目录

[基本介绍 2](#_Toc450)

[服务限流 2](#_Toc860)

[服务熔断 2](#_Toc30795)

[服务降级 2](#_Toc8569)

[为什么需要限流降级 2](#_Toc28263)

[开源限流熔断框架技术实现 3](#_Toc19686)

[Sentinel 和 Hystrix 框架对比 3](#_Toc17793)

[Sentinel开源限流熔断框架 4](#_Toc13209)

[Sentinel 涉及专有名词 4](#_Toc25859)

[Sentinel 基础概念 4](#_Toc4961)

[资源 4](#_Toc24421)

[规则 4](#_Toc22828)

[Sentinel 功能和设计理念 5](#_Toc26833)

[流量控制 5](#_Toc16299)

[熔断降级 6](#_Toc29614)

[系统自适应保护 7](#_Toc4125)

[Sentinel 特征 8](#_Toc10971)

[Sentinel 架构设计 9](#_Toc32308)

[Sentinel 组成 9](#_Toc32348)

[Sentinel控制台（Dashboard） 9](#_Toc8317)

[获取Sentinel控制台（Dashboard） 9](#_Toc24256)

[启动Sentinel控制台（Dashboard） 10](#_Toc20658)

[Sentinel核心库（Java客户端） 11](#_Toc25755)

[Sentinel核心库（Java客户端）接入控制台（Dashboard） 11](#_Toc23510)

[Sentinel核心库（Java客户端）实现熔断和限流 13](#_Toc19637)

# 基本介绍

## 服务限流

当系统资源不够，不足以应对大量请求，对系统按照预设的规则进行流量限制或功能限制。

## 服务熔断

当调用目标服务的请求和调用大量超时或失败，服务调用方为避免造成长时间的阻塞造成影响其他服务，后续对该服务接口的调用不再经过进行请求，直接执行本地的默认方法。

## 服务降级

为了保证核心业务在大量请求下能正常运行，根据实际业务情况及流量，对部分服务降低优先级，有策略的不处理或用简单的方式处理。

# 为什么需要限流降级

系统承载的访问量是有限的，如果不做流量控制，会导致系统资源占满，服务超时，从而所有用户无法使用，通过服务限流控制请求的量，服务降级省掉非核心业务对系统资源的占用，最大化利用系统资源，尽可能服务更多用户。

# 开源限流熔断框架技术实现

* Hystrix是由Netflix开源的一个延迟和容错库，是SOA/微服务架构中提供服务隔离、熔断、降级机制的工具/框架。Netflix Hystrix是断路器的一种实现，用于高微服务架构的可用性，是防止服务出现雪崩的利器。可以提升系统的可用性与容错性。
* Sentinel: 分布式系统的流量防卫兵，是阿里中间件团队 2018 年 7 月开源的，面向分布式服务架构的轻量级流量控制产品，主要以流量为切入点，从流量控制、熔断降级、系统负载保护等多个维度来保护系统服务的稳定性。

# Sentinel 和 Hystrix 框架对比

|  |  |  |
| --- | --- | --- |
|  | Sentinell | Hystrix |
| 隔离策略 | 信号量隔离 | 线程池隔离/信号量隔离 |
| 熔断降级策略 | 基于响应时间/失败率 | 基于失败率 |
| 实时指标实现 | 滑动窗口 | 滑动窗口(基于RxJAVA) |
| 规则配置 | 支持多种数据源 | 支持多种数据源 |
| 拓展性 | 多个拓展点 | 插件的形式 |
| 基于注解支持 | 支持 | 支持 |
| 限流 | 基于QPS/支持基于调用关系限流 | 不支持 |
| 流量整形 | 支持慢启动，匀速器模式 | 不支持 |
| 系统负载保护 | 支持 | 不支持 |
| 控制台/管理UI | 开箱即用，可以查看配置、监控和服务器发现 | 不完善 |
| 框架适配 | Servlet\Spring Cloud\Dunbbo等 | Servlet \SpringC Cloud Next |

综合上述开源开源限流熔断框架对比并结合项目开发的实际功能需求，推荐使用Sentinel开源限流熔断框架。

# Sentinel开源限流熔断框架

## Sentinel 涉及专有名词

* 响应时间(RT)：响应时间是指系统对请求作出响应的时间。
* 吞吐量(Throughput)：吞吐量是指系统在单位时间内处理请求的数量。
* 并发用户数：并发用户数是指系统可以同时承载的正常使用系统功能的用户的数量。
* QPS每秒查询率(Query Per Second)：每秒查询率QPS是对一个特定的查询服务器在规定时间内所处理流量多少的衡量标准，在因特网上，作为域名系统服务器的机器的性能经常用每秒查询率来衡量。对应fetches/sec，即每秒的响应请求数，也即是最大吞吐能力。

## Sentinel 基础概念

### 资源

资源是 Sentinel 的关键概念。它可以是 Java 应用程序中的任何内容，例如，由应用程序提供的服务，或由应用程序调用的其它应用提供的服务，甚至可以是一段代码。在接下来的文档中，我们都会用资源来描述代码块。

只要通过 Sentinel API 定义的代码，就是资源，能够被 Sentinel 保护起来。大部分情况下，可以使用方法签名，URL，甚至服务名称作为资源名来标示资源。

### 规则

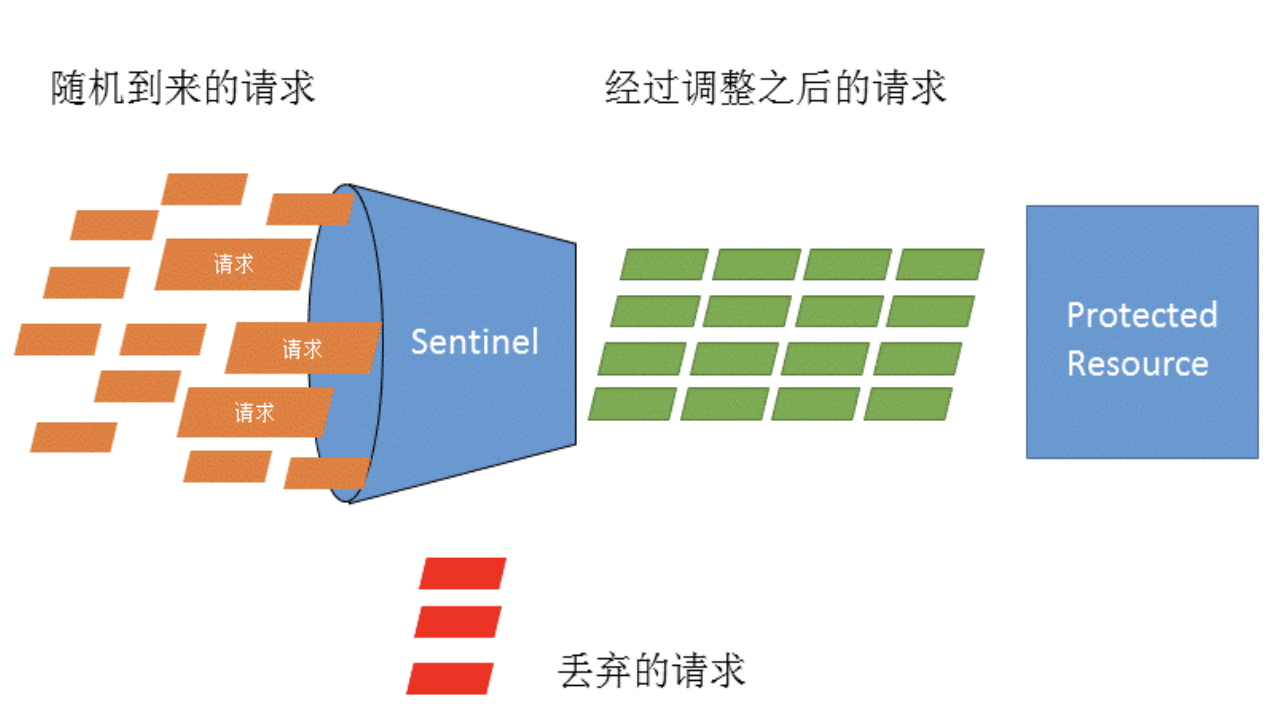
围绕资源的实时状态设定的规则，可以包括流量控制规则、熔断降级规则以及系统保护规则。所有规则可以动态实时调整。

## Sentinel 功能和设计理念

### 流量控制

#### 什么是流量控制

流量控制在网络传输中是一个常用的概念，它用于调整网络包的发送数据。然而，从系统稳定性角度考虑，在处理请求的速度上，也有非常多的讲究。任意时间到来的请求往往是随机不可控的，而系统的处理能力是有限的。我们需要根据系统的处理能力对流量进行控制。Sentinel 作为一个调配器，可以根据需要把随机的请求调整成合适的形状，如下图所示：



#### 流量控制设计理念

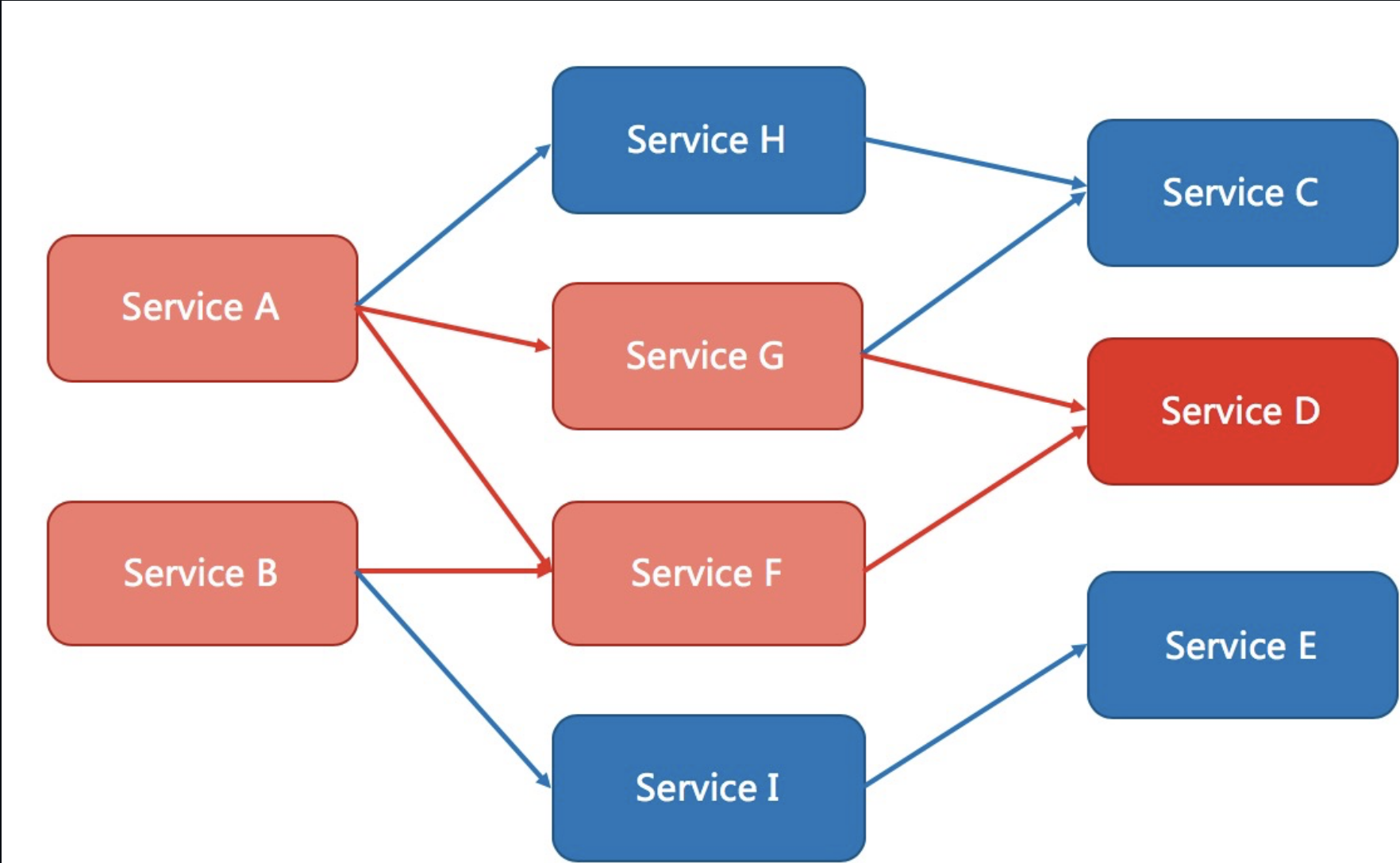
流量控制有以下几个角度:

* 资源的调用关系，例如资源的调用链路，资源和资源之间的关系；
* 运行指标，例如 QPS、线程池、系统负载等；
* 控制的效果，例如直接限流、冷启动、排队等。

### 熔断降级

#### 什么是熔断降级

除了流量控制以外，及时对调用链路中的不稳定因素进行熔断也是 Sentinel 的使命之一。由于调用关系的复杂性，如果调用链路中的某个资源出现了不稳定，可能会导致请求发生堆积，进而导致级联错误。



Sentinel 和 Hystrix 的原则是一致的: 当检测到调用链路中某个资源出现不稳定的表现，例如请求响应时间长或异常比例升高的时候，则对这个资源的调用进行限制，让请求快速失败，避免影响到其它的资源而导致级联故障。

#### 熔断降级设计理念

在限制的手段上，Sentinel 和 Hystrix 采取了完全不一样的方法。

Hystrix 通过 线程池隔离 的方式，来对依赖（在 Sentinel 的概念中对应 资源）进行了隔离。这样做的好处是资源和资源之间做到了最彻底的隔离。缺点是除了增加了线程切换的成本（过多的线程池导致线程数目过多），还需要预先给各个资源做线程池大小的分配。

##### Sentinel 对这个问题采取了两种手段:

* 通过并发线程数进行限制

和资源池隔离的方法不同，Sentinel 通过限制资源并发线程的数量，来减少不稳定资源对其它资源的影响。这样不但没有线程切换的损耗，也不需要您预先分配线程池的大小。当某个资源出现不稳定的情况下，例如响应时间变长，对资源的直接影响就是会造成线程数的逐步堆积。当线程数在特定资源上堆积到一定的数量之后，对该资源的新请求就会被拒绝。堆积的线程完成任务后才开始继续接收请求。

* 通过响应时间对资源进行降级

除了对并发线程数进行控制以外，Sentinel 还可以通过响应时间来快速降级不稳定的资源。当依赖的资源出现响应时间过长后，所有对该资源的访问都会被直接拒绝，直到过了指定的时间窗口之后才重新恢复。

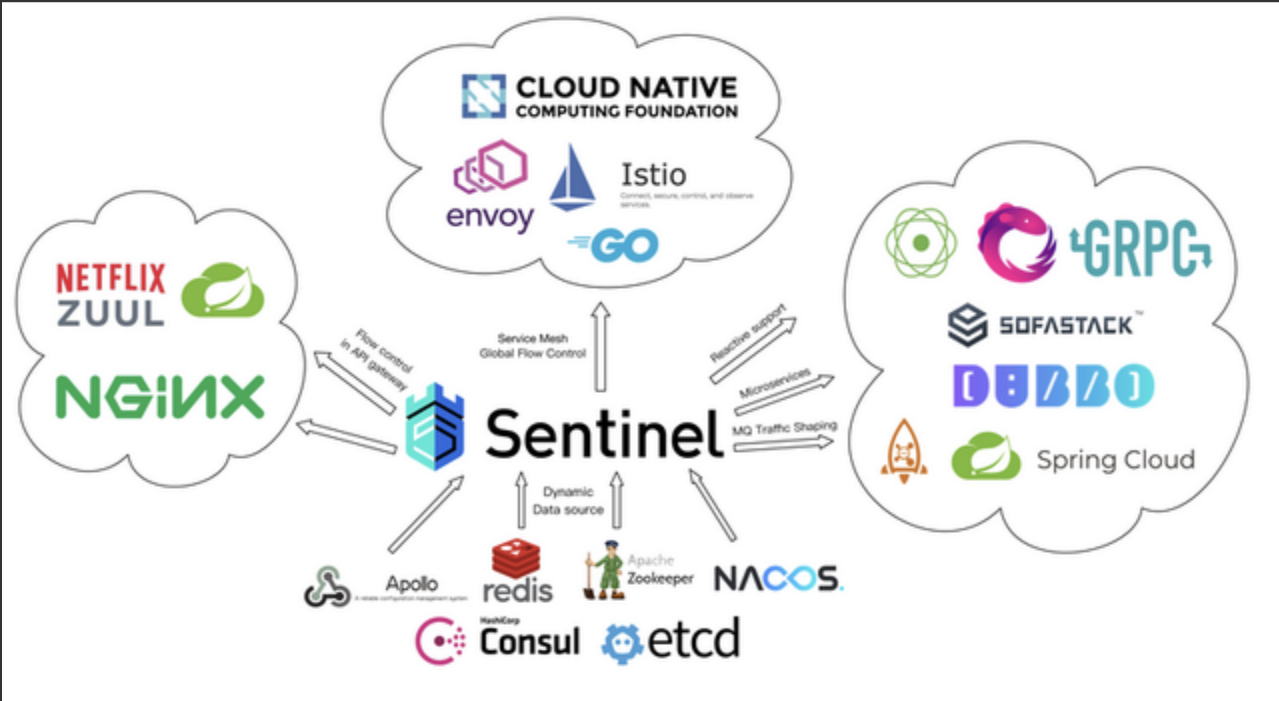
### 系统自适应保护

Sentinel 同时提供系统维度的自适应保护能力。防止雪崩，是系统防护中重要的一环。当系统负载较高的时候，如果还持续让请求进入，可能会导致系统崩溃，无法响应。在集群环境下，网络负载均衡会把本应这台机器承载的流量转发到其它的机器上去。如果这个时候其它的机器也处在一个边缘状态的时候，这个增加的流量就会导致这台机器也崩溃，最后导致整个集群不可用。

针对这个情况，Sentinel 提供了对应的保护机制，让系统的入口流量和系统的负载达到一个平衡，保证系统在能力范围之内处理最多的请求。

## Sentinel 特征

* **丰富的应用场景** ：Sentinel 承接了阿里巴巴近 10 年的双十一大促流量的核心场景，例如秒杀（即突发流量控制在系统容量可以承受的范围）、消息削峰填谷、集群流量控制、实时熔断下游不可用应用等。  
  **完备的实时监控** ：Sentinel 同时提供实时的监控功能。您可以在控制台中看到接入应用的单台机器秒级数据，甚至 500 台以下规模的集群的汇总运行情况。  
  **广泛的开源生态** ：Sentinel 提供开箱即用的与其它开源框架/库的整合模块，例如与 SpringCloud、Dubbo、gRPC 的整合。您只需要引入相应的依赖并进行简单的配置即可快速地接入Sentinel。
* **完善的 SPI 扩展点**：Sentinel 提供简单易用、完善的 SPI 扩展接口。您可以通过实现扩展接口来快速地定制逻辑。例如定制规则管理、适配动态数据源等。
* **Sentinel的生态圈**



## Sentinel 架构设计



## Sentinel 组成

**Sentinel 的使用可以分为两个部分:**

* 控制台（Dashboard）：控制台主要负责管理推送规则、监控、集群限流分配管理、机器发现等。
* 核心库（Java 客户端）：不依赖任何框架/库，能够运行于 Java 7 及以上的版本的运行时环境，同时对 Dubbo / Spring Cloud 等框架也有较好的支持。

Sentinel 社区官方网站：https://sentinelguard.io/

## Sentinel控制台（Dashboard）

### 获取Sentinel控制台（Dashboard）

方式一：从Sentinel Release 页面下载指版本。

访问地址:https://github.com/alibaba/Sentinel/releases

方式二：从GitHub克隆Sentinel 源码，自己本地通过Maven 指令编译源码生成jar包(mvn clean package)

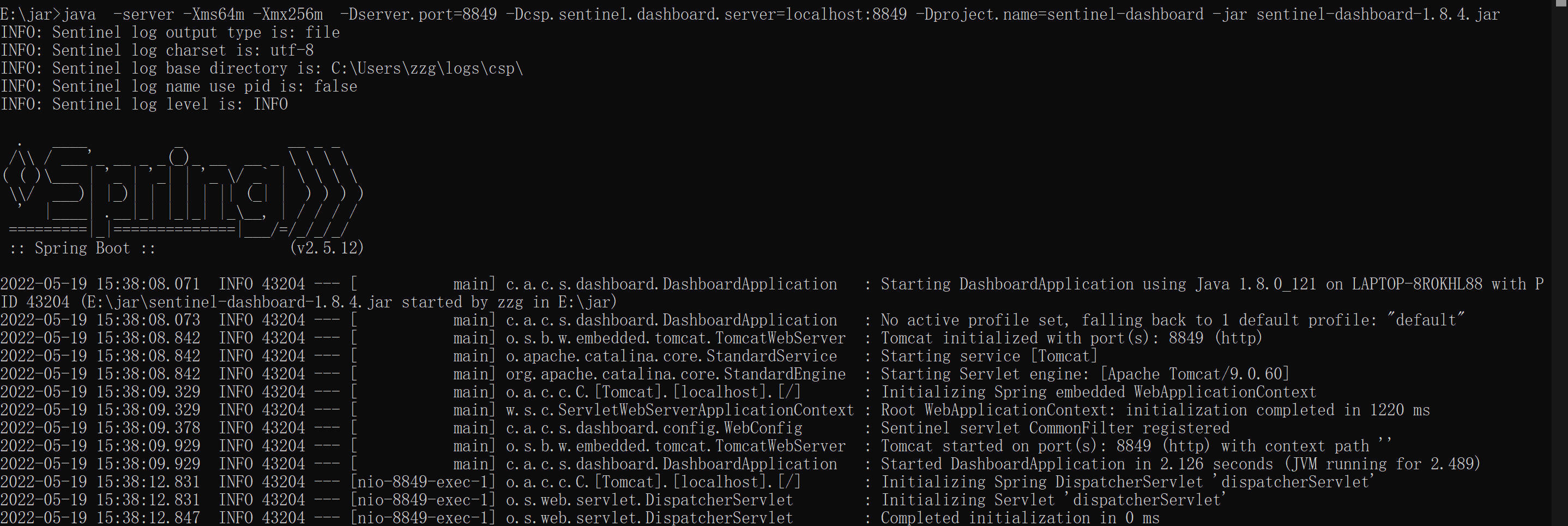
gitHub地址：<https://github.com/alibaba/Sentinel.git>

### 启动Sentinel控制台（Dashboard）

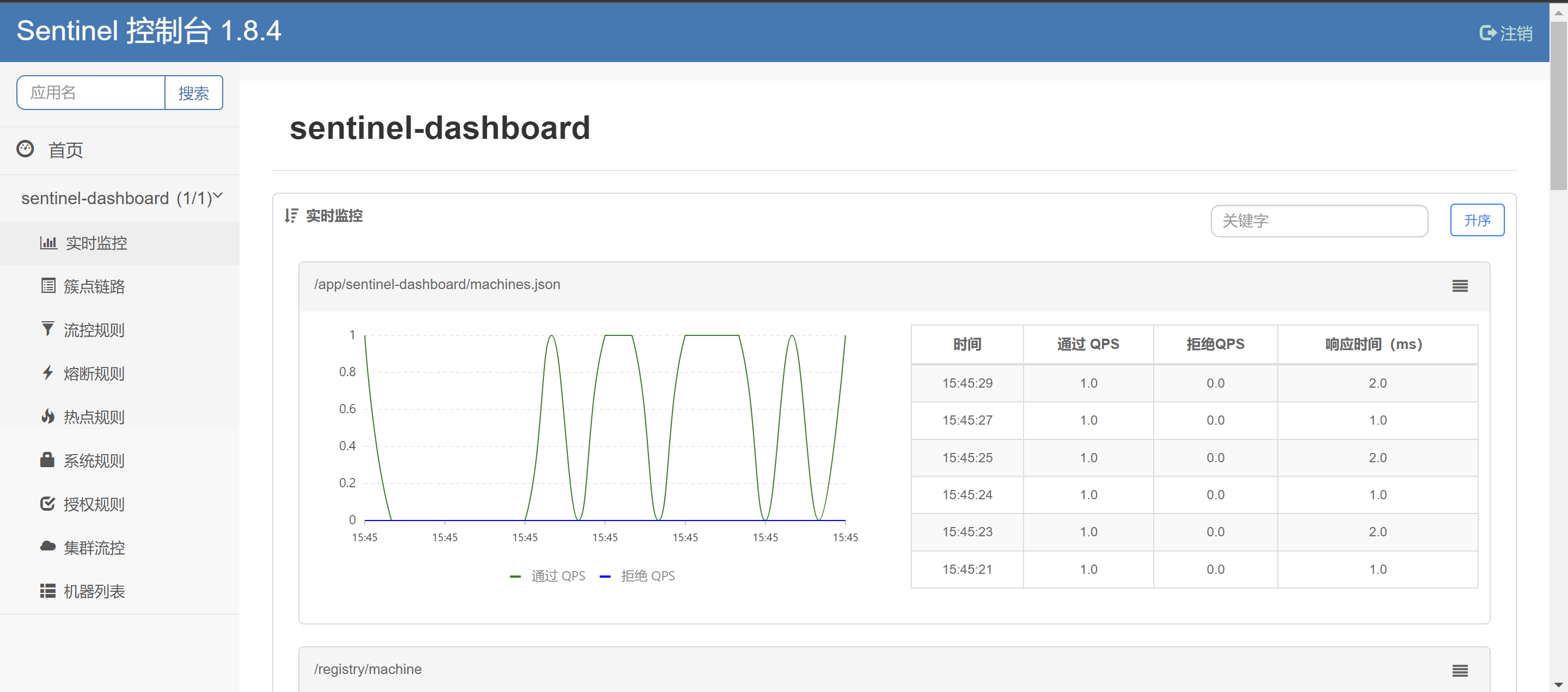
打开Windows 命令窗口，并且切换至sentinel-dashboard-\*.\*.\* jar包存放位置。执行启动Jar项目指令：

Java -server -Xms64m -Xmx256m -Dserver.port=8849 -Dcsp.sentinel.dashboard.server=localhost:8849 -Dproject.name=sentinel-dashboard -jar sentinel-dashboard-1.8.4.jar

温馨提示：从 Sentinel 1.6.0 起，Sentinel 控制台引入基本的登录功能，默认用户名和密码都是 sentinel 。







## Sentinel核心库（Java客户端）

### Sentinel核心库（Java客户端）接入控制台（Dashboard）

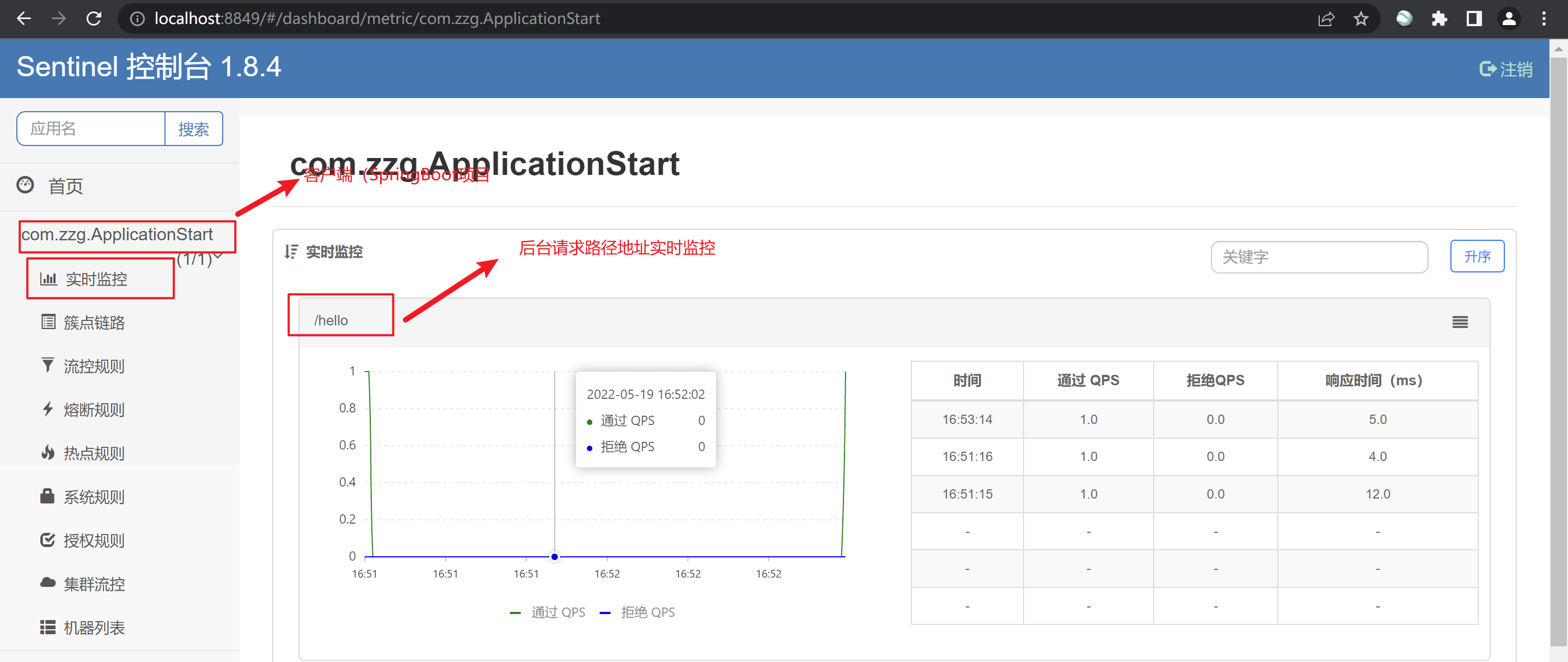
SpringBoot 项目pom.xml 添加Spring-Cloud依赖：

<dependencyManagement>  
 <dependencies>  
 <dependency>  
 <groupId>org.springframework.cloud</groupId>  
 <artifactId>spring-cloud-dependencies</artifactId>  
 <version>Finchley.SR1</version>  
 <type>pom</type>  
 <scope>import</scope>  
 </dependency>  
  
 <dependency>  
 <groupId>org.springframework.cloud</groupId>  
 <artifactId>spring-cloud-alibaba-dependencies</artifactId>  
 <version>0.2.2.RELEASE</version>  
 <type>pom</type>  
 <scope>import</scope>  
 </dependency>  
 </dependencies>  
</dependencyManagement>

SpringBoot 项目pom.xml 添加Sentinel 依赖

*<!-- 集成Sentinel-->  
<!--sentinel-->*<dependency>  
 <groupId>org.springframework.cloud</groupId>  
 <artifactId>spring-cloud-starter-alibaba-sentinel</artifactId>  
</dependency>  
*<!--sentinel end-->*

通过浏览器或PostMan模拟器向客户端发起请求。



### Sentinel核心库（Java客户端）实现熔断和限流

Java客户端使用Sentinel进行熔断保护，主要分为几个步骤：

第一：定义资源

资源：可以是任何东西（个人理解：可以通过URL或RPC请求的服务或者后台接口），举例说明：可以是Controller中的任意方法，也可以是Service 服务接口定义。

第二：定义规则

Sentinel支持的规则如下：流量控制规则、熔断降级规则、系统保护规则、来源访问控制规则和热点参数规则

第三步：校验规则是否生效

温馨提示：Sentinel 的所有规则都可以在内存态中动态地查询及修改，修改之后立即生效. 先把可能需要保护的资源定义好，之后再配置规则。