阅读本文，您将了解：

* Zuul过滤器类型与请求生命周期
* 如何编写Zuul过滤器
* 如何禁用Zuul过滤器
* Spring Cloud为Zuul编写的过滤器及其功能。

过滤器是Zuul的核心组件，本节我们来详细讨论Zuul的过滤器。

## 过滤器类型与请求生命周期

Zuul大部分功能都是通过过滤器来实现的。Zuul中定义了四种标准过滤器类型，这些过滤器类型对应于请求的典型生命周期。

(1) 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。

(2) ROUTING：这种过滤器将请求路由到微服务。这种过滤器用于构建发送给微服务的请求，并使用Apache HttpClient或Netfilx Ribbon请求微服务。

(3) 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。

(4) ERROR：在其他阶段发生错误时执行该过滤器。

除了默认的过滤器类型，Zuul还允许我们创建自定义的过滤器类型。例如，我们可以定制一种STATIC类型的过滤器，直接在Zuul中生成响应，而不将请求转发到后端的微服务。

Zuul请求的生命周期如图8-5所示，该图详细描述了各种类型的过滤器的执行顺序。

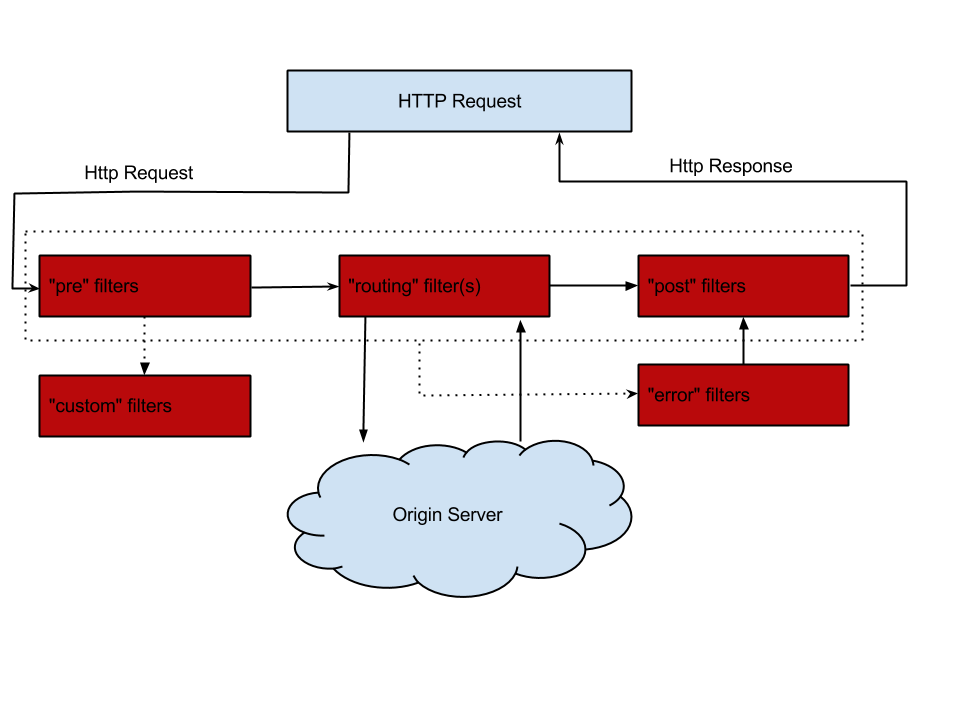


图8-5 Zuul请求的生命周期

## 编写Zuul过滤器

理解过滤器类型和请求生命周期后，我们来编写一个Zuul过滤器。编写Zuul的过滤器非常简单，我们只需继承抽象类ZuulFilter，然后实现几个抽象方法就可以了。

那么现在，我们来编写一个简单的Zuul过滤器，让该过滤器打印请求日志。

(1) 复制项目microservice-gateway-zuul，将ArtifactId修改为microservice-gateway-zuul-filter。

(2) 编写自定义Zuul过滤器

|  |
| --- |
| public class PreRequestLogFilter extends ZuulFilter {  private static final Logger LOGGER = LoggerFactory.getLogger(PreRequestLogFilter.class);  @Override  public String filterType() {  return "pre";  }  @Override  public int filterOrder() {  return 1;  }  @Override  public boolean shouldFilter() {  return true;  }  @Override  public Object run() {  RequestContext ctx = RequestContext.getCurrentContext();  HttpServletRequest request = ctx.getRequest();  PreRequestLogFilter.LOGGER.info(String.format("send %s request to %s", request.getMethod(), request.getRequestURL().toString()));  return null;  }  } |

由代码可知，自定义的Zuul Filter需实现以下几个方法：

* filterType：返回过滤器的类型。有pre、route、post、error等几种取值，分别对应上文的几种过滤器。详细可以参考com.netflix.zuul.ZuulFilter.filterType() 中的注释。
* filterOrder：返回一个int值来指定过滤器的执行顺序，不同的过滤器允许返回相同的数字。
* shouldFilter：返回一个boolean值来判断该过滤器是否要执行，true表示执行，false表示不执行。
* run：过滤器的具体逻辑。本例中，我们让它打印了请求的HTTP方法以及请求的地址。

(2) 修改启动类，为启动类添加以下内容：

|  |
| --- |
| @Bean  public PreRequestLogFilter preRequestLogFilter() {  return new PreRequestLogFilter();  } |

**测试**

(1) 启动microservice-discovery-eureka。

(2) 启动microservice-provider-user。

(3) 启动microservice-gateway-zuul-filter。

(4) 访问<http://localhost:8040/microservice-provider-user/1> ，可获得类似如下的日志。

|  |
| --- |
| [nio-8040-exec-6] c.i.c.s.filters.pre.PreRequestLogFilter : send GET request to http:*//localhost:8040//microservice-provider-user/1* |

说明我们编写的自定义Zuul过滤器被执行了。

## 禁用Zuul过滤器

Spring Cloud默认为Zuul编写并启用了一些过滤器，例如DebugFilter、FormBodyWrapperFilter、PreDecorationFilter等。这些过滤器都存放在spring-cloud-netflix-core这个Jar包的org.springframework.cloud.netflix.zuul.filters包中。

一些场景下，我们想要禁用掉部分过滤器，此时该怎么办呢？

答案非常简单，只需设置zuul.<SimpleClassName>.<filterType>.disable=true ，即可禁用SimpleClassName所对应的过滤器。以过滤器org.springframework.cloud.netflix.zuul.filters.post.SendResponseFilter为例，只需设置zuul.SendResponseFilter.post.disable=true ，即可禁用该过滤器。

同理，如果想要禁用《编写Zuul过滤器》一节编写的过滤器，只需设置zuul.PreRequestLogFilter.pre.disable=true 即可。

**TPS**

(1) 相关代码com.netflix.zuul.ZuulFilter.disablePropertyName()、com.netflix.zuul.ZuulFilter.isFilterDisabled()、com.netflix.zuul.ZuulFilter.runFilter()。

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

**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。