Spring Cloud Gateway 内置的过滤器工厂

内置的过滤器工厂

这里简单将Spring Cloud Gateway内置的所有过滤器工厂整理成了一张表格。如下：

过滤器工厂	作用	参数
AddRequestHeader	为原始请求添加Header	Header的名称及值
AddRequestParameter	为原始请求添加请求参数	参数名称及值
AddResponseHeader	为原始响应添加Header	Header的名称及值
DedupeResponseHeader	剔除响应头中重复的值	需要去重的Header名称及去重策略
Hystrix	为路由引入Hystrix的断路器保护	`HystrixCommand`的名称
FallbackHeaders	为fallbackUri的请求头中添加具体的异常信息	Header的名称
PrefixPath	为原始请求路径添加前缀	前缀路径
PreserveHostHeader	为请求添加一个preserveHostHeader=true的属性，路由过滤器会检查该属性以决定是否要发送原始的Host	无
RequestRateLimiter	用于对请求限流，限流算法为令牌桶	keyResolver、rateLimiter、statusCode、denyEmptyKey、emptyKeyStatus
RedirectTo	将原始请求重定向到指定的URL	http状态码及重定向的url
RemoveHopByHopHeadersFilter	为原始请求删除IETF组织规定的一系列Header	默认就会启用，可以通过配置指定仅删除哪些Header
RemoveRequestHeader	为原始请求删除某个Header	Header名称
RemoveResponseHeader	为原始响应删除某个Header	Header名称
RewritePath	重写原始的请求路径	原始路径正则表达式以及重写后路径的正则表达式
RewriteResponseHeader	重写原始响应中的某个Header	Header名称，值的正则表达式，重写后的值
SaveSession	在转发请求之前，强制执行`WebSession::save`操作	无
secureHeaders	为原始响应添加一系列起安全作用的响应头	无，支持修改这些安全响应头的值
SetPath	修改原始的请求路径	修改后的路径

过滤器工厂	作用	参数
SetResponseHeader	修改原始响应中某个Header的值	Header名称，修改后的值
SetStatus	修改原始响应的状态码	HTTP 状态码，可以是数字，也可以是字符串
StripPrefix	用于截断原始请求的路径	使用数字表示要截断的路径的数量
Retry	针对不同的响应进行重试	retries、statuses、methods、series
RequestSize	设置允许接收最大请求包的大小。如果请求包大小超过设置的值，则返回 `413 Payload Too Large`	请求包大小，单位为字节，默认值为5M
ModifyRequestBody	在转发请求之前修改原始请求体内容	修改后的请求体内容
ModifyResponseBody	修改原始响应体的内容	修改后的响应体内容
Default	为所有路由添加过滤器	过滤器工厂名称及值

Tips：每个过滤器工厂都对应一个实现类，并且这些类的名称必须以 `GatewayFilterFactory` 结尾，这是Spring Cloud Gateway的一个约定，例如 `AddRequestHeader` 对应的实现类为 `AddRequestHeaderGatewayFilterFactory`。

1、AddRequestHeader GatewayFilter Factory

为原始请求添加Header，配置示例：

```
spring:
  cloud:
    gateway:
      routes:
      - id: add_request_header_route
        uri: https://example.org
        filters:
        - AddRequestHeader=X-Request-Foo, Bar
```

为原始请求添加名为 `X-Request-Foo` ，值为 `Bar` 的请求头

2、AddRequestParameter GatewayFilter Factory

为原始请求添加请求参数及值，配置示例：

```
spring:
  cloud:
    gateway:
      routes:
      - id: add_request_parameter_route
        uri: https://example.org
        filters:
        - AddRequestParameter=foo, bar
```

为原始请求添加名为foo，值为bar的参数，即：`foo=bar`

3、AddResponseHeader GatewayFilter Factory

为原始响应添加Header，配置示例：

```
spring:
  cloud:
    gateway:
      routes:
      - id: add_response_header_route
        uri: https://example.org
        filters:
        - AddResponseHeader=X-Response-Foo, Bar
```

为原始响应添加名为 `X-Request-Foo` ，值为 `Bar` 的响应头

4、DedupeResponseHeader GatewayFilter Factory

DedupeResponseHeader可以根据配置的Header名称及去重策略剔除响应头中重复的值，这是Spring Cloud Greenwich SR2提供的新特性，低于这个版本无法使用。

我们在Gateway以及微服务上都设置了CORS（解决跨域）Header的话，如果不做任何配置，那么请求 -> 网关 -> 微服务，获得的CORS Header的值，就将会是这样的：

```
Access-Control-Allow-Credentials: true, true
Access-Control-Allow-Origin: https://musk.mars, https://musk.mars
```

可以看到这两个Header的值都重复了，若想把这两个Header的值去重的话，就需要使用到DedupeResponseHeader，配置示例：

```
spring:
  cloud:
    gateway:
      routes:
      - id: dedupe_response_header_route
        uri: https://example.org
        filters:
        # 若需要去重的Header有多个，使用空格分隔
        - DedupeResponseHeader=Access-Control-Allow-Credentials Access-Control-
Allow-Origin
```

去重策略：

- RETAIN_FIRST：默认值，保留第一个值
- RETAIN_LAST：保留最后一个值
- RETAIN_UNIQUE：保留所有唯一值，以它们第一次出现的顺序保留

若想对该过滤器工厂有个比较全面的了解的话，建议阅读该过滤器工厂的源码，因为源码里有详细的注释及示例，比官方文档写得还好：
`org.springframework.cloud.gateway.filter.factory.DedupeResponseHeaderGatewayFilterFactory`

5、Hystrix GatewayFilter Factory

为路由引入Hystrix的断路器保护，配置示例：

```
spring:
  cloud:
    gateway:
      routes:
      - id: hystrix_route
        uri: https://example.org
        filters:
        - Hystrix=myCommandName
```

Hystrix是Spring Cloud第一代容错组件，不过已经进入维护模式，未来Hystrix会被Spring Cloud移除掉，取而代之的是Alibaba Sentinel/Resilience4J。所以本文不做详细介绍了，感兴趣的话可以参考官方文档：

- Hystrix GatewayFilter Factory

6、FallbackHeaders GatewayFilter Factory

同样是对Hystrix的支持，上一小节所介绍的过滤器工厂支持一个配置参数：`fallbackUri`，该配置用于当发生异常时将请求转发到一个特定的uri上。而`FallbackHeaders`这个过滤工厂可以在转发请求到该uri时添加一个Header，这个Header的值为具体的异常信息。配置示例：

```
spring:
  cloud:
    gateway:
      routes:
      - id: ingredients
        uri: lb://ingredients
        predicates:
        - Path=//ingredients/**
        filters:
        - name: Hystrix
          args:
            name: fetchIngredients
            fallbackUri: forward:/fallback
      - id: ingredients-fallback
        uri: http://localhost:9994
        predicates:
        - Path=/fallback
        filters:
        - name: FallbackHeaders
          args:
```

```
            executionExceptionTypeHeaderName: Test-Header
```

这里也不做详细介绍了，感兴趣可以参考官方文档：

- FallbackHeaders GatewayFilter Factory

7、PrefixPath GatewayFilter Factory

为原始的请求路径添加一个前缀路径，配置示例：

```
spring:
  cloud:
    gateway:
      routes:
      - id: prefixpath_route
        uri: https://example.org
        filters:
        - PrefixPath=/mypath
```

该配置使访问 `${GATEWAY_URL}/hello` 会转发到 `https://example.org/mypath/hello`

8、PreserveHostHeader GatewayFilter Factory

为请求添加一个preserveHostHeader=true的属性，路由过滤器会检查该属性以决定是否要发送原始的Host Header。配置示例：

```
spring:
  cloud:
    gateway:
      routes:
      - id: preserve_host_route
        uri: https://example.org
        filters:
        - PreserveHostHeader
```

如果不设置，那么名为 `Host` 的Header将由Http Client控制

9、RequestRateLimiter GatewayFilter Factory

用于对请求进行限流，限流算法为令牌桶。配置示例：

```

spring:
  cloud:
    gateway:
      routes:
      - id: requestratelimiter_route
        uri: https://example.org
        filters:
        - name: RequestRateLimiter
          args:
            redis-rate-limiter.replenishRate: 10
            redis-rate-limiter.burstCapacity: 20
```

由于另一篇文章中已经介绍过如何使用该过滤器工厂实现网关限流，所以这里就不再赘述了：

- Spring Cloud Gateway - 扩展

或者参考官方文档：

- RequestRateLimiter GatewayFilter Factory

10、RedirectTo GatewayFilter Factory

将原始请求重定向到指定的Url，配置示例：

```
spring:
  cloud:
    gateway:
      routes:
      - id: redirect_route
        uri: https://example.org
        filters:
        - RedirectTo=302, https://acme.org
```

该配置使访问 `${GATEWAY_URL}/hello` 会被重定向到 `https://acme.org/hello` ，并且携带一个 `Location:http://acme.org` 的Header，而返回客户端的HTTP状态码为302

注意事项：

- HTTP状态码应为3xx，例如301
- URL必须是合法的URL，该URL会作为 `Location` Header的值

11、RemoveHopByHopHeadersFilter GatewayFilter Factory

为原始请求删除IETF组织规定的一系列Header，默认删除的Header如下：

- Connection
- Keep-Alive
- Proxy-Authenticate
- Proxy-Authorization
- TE
- Trailer
- Transfer-Encoding
- Upgrade

可以通过配置去指定仅删除哪些Header，配置示例：

```
spring:
  cloud:
    gateway:
      filter:
        remove-hop-by-hop:
          # 多个Header使用逗号（,）分隔
          headers: Connection,Keep-Alive
```

12、RemoveRequestHeader GatewayFilter Factory

为原始请求删除某个Header，配置示例：

```yaml
spring:
  cloud:
    gateway:
      routes:
      - id: removerequestheader_route
        uri: https://example.org
        filters:
        - RemoveRequestHeader=X-Request-Foo
```

删除原始请求中名为 `X-Request-Foo` 的请求头

13、RemoveResponseHeader GatewayFilter Factory

为原始响应删除某个Header，配置示例：

```yaml
spring:
  cloud:
    gateway:
      routes:
      - id: removeresponseheader_route
        uri: https://example.org
        filters:
        - RemoveResponseHeader=X-Response-Foo
```

删除原始响应中名为 `X-Request-Foo` 的响应头

14、RewritePath GatewayFilter Factory

通过正则表达式重写原始的请求路径，配置示例：

```yaml
spring:
  cloud:
    gateway:
      routes:
      - id: rewritepath_route
        uri: https://example.org
        predicates:
        - Path=/foo/**
        filters:
        # 参数1为原始路径的正则表达式，参数2为重写后路径的正则表达式
        - RewritePath=/foo/(?<segment>.*), /$\{segment}
```

该配置使得访问 `/foo/bar` 时，会将路径重写为 `/bar` 再进行转发，也就是会转发到 `https://example.org/bar`。需要注意的是：由于YAML语法，需用 `$\` 替换 `$`

15、RewriteResponseHeader GatewayFilter Factory

重写原始响应中的某个Header，配置示例：

```yaml
spring:
  cloud:
    gateway:
      routes:
      - id: rewriteresponseheader_route
        uri: https://example.org
        filters:
        # 参数1为Header名称，参数2为值的正则表达式，参数3为重写后的值
        - RewriteResponseHeader=X-Response-Foo, password=[^&]+, password=***
```

该配置的意义在于：如果响应头中`X-Response-Foo`的值为`/42?user=ford&password=omg!what&flag=true`，那么就会被按照配置的值重写成`/42?user=ford&password=***&flag=true`，也就是把其中的`password=omg!what`重写成了`password=***`

16、SaveSession GatewayFilter Factory

在转发请求之前，强制执行`WebSession::save`操作，配置示例：

```yaml
spring:
  cloud:
    gateway:
      routes:
      - id: save_session
        uri: https://example.org
        predicates:
        - Path=/foo/**
        filters:
        - SaveSession
```

主要用在那种像 Spring Session 延迟数据存储（数据不是立刻持久化）的，并希望在请求转发前确保session状态保存情况。如果你将Spring Secutiry于Spring Session集成使用，并想确保安全信息都传到下游机器，你就需要配置这个filter。

17、secureHeaders GatewayFilter Factory

secureHeaders过滤器工厂主要是参考了这篇博客中的建议，为原始响应添加了一系列起安全作用的响应头。默认会添加如下Headers（包括值）：

- `X-Xss-Protection:1; mode=block`
- `Strict-Transport-Security:max-age=631138519`
- `X-Frame-Options:DENY`
- `X-Content-Type-Options:nosniff`
- `Referrer-Policy:no-referrer`
- `Content-Security-Policy:default-src 'self' https:; font-src 'self' https: data:; img-src 'self' https: data:; object-src 'none'; script-src https:; style-src 'self' https: 'unsafe-inline'`
- `X-Download-Options:noopen`
- `X-Permitted-Cross-Domain-Policies:none`

如果你想修改这些Header的值，那么就需要使用这些Headers对应的后缀，如下：

- `xss-protection-header`

- `strict-transport-security`
- `frame-options`
- `content-type-options`
- `referrer-policy`
- `content-security-policy`
- `download-options`
- `permitted-cross-domain-policies`

配置示例：

```
spring:
  cloud:
    gateway:
      filter:
        secure-headers:
          # 修改 X-Xss-Protection 的值为 2; mode=unblock
          xss-protection-header: 2; mode=unblock
```

如果想禁用某些Header，可使用如下配置：

```
spring:
  cloud:
    gateway:
      filter:
        secure-headers:
          # 多个使用逗号（,）分隔
          disable: frame-options,download-options
```

18、SetPath GatewayFilter Factory

修改原始的请求路径，配置示例：

```
spring:
  cloud:
    gateway:
      routes:
      - id: setpath_route
        uri: https://example.org
        predicates:
        - Path=/foo/{segment}
        filters:
        - SetPath=/{segment}
```

该配置使访问 `${GATEWAY_URL}/foo/bar` 时会转发到 `https://example.org/bar` ，也就是原本的 `/foo/bar` 被修改为了 `/bar`

19、SetResponseHeader GatewayFilter Factory

修改原始响应中某个Header的值，配置示例：

```yaml
spring:
  cloud:
    gateway:
      routes:
      - id: setresponseheader_route
        uri: https://example.org
        filters:
        - SetResponseHeader=X-Response-Foo, Bar
```

将原始响应中 `X-Response-Foo` 的值修改为 `Bar`

20、SetStatus GatewayFilter Factory

修改原始响应的状态码，配置示例：

```yaml
spring:
  cloud:
    gateway:
      routes:
      - id: setstatusstring_route
        uri: https://example.org
        filters:
        # 字符串形式
        - SetStatus=BAD_REQUEST
      - id: setstatusint_route
        uri: https://example.org
        filters:
        # 数字形式
        - SetStatus=401
```

SetStatusd的值可以是数字，也可以是字符串。但一定要是Spring `HttpStatus` 枚举类中的值。上面这两种配置都可以返回401这个HTTP状态码。

21、StripPrefix GatewayFilter Factory

用于截断原始请求的路径，配置示例：

```yaml
spring:
  cloud:
    gateway:
      routes:
      - id: nameRoot
        uri: http://nameservice
        predicates:
        - Path=/name/**
        filters:
        # 数字表示要截断的路径的数量
        - StripPrefix=2
```

如上配置，如果请求的路径为 `/name/bar/foo` ，那么则会截断成 `/foo` 后进行转发，也就是会截断2个路径。

22、Retry GatewayFilter Factory

针对不同的响应进行重试，例如可以针对HTTP状态码进行重试，配置示例：

```yaml
spring:
  cloud:
    gateway:
      routes:
      - id: retry_test
        uri: http://localhost:8080/flakey
        predicates:
        - Host=*.retry.com
        filters:
        - name: Retry
          args:
            retries: 3
            statuses: BAD_GATEWAY
```

可配置如下参数：

- `retries`：重试次数
- `statuses`：需要重试的状态码，取值在 `org.springframework.http.HttpStatus` 中
- `methods`：需要重试的请求方法，取值在 `org.springframework.http.HttpMethod` 中
- `series`：HTTP状态码序列，取值在 `org.springframework.http.HttpStatus.Series` 中

23、RequestSize GatewayFilter Factory

设置允许接收最大请求包的大小，配置示例：

```yaml
spring:
  cloud:
    gateway:
      routes:
      - id: request_size_route
      uri: http://localhost:8080/upload
      predicates:
      - Path=/upload
      filters:
      - name: RequestSize
        args:
          # 单位为字节
          maxSize: 5000000
```

如果请求包大小超过设置的值，则会返回 `413 Payload Too Large` 以及一个 `errorMessage`

24、Modify Request Body GatewayFilter Factory

在转发请求之前修改原始请求体内容，该过滤器工厂只能通过代码配置，不支持在配置文件中配置。代码示例：

```
@Bean
public RouteLocator routes(RouteLocatorBuilder builder) {
    return builder.routes()
        .route("rewrite_request_obj", r -> r.host("*.rewriterequestobj.org")
            .filters(f -> f.prefixPath("/httpbin")
```

```
                .modifyRequestBody(String.class, Hello.class,
MediaType.APPLICATION_JSON_VALUE,
                    (exchange, s) -> return Mono.just(new
Hello(s.toUpperCase()))))).uri(uri))
        .build();
}

static class Hello {
    String message;

    public Hello() { }

    public Hello(String message) {
        this.message = message;
    }

    public String getMessage() {
        return message;
    }

    public void setMessage(String message) {
        this.message = message;
    }
}
```

> Tips：该过滤器工厂处于 **BETA** 状态，未来API可能会变化，生产环境请慎用

25、Modify Response Body GatewayFilter Factory

可用于修改原始响应体的内容，该过滤器工厂同样只能通过代码配置，不支持在配置文件中配置。代码示例：

```
@Bean
public RouteLocator routes(RouteLocatorBuilder builder) {
    return builder.routes()
        .route("rewrite_response_upper", r ->
r.host("*.rewriteresponseupper.org")
            .filters(f -> f.prefixPath("/httpbin")
                .modifyResponseBody(String.class, String.class,
                    (exchange, s) -> Mono.just(s.toUpperCase()))).uri(uri)
        .build();
}
```

> Tips：该过滤器工厂处于 **BETA** 状态，未来API可能会变化，生产环境请慎用

26、Default Filters

Default Filters用于为所有路由添加过滤器工厂，也就是说通过Default Filter所配置的过滤器工厂会作用到所有的路由上。配置示例：

```yaml
spring:
  cloud:
    gateway:
      default-filters:
      - AddResponseHeader=X-Response-Default-Foo, Default-Bar
      - PrefixPath=/httpbin
```

官方文档：

- GatewayFilter Factories