Spring cloud

Zuul API gateway:

* Zuul api gateway will work as edge service(An **edge service** is a component which is exposed to the public internet. It acts as a gateway to all other **services**).
* Zuul can be used as dynamic routing, monitoring, resiliency and security.
* Zuul core is the main component for which it will execute and compile for the filters.
* [Zuul](https://github.com/Netflix/zuul) is a JVM-based router and server-side load balancer from Netflix
* Zuul contains 4 standard filters which are pre filter, route filter, post filter, error filter
* Pre filter: this filter run before the request is routed
* Route filter: this filter can handle the actual routing of the request
* Post filter: this filter run after the request has been routed
* Error filter: this filter run if an error occurs in the course of handling the request.

Example:

public class SimpleFilter extends ZuulFilter {

private static Logger log = LoggerFactory.getLogger(SimpleFilter.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();

log.info(String.format("%s request to %s", request.getMethod(), request.getRequestURL().toString()));

return null;

}

}

* filterType(): Returns a String that stands for the type of the filter — in this case, ”PRE”.
* filterOrder(): Gives the order in which this filter is to be run, relative to other filters.
* shouldFilter(): Contains the logic that determines when to run this filter (this particular filter is always run).
* Run(): Contains the functionality of the filter.
* Zuul filters store request and state information in (and share it by means of) the RequestContext. You can use that to get at the HttpServletRequest and then log the HTTP method and URL of the request before it is sent on its way.

zuul.routes.books.url=http://localhost:8090

ribbon.eureka.enabled=false

server.port=8080

Spring Cloud Zuul automatically sets the path to the application name. In this sample, set zuul.routes.books.url so that Zuul will proxy requests to /books to this URL.

Notice the second property in the application.properties file, Spring Cloud Netflix Zuul uses Netflix’s Ribbon to perform client-side load balancing. By default, Ribbon would use Netflix Eureka for service discovery. For this simple example, you can skip service discovery, so set ribbon.eureka.enabled to false. Since Ribbon now cannot use Eureka to look up services, we must specify a url for the book service.

Eureka:

All Micro service will 1st register in the Eureka server and this Eureka server knows all the service applications running on each port and name of that application. Eureka Server is also known as Discovery Server.

* By default Eureka server will run at port :8761
* The @EnableEurekaServer annotation is used to make your Spring Boot application acts as a Eureka Server
* By default Eureka Server will get registers itself into the discovery.
* eureka.client.registerWithEureka = false
* eureka.client.fetchRegistry = false
* server.port = 8761
* @EnableEurekaClient: this will make the service as client and it will register into the eureka server.
* eureka.client.serviceUrl.defaultZone = http://localhost:8761/eureka
* spring.application.name = eurekaclient