**网关Gateway**

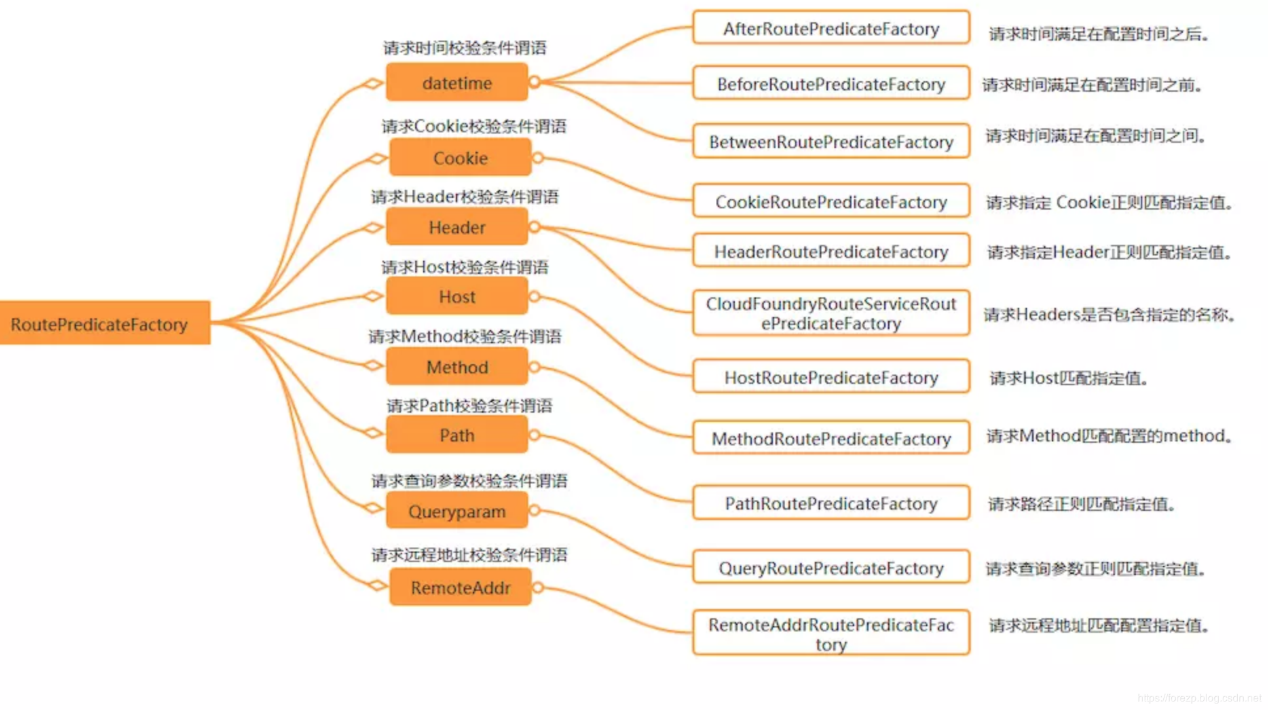
# 介绍

Spring Cloud Gateway是Spring Cloud官方推出的第二代网关框架，取代Zuul网关。网关作为流量的，在微服务系统中有着非常作用，网关常见的功能有路由转发、权限校验、限流控制等作用。

# Predict

主要路由转发功能。

各种Predicate如下图：



# Filter

Spring Cloud Gateway 的 Filter 的生命周期不像 Zuul 的那么丰富，它只有两个：“pre” 和 “post”。

PRE： 这种过滤器在请求被路由之前调用。我们可利用这种过滤器实现身份验证、在集群中选择请求的微服务、记录调试信息等。

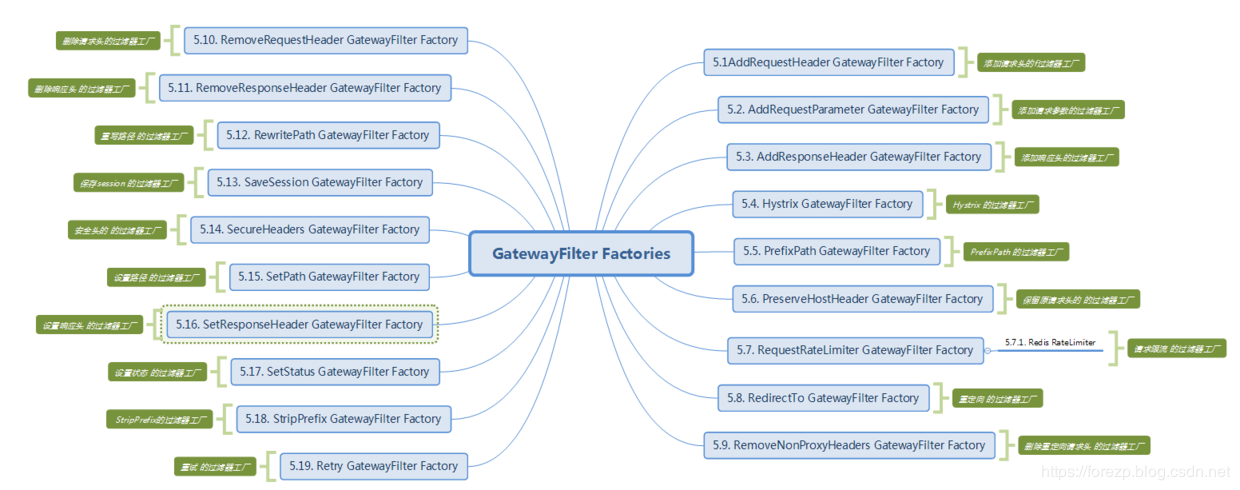
POST：这种过滤器在路由到微服务以后执行。这种过滤器可用来为响应添加标准的 HTTP Header、收集统计信息和指标、将响应从微服务发送给客户端等。

Spring Cloud Gateway 的 Filter 从作用范围可分为另外两种GatewayFilter 与 GlobalFilter。

GatewayFilter：应用到单个路由或者一个分组的路由上。（一般和Predicate组合使用）

GlobalFilter：应用到所有的路由上。

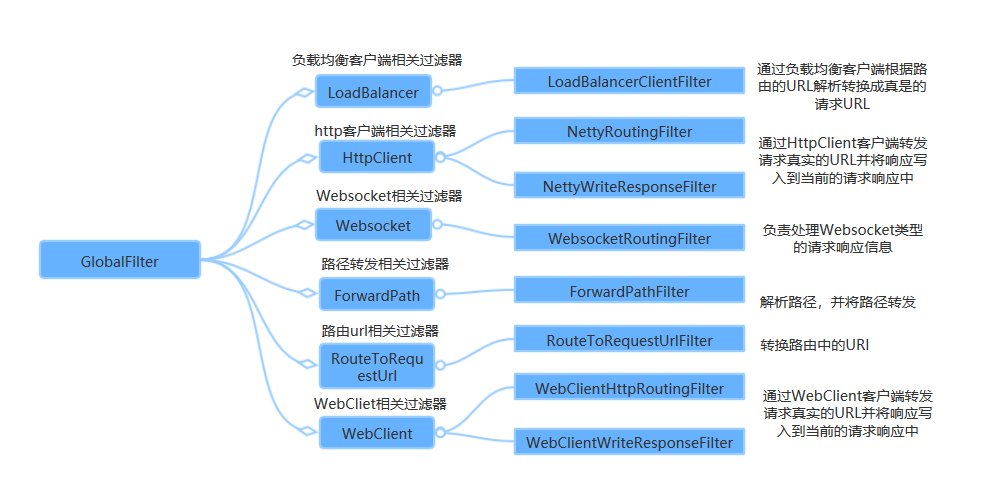
GatewayFilterFactories



自定义GatewayFilter又有两种实现方式，一种是直接实现GatewayFilter接口，另一种是自定义过滤器工厂（继承AbstractGatewayFilterFactory类） , 选择自定义过滤器工厂的方式，可以在配置文件中配置过滤器了。**推荐自定义过滤器工厂。**

路由配置和过滤器配置可以使用nacos配置中心进行配置管理，而且会实时刷新生效。

GlobalFilter



自定义GlobalFilter实现GlobalFilter，Order并交由springIOC管理即可。

|  |
| --- |
| /\*\*  \* Token 校验全局过滤器  \*/  @Component  public class AuthorizeFilter implements GlobalFilter, Ordered {  private static final Logger log = LoggerFactory.getLogger( AuthorizeFilter.class );  private static final String AUTHORIZE\_TOKEN = "token";  @Override  public Mono<Void> filter(ServerWebExchange exchange, GatewayFilterChain chain) {  String token = exchange.getRequest().getQueryParams().getFirst( AUTHORIZE\_TOKEN );  if ( StringUtils.isBlank( token )) {  log.info( "token is empty ..." );  exchange.getResponse().setStatusCode( HttpStatus.UNAUTHORIZED );  return exchange.getResponse().setComplete();  }  return chain.filter(exchange);  }  @Override  public int getOrder() {  return 0;  }  } |

# 解决问题

对服务应用有感知且重要的功能，例如鉴权。

对服务应用无感知的边缘服务，例如流控、监控、页面级缓存等。

Important

Spring Cloud Gateway is built upon Spring Boot 2.x, Spring WebFlux, and Project Reactor. As a consequence many of the familiar synchronous libraries (Spring Data and Spring Security, for example) and patterns you may not apply when using Spring Cloud Gateway. If you are unfamiliar with these projects we suggest you begin by reading their documentation to familiarize yourself with some of the new concepts before working with Spring Cloud Gateway.

Important

Spring Cloud Gateway requires the Netty runtime provided by Spring Boot and Spring Webflux. It does not work in a traditional Servlet Container or built as a WAR.

# 对比

Gateway使用的是webflux。

路由功能，负载功能ribbon。过滤，

# 实战Demo

引入依赖

|  |
| --- |
| <?xml version="1.0" encoding="UTF-8"?>  <project xmlns="http://maven.apache.org/POM/4.0.0"  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">  <modelVersion>4.0.0</modelVersion>  <groupId>com.zoe</groupId>  <artifactId>gateway-sentinel</artifactId>  <version>1.0-SNAPSHOT</version>  <parent>  <groupId>org.springframework.boot</groupId>  <artifactId>spring-boot-starter-parent</artifactId>  <version>2.1.8.RELEASE</version>  <relativePath/> <!-- lookup parent from repository -->  </parent>  <properties>  <java.version>1.8</java.version>  <spring-cloud.version>Greenwich.SR3</spring-cloud.version>  </properties>  <dependencies>  <dependency>  <groupId>com.alibaba.cloud</groupId>  <artifactId>spring-cloud-starter-alibaba-sentinel</artifactId>  <version>2.1.0.RELEASE</version>  </dependency>  <dependency>  <groupId>com.alibaba.cloud</groupId>  <artifactId>spring-cloud-alibaba-sentinel-gateway</artifactId>  <version>2.1.0.RELEASE</version>  </dependency>  <dependency>  <groupId>org.springframework.cloud</groupId>  <artifactId>spring-cloud-starter-gateway</artifactId>  </dependency>  <dependency>  <groupId>com.alibaba.csp</groupId>  <artifactId>sentinel-transport-simple-http</artifactId>  <version>1.6.3</version>  </dependency>  <dependency>  <groupId>org.springframework.cloud</groupId>  <artifactId>spring-cloud-starter-alibaba-nacos-discovery</artifactId>  <version>0.9.0.RELEASE</version>  </dependency>  <!--nacos-config-->  <dependency>  <groupId>com.alibaba.cloud</groupId>  <artifactId>spring-cloud-alibaba-nacos-config</artifactId>  <version>2.1.0.RELEASE</version>  </dependency>  <dependency>  <groupId>org.springframework.cloud</groupId>  <artifactId>spring-cloud-starter-sleuth</artifactId>  </dependency>  <dependency>  <groupId>org.springframework.cloud</groupId>  <artifactId>spring-cloud-starter-zipkin</artifactId>  </dependency>  </dependencies>  <dependencyManagement>  <dependencies>  <dependency>  <groupId>org.springframework.cloud</groupId>  <artifactId>spring-cloud-dependencies</artifactId>  <version>${spring-cloud.version}</version>  <type>pom</type>  <scope>import</scope>  </dependency>  </dependencies>  </dependencyManagement>  <build>  <plugins>  <plugin>  <groupId>org.springframework.boot</groupId>  <artifactId>spring-boot-maven-plugin</artifactId>  </plugin>  </plugins>  </build>  </project> |

* + - 1. 路由转发
      2. 权限校验

|  |
| --- |
| spring:  cloud:  gateway:  discovery:  locator:  enabled: true #开启服务发现（但是不开nacos好像也能发现）  routes:  #通过时间匹配  - id: after\_route  uri: lb://nacos-provider  predicates:  - After=2021-02-15T00:00:00+08:00[Asia/Shanghai]  - id: before\_route  uri: lb://nacos-provider  predicates:  - Before=2019-02-15T00:00:00+08:00[Asia/Shanghai]  - id: between\_route  uri: lb://nacos-provider  predicates:  - Between=2018-02-14T00:00:00+08:00[Asia/Shanghai], 2019-02-16T23:59:59+08:00[Asia/Shanghai]  #通过Cookie匹配  - id: cookie\_route  uri: lb://nacos-provider  predicates:  - Cookie=name, chengxuyuanguoguo  #通过Header属性匹配  - id: header\_route  uri: lb://nacos-provider  predicates:  - Header=X-Request-Id, \d+  #通过Host匹配  - id: host\_route  uri: http://ityouknow.com  predicates:  - Host=\*\*.ityouknow.com  #通过请求方式匹配  - id: method\_route  uri: http://httpbin.org  predicates:  - Method=GET  #通过请求路径匹配  - id: path\_route  uri: http://httpbin.org  predicates:  - Path=/foo/{segment}  #通过请求参数匹配  - id: query\_route  uri: http://httpbin.org  predicates:  - Query=foo  #通过IP地址匹配  - id: remoteaddr\_route  uri: http://httpbin.org  predicates:  - RemoteAddr=172.16.1.73/24  #组合使用  - id: group\_route  uri: http://httpbin.org  predicates:  - Host=\*\*.httpbin.com  - Header=X-Request-Id, \d+  - Cookie=name, chengxuyuanguoguo  - After=2019-02-15T00:00:00+08:00[Asia/Shanghai] |

默认是从上往下路由，一旦第一个通过的断言即进行路由。

# 参考资料

[https://yq.aliyun.com/articles/703206](https://yq.aliyun.com/articles/703206?spm=5176.10695662.1996646101.searchclickresult.20783e1bW5qi46&aly_as=-seyClPG)

<https://yq.aliyun.com/articles/703208>

<https://github.com/alibaba/Sentinel/wiki/%E7%BD%91%E5%85%B3%E9%99%90%E6%B5%81>

https://github.com/alibaba/spring-cloud-alibaba/wiki/Sentinel