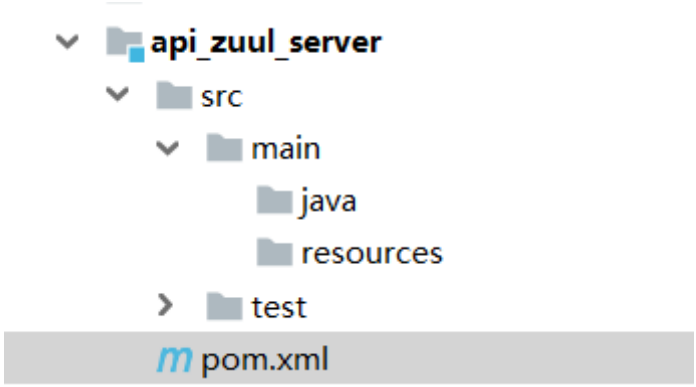
Zuul网关

1、搭建Zuul网关服务器

1.1 创建项目，导入坐标

创建子工程api_zuul_server



导入坐标

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

1.2 开发启动类

ZuulServerApplication：

```
package com.huike.zuul;

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;

@SpringBootApplication
//开启zuul网关功能
@EnableZuulProxy
public class ZuulServerApplication {

    public static void main(String[] args) {
        SpringApplication.run(ZuulServerApplication.class,args);
    }
}
```

1.3 基础配置

application.yml

```
server:
  port: 8080 #端口
spring:
  application:
    name: api-zuul-server #服务名称
```

1.4 路由配置

application.yml

```
  ##路由配置
  zuul:
    routes:
      #已商品微服务
      product-service: #路由id,随便写
        path: /product-service/** #映射路径  #localhost:8080/product-
service/sxxssds
        url: http://127.0.0.1:9001 #映射路径对应的实际微服务url地址
```

1.5 启动测试

启动，访问http://127.0.0.1:8080/product-service/product/2

2.面向服务的路由配置

2.1 导入坐标

```
    <dependency>
      <groupId>org.springframework.cloud</groupId>
        <artifactId>spring-cloud-starter-netflix-eureka-client</artifactId>
    </dependency>
```

2.2 启动类上增加服务发现注解

```
@EnableDiscoveryClient
```

2.3 增加Eureka相关配置

```
eureka:
  client:
    service-url:
      defaultZone: http://localhost:9000/eureka/
  instance:
    prefer-ip-address: true #使用ip地址注册
```

2.4 修改配置文件

```
zuul:
  routes:
  #已商品微服务
    product-service: #路由id,随便写
      path: /product-service/** #映射路径  #localhost:8080/product-service/sxxssds
#      url: http://127.0.0.1:9001 #映射路径对应的实际微服务url地址
      serviceId: service-product
```

2.5 访问测试

http://127.0.0.1:8080/product-service/product/2

3 简化的路由配置

3.1 路由简化与默认规则

```
  service-product: /product-service/**
     #zuul中的默认路由配置
     #如果当前的微服务名称 service-product , 默认的请求映射路径 /service-product/**
     #  /service-order/
```

3.2 测试

http://127.0.0.1:8080/product-service/product/2

http://127.0.0.1:8080/service-order/order/2

4 过滤器入门

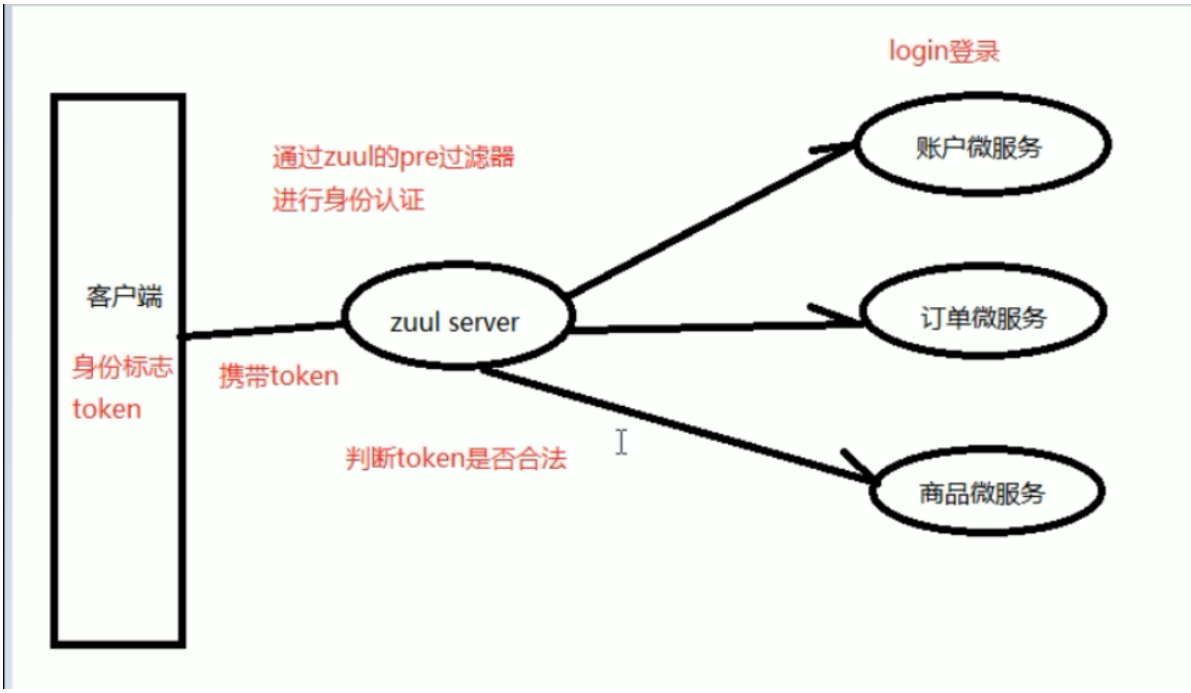
pre:转发到微服务之前执行的过滤器

routing：在路由请求时执行的过滤器

post：执行微服务获取返回值之后执行的过滤器

error：在整过阶段抛出异常的时候执行的过滤器

身份认证过滤器



4.1 LoginFilter开发

```
package com.huike.zuul.filter;

import com.netflix.zuul.ZuulFilter;
import com.netflix.zuul.context.RequestContext;
import com.netflix.zuul.exception.ZuulException;
import org.springframework.http.HttpStatus;
import org.springframework.stereotype.Component;

import javax.servlet.http.HttpServletRequest;

/**
 * 自定义的zuul过滤器
 *  继承抽象父类
 */
@Component
public class LoginFilter extends ZuulFilter {

    /**
     * 定义过滤器类型
     *  pre
     *  routing
     *  post
     *  error
     */
    public String filterType() {
        return "pre";
    }


    /**
     *  指定过滤器的执行顺序
     *      返回值越小,执行顺序越高
     */
    public int filterOrder() {
        return 1;
```

```
    }


    /**
     * 当前过滤器是否生效
     *  true : 使用此过滤器
     *  flase : 不使用此过滤器
     */
    public boolean shouldFilter() {
        return true;
    }


    /**
     * 指定过滤器中的业务逻辑
     *  身份认证:
     *      1.所有的请求需要携带一个参数 : access-token
     *      2.获取request请求
     *      3.通过request获取参数access-token
     *      4.判断token是否为空
     *      4.1 token==null : 身份验证失败
     *      4.2 token!=null : 执行后续操作
     *  在zuul网关中,通过RequestContext的上下问对象,可以获取对象request对象
     */
    public Object run() throws ZuulException {
        System.out.println("执行了过滤器");
        //1.获取zuul提供的上下文对象RequestContext
        RequestContext ctx = RequestContext.getCurrentContext();
        //2.从RequestContext中获取request
        HttpServletRequest request = ctx.getRequest();
        //3.获取请求参数access-token
        String token = request.getParameter("access-token");
        //4.判断
        if (token ==null) {
            //4.1 如果token==null ,拦截请求,返回认证失败
            ctx.setSendZuulResponse(false); // 拦截请求
            ctx.setResponseStatusCode(HttpStatus.UNAUTHORIZED.value());
        }
        //4.2 如果token!=null ,继续后续操作
        return null;
    }
}
```

4.2 测试

http://127.0.0.1:8080/product-service/product/2?access-token=1

http://127.0.0.1:8080/product-service/product/2