zuul网关

学习目标

1.什么是网关

2.网关能做什么,为什么要使用网关

3.网关有哪些

4.网关的执行流程与原理

5.网关的使用

6.使用网关可能会遇到的问题

7.自定义网关实现登录的鉴别

8.网关的配置与使用

1.什么是网关

1.1.1 网关

可以理解为服务的入口，通常情况下集中管理服务的 ip地址，可以进行域名访问，负载均衡等一些列操作

1.1.2 网关的定义

网关是系统唯一对外的入口，介于客户端和服务器之间的中间层，处理非业务功能，提供路由请求，鉴权，监控，缓存，限流等功能． 它将1对N 的问题转换成了 1对1 的问题，通过服务路由的功能，可以在对外提供服务的时候，只暴露网关中配置的调用地址，而调用方就不需要了解后端具体的微服务主机

2.网关能做什么，为什么要使用网关

2.1.1网关能做什么

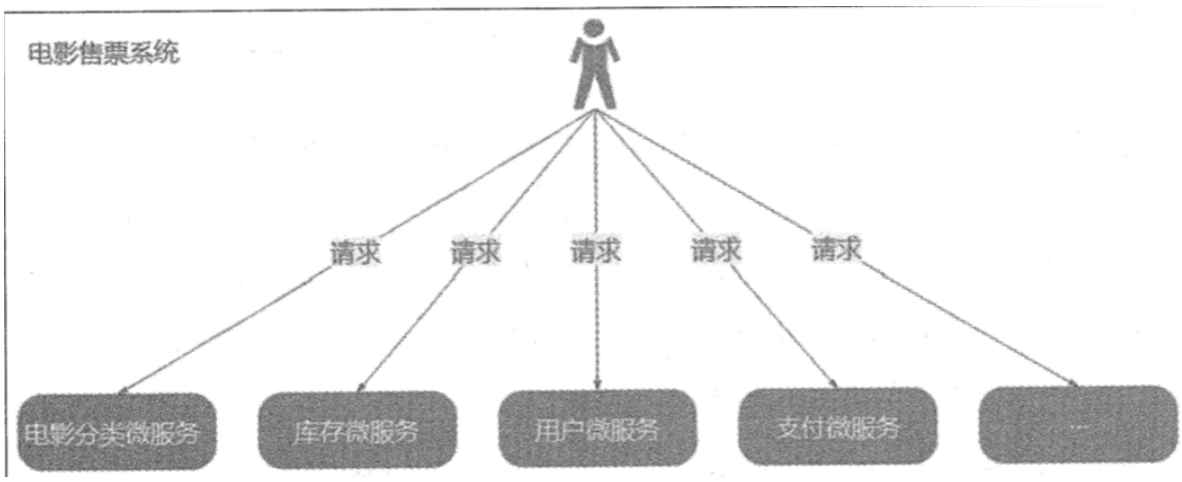
不同的微服务一般会有不同的网络地址，而客户端可能需要调用多个服务接口才能完成一个业务需求，若让客户端直接与各个微服务通信可能会有以下的问题：

（1）客户端会多次请求不同微服务，增加了客户端复杂性

（2）存在跨域请求，处理相对复杂

（3）认证复杂，每个服务都需要独立认证

（4）难以重构，多个服务可能将会合并成一个或拆分成多个



2.1.2 网关的优点

微服务网关介于服务端与客户端的中间层，所有外部服务请求都会先经过微服务网关客户只能跟微服务网关进行交互，无需调用特定微服务接口，使得开发得到简化

可以处理每一个微服务中公共的业务逻辑，针对微服务做负载均衡

可以为访问的集群服务做负载均衡，不过底层也是通过Ribbon

服务网关 = 路由转发 + 过滤器

（1）路由转发：接收一切外界请求，转发到后端的微服务上去。

（2）过滤器：在服务网关中可以完成一系列的横切功能，例如权限校验、限流以及监控等，这些都可以通过过滤器完成（其实路由转发也是通过过滤器实现的）。

2.1.3 详细补充

我们知道我们要进入一个服务本身，很明显我们没有特别好的办法，直接输入IP地址+端口号，我们知道这样的做法很糟糕的，这样的做法大有问题，首先暴露了我们实体机器的IP地址，别人一看你的IP地址就知道服务部署在哪里，让别人很方便的进行攻击操作。

第二，我们这么多服务，我们是不是要挨个调用它呀，我们这里假设做了个权限认证，我们每一个客户访问的都是跑在不同机器上的不同的JVM上的服务程序，我们每一个服务都需要一个服务认证，这样做烦不烦呀，明显是很烦的。

那么我们这时候面临着这两个极其重要的问题，这时我们就需要一个办法解决它们。首先，我们看IP地址的暴露和IP地址写死后带来的单点问题，我是不是对这么服务本身我也要动态的维护它服务的列表呀，我需要调用这服务本身，是不是也要一个负载均衡一样的玩意，

还有关于IP地址暴露的玩意，我是不是需要做一个代理呀，像Nginx的反向代理一样的东西，还有这玩意上部署公共的模块，比如所有入口的权限校验的东西。因此我们现在需要Zuul API网关。它就解决了上面的问题，你想调用某个服务，它会给你映射，把你服务的IP地址映射成

某个路径，你输入该路径，它匹配到了，它就去替你访问这个服务，它会有个请求转发的过程，像Nginx一样，服务机器的具体实例，它不会直接去访问IP，它会去Eureka注册中心拿到服务的实例ID，即服务的名字。我再次使用客户端的负载均衡ribbon访问其中服务实例中的一台。

3.网关有哪些

zuul

kong

nginx + lua

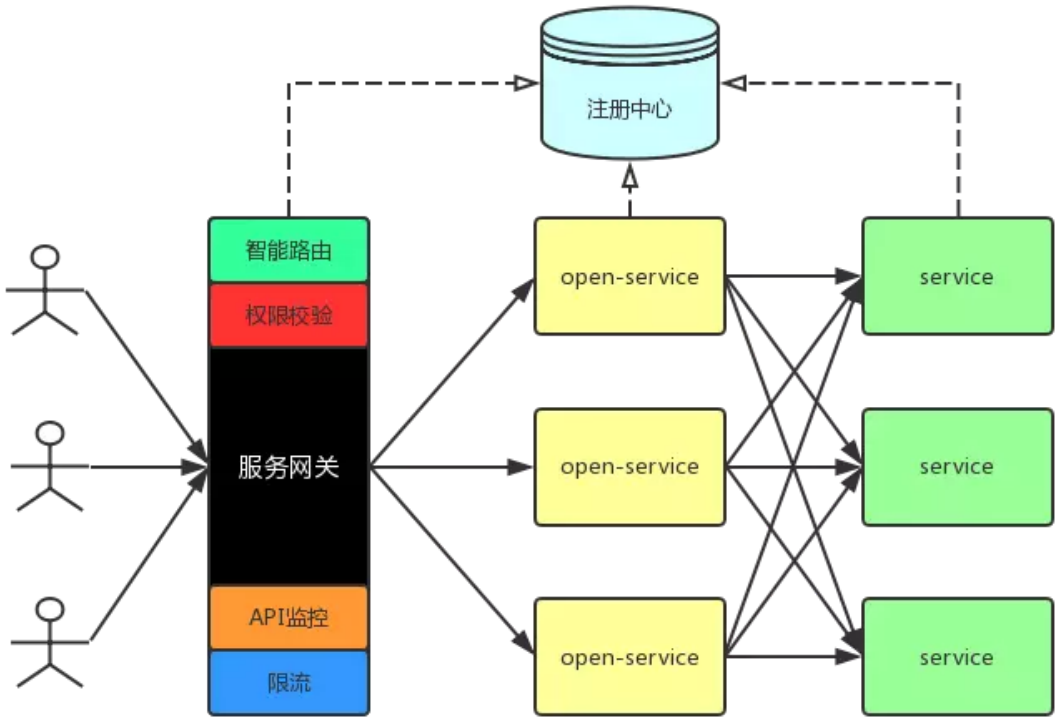
3.1.1 对比

从性能上讲 nginx + lua是性能最好的 , 不过由于要接触一个新的语言 , 开发起来很麻烦 , 而zuul是被SpringCloud集成的 , 生态好 , 易于开发 和Eureka,Hystrix等可以结合使用

4. 网关的执行流程与原理

这张图简单介绍了网关在微服务之中的地位与功能

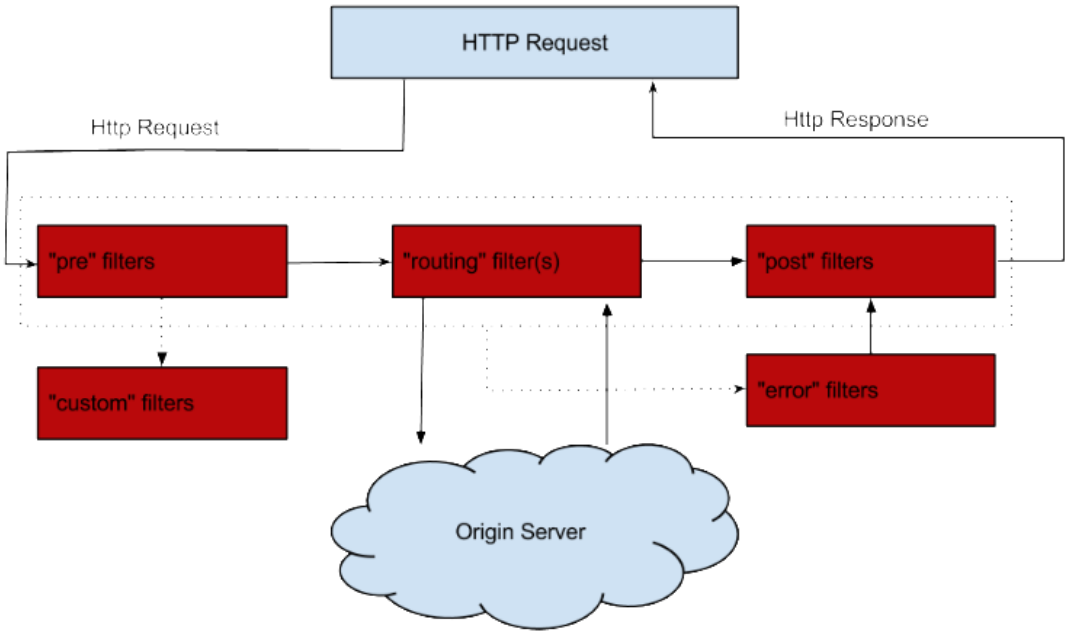
4.1.1 网关在微服务中的位置



4.1.2 网关的底层就是一个servlet

网关的底层就是一个servlet , 通过用过滤器来实现不同的功能

网关底层的过滤执行流程图 :



Zuul的中心是一系列过滤器，它们能够在HTTP请求和响应的路由期间执行一系列操作。

我们可以通过自定义过滤器 继承 ZuulFilter来实现网关的过滤机制，可以用来做**登录健全**

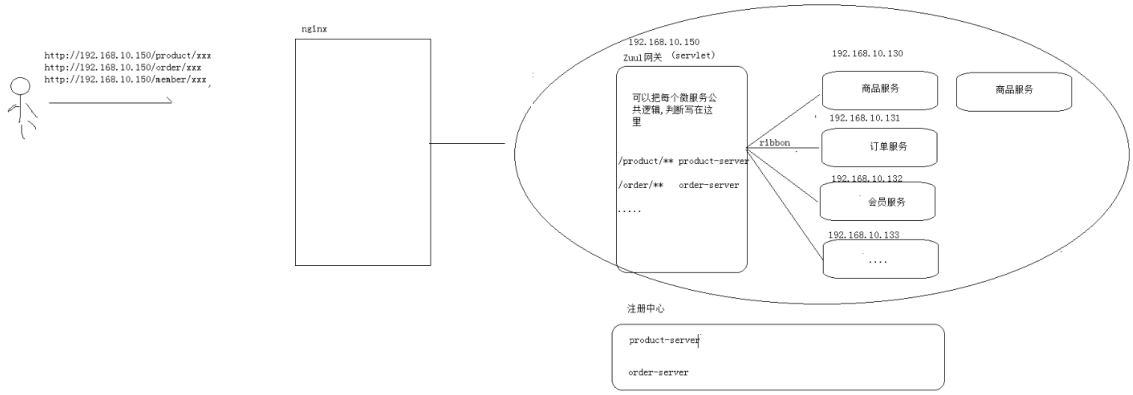
4.1.3 网关的执行流程

大体的执行流程

在生产者将服务放到了注册中心的服务列表的时候，我们的请求通过进入网关，网关做智能路由转发(比如服务发现，负载均衡) 到我们的消费者上面，这其中包括了，权限校检，监控，限流等操作

如果有注册中心的话，网关通过访问注册中心获取到注册中心上面服务列表中服务的端口号，然后回到网关，底层做 url拼接，然后用拼接好的地址，访问我们外部的服务，获取到数据后再返回给我们的客户端

4.1.4 网关的结构图



通常情况下我们的网关也是需要做集群处理的，在外部使用nginx来进行请求的分发，将不同的请求负载均衡到我们的网关上面，然后网关在从注册中心获取地址拼接，再负载均衡进行请求分发

5.网关的使用

5.1.1 网关的使用方式

Zuul可以通过加载动态过滤机制，从而实现以下各项功能：

1.验证与安全保障: 识别面向各类资源的验证要求并拒绝那些与要求不符的请求。

2.审查与监控: 在边缘位置追踪有意义数据及统计结果，从而为我们带来准确的生产状态结论。

3.动态路由: 以动态方式根据需要将请求路由至不同后端集群处。

4.压力测试: 逐渐增加指向集群的负载流量，从而计算性能水平。

5.负载分配: 为每一种负载类型分配对应容量，并弃用超出限定值的请求。

6.静态响应处理: 在边缘位置直接建立部分响应，从而避免其流入内部集群。

7.多区域弹性: 跨越AWS区域进行请求路由，旨在实现ELB使用多样化并保证边缘位置与使用者尽可能接近。

6.使用网关可能会遇到的问题

在使用网关的时候，网关默认是会把Cookie拦截的，我们可以通过配置文件来配置 不拦截Cookie

默认情况，网关会把Cookie", "Set-Cookie", "Authorization"这三个请求头过滤掉，下游的服务是获取不到这几个请求头的。

```
sensitiveHeaders
```

7.自定义网关实现登录的鉴别

```
@Component
public class AuthZuulFilter extends ZuulFilter {
    //过滤器的类型
    @Override
    public String filterType() {
        return FilterConstants.PRE_TYPE;
    }
    //过滤的顺序  这里默认是0
    @Override
    public int filterOrder() {
        return 1;
    }
    //要过滤的地址,是否需要拦截
    @Override
    public boolean shouldFilter() {
        RequestContext requestContext = RequestContext.getCurrentContext();
        HttpServletRequest request = requestContext.getRequest();
        if(request.getRequestURI().indexOf("/order/api/")>=0){
```

```
            return true;
        }
        return false;
    }
    //拦截的业务逻辑
    @Override
    public Object run() throws ZuulException {
        //从请求上下文中获取上下文对象
        RequestContext requestContext = RequestContext.getCurrentContext();
        HttpServletRequest request = requestContext.getRequest();
        String cookie = request.getHeader("Cookie");
        if(StringUtils.isEmpty(cookie)){
            cookie = request.getParameter("Cookie");
        }
        if(StringUtils.isEmpty(cookie)){
            //不要做下一步操作
            requestContext.setSendZuulResponse(false);
            //设置响应码 401没有权限

requestContext.setResponseStatusCode(HttpStatus.UNAUTHORIZED.value());
        }
        return null;
    }
}
```

8 网关的配置

1.创建一个springboot的项目 , 选择 Cloud Discover->Eureka Discover , Cloud Rounting -> Zuul

2.添加application.yml配置文件并添加相关的配置信息.

```
server:
  port: 9000
spring:
  application:
    name: zuul-server
eureka:
  client:
    serviceUrl:
      defaultZone: http://localhost:8761/eureka/
```

3.在启动类上贴上@EnableZuulProxy注解

`@EnableZuulProxy`

通知Spring的配置文件 开启网关

4.定义路由器的规则

```yaml
zuul:
  ignoredPatterns: /*-server/**
  routes:
    order-server-route:
      path: /order/**
      serviceId: order-server
```

5.网关超时时间.....