @return
    */
   @override
   public String filterType() {
       return "pre";
    * 过滤器优先级
    * @return
    */
   @override
   public int filterOrder() {
       return 0;
   }
   /**
    * 是否过滤
    * @return
```

```
@override
   public boolean shouldFilter() {
       return true:
   /**
    * 核心的过滤逻辑写在这里
     * @return 这个方法虽然有返回值,但是这个返回值目前无所谓
     * @throws ZuulException
    */
   @override
   public Object run() throws ZuulException {
       RequestContext ctx = RequestContext.getCurrentContext();
       HttpServletRequest request = ctx.getRequest();//获取当前请求
       String username = request.getParameter("username");
       String password = request.getParameter("password");
       if (!"javaboy".equals(username) || !"123".equals(password)) {
           //如果请求条件不满足的话,直接从这里给出响应
           ctx.setSendZuulResponse(false);
           ctx.setResponseStatusCode(401);
           ctx.addZuulResponseHeader("content-type","text/html;charset=utf-8");
           ctx.setResponseBody("非法访问");
       }
       return null;
   }
}
```

重启 Zuul,接下来,发送请求必须带上 username 和 password 参数,否则请求不通过。

http://localhost:2020/javaboy-a/hello?username=javaboy&password=123

11.2.3 Zuul 中的其他配置

匹配规则

例如有两个服务,一个叫 consumer,另一个叫 consumer-hello,在做路由规则设置时,假如出现了如下配置:

```
zuul.routes.consumer=/consumer/**
zuul.routes.consumer-hello=/consumer/hello/**
```

此时,如果访问一个地址: http://localhost:2020/consumer/hello/123, 会出现冲突。实际上,这个地址是希望和 consumer-hello 这个服务匹配的,这个时候,只需要把配置文件改为 yml 格式就可以了。

忽略路径

默认情况下,zuul 注册到 eureka 上之后,eureka 上的所有注册服务都会被自动代理。如果不想给某一个服务做代理,可以忽略该服务,配置如下:

```
zuul.ignored-services=provider
```

上面这个配置表示忽略 provider 服务,此时就不会自动代理 provider 服务了。

也可以忽略某一类地址:

```
zuul.ignored-patterns=/**/hello/**
```

这个表示请求路径中如果包含 hello,则不做代理。

前缀

也可以给路由加前缀。

zuul.prefix=/javaboy

这样,以后所有的请求地址自动多了前缀,/javaboy

