trade-order

说明

- 项目编码：UTF-8
- JDK:1.8
- IDE：不限
- 项目开发方式
  - 分支开发，发布时合并主干
  - 环境隔离（profile：local、dev、uat、prod）

模块

- biz：公共业务模块（供其它模块调用）
- biz-base:基础业务
- biz-req-trace:请求链路跟踪
- biz-glue:
- trade-order-api：业务代码
- trade-order-gateway：路由服务

功能一 实现类似于Nginx请求转发功能

1.通过applications.yml读取负载均衡的URL列表，actions如下

```
actions: http://campus.query1.ksyun.com:8089,http://campus.query2.ksyun.com:8089
```

2.通过随机负载均衡算法将trade-order-gateway的请求打至trade-order-api

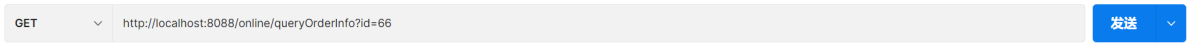
3.采用postman发送http请求。

4.**结果**

系统报的日志如下，可以看到两次请求发送去了两个不同的url

功能二 查询订单详情

1.通过postman发送请求

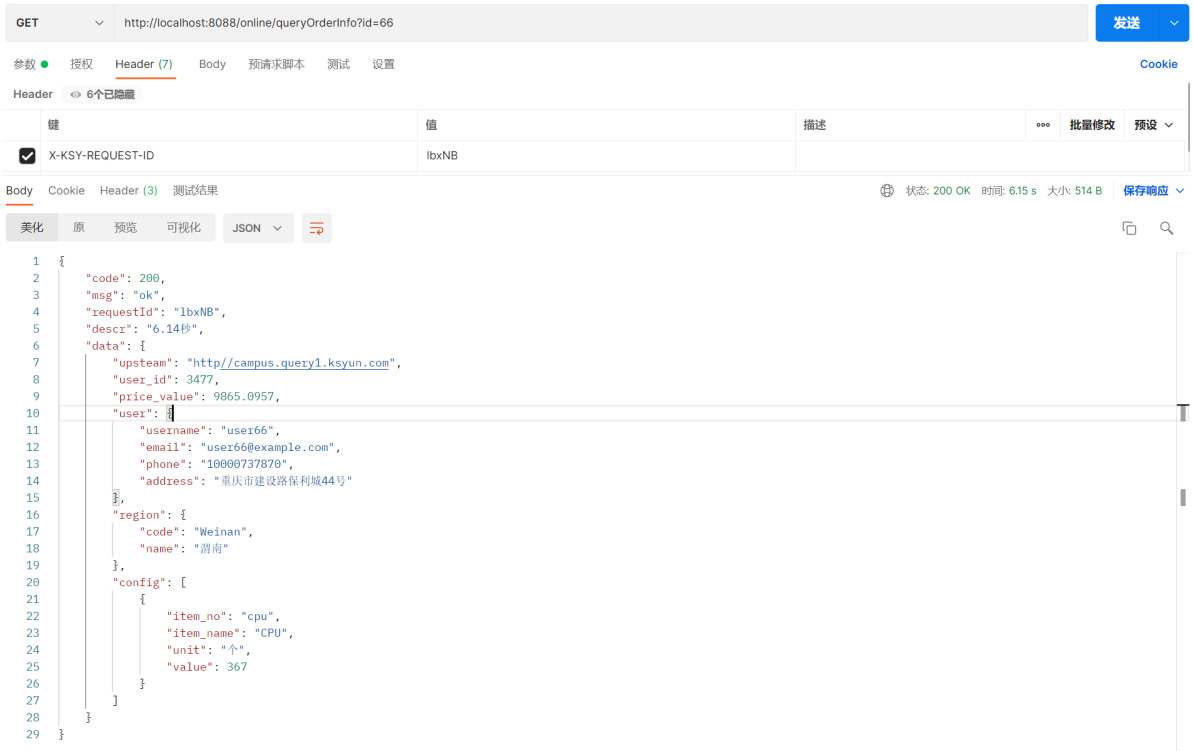


2.通过随机负载均衡算法去打请求

3.用到了多级缓存，先差本地缓存，再查redis缓存，没有就最后再去查数据库

第一次查询结果如下，本地和redis都没有，去查数据库和第三方接口共用了6.14s：

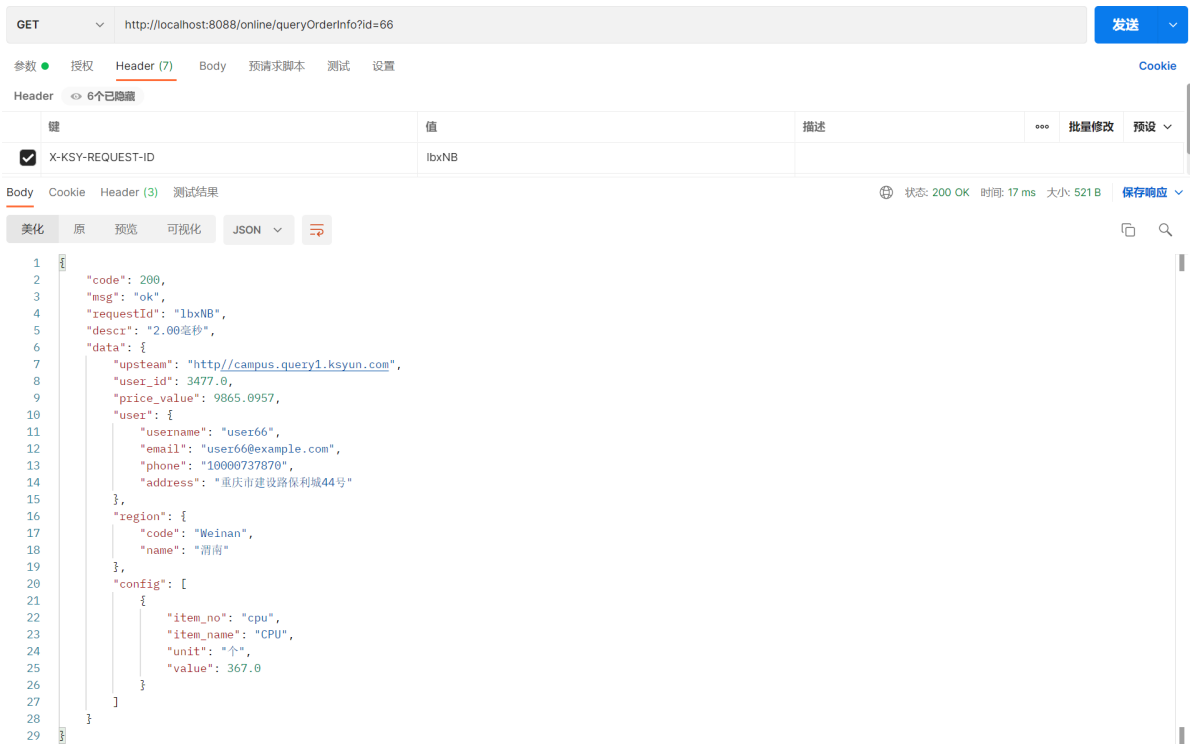
```
2023-07-24 15:58:22.684 |[请求的requestId为:lbxNB]|  INFO | TradeOrderController | http-nio-8089-exec-9  | | 本地和redis都没有，去查数据库！！！
```




```
{
    "code": 200,
    "msg": "ok",
    "requestId": "lbxNB",
    "descr": "6.14秒",
    "data": {
        "upsteam": "http//campus.query1.ksyun.com",
        "user_id": 3477,
        "price_value": 9865.0957,
        "user": {
            "username": "user66",
            "email": "user66@example.com",
            "phone": "10000737870",
            "address": "重庆市建设路保利城44号"
        },
        "region": {
            "code": "Weinan",
            "name": "渭南"
        },
        "config": [
            {
                "item_no": "cpu",
                "item_name": "CPU",
                "unit": "个",
                "value": 367
            }
        ]
    }
}
```


第二次结果如下，只用了2ms，用到了本地缓存

```
2023-07-24 15:58:22.684 |[请求的requestId为:lbxNB]|  INFO | TradeOrderController | http-nio-8089-exec-9  | | 本地和redis都没有，去查数据库！！！
```




```
{
    "code": 200,
    "msg": "ok",
    "requestId": "lbxNB",
    "descr": "2.00毫秒",
    "data": {
        "upsteam": "http//campus.query1.ksyun.com",
        "user_id": 3477.0,
        "price_value": 9865.0957,
        "user": {
            "username": "user66",
            "email": "user66@example.com",
            "phone": "10000737870",
            "address": "重庆市建设路保利城44号"
        },
        "region": {
            "code": "Weinan",
            "name": "渭南"
        },
        "config": [
            {
                "item_no": "cpu",
                "item_name": "CPU",
                "unit": "个",
                "value": 367.0
            }
        ]
    }
}
```

重启程序，会用到redis缓存

```
2023-07-24 16:03:12.902 |[请求的requestId为:lbxNB]|  INFO | TradeOrderController | http-nio-8089-exec-1  | | 用到了redis缓存！！
2023-07-24 16:03:12.906 |[请求的requestId为:lbxNB]|  INFO | webLogger | http-nio-8089-exec-1  | | 方法:TradeOrderController_query 执行耗时:41.18 ms !
```

功能三 根据机房Id查询机房名称

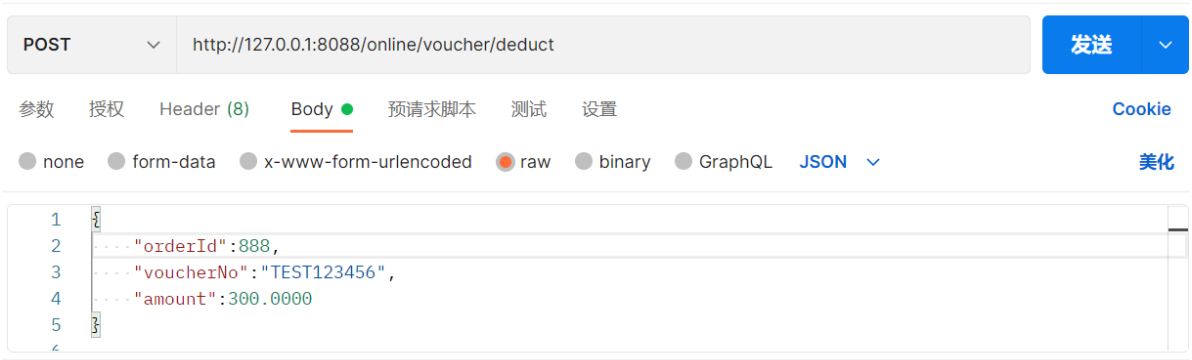
本功能用到的知识点：重试，我用的guava-retrying（第三方接口不稳定，我这里重试次数设置为10次）

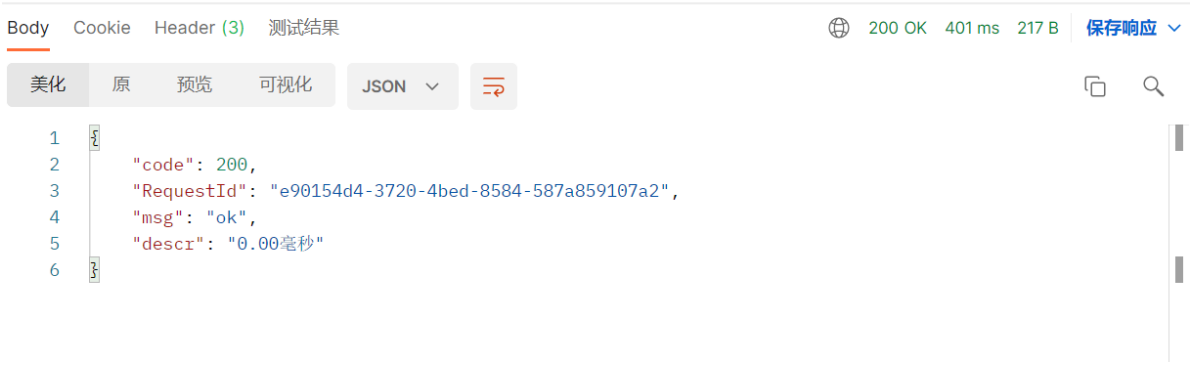
1.postman获得的结果如下

这次运气不好，一共重试了五次才获取成功，哈哈哈。

功能四 订单优惠券抵扣公摊

用postman的http的post请求发送，，结果如下，可以抵扣多次





对象 ksc_voucher_deduct @test_trade (lb...

开始事务 文本 筛选 排序 导入 导出

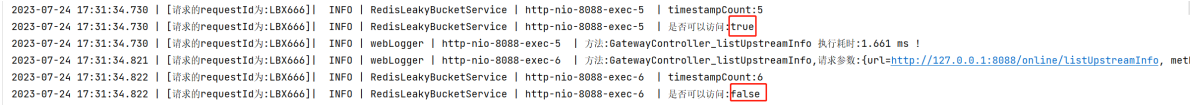
id	order_id	voucher_no	amount	before_deduct_a	after_deduct_am	create_time	update_time
1	1	TEST123456	10.0000	2000.0000	1990.0000	2023-07-23 04:02	2023-07-23 04:02:08
65	1	TEST123456	20.0000	1990.0000	1970.0000	2023-07-23 23:45	2023-07-23 23:45:36
66	1	TEST123456	0000.0000	0.0000	0.0000	2023-07-23 23:45	2023-07-23 23:45:48
67	1	TEST123456	2.0000	0.0000	0.0000	2023-07-23 23:45	2023-07-23 23:45:57
68	1	TEST123456	2.0000	0.0000	0.0000	2023-07-23 23:47	2023-07-23 23:47:50
69	1	TEST123456	2.0000	0.0000	0.0000	2023-07-23 23:47	2023-07-23 23:47:51
70	10	TEST123456	2.0000	9406.5013	9404.5013	2023-07-24 00:59	2023-07-24 00:59:11
71	39	TEST123456	2.0000	9842.3585	9840.3585	2023-07-24 00:59	2023-07-24 00:59:47
72	39	TEST123456	2.0000	9840.3585	9838.3585	2023-07-24 01:00	2023-07-24 01:00:00
73	39	TEST123456	10.0000	9838.3585	9828.3585	2023-07-24 01:00	2023-07-24 01:00:12
74	500	TEST123456	0000.0000	4901.4077	0.0000	2023-07-24 01:01	2023-07-24 01:01:30
75	5000	TEST123456	0000.0000	5890.0230	0.0000	2023-07-24 01:04	2023-07-24 01:04:16
76	5001	TEST123456	0000.0000	11861.4906	0.0000	2023-07-24 01:06	2023-07-24 01:06:06
77	5001	TEST123456	0000.0000	0.0000	0.0000	2023-07-24 01:07	2023-07-24 01:07:16
78	777	TEST123456	300.0000	17179.8338	16879.8338	2023-07-24 15:38	2023-07-24 15:38:52
79	888	TEST123456	300.0000	12151.7954	11851.7954	2023-07-24 17:01	2023-07-24 17:01:42
80	888	TEST123456	300.0000	11851.7954	11551.7954	2023-07-24 17:02	2023-07-24 17:02:44

功能五 基于Redis实现漏桶限流算法，并在API调用上体现

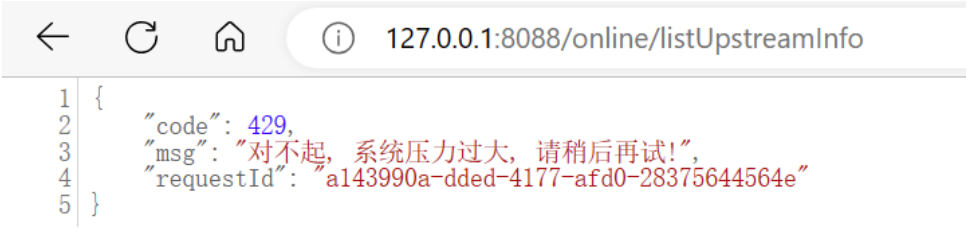
1.实现思路

1.写了一漏桶限流类 `RedisLeakyBucketService`，
2.定义了`INTERVAL_TIME = 1000;`表示一秒
3.定义了`jedis`的`list`变量，往里面存储时间戳，通过计算`1s`内时间戳的数量来判断当前有多少给`http`请求，如果请求数量大于设定的阈值`(5)`,就返回`false`，表示现在`http`请求已经满了，不可访问了

用postman发送请求，我一次发了100个，从下图中可以看出在数量为6时就不可以访问了



postman发送批量请求的，所以我在浏览器访问结果如下，所以是访问不了的



简单的链路跟踪实现

1.在postman发送http请求的时候带X-KSY-REQUEST-ID及requestId

2. 通过HttpServletRequest获取请求头的requestId，并放入MDC.put(traceId,requestId)放入日志中
3. 再通过把traceId放入logback的配置文件中就可以了

在日志中体现如下图：