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

自定义限流处理（网关熔断）

发生限流之后的处理流程：

- 发生限流之后可自定义返回参数，通过实现 `ZuulBlockFallbackProvider` 接口，默认的实现是 `DefaultBlockFallbackProvider`。
- 默认的 fallback route 的规则是 route ID 或自定义的 API 分组名称。

编写限流处理类

OrderBlockFallbackProvider.java

```
package com.example.fallback;

import com.alibaba.csp.sentinel.adapter.gateway.zuul.fallback.BlockResponse;
import com.alibaba.csp.sentinel.adapter.gateway.zuul.fallback.ZuulBlockFallbackProvider;
import com.alibaba.csp.sentinel.slots.block.BlockException;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;

/**
 * 对订单服务做服务容错处理
 */
public class OrderBlockFallbackProvider implements ZuulBlockFallbackProvider {

    private Logger logger = LoggerFactory.getLogger(OrderBlockFallbackProvider.class);

    @Override
    public String getRoute() {
        return "order-service"; // 服务名称
    }

    @Override
    public BlockResponse fallbackResponse(String route, Throwable cause) {
        logger.error("{} 服务触发限流", route);
        if (cause instanceof BlockException) {
            return new BlockResponse(429, "服务访问压力过大，请稍后再试。", route);
        } else {
            return new BlockResponse(500, "系统错误，请联系管理员。", route);
        }
    }

}
```

将限流处理类注册至 Zuul 容器

ZuulConfig.java

```
// Spring 容器初始化的时候执行该方法
@PostConstruct
public void doInit() {
    // 注册 FallbackProvider
    ZuulBlockFallbackManager.registerProvider(new OrderBlockFallbackProvider());
    // 加载网关限流规则
    initGatewayRules();
}
```

多次访问：http://localhost:9001/order-service/order/1 触发限流后返回自定义提示：