sentinel-springcloudalibaba

1.注意点

```
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.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.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.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.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

```xml
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 https://maven.apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>
    <!-- Spring Boot 父工程 -->
    <parent>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-parent</artifactId>
        <version>2.7.5</version>
        <relativePath/> <!-- lookup parent from repository -->
    </parent>
    <groupId>com.example.study</groupId>
    <artifactId>springcloudalibaba-sentinel</artifactId>
    <version>0.0.1-SNAPSHOT</version>
    <name>springcloudalibaba-sentinel</name>
    <description>Demo project for Spring Boot</description>


    <properties>
        <java.version>1.8</java.version> <!-- 指定 JDK 8 -->
        <spring-cloud.version>2021.0.5</spring-cloud.version>
    </properties>

    <!-- 依赖版本管理（锁定 Spring Cloud 和 Spring Cloud Alibaba 版本） -->
    <dependencyManagement>
        <dependencies>
            <!-- Spring Cloud 依赖管理 -->
            <dependency>
                <groupId>org.springframework.cloud</groupId>
                <artifactId>spring-cloud-dependencies</artifactId>
                <version>${spring-cloud.version}</version>
                <type>pom</type>
                <scope>import</scope>
            </dependency>
            <!-- Spring Cloud Alibaba 依赖管理（内置 Sentinel 1.8.6） -->
            <dependency>
                <groupId>com.alibaba.cloud</groupId>
                <artifactId>spring-cloud-alibaba-dependencies</artifactId>
                <version>2021.0.5.0</version>
                <type>pom</type>
                <scope>import</scope>
            </dependency>
        </dependencies>
    </dependencyManagement>

    <dependencies>
        <!-- Spring Boot Web 依赖（提供 Web 环境） -->
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-web</artifactId>
        </dependency>

        <!-- https://mvnrepository.com/artifact/com.alibaba.cloud/spring-cloud-starter-alibaba-sentinel -->
        <dependency>
            <groupId>com.alibaba.cloud</groupId>
            <artifactId>spring-cloud-starter-alibaba-sentinel</artifactId>
            <version>2.2.8.RELEASE</version>
            <scope>compile</scope>
        </dependency>
        <dependency>
            <groupId>junit</groupId>
            <artifactId>junit</artifactId>
            <scope>test</scope>
        </dependency>

        <!-- 5. Spring Boot 测试依赖（可选，用于接口测试） -->
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-test</artifactId>
            <scope>test</scope>
        </dependency>
    </dependencies>

    <build>
        <plugins>
            <plugin>
                <groupId>org.springframework.boot</groupId>
                <artifactId>spring-boot-maven-plugin</artifactId>
                <version>2.7.5</version>
            </plugin>
        </plugins>
    </build>

</project>

```

2.yam

```yaml

server:
  port: 7000 # 项目端口（避免与 Sentinel 控制台默认 8080 冲突）
spring:
  application:
    name: springcloudalibaba-sentinel # 应用名称（控制台显示）
  cloud:
    sentinel:
      transport:
        dashboard: localhost:8080 # Sentinel 控制台地址（匹配 1.8.6 版本）
        port: 8719 # 应用与控制台通信端口（默认，被占用自动递增）
      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）
      web-context-unify: true # 统一 Web 上下文，适配 Spring Boot 2.7.5 Web 环境




```

3.controller

```java
package com.example.study.springcloudalibabasentinel.controller;

import com.alibaba.csp.sentinel.annotation.SentinelResource;
import com.alibaba.csp.sentinel.slots.block.BlockException;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.RestController;

@RestController
public class HelloController {

    /**
     * 被 Sentinel 保护的接口资源（兼容 JDK 8 + Spring Boot 2.7.5）
     * @param id 路径参数
     * @return 响应结果
     */
    @GetMapping("/hello/{id}")
    @SentinelResource(
            value = "helloResource", // 资源名称（控制台标识）
            blockHandler = "helloBlockHandler", // 限流/熔断兜底方法
            fallback = "helloFallback" // 业务异常兜底方法
    )
    public String hello(@PathVariable Integer id) {
        // 模拟业务异常（JDK 8 支持的语法）
        if (id == 0) {
            throw new RuntimeException("id 不能为 0（JDK 8 兼容）");
        }
        String s = "Hello Sentinel 1.8.6 + Spring Boot 2.7.5, id: " + id;
        System.out.println(s);
        return s;
    }

    /**
     * 限流/熔断兜底方法（JDK 8 兼容，支持 BlockException 入参）
     */
    public String helloBlockHandler(Integer id, BlockException e) {
        String s = "当前请求过于频繁，请稍后再试（限流/熔断保护），id: " + id;
        System.out.println(s);
        return s;
    }

    /**
     * 业务异常兜底方法（JDK 8 兼容，支持 Throwable 入参）
     */
    public String helloFallback(Integer id, Throwable e) {
        String s = "业务处理失败：" + e.getMessage() + ", id: " + id;
        System.out.println(s);
        return s;
    }
}
```

2.dashboard

1.下载安装

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

2.启动访问

```
1.下载好jar后
执行 java  -jar sentinel-dashboard-1.8.9.jar

2.启动刚写的服务8088的

3.观察dashboard
http://localhost:8080
默认账号密码
sentinel/sentinel
```

3.jmeter

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

4.hello测试

```
效果
Hello Sentinel 1.8.6 + Spring Boot 2.7.5, id: 1
Hello Sentinel 1.8.6 + Spring Boot 2.7.5, id: 1
当前请求过于频繁，请稍后再试（限流/熔断保护），id: 1
当前请求过于频繁，请稍后再试（限流/熔断保护），id: 1
当前请求过于频繁，请稍后再试（限流/熔断保护），id: 1
当前请求过于频繁，请稍后再试（限流/熔断保护），id: 1
Hello Sentinel 1.8.6 + Spring Boot 2.7.5, id: 1
当前请求过于频繁，请稍后再试（限流/熔断保护），id: 1
当前请求过于频繁，请稍后再试（限流/熔断保护），id: 1
当前请求过于频繁，请稍后再试（限流/熔断保护），id: 1
当前请求过于频繁，请稍后再试（限流/熔断保护），id: 1
Hello Sentinel 1.8.6 + Spring Boot 2.7.5, id: 1
当前请求过于频繁，请稍后再试（限流/熔断保护），id: 1
当前请求过于频繁，请稍后再试（限流/熔断保护），id: 1
当前请求过于频繁，请稍后再试（限流/熔断保护），id: 1
当前请求过于频繁，请稍后再试（限流/熔断保护），id: 1
Hello Sentinel 1.8.6 + Spring Boot 2.7.5, id: 1
当前请求过于频繁，请稍后再试（限流/熔断保护），id: 1
当前请求过于频繁，请稍后再试（限流/熔断保护），id: 1
当前请求过于频繁，请稍后再试（限流/熔断保护），id: 1
当前请求过于频繁，请稍后再试（限流/熔断保护），id: 1
Hello Sentinel 1.8.6 + Spring Boot 2.7.5, id: 1
当前请求过于频繁，请稍后再试（限流/熔断保护），id: 1
当前请求过于频繁，请稍后再试（限流/熔断保护），id: 1
当前请求过于频繁，请稍后再试（限流/熔断保护），id: 1
当前请求过于频繁，请稍后再试（限流/熔断保护），id: 1
Hello Sentinel 1.8.6 + Spring Boot 2.7.5, id: 1
当前请求过于频繁，请稍后再试（限流/熔断保护），id: 1
当前请求过于频繁，请稍后再试（限流/熔断保护），id: 1
当前请求过于频繁，请稍后再试（限流/熔断保护），id: 1
当前请求过于频繁，请稍后再试（限流/熔断保护），id: 1

```

5.引入nacos持久化

1.pom

```xml
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 https://maven.apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>
    <!-- Spring Boot 父工程 -->
    <parent>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-parent</artifactId>
        <version>2.7.5</version>
        <relativePath/> <!-- lookup parent from repository -->
    </parent>
    <groupId>com.example.study</groupId>
    <artifactId>springcloudalibaba-sentinel</artifactId>
    <version>0.0.1-SNAPSHOT</version>
    <name>springcloudalibaba-sentinel</name>
    <description>Demo project for Spring Boot</description>


    <properties>
        <java.version>1.8</java.version> <!-- 指定 JDK 8 -->
        <spring-cloud.version>2021.0.5</spring-cloud.version>
    </properties>

    <!-- 依赖版本管理（锁定 Spring Cloud 和 Spring Cloud Alibaba 版本） -->
    <dependencyManagement>
        <dependencies>
            <!-- Spring Cloud 依赖管理 -->
            <dependency>
                <groupId>org.springframework.cloud</groupId>
                <artifactId>spring-cloud-dependencies</artifactId>
                <version>${spring-cloud.version}</version>
                <type>pom</type>
                <scope>import</scope>
            </dependency>
            <!-- Spring Cloud Alibaba 依赖管理（内置 Sentinel 1.8.6） -->
            <dependency>
                <groupId>com.alibaba.cloud</groupId>
                <artifactId>spring-cloud-alibaba-dependencies</artifactId>
                <version>2.1.0.RELEASE</version>
                <type>pom</type>
                <scope>import</scope>
            </dependency>
        </dependencies>
    </dependencyManagement>

    <dependencies>
        <!-- Spring Boot Web 依赖（提供 Web 环境） -->
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-web</artifactId>
        </dependency>

        <!-- https://mvnrepository.com/artifact/com.alibaba.cloud/spring-cloud-starter-alibaba-sentinel -->
        <dependency>
            <groupId>com.alibaba.cloud</groupId>
            <artifactId>spring-cloud-starter-alibaba-sentinel</artifactId>
            <version>2.2.8.RELEASE</version>
            <scope>compile</scope>
        </dependency>
        <dependency>
            <groupId>junit</groupId>
            <artifactId>junit</artifactId>
            <scope>test</scope>
        </dependency>

        <!-- 5. Spring Boot 测试依赖（可选，用于接口测试） -->
        <dependency>
```

```
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-test</artifactId>
            <scope>test</scope>
        </dependency>

        <!-- Sentinel 规则持久化到 Nacos 的核心适配依赖 -->
        <dependency>
            <groupId>com.alibaba.csp</groupId>
            <artifactId>sentinel-datasource-nacos</artifactId>
        </dependency>

        <!-- 确保已存在 Nacos 核心依赖（服务注册+配置中心），无需重复添加 -->
        <dependency>
            <groupId>com.alibaba.cloud</groupId>
            <artifactId>spring-cloud-starter-alibaba-nacos-discovery</artifactId>
        </dependency>
        <dependency>
            <groupId>com.alibaba.cloud</groupId>
            <artifactId>spring-cloud-starter-alibaba-nacos-config</artifactId>
        </dependency>
    </dependencies>

    <build>
        <plugins>
            <plugin>
                <groupId>org.springframework.boot</groupId>
                <artifactId>spring-boot-maven-plugin</artifactId>
                <version>2.7.5</version>
            </plugin>
        </plugins>
    </build>

</project>

```

2.yaml

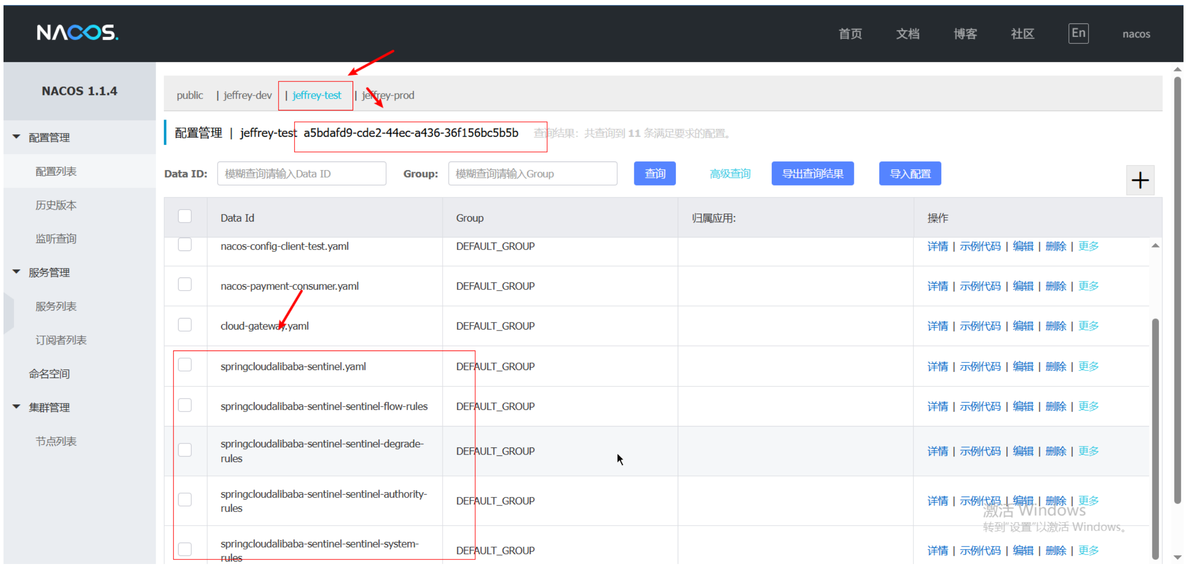
```yaml
myNameSpace: a5bdafd9-cde2-44ec-a436-36f156bc5b5b
server:
  port: 7000
spring:
  config:
    import: optional:nacos:${spring.application.name}.yaml
  application:
    name: springcloudalibaba-sentinel # 应用名称，用于拼接 Nacos 配置 ID
  cloud:
    # Nacos 基础配置（服务地址）
    nacos:
      discovery:
        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）
        namespace: ${myNameSpace}
      config:
        server-addr: localhost:8848 #nacos作为配置中心地址
        file-extension: yaml #指定yaml格式的配置
        group: DEFAULT_GROUP #指定分组
        namespace: ${myNameSpace} #指定spacename
    sentinel:
      transport:
        dashboard: localhost:8080 # Sentinel 控制台地址
        port: 8719 # 应用与控制台通信端口
        app-name: ${spring.application.name}
      eager: true # 开启非懒加载，项目启动直接注册到控制台
      web-context-unify: true # 统一 Web 上下文，适配 Spring Boot 2.7.5 Web 环境
      # 核心：配置 Sentinel 多规则类型的 Nacos 数据源（持久化核心配置）
      datasource:
        # 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）
        flow:
          nacos:
            server-addr: localhost:8848 # Nacos 服务地址（与上方一致）
            data-id: ${spring.application.name}-sentinel-flow-rules # 配置 ID（唯一标识）
            group-id: DEFAULT_GROUP # 配置分组（默认 DEFAULT_GROUP，可自定义）
            data-type: json # 规则数据格式（必须为 JSON，Sentinel 仅支持 JSON 解析）
            rule-type: flow # 规则类型（flow=限流，degrade=熔断，authority=授权，system=系统）
            namespace: ${myNameSpace} # 新增：Sentinel 数据源命名空间（与上方一致）
        # 2. 熔断降级规则（degrade）- 可选，按需配置
        degrade:
          nacos:
            server-addr: localhost:8848
            data-id: ${spring.application.name}-sentinel-degrade-rules
            group-id: DEFAULT_GROUP
            data-type: json
            rule-type: degrade
            namespace: ${myNameSpace} # 新增：Sentinel 数据源命名空间（与上方一致）
        # 3. 其他规则（授权、系统）- 按需添加，配置格式同上
        authority:
          nacos:
            server-addr: localhost:8848
            data-id: ${spring.application.name}-sentinel-authority-rules
            group-id: DEFAULT_GROUP
            data-type: json
            rule-type: authority
            namespace: ${myNameSpace} # 新增：Sentinel 数据源命名空间（与上方一致）
        # 4. 系统规则（system）- 全局应用保护（CPU/负载/QPS 等）
        system:
          nacos:
            server-addr: localhost:8848
            data-id: ${spring.application.name}-sentinel-system-rules
            group-id: DEFAULT_GROUP
            data-type: json
            rule-type: system
            namespace: ${myNameSpace} # 新增：Sentinel 数据源命名空间（与上方一致）
        # 5. 热点参数规则（param-flow） - 针对接口热点参数的精准限流
        param-flow:
          nacos:
            server-addr: localhost:8848
            data-id: ${spring.application.name}-sentinel-param-flow-rules
            group-id: DEFAULT_GROUP
            data-type: json
            rule-type: param-flow
            namespace: ${myNameSpace} # 新增: Sentinel 数据源命名空间（与上方一致）






```

3.nacos



NACOS. 首页 文档 博客 社区 En nacos

NACOS 1.1.4

- 配置管理
  - 配置列表
  - 历史版本
  - 监听查询
- 服务管理
  - 服务列表
  - 订阅者列表
- 命名空间
- 集群管理
  - 节点列表

public | jeffrey-dev | jeffrey-test | jeffrey-prod

配置管理 | jeffrey-test a5bdafd9-cde2-44ec-a436-36f156bc5b5b 查询结果：共查询到 11 条满足要求的配置。

Data ID: 模糊查询请输入Data ID Group: 模糊查询请输入Group 查询 高级查询 导出查询结果 导入配置

Data Id	Group	归属应用:	操作
nacos-config-client-test.yaml	DEFAULT_GROUP		详情 | 示例代码 | 编辑 | 删除 | 更多
nacos-payment-consumer.yaml	DEFAULT_GROUP		详情 | 示例代码 | 编辑 | 删除 | 更多
cloud-gateway.yaml	DEFAULT_GROUP		详情 | 示例代码 | 编辑 | 删除 | 更多
springcloudalibaba-sentinel.yaml	DEFAULT_GROUP		详情 | 示例代码 | 编辑 | 删除 | 更多
springcloudalibaba-sentinel-sentinel-flow-rules	DEFAULT_GROUP		详情 | 示例代码 | 编辑 | 删除 | 更多
springcloudalibaba-sentinel-sentinel-degrade-rules	DEFAULT_GROUP		详情 | 示例代码 | 编辑 | 删除 | 更多
springcloudalibaba-sentinel-sentinel-authority-rules	DEFAULT_GROUP		详情 | 示例代码 | 编辑 | 删除 | 更多
springcloudalibaba-sentinel-sentinel-system-rules	DEFAULT_GROUP		详情 | 示例代码 | 编辑 | 删除 | 更多

激活 Windows
转到"设置"以激活 Windows。

3.常见测试

```
nacos中的flow规则

[
  {
    "resource": "/threshold/qps",
    "limitApp": "default",
    "grade": 1,
    "count": 2,
    "strategy": 0,
    "controlBehavior": 0,
    "clusterMode": false
  },
  {
    "resource": "/threshold/thread",
    "limitApp": "default",
    "grade": 0,
    "count": 3,
    "strategy": 0,
    "controlBehavior": 0,
    "clusterMode": false
  },
  {
    "resource": "helloResource",
    "limitApp": "default",
    "grade": 1,
    "count": 2,
    "strategy": 0,
    "controlBehavior": 0,
    "clusterMode": false
  },
  {
    "resource": "/flowControlEffect/warmUp",
    "limitApp": "default",
    "grade": 1,
    "count": 4,
    "strategy": 0,
    "controlBehavior": 0,
    "clusterMode": false
  },
  {
    "resource": "/flowControlEffect/queueWait",
    "limitApp": "default",
    "grade": 1,
    "count": 5,
    "strategy": 0,
    "controlBehavior": 0,
    "clusterMode": false
  },
  {
    "resource": "/flowControlEffect/failFast",
    "limitApp": "default",
    "grade": 1,
    "count": 3,
    "strategy": 0,
    "controlBehavior": 0,
    "clusterMode": false
  },
  {
  "resource": "/flowControlMode/direct",
  "limitApp": "default",
  "grade": 1,
  "count": 2,
  "strategy": 0,
  "controlBehavior": 0,
  "clusterMode": false
  },
  {
  "resource": "/flowControlMode/association",
  "limitApp": "default",
  "grade": 1,
  "count": 3,
  "strategy": 1,
  "refResource": "/flowControlMode/associationRef",
  "controlBehavior": 0,
  "clusterMode": false
  },
  {
  "resource": "serviceLink",
  "limitApp": "default",
  "grade": 1,
  "count": 2,
  "strategy": 2,
  "refResource": "/flowControlMode/linkEnter",
  "controlBehavior": 0,
  "clusterMode": false
  },
  {
  "resource": "flowControlMode/linkController",
  "limitApp": "default",
  "grade": 1,
  "count": 2,
  "strategy": 2,
  "refResource": "/flowControlMode/linkEnterController",
  "controlBehavior": 0,
  "clusterMode": false
  }

]
```

1.threshold

1.qps

```
【qps-成功】
【qps-成功】
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}
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}
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}
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}
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}
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}
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}
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}
```

```
【qps-成功】
【qps-成功】
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}

```

2.thread

```
使用单个线程发送很多请求成功才是正常的
【thread-成功】
【thread-成功】
【thread-成功】
【thread-成功】
【thread-成功】
【thread-成功】
【thread-成功】
【thread-成功】
【thread-成功】
【thread-成功】
【thread-成功】
【thread-成功】
【thread-成功】
【thread-成功】
【thread-成功】
【thread-成功】
【thread-成功】
【thread-成功】
【thread-成功】

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

【thread-成功】
【thread-成功】
【thread-成功】
【thread-成功】
【thread-成功】
【thread-成功】
【thread-成功】
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}
【thread-成功】
【thread-成功】
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}
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}
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}
```

```
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}
【thread-成功】
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}
【thread-成功】
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}
【thread-成功】
【thread-成功】
【thread-成功】
【thread-成功】
```

2.flowControlEffect

1.failfast

```
【failFast-成功】
【failFast-成功】
【failFast-成功】
响应结果：{"code":429,"errorType":"FLOW_LIMITING","allRequestParams":{},"requestMethod":"GET","requestPath":"/flowControlEffect/failFast","errorMsg":"请求过于频繁，触发流量控制（限流）","timestamp":1768133313578}
响应结果：{"code":429,"errorType":"FLOW_LIMITING","allRequestParams":{},"requestMethod":"GET","requestPath":"/flowControlEffect/failFast","errorMsg":"请求过于频繁，触发流量控制（限流）","timestamp":1768133313682}
响应结果：{"code":429,"errorType":"FLOW_LIMITING","allRequestParams":{},"requestMethod":"GET","requestPath":"/flowControlEffect/failFast","errorMsg":"请求过于频繁，触发流量控制（限流）","timestamp":1768133313781}
响应结果：{"code":429,"errorType":"FLOW_LIMITING","allRequestParams":{},"requestMethod":"GET","requestPath":"/flowControlEffect/failFast","errorMsg":"请求过于频繁，触发流量控制（限流）","timestamp":1768133313878}
响应结果：{"code":429,"errorType":"FLOW_LIMITING","allRequestParams":{},"requestMethod":"GET","requestPath":"/flowControlEffect/failFast","errorMsg":"请求过于频繁，触发流量控制（限流）","timestamp":1768133313980}
【failFast-成功】
【failFast-成功】
【failFast-成功】
响应结果：{"code":429,"errorType":"FLOW_LIMITING","allRequestParams":{},"requestMethod":"GET","requestPath":"/flowControlEffect/failFast","errorMsg":"请求过于频繁，触发流量控制（限流）","timestamp":1768133314379}
响应结果：{"code":429,"errorType":"FLOW_LIMITING","allRequestParams":{},"requestMethod":"GET","requestPath":"/flowControlEffect/failFast","errorMsg":"请求过于频繁，触发流量控制（限流）","timestamp":1768133314479}
```

2.warmUp

```
【warmUp-成功】
【warmUp-成功】
```

```
【warmUp-成功】
【warmUp-成功】
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}
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}
【warmUp-成功】
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}
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}
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}
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}
【warmUp-成功】
【warmUp-成功】
【warmUp-成功】
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

```
【queueWait-成功】
【queueWait-成功】
【queueWait-成功】
【queueWait-成功】
【queueWait-成功】
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}
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}
【queueWait-成功】
【queueWait-成功】
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}
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}
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}
【queueWait-成功】
【queueWait-成功】
【queueWait-成功】
```

```
```

3.flowControlMode

1.direct

```
【direct-成功】
【direct-成功】
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}
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}
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}
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}
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}
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}
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}
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}
【direct-成功】
【direct-成功】
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}
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.assocication

```
注意点
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，然后单独用了一个没有规则的，让ref能正常并发不被限流

【association-成功】
【associationRef-成功】
【association-成功】
【associationRef-成功】
【associationRef-成功】
【associationRef-成功】
```

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

3.link

```
单独调用link,只有成功的才对
【service-link-成功】
【service-link-成功】
【service-link-成功】
【service-link-成功】
【service-link-成功】
【service-link-成功】
【service-link-成功】
【service-link-成功】
【service-link-成功】
【service-link-成功】
【service-link-成功】
【service-link-成功】
【service-link-成功】
【service-link-成功】
【service-link-成功】
【service-link-成功】


调用linkEnter

linkEnter-【service-link-成功】
linkEnter-【service-link-成功】
linkEnter-【service-link-降级】
linkEnter-【service-link-降级】
linkEnter-【service-link-降级】
linkEnter-【service-link-降级】
linkEnter-【service-link-降级】
linkEnter-【service-link-降级】
linkEnter-【service-link-降级】
linkEnter-【service-link-降级】
linkEnter-【service-link-降级】
linkEnter-【service-link-降级】
linkEnter-【service-link-降级】
linkEnter-【service-link-成功】
linkEnter-【service-link-成功】
linkEnter-【service-link-降级】
linkEnter-【service-link-降级】
linkEnter-【service-link-降级】
linkEnter-【service-link-降级】
linkEnter-【service-link-降级】
linkEnter-【service-link-降级】

```

```

框架系统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
```

4.hotParam

```
nacos中的规则

[
  {
  "resource": "/hotParam/single",
  "grade": 1,
  "count": 2,
  "paramIdx": 0,
  "durationInSec": 1,
  "controlBehavior": 0,
  "clusterMode": false,
  "paramFlowItemList": []
  } ,
  {
  "resource": "/hotParam/singleWithItem",
  "grade": 1,
  "count": 2,
  "paramIdx": 0,
  "durationInSec": 1,
  "controlBehavior": 0,
  "clusterMode": false,
  "paramFlowItemList": [
    {
      "object": "999",
      "classType": "java.lang.String",
      "count": 200
    }
  ]
  },
  {
  "resource": "/hotParam/mutil",
  "grade": 1,
  "count": 2,
  "paramIdx": 0,
  "durationInSec": 1,
  "controlBehavior": 0,
  "clusterMode": false,
  "paramFlowItemList": [],
  "additionalParamIdxList": []
  },
  {
  "resource": "/hotParam/mutil",
  "grade": 1,
  "count": 1,
  "paramIdx": 1,
  "durationInSec": 1,
  "controlBehavior": 0,
  "clusterMode": false,
  "paramFlowItemList": [],
  "additionalParamIdxList": []
  },
  {
  "resource": "/hotParam/mutilWithItem",
  "grade": 1,
  "count": 2,
  "paramIdx": 0,
  "durationInSec": 1,
  "controlBehavior": 0,
  "clusterMode": false,
  "paramFlowItemList": [
    {
      "object": "999",
      "classType": "java.lang.String",
      "count": 200
    }
  ]
  } ,
  {
  "resource": "/hotParam/mutilWithItem",
  "grade": 1,
  "count": 1,
  "paramIdx": 1,
  "durationInSec": 1,
  "controlBehavior": 0,
  "clusterMode": false,
  "paramFlowItemList": [
    {
      "object": "jeffrey",
      "classType": "java.lang.String",
      "count": 200
    }
  ]
  }

]
```

1.single

```
【single-成功】-id:1001
【single-成功】-id:1001
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}
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}
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}
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}
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}
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}
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}
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}
```

```
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}
【single-成功】-id:1001
【single-成功】-id:1001
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}
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}
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

```
【singleWithItem-成功】-id:999
【singleWithItem-成功】-id:999
【singleWithItem-成功】-id:999
【singleWithItem-成功】-id:999
【singleWithItem-成功】-id:999
【singleWithItem-成功】-id:999
【singleWithItem-成功】-id:999
【singleWithItem-成功】-id:999
【singleWithItem-成功】-id:999
【singleWithItem-成功】-id:999
```

3.mutil

```
【mutil-成功】-id:1001name:luck
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}
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}
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}
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}
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}
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}
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}
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}
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}
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}
【mutil-成功】-id:1001name:luck
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}
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}
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}
```

```
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

```
【mutilWithItem-成功】-id:999name:jeffrey
【mutilWithItem-成功】-id:999name:jeffrey
【mutilWithItem-成功】-id:999name:jeffrey
【mutilWithItem-成功】-id:999name:jeffrey
【mutilWithItem-成功】-id:999name:jeffrey
【mutilWithItem-成功】-id:999name:jeffrey
【mutilWithItem-成功】-id:999name:jeffrey
【mutilWithItem-成功】-id:999name:jeffrey
【mutilWithItem-成功】-id:999name:jeffrey
【mutilWithItem-成功】-id:999name:jeffrey
【mutilWithItem-成功】-id:999name:jeffrey
【mutilWithItem-成功】-id:999name:jeffrey
【mutilWithItem-成功】-id:999name:jeffrey
【mutilWithItem-成功】-id:999name:jeffrey
【mutilWithItem-成功】-id:999name:jeffrey
【mutilWithItem-成功】-id:999name:jeffrey

```

5.authority

```
nacos authority的json格式的配置

[
  {
    "resource": "/authority/whiteList",
    "limitApp": "inner-app,special-ip",
    "strategy": 0
  },
  {
    "resource": "/authority/blackList",
    "limitApp": "black-app,spec-black-ip",
    "strategy": 1
  }
]
```

1.whiteList

```
1.白名单内的
【whiteList-成功】,X-Sentinel-App:inner-app
【whiteList-成功】,X-Sentinel-App:inner-app
【whiteList-成功】,X-Sentinel-App:inner-app
【whiteList-成功】,X-Sentinel-App:inner-app
【whiteList-成功】,X-Sentinel-App:inner-app
【whiteList-成功】,X-Sentinel-App:inner-app
【whiteList-成功】,X-Sentinel-App:inner-app
【whiteList-成功】,X-Sentinel-App:inner-app
```

```
【whiteList-成功】,X-Sentinel-App:inner-app
【whiteList-成功】,X-Sentinel-App:inner-app
【whiteList-成功】,X-Sentinel-App:inner-app

2.白名单外的
响应结果：{"code":403,"errorType":"AUTHORITY_DENY","allRequestParams":{"id":"1001"},"requestMethod":"GET","requestPath":"/authority/whiteList","errorMsg":"请求被拒绝，不符合授权规则（黑白名单限制）","timestamp":1768150430620}
响应结果：{"code":403,"errorType":"AUTHORITY_DENY","allRequestParams":{"id":"1001"},"requestMethod":"GET","requestPath":"/authority/whiteList","errorMsg":"请求被拒绝，不符合授权规则（黑白名单限制）","timestamp":1768150430659}
响应结果：{"code":403,"errorType":"AUTHORITY_DENY","allRequestParams":{"id":"1001"},"requestMethod":"GET","requestPath":"/authority/whiteList","errorMsg":"请求被拒绝，不符合授权规则（黑白名单限制）","timestamp":1768150430759}
响应结果：{"code":403,"errorType":"AUTHORITY_DENY","allRequestParams":{"id":"1001"},"requestMethod":"GET","requestPath":"/authority/whiteList","errorMsg":"请求被拒绝，不符合授权规则（黑白名单限制）","timestamp":1768150430859}
响应结果：{"code":403,"errorType":"AUTHORITY_DENY","allRequestParams":{"id":"1001"},"requestMethod":"GET","requestPath":"/authority/whiteList","errorMsg":"请求被拒绝，不符合授权规则（黑白名单限制）","timestamp":1768150430959}
响应结果：{"code":403,"errorType":"AUTHORITY_DENY","allRequestParams":{"id":"1001"},"requestMethod":"GET","requestPath":"/authority/whiteList","errorMsg":"请求被拒绝，不符合授权规则（黑白名单限制）","timestamp":1768150431059}
响应结果：{"code":403,"errorType":"AUTHORITY_DENY","allRequestParams":{"id":"1001"},"requestMethod":"GET","requestPath":"/authority/whiteList","errorMsg":"请求被拒绝，不符合授权规则（黑白名单限制）","timestamp":1768150431158}

```

2.blackList

```
1.黑名单内
响应结果：{"code":403,"errorType":"AUTHORITY_DENY","allRequestParams":{"id":"1001"},"requestMethod":"GET","requestPath":"/authority/blackList","errorMsg":"请求被拒绝，不符合授权规则（黑白名单限制）","timestamp":1768150196125}
响应结果：{"code":403,"errorType":"AUTHORITY_DENY","allRequestParams":{"id":"1001"},"requestMethod":"GET","requestPath":"/authority/blackList","errorMsg":"请求被拒绝，不符合授权规则（黑白名单限制）","timestamp":1768150196178}
响应结果：{"code":403,"errorType":"AUTHORITY_DENY","allRequestParams":{"id":"1001"},"requestMethod":"GET","requestPath":"/authority/blackList","errorMsg":"请求被拒绝，不符合授权规则（黑白名单限制）","timestamp":1768150196277}
响应结果：{"code":403,"errorType":"AUTHORITY_DENY","allRequestParams":{"id":"1001"},"requestMethod":"GET","requestPath":"/authority/blackList","errorMsg":"请求被拒绝，不符合授权规则（黑白名单限制）","timestamp":1768150196378}

2.黑名单外
【blackList-成功】,X-Sentinel-App:spec-black-ip11
【blackList-成功】,X-Sentinel-App:spec-black-ip11
【blackList-成功】,X-Sentinel-App:spec-black-ip11
【blackList-成功】,X-Sentinel-App:spec-black-ip11
【blackList-成功】,X-Sentinel-App:spec-black-ip11
【blackList-成功】,X-Sentinel-App:spec-black-ip11
【blackList-成功】,X-Sentinel-App:spec-black-ip11
【blackList-成功】,X-Sentinel-App:spec-black-ip11
【blackList-成功】,X-Sentinel-App:spec-black-ip11
【blackList-成功】,X-Sentinel-App:spec-black-ip11
```

```
【blackList-成功】,X-Sentinel-App:spec-black-ip11
```