Spring Cloud内置的Zuul过滤器详解

 Spring Cloud  Spring Cloud  2017/02/14

Spring Cloud默认为Zuul编写并启用了一些过滤器，这些过滤器有什么作用呢？我们不妨按照@EnableZuulServer、@EnableZuulProxy两个注解进行展开，相信大家对这两个注解都不陌生（**至少都见过吧**）。如果觉得陌生也没有关系，可将@EnableZuulProxy简单理解为@EnableZuulServer的增强版。事实上，当Zuul与Eureka、Ribbon等组件配合使用时，@EnableZuulProxy是我们常用的注解。

在Spring Cloud的官方文档中，只说@EnableZuulServer是一个“空白”的Zuul，那么究竟空白在什么地方呢？与@EnableZuulProxy到底有什么区别呢？诸多问题，都将在本文找到答案。

在此之前，我们先理解什么是RequestContext：

RequestContext：用于在过滤器之间传递消息。它的数据保存在每个请求的ThreadLocal中。它用于存储请求路由到哪里、错误、HttpServletRequest、HttpServletResponse都存储在RequestContext中。RequestContext扩展了ConcurrentHashMap，所以，任何数据都可以存储在上下文中。

@EnableZuulServer过滤器

**一、pre类型过滤器**

(1) ServletDetectionFilter：该过滤器用于检查请求是否通过Spring Dispatcher。检查后，通过isDispatcherServletRequest设置布尔值。

(2) FormBodyWrapperFilter：解析表单数据，并为请求重新编码。

(3) DebugFilter：顾名思义，调试用的过滤器，可以通过zuul.debug.request=true ，或在请求时，加上debug=true的参数，例如$ZUUL\_HOST:ZUUL\_PORT/path?debug=true 开启该过滤器。这样，该过滤器就会把RequestContext.setDebugRouting() 、RequestContext.setDebugRequest() 设为true。

**二、route类型过滤器**

SendForwardFilter：该过滤器使用Servlet RequestDispatcher转发请求，转发位置存储在RequestContext.getCurrentContext().get("forward.to") 中。可以将路由设置成：

|  |
| --- |
| zuul:  routes:  abc:  path: /abc/\*\*  url: forward:/abc |

然后访问$ZUUL\_HOST:ZUUL\_PORT/abc ，观察该过滤器的执行过程。

**三、post类型过滤器**

SendResponseFilter：将Zuul所代理的微服务的的响应写入当前响应。

**四、error类型过滤器**

SendErrorFilter：如果RequestContext.getThrowable() 不为null，那么默认就会转发到/error，也可以设置error.path属性修改默认的转发路径。

@EnableZuulProxy过滤器

如果使用注解@EnableZuulProxy，那么除上述过滤器之外，Spring Cloud还会安装以下过滤器：

**一、pre类型过滤器**

PreDecorationFilter：该过滤器根据提供的RouteLocator确定路由到的地址，以及怎样去路由。该路由器也可为后端请求设置各种代理相关的header。

**二、route类型过滤器**

(1) RibbonRoutingFilter：该过滤器使用Ribbon，Hystrix和可插拔的HTTP客户端发送请求。serviceId在RequestContext.getCurrentContext().get("serviceId") 中。该过滤器可使用不同的HTTP客户端，例如

* Apache HttpClient：默认的HTTP客户端
* Squareup OkHttpClient v3：如需使用该客户端，需保证com.squareup.okhttp3的依赖在classpath中，并设置ribbon.okhttp.enabled = true 。
* Netflix Ribbon HTTP client：设置ribbon.restclient.enabled = true 即可启用该HTTP客户端。需要注意的是，该客户端有一定限制，例如不支持PATCH方法，另外，它有内置的重试机制。

(2) SimpleHostRoutingFilter：该过滤器通过Apache HttpClient向指定的URL发送请求。URL在RequestContext.getRouteHost() 中。

**本文链接**：[Spring Cloud内置的Zuul过滤器详解](http://www.itmuch.com/spring-cloud/zuul/zuul-filter-in-spring-cloud/)

**转载声明**：本博客由周立创作，采用 [CC BY 3.0 CN](https://creativecommons.org/licenses/by-nc-nd/3.0/)许可协议。可自由转载、引用，但需署名作者且注明文章出处。如转载至微信公众号，请在文末添加作者公众号二维码。