# 基本概念

Guava 是 Google 开源的一个 Java 工具库，提供了大量高效、简洁的工具类和方法，涵盖了集合、缓存、并发、I/O、函数式编程等场景。

# 实现接口限流

## 基本概念

接口限流（Rate Limiting）用于控制接口的请求频率，防止突发流量导致系统过载。Guava 的 RateLimiter 基于 **令牌桶算法**，可以平滑地限制请求速率。

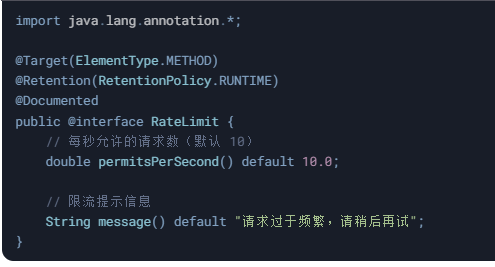
## 实现步骤

### 1、添加guava依赖



### 2、创建自定义限流注解

定义一个注解 @RateLimit，用于标记需要限流的接口方法：



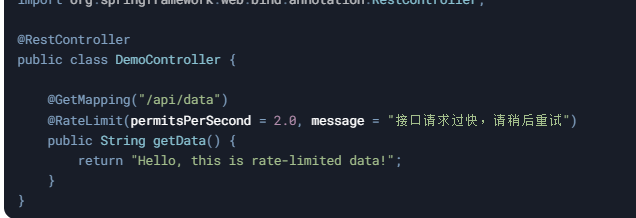
### 3、实现限流切面（AOP）

使用 AOP 拦截带有 @RateLimit 注解的方法，并进行限流控制：



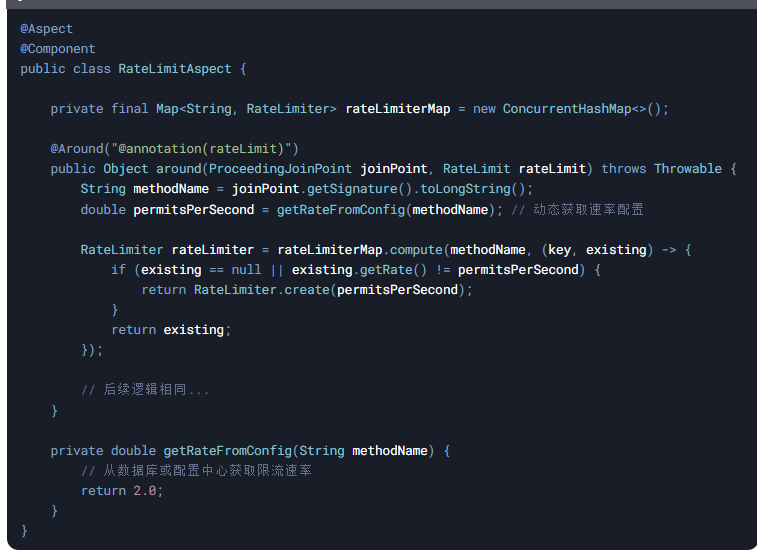
### 4、在 Controller中使用限流注解

在需要限流的接口方法上添加 @RateLimit 注解



### 5、高级配置

若需动态修改限流速率的配置（如从数据库读取），可以扩展切面逻辑



### 注意事项

**1、线程安全**：

RateLimiter 实例需要缓存，避免重复创建。使用 ConcurrentHashMap 确保线程安全。

**2、异常处理**：

统一处理限流异常，避免直接返回不友好的错误信息。

**3、性能影响**：

RateLimiter 本身性能较高，但需注意切面的执行效率。

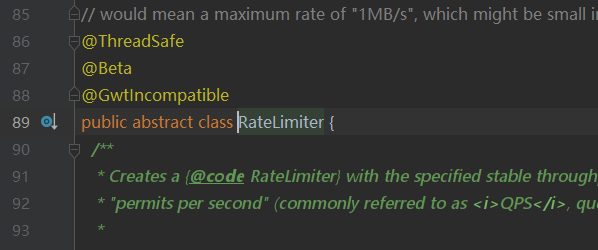
# 类

## RateLimiter

### 基本概念

RateLimiter 是 Google Guava 库提供的一个限流工具，基于**令牌桶算法**实现，用于控制资源访问的速率。它支持平滑的请求限流，并能处理突发流量，适用于高并发场景下的流量整形与资源保护。

### 源码



### 核心原理

**1、令牌生成**：系统以固定速率向桶中添加令牌（例如每秒添加 5 个令牌）。

**2、请求消费**：请求到达时需获取令牌，获取成功则放行，否则阻塞或拒绝。

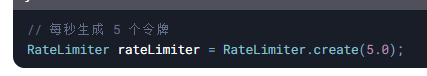
**3、允许突发**：桶中可累积令牌，允许一次性处理突发请求（如桶满时，可瞬间处理多个请求）。

### 创建RateLimiter

通过静态工厂方法创建实例，支持两种模式：

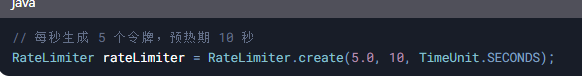
#### ****1、平滑突发模式 (****SmoothBursty****)****

特点：允许突发流量，适用于对延迟不敏感的场景。



#### **2、**平滑预热模式 (**SmoothWarmingUp**)****

**特点**：系统启动时逐步增加令牌生成速率至设定值，避免冷启动过载。



### 方法

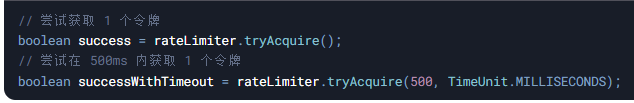
#### public double acquire()

请求令牌，若不足则阻塞，直到获取成功。



#### public boolean tryAcquire()

立即返回是否成功获取令牌，不阻塞。



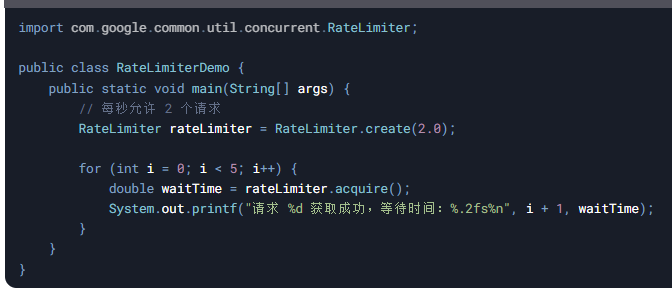
#### public final void setRate(double permitsPerSecond)

动态修改令牌生成速率。

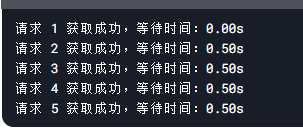


### 代码示例

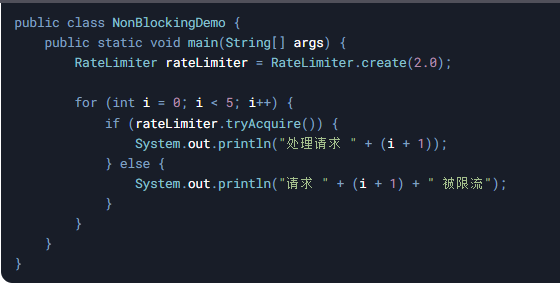
#### 1、基础限流



输出：



#### 2、非阻塞请求处理



输出：



### 使用场景

#### 1、API 限流

防止外部服务调用频率过高。

#### 2、资源保护

避免数据库、文件系统等资源被过载。

#### 3、流量整形

平滑突发流量，提升系统稳定性。

#### 4、任务调度

控制后台任务的处理速率。

### 注意事项

**1、线程安全**：RateLimiter 实例是线程安全的，可跨线程共享。

**2、分布式限流**：Guava 的 RateLimiter 仅适用于单机限流，分布式系统需结合 Redis 或中间件（如 Sentinel）。

**3、资源释放**：未使用的令牌不会累积超过桶容量，需合理设置桶大小。

### 与其他限流方案对比

