# SpringCould[Zuul服务路由]

|  |
| --- |
| //此注解表示该类是一个路由的JAVA类,并且开启了路由的功能  /\*\*  \* **@EnableZuulProxy**  \* 路由的核心动词为:转发,过滤.  \* zuul默认和Rubbon实现了负载均衡的效果  \*  \*/  @EnableZuulProxy  @EnableEurekaClient //表示该类为服务提供者,去服务中心注册服务(配置中体现)  @SpringBootApplication  **public** **class** ServiceZuulApplication {  **public** **static** **void** main(String[] args) {  SpringApplication.*run*(ServiceZuulApplication.**class**, args);  }  } |
| \*\*  \***@Component** 把普通pojo实例化到spring容器中  \* ZuulFilter:为Zuul的过滤器  \*/  @Component  **public** **class** MyFilter **extends** ZuulFilter{  **private** **static** Logger *log* = LoggerFactory.*getLogger*(MyFilter.**class**);  /\*\*  \* filterType：返回一个字符串代表过滤器的类型，在zuul中定义了四种不同生命周期的过滤器类型  \* pre：路由之前  \* routing：路由之时  \* post： 路由之后  \* error：发送错误调用  \* filterOrder：过滤的顺序  \* shouldFilter：这里可以写逻辑判断，是否要过滤，本文true,永远过滤。  \* run：过滤器的具体逻辑。可用很复杂，包括查sql，nosql去判断该请求到底有没有权限访问。  \*/  @Override  **public** String filterType() {  **return** "pre";  }  @Override  **public** **int** filterOrder() {  **return** 0;  }  @Override  **public** **boolean** shouldFilter() {  **return** **true**;  }  @Override  **public** Object run() {  RequestContext ctx = RequestContext.*getCurrentContext*();  HttpServletRequest request = ctx.getRequest();  *log*.info(String.*format*("%s >>> %s", request.getMethod(), request.getRequestURL().toString()));  Object accessToken = request.getParameter("token");  **if**(accessToken == **null**) {  *log*.warn("token is empty");  ctx.setSendZuulResponse(**false**);  ctx.setResponseStatusCode(401);  **try** {  ctx.getResponse().getWriter().write("token is empty");  }**catch** (Exception e){}  **return** **null**;  }  *log*.info("ok");  **return** **null**;  }  } |
| eureka:  client:  serviceUrl:  defaultZone: http://localhost:8761/eureka/  server:  port: 8769  spring:  application:  name: service-zuul  zuul:  routes:  api-a:  path: /api-a/\*\* ##如果URL是/api-a/的话,转发给服务名为service-ribbon  serviceId: service-ribbon  api-b:  path: /api-b/\*\* ##如果URL是/api-a/的话,转发给服务名为service-feign  serviceId: service-feign |