Date : 02-Jun-21
Spring Boot 9AM
Mr. RAGHU

Zuul Filters:-
=> Filters executed at API Gateway only.
=> These are not regular filters (javax.servlet Filter not this one)
=> Types of Filters (4)
 a. PRE-FILTER
 b. ROUTE-FILTER
 c. ERROR-FILTER
 d. POST-FILTER

*) PRE-FILTER : Is used to track/find all details like Path,URL, Header params,
Body Type...etc

*) ROUTE-FILTER is executed over Zuul Routing process, ie Goto Eureka, get Instance by using ServiceId and call MS#.

?? ROUTE-FILTER is used to modify route details (or) change route path.
Ex: If Current Request is having Authrization header then continue to same
routing, else modify Routing Service As LoginService.

*) ERROR-FILTER : Zuul API Gateway has provided a pre-defined error filter.
name: SendErrorFilter
package: [org.springframework.cloud.netflix.zuul.filters.post]

SendErrorFilter : Will dispatch request to BasicErrorController which process
Error and creates one WhiteLabelErrorPage.

=> Create one Custom Error Filter with order is -ve number. Bcoz order#0 is
set to SendErrorFilter.

##Error and Excption handling in MVC and REST##
https://www.youtube.com/watch?v=AFq9eK2OoGU
https://www.youtube.com/watch?v=tBVAybXMKzY
https://www.youtube.com/watch?v=M-LRfrYHWrk

---

d. POST-FILTER : Once, response is given by MS#, then this filter is executed.

> It is used to modify reponse like Encode/Decode, Add Custom Header Param,
Append Final Message to Response Body..etc

*) Zuul also one type of MS#, it must registered with Eureka,

then only it can fetch register and able to call other MS#

Q) What is Zuul Proxy ? Why it is required?
A) For Every MS# registered in Eureka,
one Client code is generated internally using
Ribbon LoadBalancer Client.
These generated classes supports for routing
ie goto MS# and execute them.
Such classes are called as Dynamic Zuul Proxy.
For that add: @EnableZuulProxy

Q) What is Zuul Service Register? Why it is?
A) If we provide routing details like path and serviceId,
then Zuul Routing/Service Register is created.
Once Client made request then based on this Register one ServiceId
choosen same MS# is executed.

==========Zuul Application end to end coding==========

1. Create Eureka Server Project
2. Create MS# Project

3. Create Zuul Project
a. create Project
Name : SpringCloudZuulProject
Dep : Zuul, Web, Eureka Discovery Client

b. At Starter class:

```
@EnableZuulProxy
@EnableEurekaClient
```

c. application.properties

```
server.port=80
spring.application.name=ZUUL-PROXY
eureka.client.service-url.defaultZone=http://localhost:8761/eureka

# Provide all MS# Common(API) Path ---ServiceId
zuul.routes.vendor.path=/vendor-api/**
zuul.routes.vendor.service-id=VENDOR-SERVICE

#zuul.routes.prod.path=/product-api/**
#zuul.routes.prod.service-id=PRODUCT-SERVICE
```

---

*) 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  (Abstrct class)

+ 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

Q) Why and when should we provide order for ZuulFilter?
A) We can define multiple filters of same type.
   [Modularity -- Lengthy code into multiple same filters]
   in that case order is required, else any one dummy number is fine.

   MyRequsestFilterOne   --- order # 0
   MyRequsestFilterTwo   --- order # 1
   MyRequsestFilterThree --- order # 2

Q) How can we read data from InputStream into String object?
A) use InputStreamReader
   Convert into CharStream
   Convert into String format

===Zull Filtes 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("PRE-FILTER DATA {},{},{}",
                                request.getRequestURL(),
                                request.getHeaderNames(),
                                request.getContextPath());
                return null;
        }

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

        public int filterOrder() {
                return 0;
        }

}
```

---

2. Route 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 org.springframework.util.StringUtils;

import com.netflix.zuul.ZuulFilter;
import com.netflix.zuul.context.RequestContext;
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 {
                RequestContext context =
RequestContext.getCurrentContext();
                HttpServletRequest request = context.getRequest();
                String auth = request.getHeader("Authorization");
                if(!StringUtils.hasText(auth)) {
                        LOG.warn("SEND TO LOGIN SERVICE");
                        //RequestDispatcher dispatcher =
```

```
request.getRequestDispatcher(loginPath);
                } else {
                        LOG.info("Nice! Authorization Found!!");
                }

                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.cloud.netflix.zuul.filters.support.FilterConstants
;
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 FilterConstants.ERROR_TYPE;
        }

        public int filterOrder() {
                return -1;
        }

}
-------------
4. Response 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()) {
                        if(responseDataStream!=null) {
                                String responseData =
CharStreams.toString(new InputStreamReader(responseDataStream, "UTF-
8"));

if(!responseData.contains("notworking")) {
                                        responseData = responseData +
" , DATA IS 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;
        }

}
```