罗一（LY） 关注

Guava实现限流

原创 罗一（LY） 已于 2023-09-10 14:23:47 修改 阅读量1.6k 收藏 8 点赞数
分类专栏： 高可用 文章标签： guava 系统架构

目录

前言
一、硬编码方式
1.引入guava依赖包
2.给接口添加限流逻辑
3.测试结果
二、注解方式
1.加入AOP依赖
2.自定义限流注解
3.使用AOP切面拦截
4.给接口添加注解
5.测试结果
三、其他限流方式

前言

网上讲解限流算法的很多，自己实践一下印象会更深刻，所以在自己项目上实现了引入 Guava 做限流。

一、硬编码方式

1.引入guava依赖包

```
        <dependency>
            <groupId>com.google.guava</groupId>
            <artifactId>guava</artifactId>
            <version>30.1-jre</version><!--选择自己需要的版本-->
        </dependency>
```

2.给接口添加限流逻辑

```
@Controller
@RequestMapping("/area")
public class AreaController {
    protected final Logger logger = LoggerFactory.getLogger(this.getClass());
    @Autowired
    private IAreaService areaService;

    //每秒钟获3个请求
    private final RateLimiter limiter = RateLimiter.create(3);

    @RequestMapping("/testLimit")
    @ResponseBody
    public ResponseEntity testLimit(){

        try {
            //添加限流的逻辑
            //500毫秒没获取到令牌就返回失败
            boolean success = limiter.tryAcquire(500, TimeUnit.MILLISECONDS);
            if(!success){
                logger.info("硬编码接口未获取到令牌，返回");
                return new ResponseEntity(false, "系统繁忙，请稍后重试");
            }
```

```
            logger.info("硬编码接口获取到了令牌，执行业务");24
            schoolIdList.add("1");
            return areaService.queryCount(schoolIdList);
        } catch (Exception e) {
            logger.error("查询校区异常", Throwables.getStackTraceAsString(e));
            return new ResponseEntity(false, "查询异常");
        }
    }

}
```

```
List<String> schoolIdList = new ArrayList<>();
```

因为是在自己真实项目上引入的，所以有一些业务代码，请忽略。

重要方法介绍：

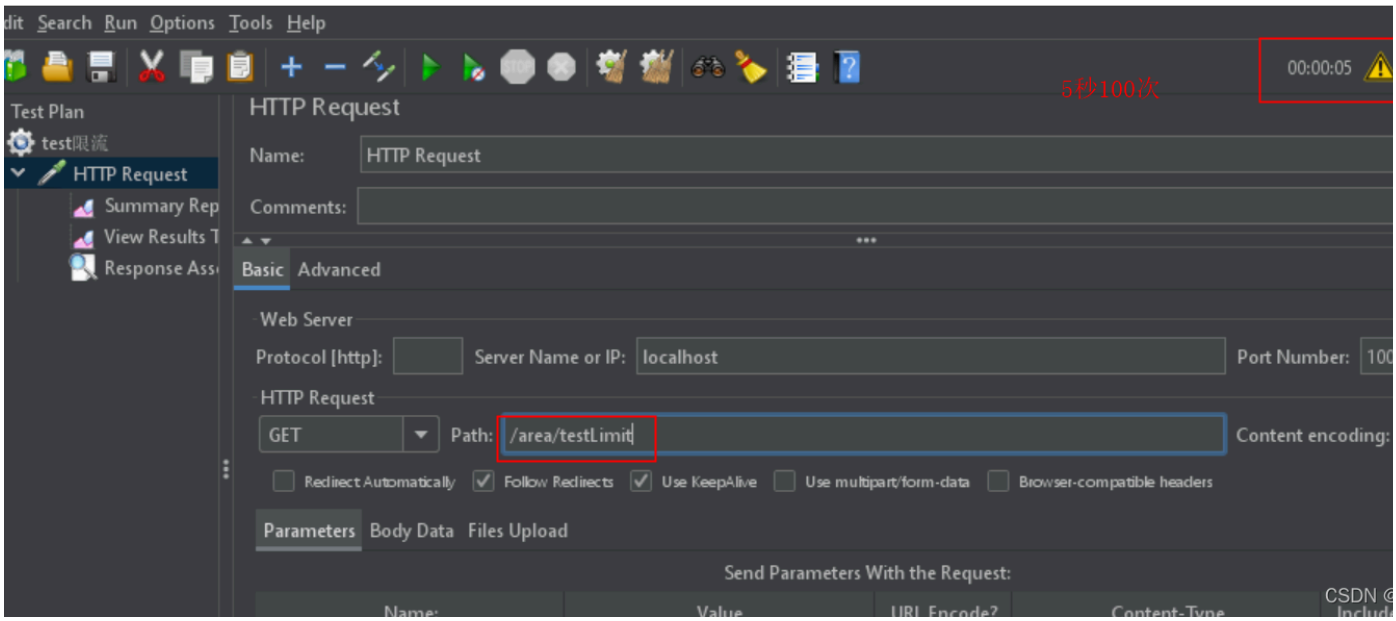
- acquire() 获取一个令牌, 该方法会阻塞直到获取到这一个令牌, 返回值为获取到这个令牌花费的时间
- acquire(int permits) 获取指定数量的令牌, 该方法也会阻塞, 返回值为获取到这 N 个令牌花费的时间
- tryAcquire() 判断是否能获取到令牌, 如果不能获取立即返回 false
- tryAcquire(int permits) 获取指定数量的令牌, 如果不能获取立即返回 false
- tryAcquire(long timeout, TimeUnit unit) 判断能否在指定时间内获取到令牌, 如果不能获取立即返回 false
- tryAcquire(int permits, long timeout, TimeUnit unit) 同上

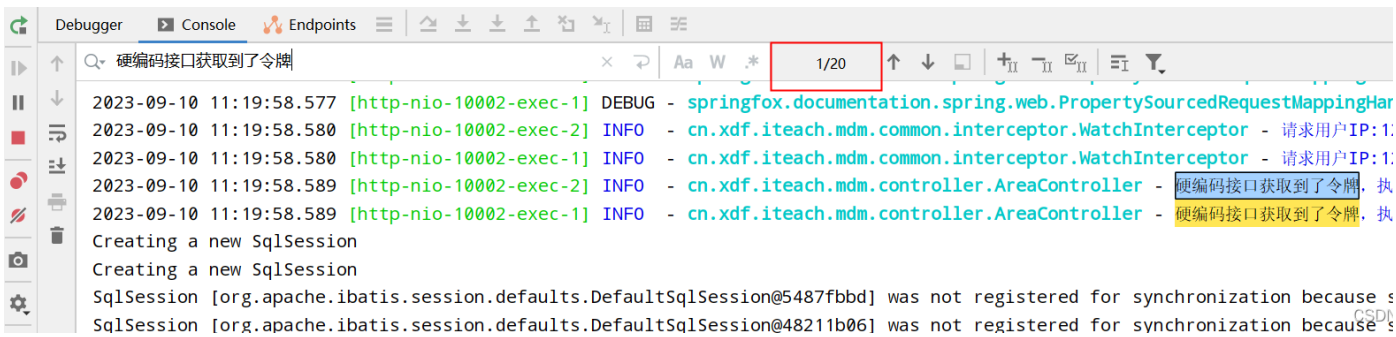
3.测试结果

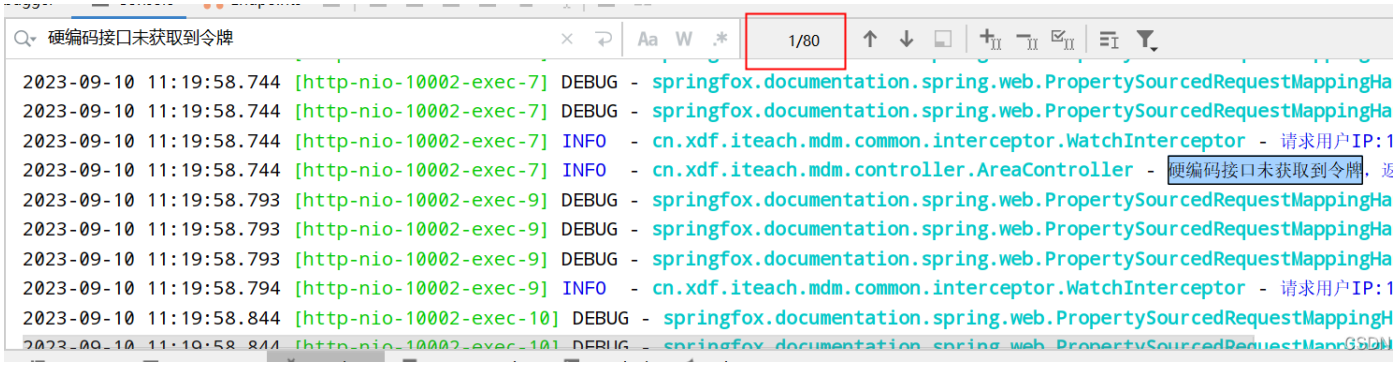
参数设置：设置每秒获取三个令牌，接口测试5秒执行100次请求。

结果：20次获取到了令牌，80次没有获取到令牌。

误差分析：标准结果应该是15次获取到了令牌，分析误差产生的可能原因有两点：1）采用单机发送请求，跟真实场景有差异。2)等待时间500毫秒也果。







二、注解方式

实际使用的时候我们肯定不能每次都硬编码，可以使用注解的方式。

自己写注解，基于aop 实现。

1.加入AOP依赖

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

2.自定义限流注解

```
import java.lang.annotation.ElementType;
import java.lang.annotation.Retention;
import java.lang.annotation.RetentionPolicy;
import java.lang.annotation.Target;
import java.util.concurrent.TimeUnit;

@Retention(RetentionPolicy.RUNTIME)//修饰注解，用来表示注解的生命周期
@Target({ElementType.METHOD})//注解的作用目标，这个表示注解到方法
public @interface Limit {

```

3.使用AOP切面拦截

```
@Slf4j
@Aspect//定义为切面
@Component
public class LimitAspect {

    /**
     * 不同的接口，不同的流量控制，map的key为Limit.key
     */
    private final Map<String , RateLimiter> limiterMap = Maps.newConcurrentMap();

    @Around("@annotation(cn.xdf.iteach.mdm.common.annotation.Limit)")
    public Object around(ProceedingJoinPoint joinPoint) throws Throwable {
        MethodSignature signature = (MethodSignature)joinPoint.getSignature();
        Method method = signature.getMethod();
        Limit limit = method.getAnnotation(Limit.class);//获取到方法上的Limit注解

        if(limit != null){
            RateLimiter rateLimiter = limiterMap.get(limit.key());
            if(null == rateLimiter){
                //创建当前key的RateLimiter
                rateLimiter = RateLimiter.create(limit.permitsPerSecond());
```

```
                limiterMap.put(limit.key(),rateLimiter);
                log.info("创建了新的令牌桶，key={},容量={}",limit.key(),limit.permitsPerSecond());
            }
            boolean success = rateLimiter.tryAcquire(0, limit.timeUnit());
            if(!success){
                log.info("获取令牌失败,key={}",limit.key());
                this.responseFail(limit.message());
                return null;
            }
            log.info("获取令牌成功,key={}",limit.key());

        }
        return joinPoint.proceed();
    }

    private void responseFail(String message) throws IOException {
        HttpServletResponse response=((ServletRequestAttributes) RequestContextHolder.getRequestAttributes()).getResponse();
        response.setCharacterEncoding("UTF-8");
        response.setContentType("application/json;charset=UTF-8");
        PrintWriter pw = response.getWriter();
        JSONObject object = new JSONObject();
        object.put("success",false);
        object.put("code",2001);
        object.put("message",message);
        pw.write(object.toJSONString());
        pw.flush();
        pw.close();
    }
}
```

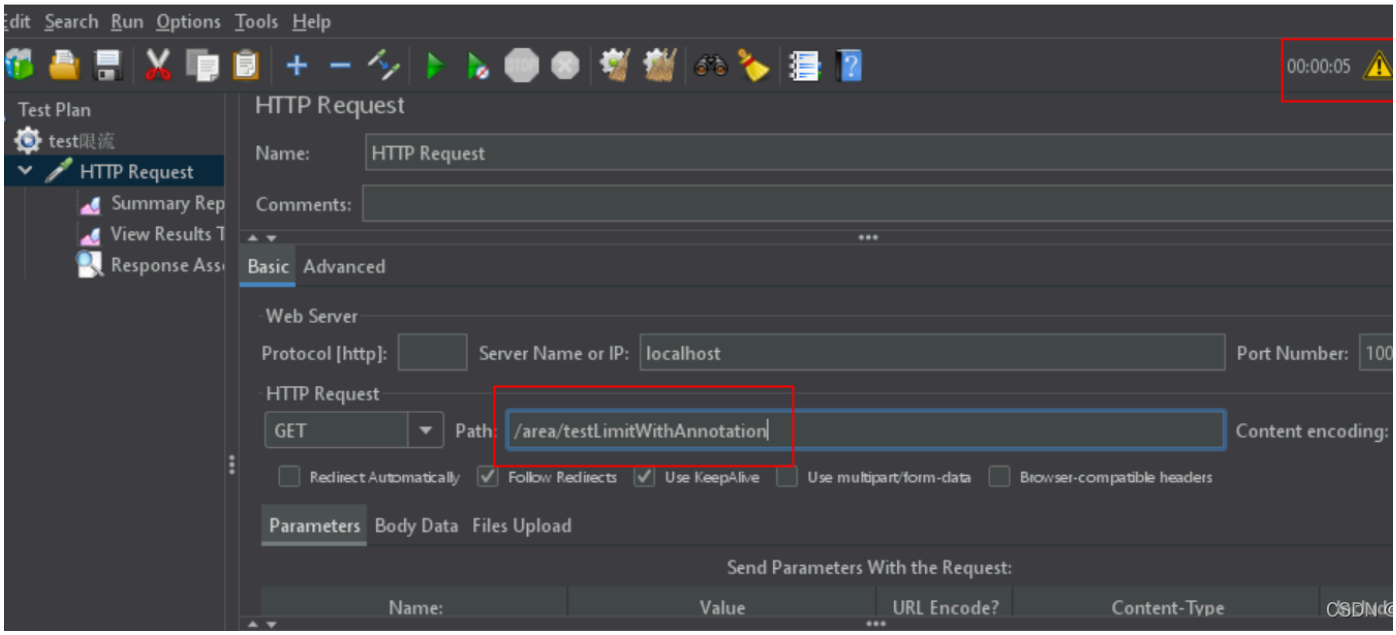
4.给接口添加注解

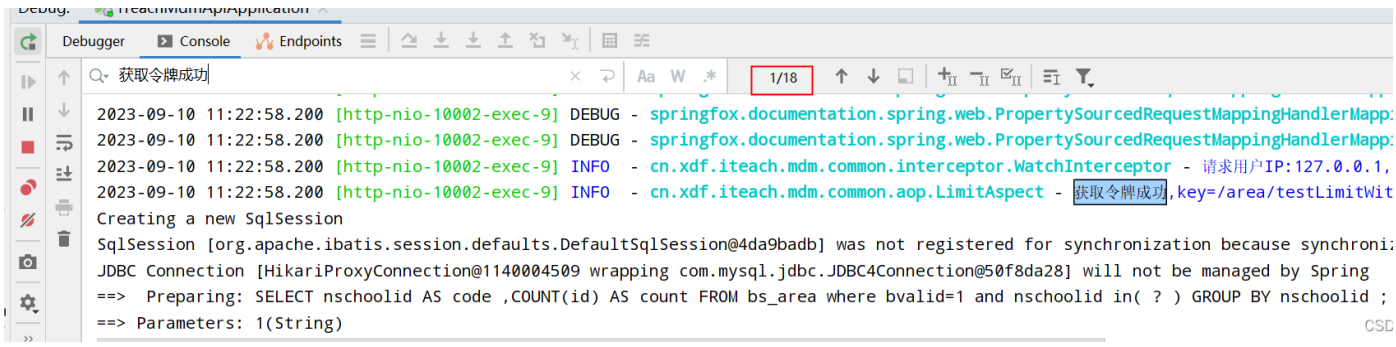
```
@Controller
@RequestMapping("/area")
public class AreaController {
    protected final Logger logger = LoggerFactory.getLogger(this.getClass());
    @Autowired
    private IAreaService areaService;



    @RequestMapping("/testLimitWithAnnotation")
    @ResponseBody
    @Limit(key="/area/testLimitWithAnnotation",permitsPerSecond = 3,timeOut = 500L)
    public ResponseEntity testLimitWithAnnotation(){

        try {
            List<String> schoolIdList = new ArrayList<>();
            schoolIdList.add("1");
            return areaService.queryCount(schoolIdList);
        } catch (Exception e) {
            logger.error("查询校区异常", Throwables.getStackTraceAsString(e));
            return new ResponseEntity(false, "查询异常");
        }
    }


}
```

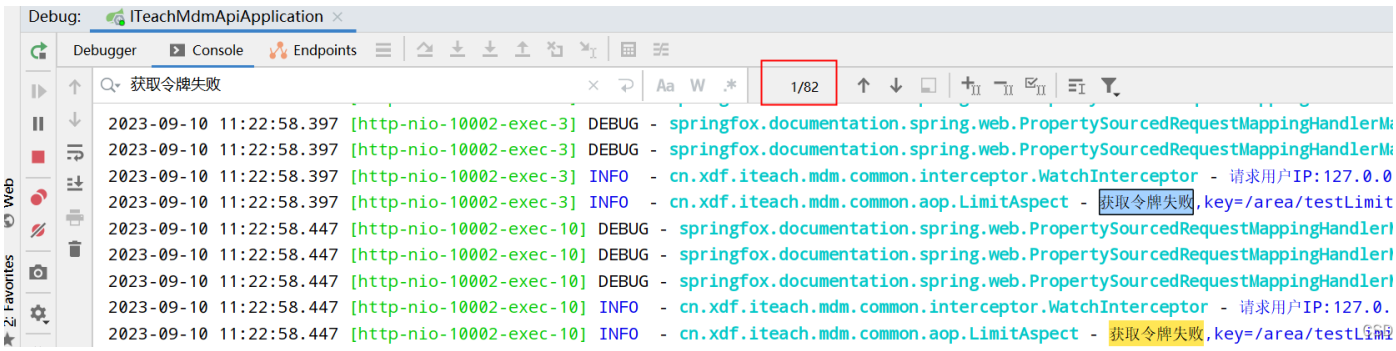
5.测试结果

参数设置：设置每秒获取三个令牌，接口测试5秒执行100次请求。

结果：18次获取到了令牌，82次没有获取到令牌。







三、其他限流方式

Guava是单机版的限流方式，还有一些其他方式的限流

1）基于sentinel限流（分布式版）

2）基于redis+lua限流（分布式版）

3）网关限流（分布式版）

有兴趣的同学可以尝试一下。

分布式版是控制的是接口的所有部署节点，准确性上肯定是更高的。但因为跨网络，性能上会稍差一些。

参考文档

1.SpringBoot 中使用Guava实现单机令牌桶限流 - 知乎

2.【限流】4 种常见的限流实现方案_限流怎么实现-CSDN博客

Guava限流器原理浅析 徐小陌的
本文并不过多深度剖析源码和原理。旨在以初学者的角度窥探Guava限流器的限流实现思路，并解答一些理解中存在的疑惑。

1 条评论  CSDN-Ada助手 热评 恭喜您写了第三篇博客！标题“Guava实现限流”听起来很有趣。非常感谢您分享关于Guava的实现限流的经验。我觉得…

guava限流器RateLimiter使用简介(Springboot实现)_guava ratelimiter...
限制时间窗口内的平均速率(如Guava的RateLimiter、nginx的limitreq模块,限制每秒的平均速率) 其他:比如如限制远程接口调用速率、限制MQ的消费速率。另外还可以根据

使用Guava实现限流器_guava 限流
使用Guava实现限流器 本文介绍了如何利用Google Guava库中的RateLimiter实现限流功能,基于令牌桶算法,创建全局限流器限制请求QPS,并在Spring MVC中添加自定义拦

SpringBoot使用 guava限流器RateLimiter (自定义注解+AOP实现) 最新发布
guava限流器RateLimiter使用简介(Springboot实现)_guava ratelimiter-CSDN博客

Guava限流 weixin_43589025的
Guava限流 文章目录Guava限流为什么要做限流原理漏桶算法令牌桶算法实战 为什么要做限流 通常我们的应用在部署之前都会先进行评估，有多少的调用量，需要多少台

guava之限流RateLimiter_guava ratelimiter
guava单机限流RateLimiter 参考:https://mp.weixin.qq.com/s/GOBmSOvWqpmLp2rijZ6q4w RateLimiter是基于令牌桶算法实现的一个限流组件,其代码看起来很简单,一共就

【项目实战】限流框架介绍 - 使用Guava RateLimiter限制请求速率
一、Guava入门介绍 Guava是一个流行的Java库,提供了许多实用程序类和函数。 【项目实战】Google Guava入门介绍 二、RateLimiter类介绍 其中一个实用程序是RateL

guava实现限流 qq_35137375的
在秒杀中，我们需要实现限制客户端的流量，常用的实现限流的方式有:hystrix、nginx、guava等方式。下面我们介绍下guava如何实现限流 1.maven引入 <dependency>

【Guava】使用Guava的RateLimiter做限流 weixin_34191845的
2019独角兽企业重金招聘Python工程师标准>>> ...

限流-Guava-RateLimiter_guava ratelimiter
3 Guava-RateLimiter的使用 3.1 概述 Google开源工具包Guava提供了限流工具类RateLimiter,该类基于令牌桶算法实现流量限制,使用十分方便,而且十分高效。 3.2 RateLi

Guava限流神器:RateLimiter使用指南
Guava限流神器:RateLimiter使用指南 1. 引言 可能有些小伙伴听到“限流”这个词就觉得头大,感觉像是一个既复杂又枯燥的话题。别急,小黑今天就要用轻松易懂的方式,带咱

guava限流 weixin_34318956的
<!--google guava 限流--><dependency> <groupId>com.google.guava</groupId> <artifactId>guava</artifactId> <version>23.0</version></dependency> ...

基于Zookeeper和guava动态限流 源码
总结来说，结合Zookeeper和Guava实现的动态限流方案，提供了灵活性和可扩展性，能够适应不断变化的系统需求。通过对"基于Zookeeper和guava动态限流 源码"的深

SpringCloud Zuul过滤器和谷歌Gauva实现限流
SpringCloud Zuul 过滤器和谷歌 Guava 实现限流 在本篇文章中，我们将主要介绍如何使用 SpringCloud Zuul 过滤器和谷歌 Guava 实现限流。限流是指对服务的并发请求

java实现令牌桶限流
限流是对某一时间窗口内的请求数进行限制，保持...常用的限流算法有令牌桶和和漏桶，而Google开源项目Guava中的RateLimiter使用的就是令牌桶控制算法。 在开发高

基于Redis的限流器的实现(示例讲解)
限流器在软件开发中扮演着至关重要的角色，它能够防止系统被大量请求淹没，确保服务的...在设计和实现限流器时，应考虑到系统的整体架构、并发量以及对服务稳定性

springboot结合自定义注解aop实现限流
Springboot结合自定义注解的使用 Springboot结合aop使用 接口限流思路

Guava(二)限流算法的使用 姑
Guava限流算法的具体分析

guava之限流RateLimiter 热门推荐 sinat_14913533的
常用的限流方式和场景有： 限制总并发数（比如数据库连接池、线程池） 限制瞬时并发数（如nginx的limitconn模块，用来限制瞬时并发连接数，Java的Semaphore也可

使用Guava实现限流器 m0_67596808的
@author linzhiqiang @date 2019/4/17 */ public abstract class AbstractInterceptor extends HandlerInterceptorAdapter { private Logger logger = LoggerFactory.getLogger(

限流模式-Guava的RateLimiter loredp的
目前有几种常见的限流方式： 1、通过限制单位时间段内调用量来限流 2、通过限制系统的并发调用程度来限流 3、使用漏桶（Leaky Bucket）算法来进行限流 4、使用令

Guava——平滑限流 ShiXueTanLang的
1.常用限流方法对于一个应用系统来说一定会有极限并发/请求数，即总有一个TPS/QPS阈值，如果超了阈值则系统就会不响应用户请求或响应的非常慢，因此我们最好进

Guava中常用的4种经典限流算法介绍 gb4215287的
限流算法

Spring Cloud Gateway如何实现限流?

Spring Cloud Gateway可以通过集成限流组件来实现限流功能。在Spring Cloud Gateway中，可以使用Redis、Guava、Bucket4j等组件进行限流。 下面以Redis实现限流

罗一（LY）
码龄1年 暂无认证

10	117万+	10万+	4343	
原创	周排名	总排名	访问	等级

135	33	34	10	51
积分	粉丝	获赞	评论	收藏

私信 关注

搜博主文章

热门文章

Guava实现限流 1631
Netty(一) netty实现服务端小Demo 531
mysql的S锁和X锁小测试 396
Netty(二) netty必备前置基础知识 383
java实现singleflight 360

分类专栏

Netty 2篇
mysql 2篇
JAVA 1篇
缓存 1篇
高可用 1篇

最新评论

upsert小验证
CSDN-Ada助手: MySQL入门 技能树或许可以帮到你：https://edu.csdn.net/skill/my ...

随机IO小测试
CSDN-Ada助手: 恭喜博主发布了第9篇博客“随机IO小测试”，内容精彩纷呈！继续伪 ...

java零拷贝小测试
普通网友: 支持一下！我也写了一篇获取【大厂面试真题解析、核心开发学习笔记 ...

java零拷贝小测试
CSDN-Ada助手: 推荐 Java 技能树：https://edu.csdn.net/skill/java?utm_source= ...

Netty(一) netty实现服务端小Demo
CSDN-Ada助手: 恭喜作者发布了第6篇博客，内容涉及Netty实现服务端小Demo， ...

最新文章

upsert小验证

随机IO小测试

java零拷贝小测试

2024年 3篇 2023年 7篇

目录

前言
一、硬编码方式
1.引入guava依赖包
2.给接口添加限流逻辑
3.测试结果
二、注解方式
1.加入AOP依赖
2.自定义限流注解
3.使用AOP切面拦截
4.给接口添加注解
5.测试结果
三、其他限流方式
参考文档

罗一（LY） 关注