一　zuul 简介

网关组件,实现整个微服务集群对外访问的唯一入口.

二　zuul 功能

1　路由

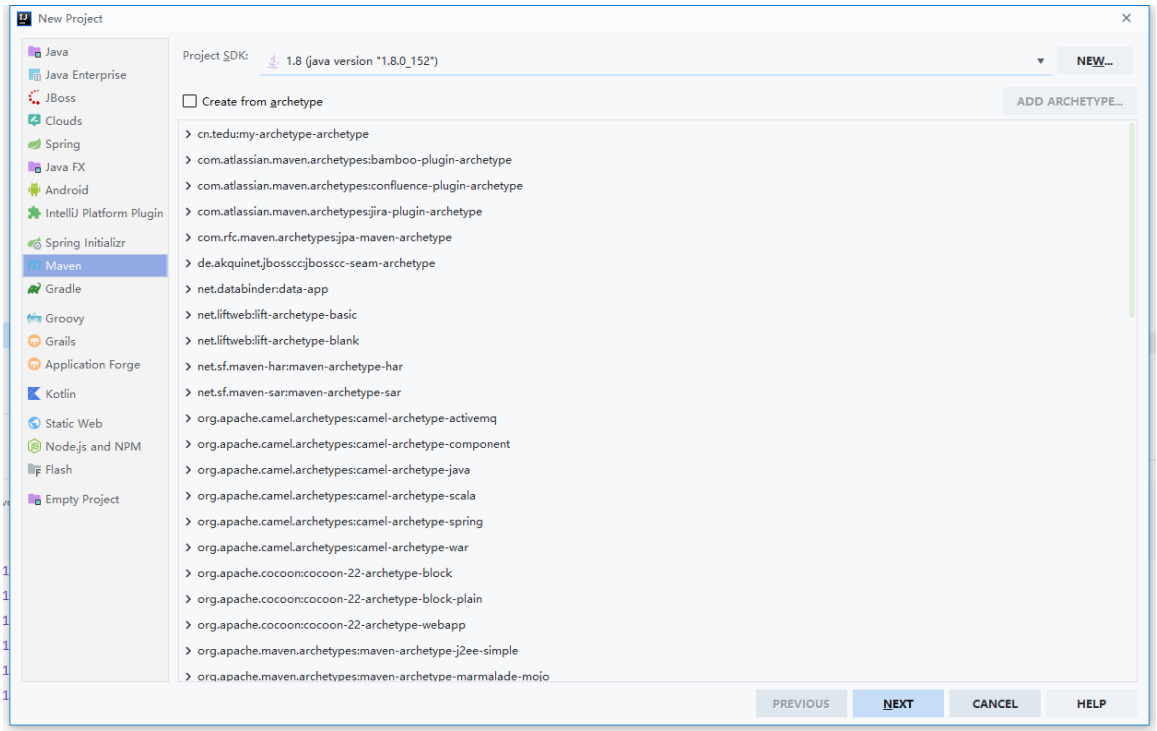
- 根据请求地址不同,网关工程会计算路由然后调用后端的不同微服务

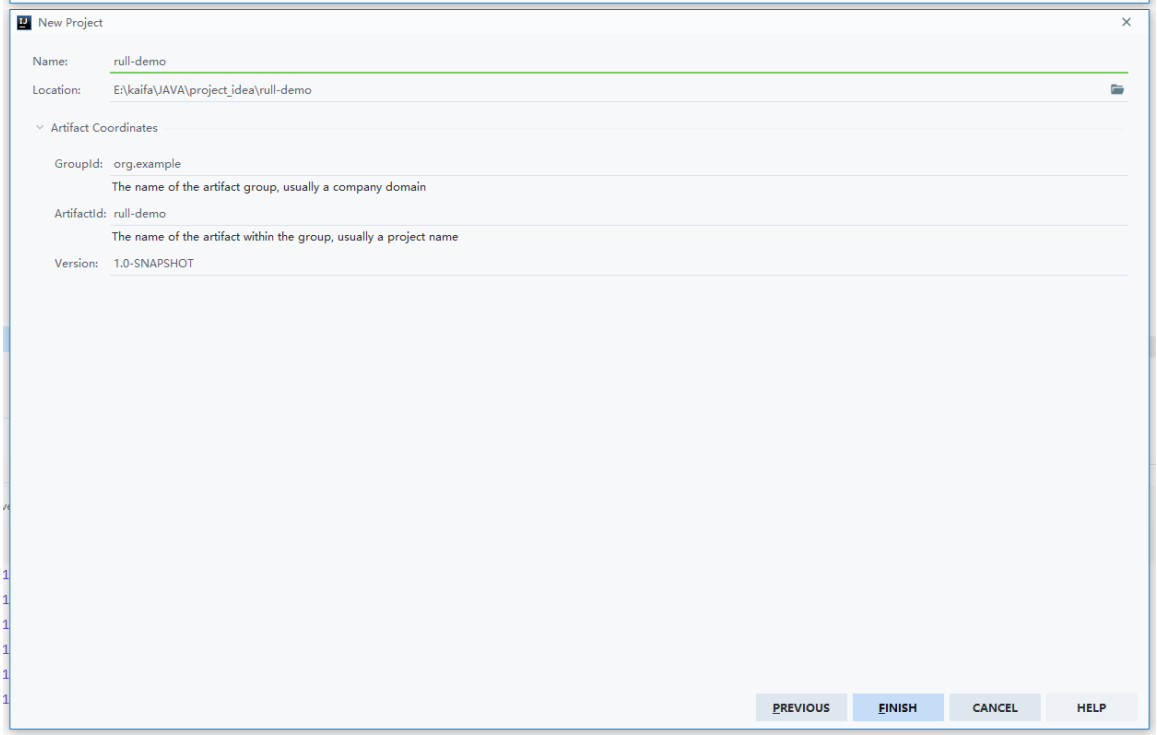
2　过滤（拦截）

- 网关中实现请求的鉴权工作(符不符合参数要求,是否携带身份信息,是否合法)

三　网关工程的创建

1　创建项目





2 pom.xml

2.A 继承（spring-boot-parent）

```
<!--继承springboot-->
<parent>
    <artifactId>spring-boot-starter-parent</artifactId>
    <groupId>org.springframework.boot</groupId>
    <version>1.5.9.RELEASE</version>
</parent>
```

2.B 依赖

- eureka-client（通过网关路由到微服务，就必须要通过客户端从注册中心获取微服务信息）
- zuul 依赖（内部包括了 ribbon）

```
<dependencies>
    <dependency>
        <groupId>org.springframework.cloud</groupId>
        <artifactId>spring-cloud-starter-eureka</artifactId>
    </dependency>
    <dependency>
        <groupId>org.springframework.cloud</groupId>
        <artifactId>spring-cloud-starter-zuul</artifactId>
    </dependency>
</dependencies>
```

3 application.yml

- 端口:8103
- 服务名称:
- ip-prefer
- 注册中心地址
- 路由规则

#zuul 网关的路由配置

zuul.routes.api-a.path=/zuul-a/**

zuul.routes.api-a.service-id=service-hi

```
server:
  port: 8763
  #给功能起一个
务
称
spring:
  application:
```

```
    name: gate-way
eureka:
  instance:
    #ip优先相互通信
  prefer-ip-address: true

    #访问的注册中心接口 会使用name到注册中心实现
```

```
务注册
 client:
    service-url:
      defaultZone: http://127.0.0.1:8761/eureka
#网关路由
zuul:
  routes:
    api-a: #自定义的名称
   path: /zuul-user/** #外部访问路径匹配
   service-id: ouu-user #匹配后找对应的服务
  api-b: #自定义的名称
   path: /zuul-order/** #外部访问路径匹配
   service-id: ouu-order #匹配后找对应的服务
```

4 创建启动类

```
@SpringBootApplication
@EnableEurekaClient
@EnableZuulProxy//开启zuul代理 目的就是引入zuul的过滤逻辑
//封装了ribbon /restTemplate
public class StarterGateway {
    public static void main(String[] args) {
        SpringApplication.run(StarterGateway.class,args);
    }

}
```

四 网关路由配置详解

```
#网关路由
zuul:
  routes:
    api-a: #自定义的名称
   path: /zuul-user/** #外部访问路径匹配
   service-id: ouu-user #匹配后找对应的服务
  api-b: #自定义的名称
   path: /zuul-order/** #外部访问路径匹配
   service-id: ouu-order #匹配后找对应的服务
```

1 zuul.routes

固定的路由配置前缀

2 api-a/api-b:

自定义的路由名称，一般这里配置的路由名称和调用的功能有关,一对路由配置这个名字保持一致

3 path

ant 匹配规范,匹配的是访问到网关的请求 uri 地址是否满足这里的规范
/zuul-a/**表示只要请求到网关的 uri 地址是以/zuul-a/开始的就满足匹配
例如/zuul-a/haha,/zuul-a/a/b,/zuul-a/.其中 ANT 匹配规范如下

?	表示匹配一级(/为 1 级)单个字符，例如：path=/zuul-a/?,可以匹配到/zuul-a/a,/zuul-a/b,不能匹配/zuul-a/abc,/zuul-a/a/b/c
*	表示匹配一级任意字符串，例如：path=/zuul-a/*,可以匹配到/zuul-a/a,/zuul-a/abc,不能匹配/zuul-a/a/b/c
**	表示匹配任意多级任意字符串，例如：path=/zuul-a/**,可以匹配到/zuul-a/a,/zuul-a/abc,/zuul-a/a/b/c.没有不能匹配的

4 serviceId（微服务名称）

路由匹配在 zuul 中一对一对出现,path 匹配请求地址,一旦匹配上将会调用 serviceId 的微服务

5 优先级问题

如果功能非常复杂,多个微服务配置路由时 path 有包含关系,相当于一个请求到网关的功能匹配到多个 path(优先级)

例如:

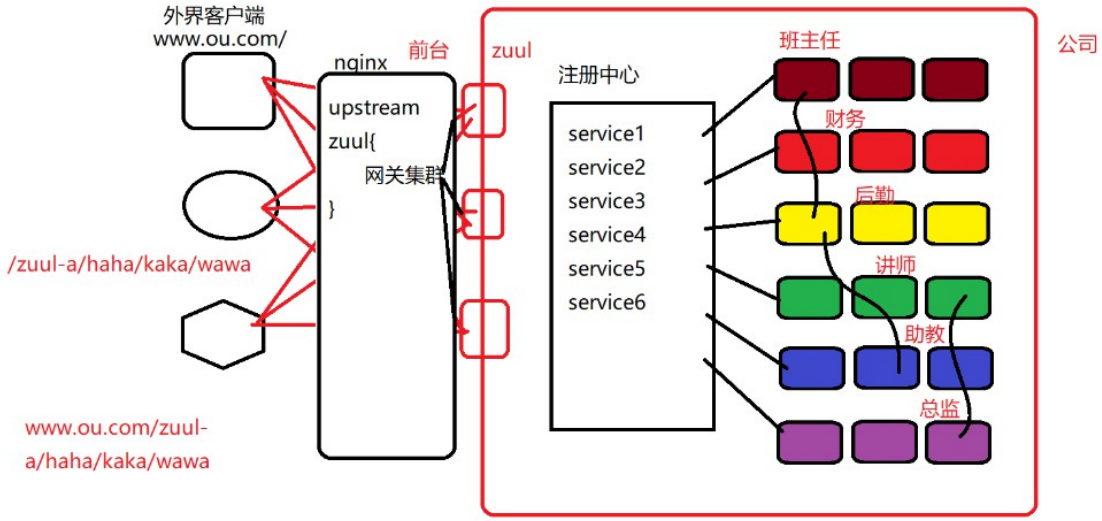
```
zuul.routes.api-a.path=/zuul/*
zuul.routes.api-a.service-id=service1

zuul.routes.api-b.path=/zuul/**
zuul.routes.api-b.service-id=service2
```

请求网关:
localhost:8103/zuul/abc 到底在网关当中调用是 service1 service2
有可能是 service1 调用,也有可能是 service2(properties 格式)

zuul 中对应包含关系的匹配规则 path 具有优先级的判断,先加载到内存的路由优先级高于后加载的.properties 的格式文件,先后加载和手写顺序未必一致(hashMap).可以通过 yml 格式编写,定义优先级,yml 格式就是按照先写先加载的逻辑,读取属性.
但是实际场景中,不太可能使用到有包含关系的 path.

五　网关集群和结构



最终经过 eureka 治理组件和 ribbon 组件形成微服务集群,并且实现服务调用服务的功能

通过网关组件实现所有的微服务功能,必须经过网关才能访问

引入 nginx 实现外界访问入口网关的集群负载均衡

这种结构才是 springcloud 最基本的完整的结构

六　Js 请求到微服务整体流程

1　js 请求起始地址：

http://www.ou.com/user/query/point

|经过 hosts 文件 ip 地址映射找到 127.0.0.1

http://127.0.0.1/user/query/point

2 nginx 接收请求

|经过监听 80 端口 监听 www.ou.com 满足要求，进入 server 的 location

|location =/user/query/point 匹配到这个 location 剩余 uri 字符串/

|proxy_pass http://127.0.0.1:8103/zuul-tuser/user/query/point/

|从 nginx 发送出去的请求 zuul 网关

3 进入 zuul 网关

|http://127.0.0.1:8103/zuul-tuser/user/query/point/

|path 匹配路径 /zuul-tuser/** 剩余 url 地址 /user/query/point/

|path 匹配成功找到路由规则 api-a 对应找到了服务名称 ouu-user

|http:// ouu-user/user/query/point

zuul 网关内部 ribbon 拦截生效将 ouu-userr 找到对应实例(负载均衡)详细信息 ip:port 访问后端微服务

从 zuul 网关出去:

http://127.0.0.1:9001/user/query/point

4 URL 变化

进入 nginx 请求

http://www.ou.com/user/query/point

进入 zuul 网关请求

http://127.0.0.1:8103/zuul-tuser/user/query/point

进入服务提供者请求

http://127.0.0.1:9001/user/query/point

七　zuul 的过滤功能

网关作为微服务的集群唯一入口,非法的请求经过过滤,没有身份的请求拦截.

过滤：判断力度较小

拦截：力度较大

应用场景：鉴权作用.

本身网关的所有功能就是一套过滤机制.自定义过滤器 class 可以直接添加到网关系统

1　编写实现类并重写相关方法

下例中判断请求的 url 是否以/zuul-a/开始，若是则判断是否有请求参数 name，有则放行，否则自定义消息返回

使得过滤器的类生效,必须让他成为容器的一个 bean

①添加类的注解@Component

②或配置类的方法上@Bean 生成

```
/**
 * @作者 舒新胜
 * @项目 easymall-2002-all
 * @创建时间 2020/6/12 11:32
 */
@Component
public class MyZuulFilter extends ZuulFilter {
    /**
     * 过滤器类型
     * @return
     */
    @Override
    public String filterType() {
        return null;
    }

    /**
     *多个过滤器顺序
     * @return
     */
    @Override
    public int filterOrder() {
        return 0;
    }

    /**
     * 是否启动过滤
     * @return
     */
    @Override
    public boolean shouldFilter() {
        //获取 zuul 上下文
        RequestContext context=RequestContext.getCurrentContext();
        //请求对象
        HttpServletRequest request = context.getRequest();
        //除去域名的上下文请求地址
        String requestURI = request.getRequestURI();
        //是否以/zuul-a/开始
        boolean b = requestURI.startsWith("/zuul-a/");
        return b;
    }
```

```
    @Override
    public Object run() {
        //过滤逻辑,判断 name 是否为参数
        //拿到当前请求 request,拦截,获取 respone
        //获取上下文
        RequestContext context = RequestContext.getCurrentContext();
        //请求对象
        HttpServletRequest request = context.getRequest();
        //响应对象
        HttpServletResponse response = context.getResponse();
        //从 request 拿到参数
        String name = request.getParameter("name");
        if(name==null){
            //请求中没有任何形式的 name 的参数
            //进行拦截,防止后续调用逻辑

            //请求是否执行后续逻辑
            context.setSendZuulResponse(false);
            //设置响应码
            context.setResponseStatusCode(403);
            //设置响应编码
            response.setContentType("text/html;charset=utf-8");
            //手动响应返回数据
            context.setResponseBody
                    ("{\"status\":\"201\",\"msg\":\"参数数据格式不正确\"}");
        }
        return null;
    }
}
```

2 过滤器重写的四个方法介绍

2.A String filterType

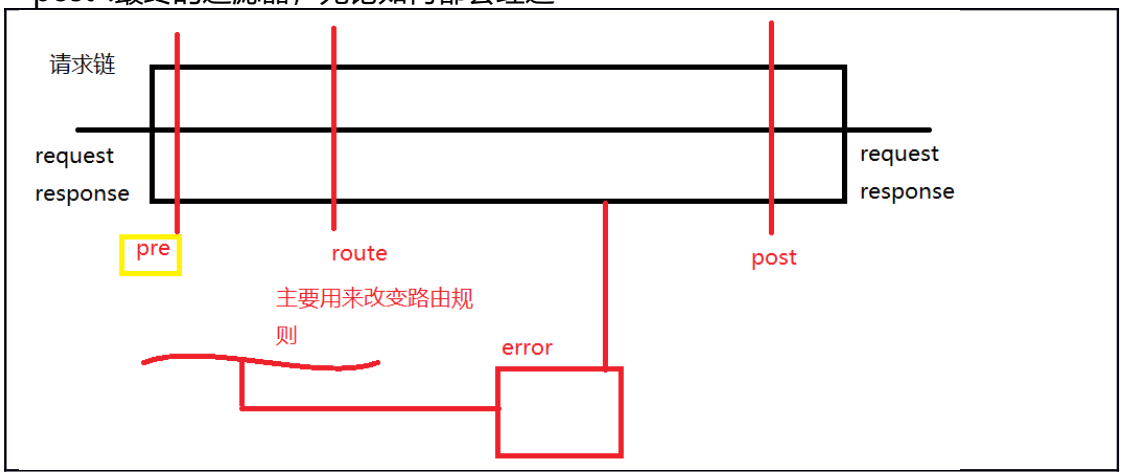
返回值表示过滤器的类型,类型不同过滤器使用的位置不同

"pre":进入 zuul 网关时加载的过滤器

"route":计算路由时记载的过滤器

"error":在上述 2 个过滤器中任何一个出现问题异常时，会进入这个过滤器

"post":最终的过滤器，无论如何都会经过



2.B int filterOrder()

如果自定义了多个同一种类型的过滤器,可以通过这里的返回值定义执行顺序，返回值可正可负,值越小，顺序执行越靠前

2.C boolean shouldFilter()

主要是判断当前请求该不该进入到过滤逻辑 run()方法.例如:
鉴权工作,并不是所有的请求都要执行鉴权的
返回 false:表示不需要过滤逻辑执行
返回 true:表示需要过滤逻辑执行,进入到 run 方法

2.D Object run()

过滤的核心逻辑对应的方法,返回值没有任何效果,一般都是在 run 里处理 request 和 response

八 微服务框架下的错误排查

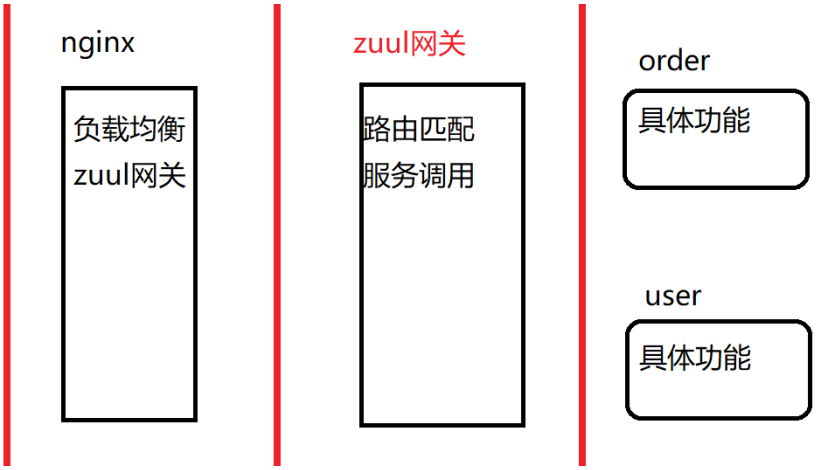
1 问题定位

在微服务框架下启动的程序功能如果出现访问错误，应该从不同层面先去排查各自的问题。目前结构中由 3 个层面

第一层：微服务提供者，单独访问出现问题，不用在看网关 nginx
http://127.0.0.1:9001/user/query/point?userId=1
保证这里返回的数据，执行的功能是正确的，如果不正确检查代码项目

第二层：添加网关后，通过路由匹配，服务调用访问后端微服务提供者
http://127.0.0.1:8103/zuul-tuser/user/query/point?userId=1
保证单独访问网关能够正常调用后端微服务，如果出错，一定路由匹配错误，测试请求地址和路由匹配不配套（检查 path，检查 serviceId 值，检查访问网关是否匹配 path）

第三层：nginx 做负载均衡访问网关，单独测试 nginx 请求，就是 js 请求地址
http://www.ou.com/user/query/point?userId
保证返回值也是正确的，如果出现错误，检查 nginx. conf(proxy_pass 是否正确访问网关的路由匹配规则)，hosts



2 核心能力

- 了解所有功能的正确流转过程
  - 起始地址 js 发起的请求 F12 看浏览器开发者模式 network
  - 请求进入 nginx 如何发送出
  - 进入 zuul 网关如调用哪个微服务哪个功能

3 常见的问题

3.A Forwarding error/ Load balancer does not have available server for client: test-use

zuul 网关负载均衡无法找到一个微服务名称叫做 test-use

原因：当进入 zuul 网关经过路由匹配确实成功了，但是无法找对应调用的微服务，zuul 启动时从 eureka 没有抓取这个服务

- application.properties 微服务 serviceId 拼错了
- zuul 网关在微服务之前启动（等一段时间就好了）

3.B read time out 访问超时

原因：启动所有工程后，相互间的联系通过 eureka 还没有建立稳定

等一段时间/重启一下网关进程

- 测试时访问不同层此的 url 本身就是错的

3.C com.sun.jersey.api.client.ClientHandlerException: java.net.ConnectException: Connection refused: connect

3.D com.netflix.discovery.shared.transport.TransportException: Cannot execute request on any known server

微服务提供者启动时,作为 eureka-client 携带自己的信息,到注册中心注册,每隔 30 秒心跳,每隔 30 秒中抓取注册中心服务
访问注册中心地址,同时一个属性指定
eureka.client.service-url.defaultZone=http://127.0.0.1:8761/eureka

微服务提供者报上述异常:

- 注册中心没启动
- 微服务提供者配置属性地址错了

注册中心报上述异常

- 高可用注册中心相互注册,必定至少报一次这个异常--正常的
- 注册中心把自己当成客户端在自己注册,必定会报异常至少一次---正常