乐字节教育高级架构课程

正所谓“授人以鱼不如授人以渔”，你们想要的 Java 学习资料来啦！

不管你是学生，还是已经步入职场的同行，希望你们都要珍惜眼前的学习机会，奋斗没有终点，知识永不过时。

扫描下方二维码即可领取

乐字节晓啡

乐字节官方交流群

自定义限流策略

如果希望自己控制限流策略，可以通过自定义 `RateLimitKeyGenerator` 的实现来增加自己的策略逻辑。

修改商品服务控制层代码如下，添加 `/product/single`：

```
package com.example.controller;

import com.example.pojo.Product;
import com.example.service.ProductService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;

@RestController
@RequestMapping("/product")
public class ProductController {

    @Autowired
    private ProductService productService;

    /**
     * 根据主键查询商品
     *
     * @param id
     * @return
     */
    @GetMapping("/{id}")
    public Product selectProductById(@PathVariable("id") Integer id) {
        return productService.selectProductById(id);
    }

    /**
     * 根据主键查询商品
     *
     * @param id
     * @return
     */
    @GetMapping("/single")
    public Product selectProductSingle(Integer id) {
        return productService.selectProductById(id);
    }

}
```

自定义限流策略类。

```
package com.example.ratelimit;

import com.marcosbarbero.cloud.autoconfigure.zuul.ratelimit.config.RateLimitUtils;
import
com.marcosbarbero.cloud.autoconfigure.zuul.ratelimit.config.properties.RateLimitProperties;
import
com.marcosbarbero.cloud.autoconfigure.zuul.ratelimit.support.DefaultRateLimitKeyGenerator;
import org.springframework.cloud.netflix.zuul.filters.Route;
import org.springframework.stereotype.Component;

import javax.servlet.http.HttpServletRequest;

/**
 * 自定义限流策略
 */
@Component
public class RateLimitKeyGenerator extends DefaultRateLimitKeyGenerator {

    public RateLimitKeyGenerator(RateLimitProperties properties, RateLimitUtils
rateLimitUtils) {
        super(properties, rateLimitUtils);
    }

    /**
     * 限流逻辑
     *
     * @param request
     * @param route
     * @param policy
     * @return
     */
    @Override
    public String key(HttpServletRequest request, Route route, RateLimitProperties.Policy
policy) {
        // 对请求参数中相同的 id 值进行限流
        return super.key(request, route, policy) + ":" + request.getParameter("id");
    }

}
```

多次访问：http://localhost:9000/api/product-service/product/single?token=abc123&id=1 被限流后，马上更换 `id=2` 重新访问发现服务任然可用，再继续多次访问，发现更换过的 `id=2` 也被限流了。Redis 信息如下：

```
127.0.0.1:6379> keys *
1) "zuul-server:product-service:0:0:0:0:0:0:0:1:/product/single:anonymous:1"
2) "zuul-server:product-service:0:0:0:0:0:0:0:1:/product/single:anonymous:2"
```