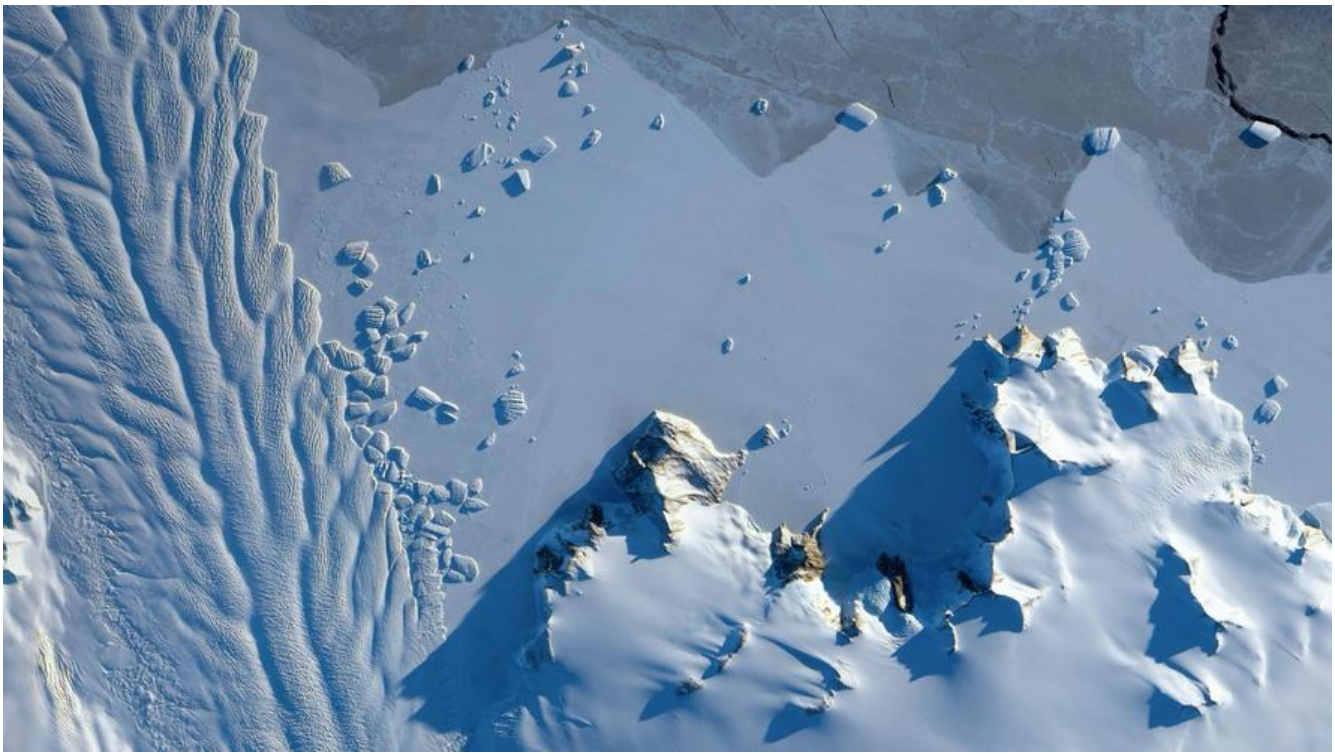
homepage 第二期：服务网关模块开发

作者：ChenforCode
原文链接：https://ld246.com/article/1573382605343
来源网站：链滴



1.在今后的实验中，都采用单节点的eureka进行开发，所以把bootstrap的配置全部注释掉，重新启用pplication的配置。

2.创建homepage-zuul模块，完成pom文件配置，比较注意的一点是，上一节我们做的是erueka server的开发，那么接下来的所有模块都应该属于erueka client，因此需要在pom文件中加入eureka client的依赖

```
<?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">
    <parent>
        <artifactId>imooc-homepage</artifactId>
        <groupId>cn.chenforcode.homepage</groupId>
        <version>1.0-SNAPSHOT</version>
    </parent>
    <modelVersion>4.0.0</modelVersion>

    <artifactId>homepage-zuul</artifactId>
    <version>1.0-SNAPSHOT</version>
    <packaging>jar</packaging>

    <!--  模块信息描述  -->
    <name>homepage-zuul</name>
    <description>Spring Cloud Gateway</description>

    <dependencies>
        <!--
          Eureka客户端，客户端向Eureka server注册的时候会提供一系列的元数据信息，如主机，端
，健康检查url等
          Eureka Server接收每个客户端的心跳信息，如果在某个配置的超时时间内未收到心跳信息，
```

```
例会被从注册列表中移除
        -->
        <dependency>
            <groupId>org.springframework.cloud</groupId>
            <artifactId>spring-cloud-starter-netflix-eureka-client</artifactId>
        </dependency>

        <!-- 服务网关 -->
        <dependency>
            <groupId>org.springframework.cloud</groupId>
            <artifactId>spring-cloud-starter-netflix-zuul</artifactId>
        </dependency>

        <!-- apache工具类 -->
        <dependency>
            <groupId>commons-io</groupId>
            <artifactId>commons-io</artifactId>
            <version>1.3.2</version>
        </dependency>
    </dependencies>

    <build>
        <plugins>
            <plugin>
                <groupId>org.springframework.boot</groupId>
                <artifactId>spring-boot-maven-plugin</artifactId>
            </plugin>
        </plugins>
    </build>
</project>
```

3.在src目录下建立与server相类似的结构，建立包和相应的启动类ZuulGatewayApplication

```
package cn.chenforcode.homepage;

import org.springframework.boot.SpringApplication;
import org.springframework.cloud.client.SpringCloudApplication;
import org.springframework.cloud.netflix.zuul.EnableZuulProxy;

/**
 * @author <a href="http://www.chenforcode.cn">PKUCoder</a>
 * @date 2019/11/10 6:00 下午
 * @description 服务网关的启动类
 * EnableZuulProxy用来标示这个类为zuul server
 * SpringCloudApplication 是一个组合注解，用来简化配置，里边有额外的一些注解
 */
@EnableZuulProxy
@SpringCloudApplication
public class ZuulGatewayApplication {
    public static void main(String[] args) {
        SpringApplication.run(ZuulGatewayApplication.class, args);
    }
}
```

4.实现自定义过滤器，建立filter包，建立PreRequestFliter

```
package cn.chenforcode.homepage.fliter;

import com.netflix.zuul.ZuulFilter;
import com.netflix.zuul.context.RequestContext;
import com.netflix.zuul.exception.ZuulException;
import org.springframework.cloud.netflix.zuul.filters.support.FilterConstants;
import org.springframework.stereotype.Component;

/**
 * @author <a href="http://www.chenforcode.cn">PKUCoder</a>
 * @date 2019/11/10 6:12 下午
 * @description 自定义过滤器，记录客户端发起请求的时间戳
 */
@Component
public class PreRequestFliter extends ZuulFilter {
    //定义过滤器的类型，代表过滤器在请求的什么时候进行调用，由于记录请求时间，所以在请求带
前记录时间
    @Override
    public String filterType() {
        return FilterConstants.PRE_TYPE;
    }

    //定义过滤器的执行顺序，数值越小代表优先级越高
    @Override
    public int filterOrder() {
        return 0;
    }

    //定义是否启用该过滤器，true代表启用，false代表不启用
    @Override
    public boolean shouldFilter() {
        return false;
    }

    @Override
    public Object run() throws ZuulException {
        //获取请求上下文，用与在过滤器之间传递消息，它的本质是一个concurrentHashmap，所以
储的时候以key-value的形式存储
        RequestContext ctx = RequestContext.getCurrentContext();
        ctx.set("startTime", System.currentTimeMillis());
        return null;
    }
}
```

5.建立AccessLogFliter

```
package cn.chenforcode.homepage.fliter;

import com.netflix.zuul.ZuulFilter;
import com.netflix.zuul.context.RequestContext;
import com.netflix.zuul.exception.ZuulException;
```

```
import lombok.extern.slf4j.Slf4j;
import org.springframework.cloud.netflix.zuul.filters.support.FilterConstants;
import org.springframework.stereotype.Component;

import javax.servlet.http.HttpServletRequest;

/**
 * @author <a href="http://www.chenforcode.cn">PKUCoder</a>
 * @date 2019/11/10 6:22 下午
 * @description 记录请求执行时间
 */
@Component
@Slf4j
public class AccessLogFliter extends ZuulFilter {
    //在请求之后执行
    @Override
    public String filterType() {
        return FilterConstants.POST_TYPE;
    }

    //这个常量是最低的，但是order设置成这个值的时候过滤器失效，所以在他之前执行，比最后一个
rder的优先级略高
    @Override
    public int filterOrder() {
        return FilterConstants.SEND_RESPONSE_FILTER_ORDER - 1;
    }

    @Override
    public boolean shouldFilter() {
        return true;
    }

    //打印日志，记录每个请求的时间
    @Override
    public Object run() throws ZuulException {
        RequestContext ctx = RequestContext.getCurrentContext();
        Long startTime = (Long) ctx.get("startTime");

        HttpServletRequest request = ctx.getRequest();
        String uri = request.getRequestURI();
        long duration = System.currentTimeMillis() - startTime;

        log.info("uri: {}, duration: {}", uri, duration / 100);
        return null;
    }
}
```

在这里简述一下自己的疑问：

如果在第一个请求过来的时候，设置了一下startTime，然后在这个请求没有执行到第二个过滤器的候，又来了一个请求，那不就会又设置了一下startTime了吗？这个时候第一个请求最后算出来的时就不准了。

所以我觉得不应该使用同一个名字记录这个startTime。应该在记录开始时间的时候也拿到相应的uri

然后用一个uriStartTime来存储他的时间，也就是把自己的时间key能够标识出来。

6.完成配置文件application.yml文件

```
server:
  port: 9000
spring:
  application:
    name: homepage-zuul
eureka:
  client:
    service-url:
      defaultZone: http://server1:8000/eureka/
```

注意这里的eureka配置要写成client，不再是server了