**Hystrix服务容错**

基于Netflix对应的Hystrix。

**服务降级**

1 优先核心服务。非核心服务不可用或者弱可用

2 通过HystrixCommand注解指定

3 fallbackMethod(回退函数)中具体实现降级逻辑

**使用方法**

**1 引入依赖**

**2 启动类也要加注解**

**3 如下**

@HystrixCommand(fallbackMethod = "fallback")  
@GetMapping("/getProductInfoList")  
public String getProductInfoList(){  
 RestTemplate restTemplate = new RestTemplate();  
 return restTemplate.postForObject("http://127.0.0.1:8082/product/listForOrder", Arrays.*asList*("1234568"),String.class);  
}  
  
  
private String fallback(){  
 return "太拥挤了，稍后再试";  
}

当目标服务无法访问的时候，或者自己的服务需要降级处理的时候用。

4 可以使用DefaultProperties(defaultFallback = “defaultFallBack”)，指定默认的方法

5 指定超时时间，默认是1秒，1秒钟无响应结果的话自动执行默认方法。所以一般要设置一下超时时间

6 docker使用仓闭模式实现进程隔离，但是Hystrix实现的是线程池的隔离

7 **线程熔断**：当服务失败次数累计到一定程度，自行触发失败机制。打开一个时间窗，在时间窗范围内请求全部失败。当时间窗时间到了之后，进入到半打开状态，然后进行一次请求，若请求成功，则恢复到原来的状态，若是失败，主逻辑依然闭合。则再次进入到时间窗。开始计时。

8 **feign集成**

**1 加上配置**

2之前的HystrixCommand注解都是注解在类或者方法上，当和feign组件联合使用注解在接口上的时候，这么使用。

@FeignClient(name = "product" ,fallback = ProductClient.ProductClientFallback.class)  
public interface ProductClient {  
  
 @PostMapping("/product/listForOrder")  
 List<ProductInfoOutput> listForOrder(@RequestBody List<String> productIdList);  
  
 @PostMapping("/product/decreaseStock")  
 void decreaseStock(@RequestBody List<DecreaseStockInput> decreaseStockInputList);  
  
  
 @Component  
 static class ProductClientFallback implements ProductClient{  
  
 @Override  
 public List<ProductInfoOutput> listForOrder(List<String> productIdList) {  
 return null;  
 }  
  
 @Override  
 public void decreaseStock(List<DecreaseStockInput> decreaseStockInputList) {  
  
 }  
 }  
}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*暂时挖了一个坑\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

在order项目中Springcloud的版本和Hystrix版本不统一，