

# sentinel-springcloudalibaba

## 1.注意点

- 1 1.dashboard动态的规则，添加的资源不需要使用注解@SentinelResource标识，它只是单独用其他名称不一定用@RequestMapping里的路径
- 2 2.dashboard配置的规则默认是存在内存中的，可以持久化到nacos等第三方持久化中
- 3 3.持久化的方向  
只能从nacos读取，然后加载到dashboard；不能动态添加规则，然后同步到nacos中
- 4 4.链路型的bug  
流控模式为链路的，被访问的只能是非controller层的，且需要手动编码（我用的是注解）进行资源命名，才有效
- 5 5.GlobalSentinelExceptionHandler  
解决只有注解中value,不指定blockException,抛出错误UndeclaredThrowableException，不走全局block的，另外单独写的

## 2.整合基础环境

### 1.客户端项目

#### 1.pom

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <project xmlns="http://maven.apache.org/POM/4.0.0"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
3     xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
  http://maven.apache.org/xsd/maven-4.0.0.xsd">
4     <modelVersion>4.0.0</modelVersion>
5     <!-- Spring Boot 父工程 -->
6     <parent>
7         <groupId>org.springframework.boot</groupId>
8         <artifactId>spring-boot-starter-parent</artifactId>
9         <version>2.7.5</version>
10        <relativePath/> <!-- lookup parent from repository -->
11    </parent>
12    <groupId>com.example.study</groupId>
13    <artifactId>springcloudalibaba-sentinel</artifactId>
14    <version>0.0.1-SNAPSHOT</version>
15    <name>springcloudalibaba-sentinel</name>
16    <description>Demo project for Spring Boot</description>
17
18
19    <properties>
20        <java.version>1.8</java.version> <!-- 指定 JDK 8 -->
21        <spring-cloud.version>2021.0.5</spring-cloud.version>
22    </properties>
```

```
23
24 <!-- 依赖版本管理（锁定 Spring Cloud 和 Spring Cloud Alibaba 版本） -->
25 <dependencyManagement>
26     <dependencies>
27         <!-- Spring Cloud 依赖管理 -->
28         <dependency>
29             <groupId>org.springframework.cloud</groupId>
30             <artifactId>spring-cloud-dependencies</artifactId>
31             <version>${spring-cloud.version}</version>
32             <type>pom</type>
33             <scope>import</scope>
34         </dependency>
35         <!-- Spring Cloud Alibaba 依赖管理（内置 Sentinel 1.8.6） -->
36         <dependency>
37             <groupId>com.alibaba.cloud</groupId>
38             <artifactId>spring-cloud-alibaba-dependencies</artifactId>
39             <version>2021.0.5.0</version>
40             <type>pom</type>
41             <scope>import</scope>
42         </dependency>
43     </dependencies>
44 </dependencyManagement>
45
46 <dependencies>
47     <!-- Spring Boot web 依赖（提供 web 环境） -->
48     <dependency>
49         <groupId>org.springframework.boot</groupId>
50         <artifactId>spring-boot-starter-web</artifactId>
51     </dependency>
52
53     <!-- https://mvnrepository.com/artifact/com.alibaba.cloud/spring-
cloud-starter-alibaba-sentinel -->
54     <dependency>
55         <groupId>com.alibaba.cloud</groupId>
56         <artifactId>spring-cloud-starter-alibaba-sentinel</artifactId>
57         <version>2.2.8.RELEASE</version>
58         <scope>compile</scope>
59     </dependency>
60     <dependency>
61         <groupId>junit</groupId>
62         <artifactId>junit</artifactId>
63         <scope>test</scope>
64     </dependency>
65
66     <!-- 5. Spring Boot 测试依赖（可选，用于接口测试） -->
67     <dependency>
68         <groupId>org.springframework.boot</groupId>
69         <artifactId>spring-boot-starter-test</artifactId>
70         <scope>test</scope>
71     </dependency>
72 </dependencies>
73
74 <build>
75     <plugins>
76         <plugin>
77             <groupId>org.springframework.boot</groupId>
78             <artifactId>spring-boot-maven-plugin</artifactId>
79             <version>2.7.5</version>
```

```

80         </plugin>
81     </plugins>
82 </build>
83
84 </project>
85

```

## 2.yam

```

1
2 server:
3     port: 7000 # 项目端口（避免与 Sentinel 控制台默认 8080 冲突）
4 spring:
5     application:
6         name: springcloudalibaba-sentinel # 应用名称（控制台显示）
7     cloud:
8         sentinel:
9             transport:
10                dashboard: localhost:8080 # Sentinel 控制台地址（匹配 1.8.6 版本）
11                port: 8719 # 应用与控制台通信端口（默认，被占用自动递增）
12                eager: true # 开启非懒加载，项目启动后直接注册到控制台（无需首次访问接口）
13                web-context-unify: true # 统一 web 上下文，适配 Spring Boot 2.7.5 web 环
境
14
15
16
17

```

## 3.controller

```

1 package com.example.study.springcloudalibabasentinel.controller;
2
3 import com.alibaba.csp.sentinel.annotation.SentinelResource;
4 import com.alibaba.csp.sentinel.slots.block.BlockException;
5 import org.springframework.web.bind.annotation.GetMapping;
6 import org.springframework.web.bind.annotation.PathVariable;
7 import org.springframework.web.bind.annotation.RestController;
8
9 @RestController
10 public class HelloController {
11
12     /**
13      * 被 Sentinel 保护的接口资源（兼容 JDK 8 + Spring Boot 2.7.5）
14      * @param id 路径参数
15      * @return 响应结果
16      */
17     @GetMapping("/hello/{id}")
18     @SentinelResource(
19         value = "helloResource", // 资源名称（控制台标识）
20         blockHandler = "helloBlockHandler", // 限流/熔断兜底方法
21         fallback = "helloFallback" // 业务异常兜底方法
22     )

```

```

23     public String hello(@PathVariable Integer id) {
24         // 模拟业务异常 (JDK 8 支持的语法)
25         if (id == 0) {
26             throw new RuntimeException("id 不能为 0 (JDK 8 兼容)");
27         }
28         String s = "Hello Sentinel 1.8.6 + Spring Boot 2.7.5, id: " + id;
29         System.out.println(s);
30         return s;
31     }
32
33     /**
34      * 限流/熔断兜底方法 (JDK 8 兼容, 支持 BlockException 入参)
35      */
36     public String helloBlockHandler(Integer id, BlockException e) {
37         String s = "当前请求过于频繁, 请稍后再试 (限流/熔断保护), id: " + id;
38         System.out.println(s);
39         return s;
40     }
41
42     /**
43      * 业务异常兜底方法 (JDK 8 兼容, 支持 Throwable 入参)
44      */
45     public String helloFallback(Integer id, Throwable e) {
46         String s = "业务处理失败: " + e.getMessage() + ", id: " + id;
47         System.out.println(s);
48         return s;
49     }
50 }

```

## 2.dashboard

### 1.下载安装

```

1  2.dashboard地址
2  https://github.com/alibaba/Sentinel/releases

```

### 2.启动访问

```

1  1.下载好jar后
2  执行 java -jar sentinel-dashboard-1.8.9.jar
3
4  2.启动刚写的服务8088的
5
6  3.观察dashboard
7  http://localhost:8080
8  默认账号密码
9  sentinel/sentinel

```

## 3.jmeter

- 1 自行下载，项目中附带保存测试用的文件
- 2 springcloudalibaba-sentinel

## 4.hello测试

```
1 效果
2 Hello Sentinel 1.8.6 + Spring Boot 2.7.5, id: 1
3 Hello Sentinel 1.8.6 + Spring Boot 2.7.5, id: 1
4 当前请求过于频繁，请稍后再试（限流/熔断保护），id: 1
5 当前请求过于频繁，请稍后再试（限流/熔断保护），id: 1
6 当前请求过于频繁，请稍后再试（限流/熔断保护），id: 1
7 当前请求过于频繁，请稍后再试（限流/熔断保护），id: 1
8 Hello Sentinel 1.8.6 + Spring Boot 2.7.5, id: 1
9 当前请求过于频繁，请稍后再试（限流/熔断保护），id: 1
10 当前请求过于频繁，请稍后再试（限流/熔断保护），id: 1
11 当前请求过于频繁，请稍后再试（限流/熔断保护），id: 1
12 当前请求过于频繁，请稍后再试（限流/熔断保护），id: 1
13 Hello Sentinel 1.8.6 + Spring Boot 2.7.5, id: 1
14 当前请求过于频繁，请稍后再试（限流/熔断保护），id: 1
15 当前请求过于频繁，请稍后再试（限流/熔断保护），id: 1
16 当前请求过于频繁，请稍后再试（限流/熔断保护），id: 1
17 当前请求过于频繁，请稍后再试（限流/熔断保护），id: 1
18 Hello Sentinel 1.8.6 + Spring Boot 2.7.5, id: 1
19 当前请求过于频繁，请稍后再试（限流/熔断保护），id: 1
20 当前请求过于频繁，请稍后再试（限流/熔断保护），id: 1
21 当前请求过于频繁，请稍后再试（限流/熔断保护），id: 1
22 当前请求过于频繁，请稍后再试（限流/熔断保护），id: 1
23 Hello Sentinel 1.8.6 + Spring Boot 2.7.5, id: 1
24 当前请求过于频繁，请稍后再试（限流/熔断保护），id: 1
25 当前请求过于频繁，请稍后再试（限流/熔断保护），id: 1
26 当前请求过于频繁，请稍后再试（限流/熔断保护），id: 1
27 当前请求过于频繁，请稍后再试（限流/熔断保护），id: 1
28 Hello Sentinel 1.8.6 + Spring Boot 2.7.5, id: 1
29 当前请求过于频繁，请稍后再试（限流/熔断保护），id: 1
30 当前请求过于频繁，请稍后再试（限流/熔断保护），id: 1
31 当前请求过于频繁，请稍后再试（限流/熔断保护），id: 1
32 当前请求过于频繁，请稍后再试（限流/熔断保护），id: 1
33
```

## 5.引入nacos持久化

### 1.pom

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <project xmlns="http://maven.apache.org/POM/4.0.0"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
3     xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
   http://maven.apache.org/xsd/maven-4.0.0.xsd">
4     <modelVersion>4.0.0</modelVersion>
5     <!-- Spring Boot 父工程 -->
6     <parent>
7         <groupId>org.springframework.boot</groupId>
8         <artifactId>spring-boot-starter-parent</artifactId>
9         <version>2.7.5</version>
10        <relativePath/> <!-- lookup parent from repository -->
```

```
11 </parent>
12 <groupId>com.example.study</groupId>
13 <artifactId>springcloudalibaba-sentinel</artifactId>
14 <version>0.0.1-SNAPSHOT</version>
15 <name>springcloudalibaba-sentinel</name>
16 <description>Demo project for Spring Boot</description>
17
18
19 <properties>
20     <java.version>1.8</java.version> <!-- 指定 JDK 8 -->
21     <spring-cloud.version>2021.0.5</spring-cloud.version>
22 </properties>
23
24 <!-- 依赖版本管理（锁定 Spring Cloud 和 Spring Cloud Alibaba 版本） -->
25 <dependencyManagement>
26     <dependencies>
27         <!-- Spring Cloud 依赖管理 -->
28         <dependency>
29             <groupId>org.springframework.cloud</groupId>
30             <artifactId>spring-cloud-dependencies</artifactId>
31             <version>${spring-cloud.version}</version>
32             <type>pom</type>
33             <scope>import</scope>
34         </dependency>
35         <!-- Spring Cloud Alibaba 依赖管理（内置 Sentinel 1.8.6） -->
36         <dependency>
37             <groupId>com.alibaba.cloud</groupId>
38             <artifactId>spring-cloud-alibaba-dependencies</artifactId>
39             <version>2.1.0.RELEASE</version>
40             <type>pom</type>
41             <scope>import</scope>
42         </dependency>
43     </dependencies>
44 </dependencyManagement>
45
46 <dependencies>
47     <!-- Spring Boot web 依赖（提供 web 环境） -->
48     <dependency>
49         <groupId>org.springframework.boot</groupId>
50         <artifactId>spring-boot-starter-web</artifactId>
51     </dependency>
52
53     <!-- https://mvnrepository.com/artifact/com.alibaba.cloud/spring-
54 cloud-starter-alibaba-sentinel -->
55     <dependency>
56         <groupId>com.alibaba.cloud</groupId>
57         <artifactId>spring-cloud-starter-alibaba-sentinel</artifactId>
58         <version>2.2.8.RELEASE</version>
59         <scope>compile</scope>
60     </dependency>
61     <dependency>
62         <groupId>junit</groupId>
63         <artifactId>junit</artifactId>
64         <scope>test</scope>
65     </dependency>
66
67     <!-- 5. Spring Boot 测试依赖（可选，用于接口测试） -->
68     <dependency>
```

```

68         <groupId>org.springframework.boot</groupId>
69         <artifactId>spring-boot-starter-test</artifactId>
70         <scope>test</scope>
71     </dependency>
72
73     <!-- Sentinel 规则持久化到 Nacos 的核心适配依赖 -->
74     <dependency>
75         <groupId>com.alibaba.csp</groupId>
76         <artifactId>sentinel-datasource-nacos</artifactId>
77     </dependency>
78
79     <!-- 确保已存在 Nacos 核心依赖（服务注册+配置中心），无需重复添加 -->
80     <dependency>
81         <groupId>com.alibaba.cloud</groupId>
82         <artifactId>spring-cloud-starter-alibaba-nacos-
discovery</artifactId>
83     </dependency>
84     <dependency>
85         <groupId>com.alibaba.cloud</groupId>
86         <artifactId>spring-cloud-starter-alibaba-nacos-
config</artifactId>
87     </dependency>
88 </dependencies>
89
90 <build>
91     <plugins>
92         <plugin>
93             <groupId>org.springframework.boot</groupId>
94             <artifactId>spring-boot-maven-plugin</artifactId>
95             <version>2.7.5</version>
96         </plugin>
97     </plugins>
98 </build>
99
100 </project>
101

```

## 2.yaml

```

1  myNamespace: a5bda9d9-cde2-44ec-a436-36f156bc5b5b
2  server:
3      port: 7000
4  spring:
5      config:
6          import: optional:nacos:${spring.application.name}.yaml
7      application:
8          name: springcloudalibaba-sentinel # 应用名称，用于拼接 Nacos 配置 ID
9      cloud:
10         # Nacos 基础配置（服务地址）
11         nacos:
12             discovery:
13                 server-addr: localhost:8848 # Nacos 服务地址（本地部署，若为集群填写多个地
址，用逗号分隔）
14                 namespace: ${myNamespace}
15                 config:

```

```

16     server-addr: localhost:8848 #nacos作为配置中心地址
17     file-extension: yaml #指定yaml格式的配置
18     group: DEFAULT_GROUP #指定分组
19     namespace: ${myNamespace} #指定spacename
20 sentinel:
21     transport:
22         dashboard: localhost:8080 # Sentinel 控制台地址
23         port: 8719 # 应用与控制台通信端口
24         app-name: ${spring.application.name}
25     eager: true # 开启非懒加载，项目启动直接注册到控制台
26     web-context-unify: true # 统一 web 上下文，适配 Spring Boot 2.7.5 web 环
境
27     # 核心：配置 Sentinel 多规则类型的 Nacos 数据源（持久化核心配置）
28     datasource:
29         # 1. 流量控制规则（flow）- 自定义数据源名称（可任意命名，如 flow、sentinel-
flow）
30         flow:
31             nacos:
32                 server-addr: localhost:8848 # Nacos 服务地址（与上方一致）
33                 data-id: ${spring.application.name}-sentinel-flow-rules # 配置
ID（唯一标识）
34                 group-id: DEFAULT_GROUP # 配置分组（默认 DEFAULT_GROUP，可自定义）
35                 data-type: json # 规则数据格式（必须为 JSON，Sentinel 仅支持 JSON 解
析）
36                 rule-type: flow # 规则类型（flow=限流，degrade=熔断，authority=授
权，system=系统）
37                 namespace: ${myNamespace} # 新增：Sentinel 数据源命名空间（与上方一
致）
38         # 2. 熔断降级规则（degrade）- 可选，按需配置
39         degrade:
40             nacos:
41                 server-addr: localhost:8848
42                 data-id: ${spring.application.name}-sentinel-degrade-rules
43                 group-id: DEFAULT_GROUP
44                 data-type: json
45                 rule-type: degrade
46                 namespace: ${myNamespace} # 新增：Sentinel 数据源命名空间（与上方一
致）
47         # 3. 其他规则（授权、系统）- 按需添加，配置格式同上
48         authority:
49             nacos:
50                 server-addr: localhost:8848
51                 data-id: ${spring.application.name}-sentinel-authority-rules
52                 group-id: DEFAULT_GROUP
53                 data-type: json
54                 rule-type: authority
55                 namespace: ${myNamespace} # 新增：Sentinel 数据源命名空间（与上方一
致）
56         # 4. 系统规则（system）- 全局应用保护（CPU/负载/QPS 等）
57         system:
58             nacos:
59                 server-addr: localhost:8848
60                 data-id: ${spring.application.name}-sentinel-system-rules
61                 group-id: DEFAULT_GROUP
62                 data-type: json
63                 rule-type: system
64                 namespace: ${myNamespace} # 新增：Sentinel 数据源命名空间（与上方一
致）

```

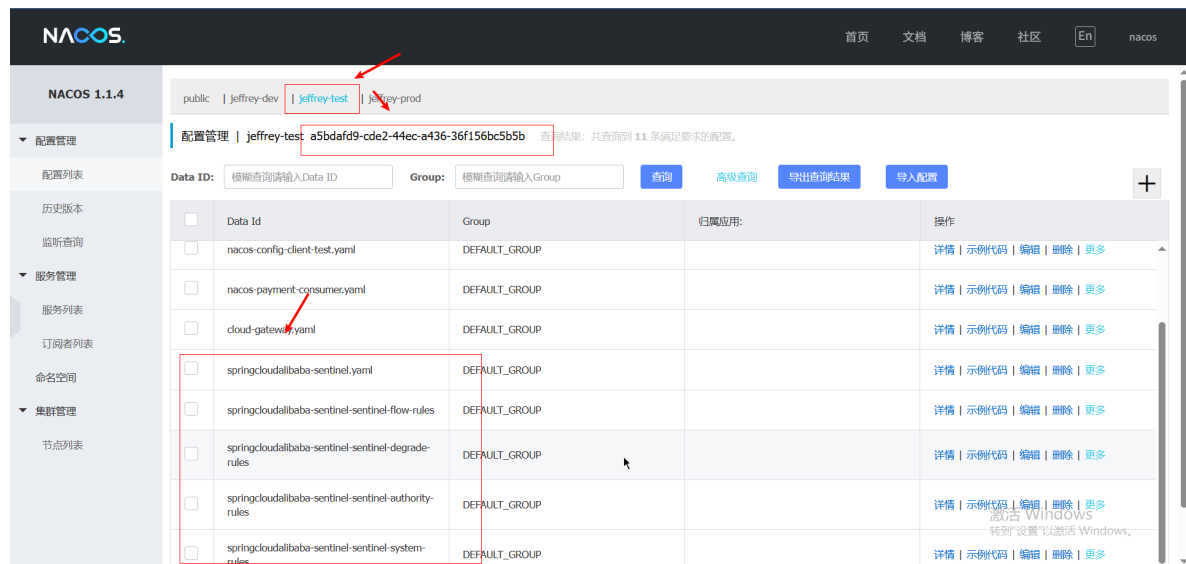


```

65 # 5. 热点参数规则 (param-flow) - 针对接口热点参数的精准限流
66 param-flow:
67     nacos:
68         server-addr: localhost:8848
69         data-id: ${spring.application.name}-sentinel-param-flow-rules
70         group-id: DEFAULT_GROUP
71         data-type: json
72         rule-type: param-flow
73         namespace: ${myNamespace} # 新增: Sentinel 数据源命名空间 (与上方一
致)
74
75
76
77
78
79

```

### 3.nacos



### 3.常见测试

```

1  nacos中的flow规则
2
3  [
4      {
5          "resource": "/threshold/qps",
6          "limitApp": "default",
7          "grade": 1,
8          "count": 2,
9          "strategy": 0,
10         "controlBehavior": 0,
11         "clusterMode": false
12     },
13     {
14         "resource": "/threshold/thread",
15         "limitApp": "default",
16         "grade": 0,

```

```
17     "count": 3,
18     "strategy": 0,
19     "controlBehavior": 0,
20     "clusterMode": false
21 },
22 {
23     "resource": "helloResource",
24     "limitApp": "default",
25     "grade": 1,
26     "count": 2,
27     "strategy": 0,
28     "controlBehavior": 0,
29     "clusterMode": false
30 },
31 {
32     "resource": "/flowControlEffect/warmUp",
33     "limitApp": "default",
34     "grade": 1,
35     "count": 4,
36     "strategy": 0,
37     "controlBehavior": 0,
38     "clusterMode": false
39 },
40 {
41     "resource": "/flowControlEffect/queueWait",
42     "limitApp": "default",
43     "grade": 1,
44     "count": 5,
45     "strategy": 0,
46     "controlBehavior": 0,
47     "clusterMode": false
48 },
49 {
50     "resource": "/flowControlEffect/failFast",
51     "limitApp": "default",
52     "grade": 1,
53     "count": 3,
54     "strategy": 0,
55     "controlBehavior": 0,
56     "clusterMode": false
57 },
58 {
59     "resource": "/flowControlMode/direct",
60     "limitApp": "default",
61     "grade": 1,
62     "count": 2,
63     "strategy": 0,
64     "controlBehavior": 0,
65     "clusterMode": false
66 },
67 {
68     "resource": "/flowControlMode/association",
69     "limitApp": "default",
70     "grade": 1,
71     "count": 3,
72     "strategy": 1,
73     "refResource": "/flowControlMode/associationRef",
74     "controlBehavior": 0,
```

```

75     "clusterMode": false
76   },
77   {
78     "resource": "serviceLink",
79     "limitApp": "default",
80     "grade": 1,
81     "count": 2,
82     "strategy": 2,
83     "refResource": "/flowControlMode/linkEnter",
84     "controlBehavior": 0,
85     "clusterMode": false
86   },
87   {
88     "resource": "flowControlMode/linkController",
89     "limitApp": "default",
90     "grade": 1,
91     "count": 2,
92     "strategy": 2,
93     "refResource": "/flowControlMode/linkEnterController",
94     "controlBehavior": 0,
95     "clusterMode": false
96   }
97 ]
98 ]

```

## 1.threshold

### 1.qps

```

1  【qps-成功】
2  【qps-成功】
3  响应结果: {"code":429,"errorType":"FLOW_LIMITING","allRequestParams":
  {}, "requestMethod":"GET", "requestPath":"/threshold/qps", "errorMsg":"请求过于
  频繁, 触发流量控制 (限流)", "timestamp":1768132425315}
4  响应结果: {"code":429,"errorType":"FLOW_LIMITING","allRequestParams":
  {}, "requestMethod":"GET", "requestPath":"/threshold/qps", "errorMsg":"请求过于
  频繁, 触发流量控制 (限流)", "timestamp":1768132425369}
5  响应结果: {"code":429,"errorType":"FLOW_LIMITING","allRequestParams":
  {}, "requestMethod":"GET", "requestPath":"/threshold/qps", "errorMsg":"请求过于
  频繁, 触发流量控制 (限流)", "timestamp":1768132425472}
6  响应结果: {"code":429,"errorType":"FLOW_LIMITING","allRequestParams":
  {}, "requestMethod":"GET", "requestPath":"/threshold/qps", "errorMsg":"请求过于
  频繁, 触发流量控制 (限流)", "timestamp":1768132425570}
7  响应结果: {"code":429,"errorType":"FLOW_LIMITING","allRequestParams":
  {}, "requestMethod":"GET", "requestPath":"/threshold/qps", "errorMsg":"请求过于
  频繁, 触发流量控制 (限流)", "timestamp":1768132425670}
8  响应结果: {"code":429,"errorType":"FLOW_LIMITING","allRequestParams":
  {}, "requestMethod":"GET", "requestPath":"/threshold/qps", "errorMsg":"请求过于
  频繁, 触发流量控制 (限流)", "timestamp":1768132425769}
9  响应结果: {"code":429,"errorType":"FLOW_LIMITING","allRequestParams":
  {}, "requestMethod":"GET", "requestPath":"/threshold/qps", "errorMsg":"请求过于
  频繁, 触发流量控制 (限流)", "timestamp":1768132425870}
10 响应结果: {"code":429,"errorType":"FLOW_LIMITING","allRequestParams":
  {}, "requestMethod":"GET", "requestPath":"/threshold/qps", "errorMsg":"请求过于
  频繁, 触发流量控制 (限流)", "timestamp":1768132425969}

```

```
11 【qps-成功】
12 【qps-成功】
13 响应结果: {"code":429,"errorType":"FLOW_LIMITING","allRequestParams":
    {"requestMethod":"GET","requestPath":"/threshold/qps","errorMsg":"请求过于
    频繁, 触发流量控制(限流)","timestamp":1768132426269}
14
```

## 2.thread

```
1 使用单个线程发送很多请求成功才是正常的
2  【thread-成功】
3  【thread-成功】
4  【thread-成功】
5  【thread-成功】
6  【thread-成功】
7  【thread-成功】
8  【thread-成功】
9  【thread-成功】
10 【thread-成功】
11 【thread-成功】
12 【thread-成功】
13 【thread-成功】
14 【thread-成功】
15 【thread-成功】
16 【thread-成功】
17 【thread-成功】
18 【thread-成功】
19 【thread-成功】
20 【thread-成功】
21
22 我们要的是多个线程的并发失败也是失败的: 如果没有测试到预期结果, 请调大线程并发量, 我自己一
    开始40, 结果看不到, 调整到500才成功
23
24 【thread-成功】
25 【thread-成功】
26 【thread-成功】
27 【thread-成功】
28 【thread-成功】
29 【thread-成功】
30 【thread-成功】
31 响应结果: {"code":429,"errorType":"FLOW_LIMITING","allRequestParams":
    {"requestMethod":"GET","requestPath":"/threshold/thread","errorMsg":"请求
    过于频繁, 触发流量控制(限流)","timestamp":1768132719847}
32 【thread-成功】
33 【thread-成功】
34 响应结果: {"code":429,"errorType":"FLOW_LIMITING","allRequestParams":
    {"requestMethod":"GET","requestPath":"/threshold/thread","errorMsg":"请求
    过于频繁, 触发流量控制(限流)","timestamp":1768132719851}
35 响应结果: {"code":429,"errorType":"FLOW_LIMITING","allRequestParams":
    {"requestMethod":"GET","requestPath":"/threshold/thread","errorMsg":"请求
    过于频繁, 触发流量控制(限流)","timestamp":1768132719851}
36 响应结果: {"code":429,"errorType":"FLOW_LIMITING","allRequestParams":
    {"requestMethod":"GET","requestPath":"/threshold/thread","errorMsg":"请求
    过于频繁, 触发流量控制(限流)","timestamp":1768132719852}
```

```
37  响应结果: {"code":429,"errorType":"FLOW_LIMITING","allRequestParams":
    {}, "requestMethod":"GET", "requestPath":"/threshold/thread", "errorMsg":"请求
    过于频繁, 触发流量控制 (限流)", "timestamp":1768132719853}
38  【thread-成功】
39  响应结果: {"code":429,"errorType":"FLOW_LIMITING","allRequestParams":
    {}, "requestMethod":"GET", "requestPath":"/threshold/thread", "errorMsg":"请求
    过于频繁, 触发流量控制 (限流)", "timestamp":1768132719854}
40  【thread-成功】
41  响应结果: {"code":429,"errorType":"FLOW_LIMITING","allRequestParams":
    {}, "requestMethod":"GET", "requestPath":"/threshold/thread", "errorMsg":"请求
    过于频繁, 触发流量控制 (限流)", "timestamp":1768132719854}
42  【thread-成功】
43  【thread-成功】
44  【thread-成功】
45  【thread-成功】
```

## 2.flowControlEffect

### 1.failfast

```
1  【failFast-成功】
2  【failFast-成功】
3  【failFast-成功】
4  响应结果: {"code":429,"errorType":"FLOW_LIMITING","allRequestParams":
    {}, "requestMethod":"GET", "requestPath":"/flowControlEffect/failFast", "error
    Msg":"请求过于频繁, 触发流量控制 (限流)", "timestamp":1768133313578}
5  响应结果: {"code":429,"errorType":"FLOW_LIMITING","allRequestParams":
    {}, "requestMethod":"GET", "requestPath":"/flowControlEffect/failFast", "error
    Msg":"请求过于频繁, 触发流量控制 (限流)", "timestamp":1768133313682}
6  响应结果: {"code":429,"errorType":"FLOW_LIMITING","allRequestParams":
    {}, "requestMethod":"GET", "requestPath":"/flowControlEffect/failFast", "error
    Msg":"请求过于频繁, 触发流量控制 (限流)", "timestamp":1768133313781}
7  响应结果: {"code":429,"errorType":"FLOW_LIMITING","allRequestParams":
    {}, "requestMethod":"GET", "requestPath":"/flowControlEffect/failFast", "error
    Msg":"请求过于频繁, 触发流量控制 (限流)", "timestamp":1768133313878}
8  响应结果: {"code":429,"errorType":"FLOW_LIMITING","allRequestParams":
    {}, "requestMethod":"GET", "requestPath":"/flowControlEffect/failFast", "error
    Msg":"请求过于频繁, 触发流量控制 (限流)", "timestamp":1768133313980}
9  【failFast-成功】
10 【failFast-成功】
11 【failFast-成功】
12 响应结果: {"code":429,"errorType":"FLOW_LIMITING","allRequestParams":
    {}, "requestMethod":"GET", "requestPath":"/flowControlEffect/failFast", "error
    Msg":"请求过于频繁, 触发流量控制 (限流)", "timestamp":1768133314379}
13 响应结果: {"code":429,"errorType":"FLOW_LIMITING","allRequestParams":
    {}, "requestMethod":"GET", "requestPath":"/flowControlEffect/failFast", "error
    Msg":"请求过于频繁, 触发流量控制 (限流)", "timestamp":1768133314479}
```

### 2.warmUp

```
1  【warmUp-成功】
2  【warmUp-成功】
```

```

3  【warmUp-成功】
4  【warmUp-成功】
5  响应结果: {"code":429,"errorType":"FLOW_LIMITING","allRequestParams":
    {"requestMethod":"GET","requestPath":"/flowControlEffect/warmUp","errorMs
    g":"请求过于频繁,触发流量控制(限流)","timestamp":1768133392367}
6  响应结果: {"code":429,"errorType":"FLOW_LIMITING","allRequestParams":
    {"requestMethod":"GET","requestPath":"/flowControlEffect/warmUp","errorMs
    g":"请求过于频繁,触发流量控制(限流)","timestamp":1768133392467}
7  【warmUp-成功】
8  响应结果: {"code":429,"errorType":"FLOW_LIMITING","allRequestParams":
    {"requestMethod":"GET","requestPath":"/flowControlEffect/warmUp","errorMs
    g":"请求过于频繁,触发流量控制(限流)","timestamp":1768133392668}
9  响应结果: {"code":429,"errorType":"FLOW_LIMITING","allRequestParams":
    {"requestMethod":"GET","requestPath":"/flowControlEffect/warmUp","errorMs
    g":"请求过于频繁,触发流量控制(限流)","timestamp":1768133392766}
10 响应结果: {"code":429,"errorType":"FLOW_LIMITING","allRequestParams":
    {"requestMethod":"GET","requestPath":"/flowControlEffect/warmUp","errorMs
    g":"请求过于频繁,触发流量控制(限流)","timestamp":1768133392867}
11 响应结果: {"code":429,"errorType":"FLOW_LIMITING","allRequestParams":
    {"requestMethod":"GET","requestPath":"/flowControlEffect/warmUp","errorMs
    g":"请求过于频繁,触发流量控制(限流)","timestamp":1768133392967}
12  【warmUp-成功】
13  【warmUp-成功】
14  【warmUp-成功】
15  响应结果: {"code":429,"errorType":"FLOW_LIMITING","allRequestParams":
    {"requestMethod":"GET","requestPath":"/flowControlEffect/warmUp","errorMs
    g":"请求过于频繁,触发流量控制(限流)","timestamp":1768133393368}

```

### 3.queueWait

```

1  【queueWait-成功】
2  【queueWait-成功】
3  【queueWait-成功】
4  【queueWait-成功】
5  【queueWait-成功】
6  响应结果: {"code":429,"errorType":"FLOW_LIMITING","allRequestParams":
    {"requestMethod":"GET","requestPath":"/flowControlEffect/queueWait","erro
    rMsg":"请求过于频繁,触发流量控制(限流)","timestamp":1768133472805}
7  响应结果: {"code":429,"errorType":"FLOW_LIMITING","allRequestParams":
    {"requestMethod":"GET","requestPath":"/flowControlEffect/queueWait","erro
    rMsg":"请求过于频繁,触发流量控制(限流)","timestamp":1768133472904}
8  【queueWait-成功】
9  【queueWait-成功】
10 响应结果: {"code":429,"errorType":"FLOW_LIMITING","allRequestParams":
    {"requestMethod":"GET","requestPath":"/flowControlEffect/queueWait","erro
    rMsg":"请求过于频繁,触发流量控制(限流)","timestamp":1768133473203}
11 响应结果: {"code":429,"errorType":"FLOW_LIMITING","allRequestParams":
    {"requestMethod":"GET","requestPath":"/flowControlEffect/queueWait","erro
    rMsg":"请求过于频繁,触发流量控制(限流)","timestamp":1768133473304}
12 响应结果: {"code":429,"errorType":"FLOW_LIMITING","allRequestParams":
    {"requestMethod":"GET","requestPath":"/flowControlEffect/queueWait","erro
    rMsg":"请求过于频繁,触发流量控制(限流)","timestamp":1768133473404}
13  【queueWait-成功】
14  【queueWait-成功】
15  【queueWait-成功】

```

## 3.flowControlMode

### 1.direct

```

1  【direct-成功】
2  【direct-成功】
3  响应结果: {"code":429,"errorType":"FLOW_LIMITING","allRequestParams":
    {}, "requestMethod":"GET", "requestPath":"/flowControlMode/direct", "errorMsg"
    : "请求过于频繁, 触发流量控制 (限流)", "timestamp":1768134396235}
4  响应结果: {"code":429,"errorType":"FLOW_LIMITING","allRequestParams":
    {}, "requestMethod":"GET", "requestPath":"/flowControlMode/direct", "errorMsg"
    : "请求过于频繁, 触发流量控制 (限流)", "timestamp":1768134396305}
5  响应结果: {"code":429,"errorType":"FLOW_LIMITING","allRequestParams":
    {}, "requestMethod":"GET", "requestPath":"/flowControlMode/direct", "errorMsg"
    : "请求过于频繁, 触发流量控制 (限流)", "timestamp":1768134396403}
6  响应结果: {"code":429,"errorType":"FLOW_LIMITING","allRequestParams":
    {}, "requestMethod":"GET", "requestPath":"/flowControlMode/direct", "errorMsg"
    : "请求过于频繁, 触发流量控制 (限流)", "timestamp":1768134396503}
7  响应结果: {"code":429,"errorType":"FLOW_LIMITING","allRequestParams":
    {}, "requestMethod":"GET", "requestPath":"/flowControlMode/direct", "errorMsg"
    : "请求过于频繁, 触发流量控制 (限流)", "timestamp":1768134396604}
8  响应结果: {"code":429,"errorType":"FLOW_LIMITING","allRequestParams":
    {}, "requestMethod":"GET", "requestPath":"/flowControlMode/direct", "errorMsg"
    : "请求过于频繁, 触发流量控制 (限流)", "timestamp":1768134396704}
9  响应结果: {"code":429,"errorType":"FLOW_LIMITING","allRequestParams":
    {}, "requestMethod":"GET", "requestPath":"/flowControlMode/direct", "errorMsg"
    : "请求过于频繁, 触发流量控制 (限流)", "timestamp":1768134396803}
10 响应结果: {"code":429,"errorType":"FLOW_LIMITING","allRequestParams":
    {}, "requestMethod":"GET", "requestPath":"/flowControlMode/direct", "errorMsg"
    : "请求过于频繁, 触发流量控制 (限流)", "timestamp":1768134396904}
11  【direct-成功】
12  【direct-成功】
13  响应结果: {"code":429,"errorType":"FLOW_LIMITING","allRequestParams":
    {}, "requestMethod":"GET", "requestPath":"/flowControlMode/direct", "errorMsg"
    : "请求过于频繁, 触发流量控制 (限流)", "timestamp":1768134397204}
14  响应结果: {"code":429,"errorType":"FLOW_LIMITING","allRequestParams":
    {}, "requestMethod":"GET", "requestPath":"/flowControlMode/direct", "errorMsg"
    : "请求过于频繁, 触发流量控制 (限流)", "timestamp":1768134397307}

```

### 2.association

```

1  注意点
2  ref的如果也加了规则, 被限流了, 就可能不命中; 我自己之前就是这样, 然后单独用了个没有规则
   的, 让ref能正常并发不被限流
3
4  【association-成功】
5  【associationRef-成功】
6  【association-成功】
7  【associationRef-成功】
8  【associationRef-成功】
9  【associationRef-成功】

```

```
10  响应结果: {"code":429,"errorType":"FLOW_LIMITING","allRequestParams":
    {}, "requestMethod":"GET", "requestPath":"/flowControlMode/association", "erro
    rMsg":"请求过于频繁, 触发流量控制 (限流)", "timestamp":1768135895488}
11  【associationRef-成功】
12  响应结果: {"code":429,"errorType":"FLOW_LIMITING","allRequestParams":
    {}, "requestMethod":"GET", "requestPath":"/flowControlMode/association", "erro
    rMsg":"请求过于频繁, 触发流量控制 (限流)", "timestamp":1768135895553}
13  【associationRef-成功】
14  【associationRef-成功】
```

### 3.link

```
1  单独调用link, 只有成功的才对
2  【service-link-成功】
3  【service-link-成功】
4  【service-link-成功】
5  【service-link-成功】
6  【service-link-成功】
7  【service-link-成功】
8  【service-link-成功】
9  【service-link-成功】
10 【service-link-成功】
11 【service-link-成功】
12 【service-link-成功】
13 【service-link-成功】
14 【service-link-成功】
15 【service-link-成功】
16 【service-link-成功】
17 【service-link-成功】
18
19
20  调用linkEnter
21
22 linkEnter- 【service-link-成功】
23 linkEnter- 【service-link-成功】
24 linkEnter- 【service-link-降级】
25 linkEnter- 【service-link-降级】
26 linkEnter- 【service-link-降级】
27 linkEnter- 【service-link-降级】
28 linkEnter- 【service-link-降级】
29 linkEnter- 【service-link-降级】
30 linkEnter- 【service-link-降级】
31 linkEnter- 【service-link-降级】
32 linkEnter- 【service-link-降级】
33 linkEnter- 【service-link-降级】
34 linkEnter- 【service-link-降级】
35 linkEnter- 【service-link-成功】
36 linkEnter- 【service-link-成功】
37 linkEnter- 【service-link-降级】
38 linkEnter- 【service-link-降级】
39 linkEnter- 【service-link-降级】
40 linkEnter- 【service-link-降级】
41 linkEnter- 【service-link-降级】
42 linkEnter- 【service-link-降级】
43
```



44  
45 框架系统bug  
46 流控模式为链路的，被访问的只能是非controller层的，且需要手动编码（我用的是注解）进行资源命名，才有效

## 4.hotParam

```
1  nacos中的规则
2
3  [
4    {
5      "resource": "/hotParam/single",
6      "grade": 1,
7      "count": 2,
8      "paramIdx": 0,
9      "durationInSec": 1,
10     "controlBehavior": 0,
11     "clusterMode": false,
12     "paramFlowItemList": []
13   },
14   {
15     "resource": "/hotParam/singlewithItem",
16     "grade": 1,
17     "count": 2,
18     "paramIdx": 0,
19     "durationInSec": 1,
20     "controlBehavior": 0,
21     "clusterMode": false,
22     "paramFlowItemList": [
23       {
24         "object": "999",
25         "classType": "java.lang.String",
26         "count": 200
27       }
28     ]
29   },
30   {
31     "resource": "/hotParam/mutil",
32     "grade": 1,
33     "count": 2,
34     "paramIdx": 0,
35     "durationInSec": 1,
36     "controlBehavior": 0,
37     "clusterMode": false,
38     "paramFlowItemList": [],
39     "additionalParamIdxList": []
40   },
41   {
42     "resource": "/hotParam/mutil",
43     "grade": 1,
44     "count": 1,
45     "paramIdx": 1,
46     "durationInSec": 1,
47     "controlBehavior": 0,
48     "clusterMode": false,
49     "paramFlowItemList": [],
50     "additionalParamIdxList": []
```

```

51 },
52 {
53   "resource": "/hotParam/mutilWithItem",
54   "grade": 1,
55   "count": 2,
56   "paramIdx": 0,
57   "durationInSec": 1,
58   "controlBehavior": 0,
59   "clusterMode": false,
60   "paramFlowItemList": [
61     {
62       "object": "999",
63       "classType": "java.lang.String",
64       "count": 200
65     }
66   ]
67 },
68 {
69   "resource": "/hotParam/mutilWithItem",
70   "grade": 1,
71   "count": 1,
72   "paramIdx": 1,
73   "durationInSec": 1,
74   "controlBehavior": 0,
75   "clusterMode": false,
76   "paramFlowItemList": [
77     {
78       "object": "jeffrey",
79       "classType": "java.lang.String",
80       "count": 200
81     }
82   ]
83 }
84
85 ]

```

## 1.single

```

1  【single-成功】-id:1001
2  【single-成功】-id:1001
3  服务器内部错误[UndeclaredThrowableException]: {"code":429,"msg":"热点参数限流限制，请求过于频繁","data":null}
4  服务器内部错误[UndeclaredThrowableException]: {"code":429,"msg":"热点参数限流限制，请求过于频繁","data":null}
5  服务器内部错误[UndeclaredThrowableException]: {"code":429,"msg":"热点参数限流限制，请求过于频繁","data":null}
6  服务器内部错误[UndeclaredThrowableException]: {"code":429,"msg":"热点参数限流限制，请求过于频繁","data":null}
7  服务器内部错误[UndeclaredThrowableException]: {"code":429,"msg":"热点参数限流限制，请求过于频繁","data":null}
8  服务器内部错误[UndeclaredThrowableException]: {"code":429,"msg":"热点参数限流限制，请求过于频繁","data":null}
9  服务器内部错误[UndeclaredThrowableException]: {"code":429,"msg":"热点参数限流限制，请求过于频繁","data":null}
10 服务器内部错误[UndeclaredThrowableException]: {"code":429,"msg":"热点参数限流限制，请求过于频繁","data":null}

```

```
11 服务器内部错误[UndeclaredThrowableException]: {"code":429,"msg":"热点参数限流限制, 请求过于频繁","data":null}
12 【single-成功】-id:1001
13 【single-成功】-id:1001
14 服务器内部错误[UndeclaredThrowableException]: {"code":429,"msg":"热点参数限流限制, 请求过于频繁","data":null}
15 服务器内部错误[UndeclaredThrowableException]: {"code":429,"msg":"热点参数限流限制, 请求过于频繁","data":null}
16 服务器内部错误[UndeclaredThrowableException]: {"code":429,"msg":"热点参数限流限制, 请求过于频繁","data":null}
```

## 2.singleWithItem

```
1  【singleWithItem-成功】-id:999
2  【singleWithItem-成功】-id:999
3  【singleWithItem-成功】-id:999
4  【singleWithItem-成功】-id:999
5  【singleWithItem-成功】-id:999
6  【singleWithItem-成功】-id:999
7  【singleWithItem-成功】-id:999
8  【singleWithItem-成功】-id:999
9  【singleWithItem-成功】-id:999
10 【singleWithItem-成功】-id:999
```

## 3.mutil

```
1  【mutil-成功】-id:1001name:luck
2  服务器内部错误[UndeclaredThrowableException]: {"code":429,"msg":"热点参数限流限制, 请求过于频繁","data":null}
3  服务器内部错误[UndeclaredThrowableException]: {"code":429,"msg":"热点参数限流限制, 请求过于频繁","data":null}
4  服务器内部错误[UndeclaredThrowableException]: {"code":429,"msg":"热点参数限流限制, 请求过于频繁","data":null}
5  服务器内部错误[UndeclaredThrowableException]: {"code":429,"msg":"热点参数限流限制, 请求过于频繁","data":null}
6  服务器内部错误[UndeclaredThrowableException]: {"code":429,"msg":"热点参数限流限制, 请求过于频繁","data":null}
7  服务器内部错误[UndeclaredThrowableException]: {"code":429,"msg":"热点参数限流限制, 请求过于频繁","data":null}
8  服务器内部错误[UndeclaredThrowableException]: {"code":429,"msg":"热点参数限流限制, 请求过于频繁","data":null}
9  服务器内部错误[UndeclaredThrowableException]: {"code":429,"msg":"热点参数限流限制, 请求过于频繁","data":null}
10 服务器内部错误[UndeclaredThrowableException]: {"code":429,"msg":"热点参数限流限制, 请求过于频繁","data":null}
11 服务器内部错误[UndeclaredThrowableException]: {"code":429,"msg":"热点参数限流限制, 请求过于频繁","data":null}
12 【mutil-成功】-id:1001name:luck
13 服务器内部错误[UndeclaredThrowableException]: {"code":429,"msg":"热点参数限流限制, 请求过于频繁","data":null}
14 服务器内部错误[UndeclaredThrowableException]: {"code":429,"msg":"热点参数限流限制, 请求过于频繁","data":null}
15 服务器内部错误[UndeclaredThrowableException]: {"code":429,"msg":"热点参数限流限制, 请求过于频繁","data":null}
```

```
16 服务器内部错误[UndeclaredThrowableException]: {"code":429,"msg":"热点参数限流限制，请求过于频繁","data":null}
```

## 4.mutilWithItem

```
1  【mutilWithItem-成功】-id:999name:jeffrey
2  【mutilWithItem-成功】-id:999name:jeffrey
3  【mutilWithItem-成功】-id:999name:jeffrey
4  【mutilWithItem-成功】-id:999name:jeffrey
5  【mutilWithItem-成功】-id:999name:jeffrey
6  【mutilWithItem-成功】-id:999name:jeffrey
7  【mutilWithItem-成功】-id:999name:jeffrey
8  【mutilWithItem-成功】-id:999name:jeffrey
9  【mutilWithItem-成功】-id:999name:jeffrey
10 【mutilWithItem-成功】-id:999name:jeffrey
11 【mutilWithItem-成功】-id:999name:jeffrey
12 【mutilWithItem-成功】-id:999name:jeffrey
13 【mutilWithItem-成功】-id:999name:jeffrey
14 【mutilWithItem-成功】-id:999name:jeffrey
15 【mutilWithItem-成功】-id:999name:jeffrey
16 【mutilWithItem-成功】-id:999name:jeffrey
17
```

## 5.authority

```
1  nacos authority的json格式的配置
2
3  [
4    {
5      "resource": "/authority/whiteList",
6      "limitApp": "inner-app,special-ip",
7      "strategy": 0
8    },
9    {
10     "resource": "/authority/blackList",
11     "limitApp": "black-app,spec-black-ip",
12     "strategy": 1
13   }
14 ]
```

## 1.whiteList

```
1  1.白名单内的
2  【whiteList-成功】,X-Sentinel-App:inner-app
3  【whiteList-成功】,X-Sentinel-App:inner-app
4  【whiteList-成功】,X-Sentinel-App:inner-app
5  【whiteList-成功】,X-Sentinel-App:inner-app
6  【whiteList-成功】,X-Sentinel-App:inner-app
7  【whiteList-成功】,X-Sentinel-App:inner-app
8  【whiteList-成功】,X-Sentinel-App:inner-app
9  【whiteList-成功】,X-Sentinel-App:inner-app
```

```
10 【whiteList-成功】,X-Sentinel-App:inner-app
11 【whiteList-成功】,X-Sentinel-App:inner-app
12 【whiteList-成功】,X-Sentinel-App:inner-app
13
14 2.白名单外的
15 响应结果: {"code":403,"errorType":"AUTHORITY_DENY","allRequestParams":
{"id":"1001"},"requestMethod":"GET","requestPath":"/authority/whiteList","e
rrorMsg":"请求被拒绝,不符合授权规则(黑白名单限制)","timestamp":1768150430620}
16 响应结果: {"code":403,"errorType":"AUTHORITY_DENY","allRequestParams":
{"id":"1001"},"requestMethod":"GET","requestPath":"/authority/whiteList","e
rrorMsg":"请求被拒绝,不符合授权规则(黑白名单限制)","timestamp":1768150430659}
17 响应结果: {"code":403,"errorType":"AUTHORITY_DENY","allRequestParams":
{"id":"1001"},"requestMethod":"GET","requestPath":"/authority/whiteList","e
rrorMsg":"请求被拒绝,不符合授权规则(黑白名单限制)","timestamp":1768150430759}
18 响应结果: {"code":403,"errorType":"AUTHORITY_DENY","allRequestParams":
{"id":"1001"},"requestMethod":"GET","requestPath":"/authority/whiteList","e
rrorMsg":"请求被拒绝,不符合授权规则(黑白名单限制)","timestamp":1768150430859}
19 响应结果: {"code":403,"errorType":"AUTHORITY_DENY","allRequestParams":
{"id":"1001"},"requestMethod":"GET","requestPath":"/authority/whiteList","e
rrorMsg":"请求被拒绝,不符合授权规则(黑白名单限制)","timestamp":1768150430959}
20 响应结果: {"code":403,"errorType":"AUTHORITY_DENY","allRequestParams":
{"id":"1001"},"requestMethod":"GET","requestPath":"/authority/whiteList","e
rrorMsg":"请求被拒绝,不符合授权规则(黑白名单限制)","timestamp":1768150431059}
21 响应结果: {"code":403,"errorType":"AUTHORITY_DENY","allRequestParams":
{"id":"1001"},"requestMethod":"GET","requestPath":"/authority/whiteList","e
rrorMsg":"请求被拒绝,不符合授权规则(黑白名单限制)","timestamp":1768150431158}
22
```

## 2.blackList

```
1 1.黑名单内
2 响应结果: {"code":403,"errorType":"AUTHORITY_DENY","allRequestParams":
{"id":"1001"},"requestMethod":"GET","requestPath":"/authority/blackList","e
rrorMsg":"请求被拒绝,不符合授权规则(黑白名单限制)","timestamp":1768150196125}
3 响应结果: {"code":403,"errorType":"AUTHORITY_DENY","allRequestParams":
{"id":"1001"},"requestMethod":"GET","requestPath":"/authority/blackList","e
rrorMsg":"请求被拒绝,不符合授权规则(黑白名单限制)","timestamp":1768150196178}
4 响应结果: {"code":403,"errorType":"AUTHORITY_DENY","allRequestParams":
{"id":"1001"},"requestMethod":"GET","requestPath":"/authority/blackList","e
rrorMsg":"请求被拒绝,不符合授权规则(黑白名单限制)","timestamp":1768150196277}
5 响应结果: {"code":403,"errorType":"AUTHORITY_DENY","allRequestParams":
{"id":"1001"},"requestMethod":"GET","requestPath":"/authority/blackList","e
rrorMsg":"请求被拒绝,不符合授权规则(黑白名单限制)","timestamp":1768150196378}
6
7 2.黑名单外
8 【blackList-成功】,X-Sentinel-App:spec-black-ip11
9 【blackList-成功】,X-Sentinel-App:spec-black-ip11
10 【blackList-成功】,X-Sentinel-App:spec-black-ip11
11 【blackList-成功】,X-Sentinel-App:spec-black-ip11
12 【blackList-成功】,X-Sentinel-App:spec-black-ip11
13 【blackList-成功】,X-Sentinel-App:spec-black-ip11
14 【blackList-成功】,X-Sentinel-App:spec-black-ip11
15 【blackList-成功】,X-Sentinel-App:spec-black-ip11
16 【blackList-成功】,X-Sentinel-App:spec-black-ip11
17 【blackList-成功】,X-Sentinel-App:spec-black-ip11
```

