创建一个gateway网关的微服务项目

1. 引依赖

<**dependency**>  
 <**groupId**>org.springframework.cloud</**groupId**>  
 <**artifactId**>spring-cloud-starter-gateway</**artifactId**>  
 </**dependency**>

1. Yml配置文件

*# 服务端口***server.port**=**80***# 服务名***spring.application.name**=**service-gateway***# nacos服务地址***spring.cloud.nacos.discovery.server-addr**=**127.0.0.1:8848***#使用服务发现路由***spring.cloud.gateway.discovery.locator.enabled**=**true***#设置路由id***spring.cloud.gateway.routes[0].id**=**service-hosp***#设置路由的uri***spring.cloud.gateway.routes[0].uri**=**lb://service-hosp***#设置路由断言,代理servicerId为auth-service的/auth/路径***spring.cloud.gateway.routes[0].predicates**= **Path=/\*/hosp/\*\****#设置路由id***spring.cloud.gateway.routes[1].id**=**service-cmn***#设置路由的uri***spring.cloud.gateway.routes[1].uri**=**lb://service-cmn***#设置路由断言,代理servicerId为auth-service的/auth/路径***spring.cloud.gateway.routes[1].predicates**= **Path=/\*/cmn/\*\***

3、添加主启动类， gateway服务起来就可以访问了

Gateway网关的三大核心概念

1. route（路由）
2. Predicate（断言）
3. Filter（过滤）
4. 自定义过滤器（主要用于token的验证）

// gateway 自定义拦截器， 微服务中统一做鉴权处理，登录处理  
@Component  
public class MyGateWayFilter implements GlobalFilter, Ordered {  
 @Override  
 public Mono<Void> filter(ServerWebExchange exchange, GatewayFilterChain chain) {  
  
 // 添加不被网关拦截的路径  
 List<String> list = new ArrayList<>();  
 list.add("/api/findTeacher");  
  
 // 获取当前的访问路径  
 String path = exchange.getRequest().getPath().pathWithinApplication().toString();  
  
 // 便利不被拦截的路径的 list， 发现访问的路径匹配上就放行，网关不拦截  
 for (int i = 0; i< list.size(); i++) {  
 if(path.equals(list.get(i))) {  
 return chain.filter(exchange); // 放行  
 }  
 }  
  
 // 获取get请求的参数(拼接在访问路径上的参数)  
 String getParams = exchange.getRequest().getQueryParams().getFirst("username");  
  
 // 获取前端请求过来的token  
 List<String> user\_token = exchange.getRequest().getHeaders().get("user\_token");  
 String token = user\_token != null ? user\_token.get(0) : "";  
  
 // redis里面取出来token(比如取出来的是下面的字符串)  
 String redisToken = "xuxiao";  
  
 if(redisToken.equals(token)) { // 匹配上  
 return chain.filter(exchange); // 放行  
 }else{ // 匹配不上  
 exchange.getResponse().setStatusCode(HttpStatus.*NOT\_ACCEPTABLE*);  
 return exchange.getResponse().setComplete();  
 }  
 }  
  
 @Override  
 public int getOrder() {  
 return 0;  
 }  
}