Date : 01-Jun-21
Spring Boot 9AM
Mr. RAGHU

---

*) Filters:-
Filter is a plugable component
(if we add/remove code, actual logic remains same),
used to execute PRE-PORCESSING logic over 'request' object
and POST-PORCESSING logic over 'response' object.

https://docs.oracle.com/javaee/6/api/javax/servlet/Filter.html
Examples that have been identified for this design are:
> Authentication Filters
> Logging and Auditing Filters
> Image conversion Filters
> Data compression Filters
> Encryption Filters
> Tokenizing Filters
> Filters that trigger resource access events
> XSL/T filters
> Mime-type chain Filter

==========
Zuul Filters
==========

=> Zuul Filters are used for ' Logging and Auditing ',
'ContentType/Mime-Type checking'.

Here, we have 4 types of filters
a. Request Filter
b. Routing Filter
c. Error Filter
d. Response Filter.

*) Error Filter is executed only in case of Routing Error,
ie unable to find MS# instance, Eureka not responding..etc

*) Remaing all filters are executed for every request.
*) These are not javax.servlet Filters. These are used for only at Zuul
level to track request and response details along with success/fail
information about routing.

*) To implement one ZuulFilter we need 4 details
a. Enable/Disable Filter
b. Filter logic
c. Filter Type (pre/route/error/post)
d. Filter Order

*) To provide filter type we are going to use FilterConstants(C).

```
class FilterConstants {
        public static final String ERROR_TYPE = "error";
        public static final String POST_TYPE = "post";
        public static final String PRE_TYPE = "pre";
        public static final String ROUTE_TYPE = "route";
```

```
}
------API Details--------------------------------------------------
com.netflix.zuul
+ IZuulFilter (I)

+ shouldFilter()
+ run()

-------------------
com.netflix.zuul
+ ZuulFilter

+ filterType()
+ filterOrder()
-------------------------------------------------------------
Netflix Zuul (not by Spring Cloud/Java Sun/Oracle) has provided Zuul
Filters.
To implement these filters
-> define one class
-> extends ZuulFilter
-> override 4 methods
-> add @Component

--Example---
package in.nareshit.raghu.filter;

import
org.springframework.cloud.netflix.zuul.filters.support.FilterConstants
;
import org.springframework.stereotype.Component;

import com.netflix.zuul.ZuulFilter;
import com.netflix.zuul.exception.ZuulException;

@Component
public class MyFilter extends ZuulFilter {

        public boolean shouldFilter() {
                return true;
        }

        public Object run() throws ZuulException {
                //logic..
                return null;
        }

        public String filterType() {
                //return "pre";
                return FilterConstants.PRE_TYPE;
        }

        public int filterOrder() {
                return 0;
        }
}

=> These are auto-executable. If we remov this code also, it will not
```

make any effect to application.

*) To get Request, Response and error details for Current request
use 'RequestContext' object created by Netflix Zuul.

Ex:
RequestContext context = RequestContext.getCurrentContext();

=> we can use methods :
context.getRequest():

Exception Handler in Spring Boot:
https://www.youtube.com/c/NareshIT/search?query=exception%20raghu

==code==========
1. pre-filter

```
package in.nareshit.raghu.filter;

import javax.servlet.http.HttpServletRequest;

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import
org.springframework.cloud.netflix.zuul.filters.support.FilterConstants
;
import org.springframework.stereotype.Component;

import com.netflix.zuul.ZuulFilter;
import com.netflix.zuul.context.RequestContext;
import com.netflix.zuul.exception.ZuulException;

@Component
public class MyRequestFilter extends ZuulFilter {

	private static final Logger LOG =
LoggerFactory.getLogger(MyRequestFilter.class);

	public boolean shouldFilter() {
		return true;
	}

	public Object run() throws ZuulException {
		RequestContext context =
RequestContext.getCurrentContext();

		HttpServletRequest request = context.getRequest();
		LOG.info("URL => " + request.getRequestURL());
		LOG.info("HEADER => " + request.getHeaderNames());
		LOG.info("METHOD => " + request.getMethod());

		return null;
		//return context;
	}

	public String filterType() {
		//return "pre";
		return FilterConstants.PRE_TYPE;
```

```
	}

	public int filterOrder() {
		return 0;
	}
}
```

2. Route Filter

```
package in.nareshit.raghu.filter;

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import
org.springframework.cloud.netflix.zuul.filters.support.FilterConstants
;
import org.springframework.stereotype.Component;

import com.netflix.zuul.ZuulFilter;
import com.netflix.zuul.exception.ZuulException;

@Component
public class MyRoutingFilter extends ZuulFilter {

	private static final Logger LOG =
LoggerFactory.getLogger(MyRoutingFilter.class);

	public boolean shouldFilter() {
		return true;
	}

	public Object run() throws ZuulException {
		LOG.info("FROM ROUTING....");
		return null;
	}

	public String filterType() {
		return FilterConstants.ROUTE_TYPE;
	}

	public int filterOrder() {
		return 0;
	}

}
```

3. Error Filter

```
package in.nareshit.raghu.filter;

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.stereotype.Component;
import org.springframework.util.ReflectionUtils;

import com.netflix.zuul.ZuulFilter;
import com.netflix.zuul.context.RequestContext;
import com.netflix.zuul.exception.ZuulException;
```

```
@Component
public class MyErrorFilter extends ZuulFilter {

	private static final Logger LOG =
LoggerFactory.getLogger(MyErrorFilter.class);

	public boolean shouldFilter() {
		return true;
	}

	public Object run() {
		try {
			RequestContext ctx =
RequestContext.getCurrentContext();
			Object e = ctx.getThrowable();

			if (e != null && e instanceof ZuulException) {
				ZuulException zuulException =
(ZuulException)e;
				LOG.error("Zuul failure detected: " +

zuulException.getMessage(), zuulException);

				ctx.remove("throwable");

				ctx.setResponseBody("{ \"code\": 500,
\"problem\": \"notworking\"}");

ctx.getResponse().setContentType("application/json");
				ctx.setResponseStatusCode(500);
			}
		}
		catch (Exception ex) {
			LOG.error("Exception filtering in custom error
filter", ex);
			ReflectionUtils.rethrowRuntimeException(ex);
		}

		return null;
	}

	public String filterType() {
		return "error";
	}

	public int filterOrder() {
		return -1;
	}
}
```

4. Post Filter

```
package in.nareshit.raghu.filter;

import java.io.IOException;
import java.io.InputStream;
import java.io.InputStreamReader;
```

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import
org.springframework.cloud.netflix.zuul.filters.support.FilterConstants
;
import org.springframework.stereotype.Component;

import com.google.common.io.CharStreams;
import com.netflix.zuul.ZuulFilter;
import com.netflix.zuul.context.RequestContext;
import com.netflix.zuul.exception.ZuulException;

@Component
public class MyResponseFilter extends ZuulFilter {

        private static final Logger LOG =
LoggerFactory.getLogger(MyResponseFilter.class);

        public boolean shouldFilter() {
                return true;
        }

        public Object run() throws ZuulException {
                LOG.info("FROM RESPONSE FILTER");
                RequestContext ctx =
RequestContext.getCurrentContext();
                try (final InputStream responseDataStream =
ctx.getResponseDataStream()) {
                        String responseData = CharStreams.toString(new
InputStreamReader(responseDataStream, "UTF-8"));
                        responseData = responseData + "MODIFIED!";
                        ctx.setResponseBody(responseData);
                } catch (IOException e) {
                        LOG.error("Error reading body",e);
                }
                return null;
        }

        public String filterType() {
                return FilterConstants.POST_TYPE;
        }

        public int filterOrder() {
                return 0;
        }
}
```