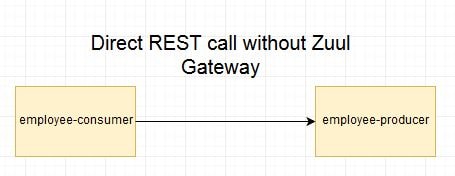
What is the Netflix Zuul? Need for it?

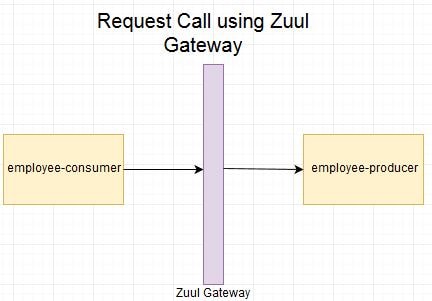
Zuul is a JVM based router and server side load balancer by Netflix.  
It provides a single entry to our system, which allows a browser, mobile app, or other user interface to consume services from multiple hosts without managing cross-origin resource sharing (CORS) and authentication for each one.

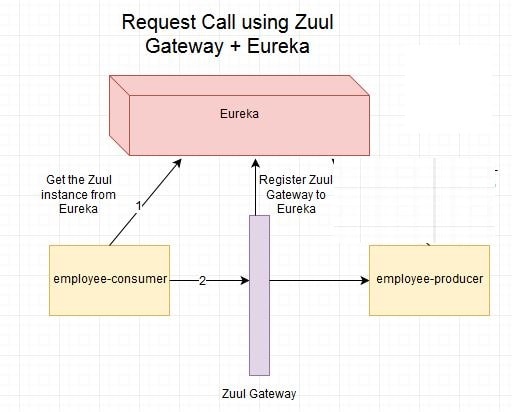
We can integrate Zuul with other Netflix projects like Hystrix for fault tolerance and Eureka for service discovery, or use it to manage routing rules, filters, and load balancing across your system.

* Microservice call without Netflix Zuul



* Microservice call with Netflix Zuul



* Microservice call with Netflix Zuul + Netflix Eureka
* 

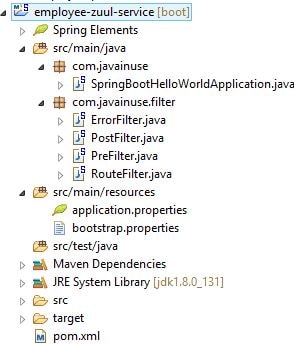
Lets Begin-

We will be creating four modules as shown in above diagram.

* employee consumer
* employee producer
* Eureka Server
* employee-zuul-service

.

* Zuul Gateway

The project structure for this module will be as follows-  
  


The pom.xml will be as follows with the **zuul dependency**.

<?xml version="1.0" encoding="UTF-8"?>

<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">

<modelVersion>4.0.0</modelVersion>

<groupId>com.javainuse</groupId>

<artifactId>employee-zuul-service</artifactId>

<version>0.0.1-SNAPSHOT</version>

<packaging>jar</packaging>

<name>SpringBootHelloWorld</name>

<description>Demo project for Spring Boot</description>

<parent>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-parent</artifactId>

<version>1.4.1.RELEASE</version>

<relativePath /> <!-- lookup parent from repository -->

</parent>

<properties>

<project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>

<project.reporting.outputEncoding>UTF-8</project.reporting.outputEncoding>

<java.version>1.8</java.version>

</properties>

<dependencies>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-web</artifactId>

</dependency>

**<dependency>**

**<groupId>org.springframework.cloud</groupId>**

**<artifactId>spring-cloud-starter-zuul</artifactId>**

**</dependency>**

<dependency>

<groupId>org.springframework.cloud</groupId>

<artifactId>spring-cloud-starter-eureka</artifactId>

</dependency>

</dependencies>

<dependencyManagement>

<dependencies>

<dependency>

<groupId>org.springframework.cloud</groupId>

<artifactId>spring-cloud-dependencies</artifactId>

<version>Camden.SR6</version>

<type>pom</type>

<scope>import</scope>

</dependency>

</dependencies>

</dependencyManagement>

<build>

<plugins>

<plugin>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-maven-plugin</artifactId>

</plugin>

</plugins>

</build>

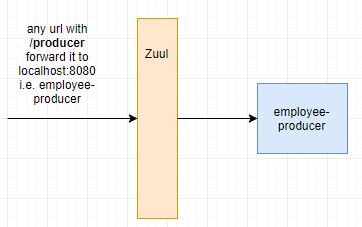
</project>

Next define the following properties in application.properties-

zuul.routes.producer.url=http://localhost:8080

eureka.client.serviceUrl.defaultZone=http://localhost:8090/eureka

server.port=8079

  
Here zuul.routes.producer.url will route incoming traffic to request for /producer to the employee-producer microservice.

Similar routes can be added for other microservices as well.  
  
Next name the application module in the bootstrap.properties file

spring.application.name=employee-zuul-service

Next we define the 4 types of filters supported by Zuul-

* **pre**
* **post**
* **route**
* **error**

Define the ErrorFilter as follows-

package com.javainuse.filter;

import com.netflix.zuul.ZuulFilter;

public class ErrorFilter extends ZuulFilter {

@Override

public String filterType() {

return "error";

}

@Override

public int filterOrder() {

return 0;

}

@Override

public boolean shouldFilter() {

return true;

}

@Override

public Object run() {

System.out.println("Using Route Filter");

return null;

}

}

Define the PostFilter as follows-

package com.javainuse.filter;

import com.netflix.zuul.ZuulFilter;

public class PostFilter extends ZuulFilter {

@Override

public String filterType() {

return "post";

}

@Override

public int filterOrder() {

return 0;

}

@Override

public boolean shouldFilter() {

return true;

}

@Override

public Object run() {

System.out.println("Using Post Filter");

return null;

}

}

Define the PreFilter as follows-

package com.javainuse.filter;

import javax.servlet.http.HttpServletRequest;

import com.netflix.zuul.ZuulFilter;

import com.netflix.zuul.context.RequestContext;

public class PreFilter extends ZuulFilter {

@Override

public String filterType() {

return "pre";

}

@Override

public int filterOrder() {

return 0;

}

@Override

public boolean shouldFilter() {

return true;

}

@Override

public Object run() {

RequestContext ctx = RequestContext.getCurrentContext();

HttpServletRequest request = ctx.getRequest();

System.out.println(

"Request Method : " + request.getMethod() + " Request URL : " + request.getRequestURL().toString());

return null;

}

}

Define the RouteFilter as follows-

package com.javainuse.filter;

import com.netflix.zuul.ZuulFilter;

public class RouteFilter extends ZuulFilter {

@Override

public String filterType() {

return "route";

}

@Override

public int filterOrder() {

return 0;

}

@Override

public boolean shouldFilter() {

return true; }

@Override

public Object run() {

System.out.println("Using Route Filter");

return null; }

}

Finally we annotate the Spring Boot Main class with **@EnableZuulProxy**.With this the module will act as a service proxy or gateway.  
Also we create the beans for the filters defined above.

package com.javainuse;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

import org.springframework.cloud.client.discovery.EnableDiscoveryClient;

import org.springframework.cloud.netflix.zuul.EnableZuulProxy;

import org.springframework.context.annotation.Bean;

import com.javainuse.filter.ErrorFilter;

import com.javainuse.filter.PostFilter;

import com.javainuse.filter.PreFilter;

import com.javainuse.filter.RouteFilter;

@SpringBootApplication

@EnableDiscoveryClient

**@EnableZuulProxy**

public class SpringBootHelloWorldApplication {

public static void main(String[] args) {

SpringApplication.run(SpringBootHelloWorldApplication.class, args);

}

@Bean

public PreFilter preFilter() {

return new PreFilter();

}

@Bean

public PostFilter postFilter() {

return new PostFilter();

}

@Bean

public ErrorFilter errorFilter() {

return new ErrorFilter();

}

@Bean

public RouteFilter routeFilter() {

return new RouteFilter();

}

}

 Code changes for employee-consumer

The changes we make for the consumer module are

* We fetch the Zuul Service instance instead of the the Employee Producer service we were doing earlier.  
  So in code we have **discoveryClient.getInstances("EMPLOYEE-ZUUL-SERVICE")** instead of **discoveryClient.getInstances("EMPLOYEE-PRODUCER")**
* Append the URL to be hit with /producer since we have defined so in the applicatio.properties above.  
  **baseUrl = baseUrl + "/producer/employee"**

package com.javainuse.controllers;

import java.io.IOException;

import java.util.List;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.cloud.client.ServiceInstance;

import org.springframework.cloud.client.discovery.DiscoveryClient;

import org.springframework.http.HttpEntity;

import org.springframework.http.HttpHeaders;

import org.springframework.http.HttpMethod;

import org.springframework.http.MediaType;

import org.springframework.http.ResponseEntity;

import org.springframework.stereotype.Controller;

import org.springframework.web.client.RestClientException;

import org.springframework.web.client.RestTemplate;

@Controller

public class ConsumerControllerClient {

@Autowired

private DiscoveryClient discoveryClient;

public void getEmployee() throws RestClientException, IOException {

List<ServiceInstance> instances = discoveryClient.getInstances("EMPLOYEE-ZUUL-SERVICE");

ServiceInstance serviceInstance = instances.get(0);

String baseUrl = serviceInstance.getUri().toString();

baseUrl = baseUrl + "/producer/employee";

RestTemplate restTemplate = new RestTemplate();

ResponseEntity<String> response = null;

try {

response = restTemplate.exchange(baseUrl, HttpMethod.GET, getHeaders(), String.class);

} catch (Exception ex) {

System.out.println(ex);

}

System.out.println(response.getBody());

}

private static HttpEntity<?> getHeaders() throws IOException {

HttpHeaders headers = new HttpHeaders();

headers.set("Accept", MediaType.APPLICATION\_JSON\_VALUE);

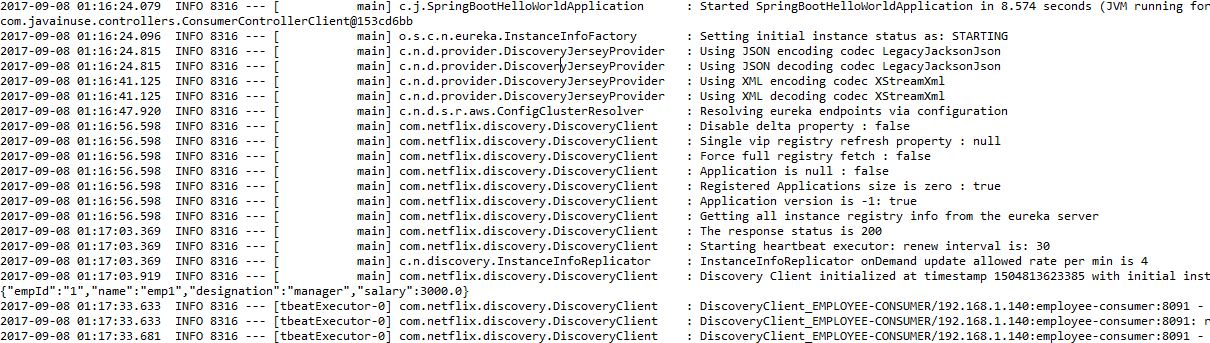
return new HttpEntity<>(headers);

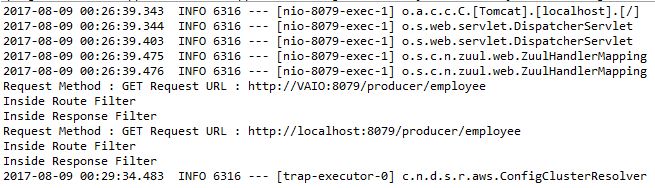
}

}

As we had done in previous posts- Start the following Spring Boot Applications-

* eureka-server
* employee-producer
* employee-zuul-service
* employee-consumer

On running the employee-consumer we get the output as follows-  


The Zuul console output is as follows-  
  
So the filters defined in the zuul gateway get executed.