Aop限流实现解决方案

01、限流

在业务场景中，为了限制某些业务的并发，造成接口的压力，需要增加限流功能。

02、限流的成熟解决方案

- guava (漏斗算法 + 令牌算法) （单机限流）
- redis + lua + ip 限流（比较推荐）（分布式限流）
- nginx 限流 （源头限流）
- .....

03、 限流的目的

- 保护服务的资源泄露
- 解决服务器的高可压，减少服务器并发

04、安装**redis**服务

安装**redis**

```
wget http://download.redis.io/releases/redis-6.0.6.tar.gz
tar xzf redis-6.0.6.tar.gz
cd redis-6.0.6
make
```

修改redis.conf

```
daemonize yes
# bind 127.0.0.1
protected-mode no
requirepass mkxiaoer1986.
```

如果你之前启动过redis服务器，请麻烦一定要先检查，把服务杀掉，在启动

```
ps -ef | grep redis
kill redispid
```

然后重启服务，一定指定配置文件启动

```
./src/redis-server ./redis.conf
```

开放端口

阿里云【安全组】开放6379端口

如果执行编译报错

如果在安装redis过程中。make报错了。不要慌张，可能是没有编译组件，系统文件有缺失，你先执行：

```
yum -y install centos-release-scl
yum -y install devtoolset-9-gcc devtoolset-9-gcc-c++ devtoolset-9-binutils
scl enable devtoolset-9 bash
```

然后在

```
make
```

05、springboto整合redis

01、添加redis依赖

```
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-web</artifactId>
</dependency>
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-aop</artifactId>
</dependency>
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-data-redis</artifactId>
</dependency>
<dependency>
    <groupId>org.projectlombok</groupId>
    <artifactId>lombok</artifactId>
</dependency>
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-test</artifactId>
    <scope>test</scope>
```

```
</dependency>
```

02、全局配置文件配置redis

在application.yml文件配置如下：

```yaml
spring:
  redis:
    host: xxxxx
    port: 6379
    database: 0
    password: xxxxxx
    lettuce:
      pool:
        max-active: 20
        max-wait: -1
        max-idle: 5
        min-idle: 0

```

03、定义redis的配置类

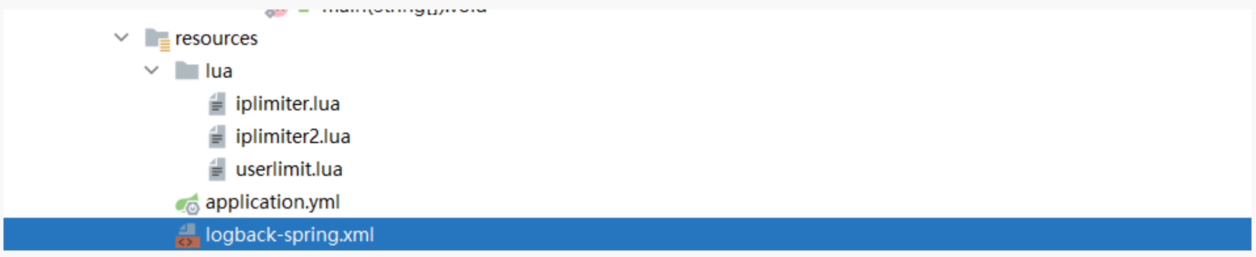
```java
package com.kuangstudy.config;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import
org.springframework.data.redis.connection.RedisConnectionFact
ory;
import org.springframework.data.redis.core.RedisTemplate;
import
org.springframework.data.redis.serializer.GenericJackson2Json
RedisSerializer;
import
org.springframework.data.redis.serializer.StringRedisSerializ
er;
```

```java

/**
 * @author 飞哥
 * @Title: 学相伴出品
 * @Description: 我们有一个学习网站：https://www.kuangstudy.com
 * @date 2021/5/20 13:16
 */
@Configuration
public class RedisConfiguration {

    /**
     * @return org.springframework.data.redis.core.RedisTemplate<java.lang.String, java.lang.Object>
     * @Author 徐柯
     * @Description 改写redistemplate序列化规则
     * @Date 13:20 2021/5/20
     * @Param [redisConnectionFactory]
     **/
    @Bean
    public RedisTemplate<String, Object> redisTemplate(RedisConnectionFactory redisConnectionFactory) {
        // 1: 开始创建一个redistemplate
        RedisTemplate<String, Object> redisTemplate = new RedisTemplate<>();
        // 2:开始redis连接工厂跪安了
        redisTemplate.setConnectionFactory(redisConnectionFactory);
        // 创建一个json的序列化方式
        GenericJackson2JsonRedisSerializer jackson2JsonRedisSerializer = new GenericJackson2JsonRedisSerializer();
        // 设置key用string序列化方式
        redisTemplate.setKeySerializer(new StringRedisSerializer());
        // 设置value用jackjson进行处理
 redisTemplate.setValueSerializer(jackson2JsonRedisSerializer
);
        // hash也要进行修改
        redisTemplate.setHashKeySerializer(new
StringRedisSerializer());
 redisTemplate.setHashValueSerializer(jackson2JsonRedisSerial
izer);
        // 默认调用
        redisTemplate.afterPropertiesSet();
        return redisTemplate;
    }
}
```

上面其实springboot本身存在RedisAutoConfiguration其实里面已经初始化好了RedisTemplate。这个redistemplate其实可以直接去使用。但是自身RedisTemplate序列化的key的时候是以Object的类型进行序列化，所以看到"\xac\xed\x00\x05t\x00\x14age1111111111111111" 不友好。所以就覆盖了。

06、定义限流lua脚本

新建一个iplimite.lua文件，放在resources目录下的lua文件夹下：



```
-- 为某个接口的请求IP设置计数器，比如：127.0.0.1请求课程接口
-- KEYS[1] = 127.0.0.1 也就是用户的IP
-- ARGV[1] = 过期时间 30m
-- ARGV[2] = 限制的次数
local limitCount = redis.call('incr',KEYS[1]);
```

```lua
if limitCount == 1 then
    redis.call("expire",KEYS[1],ARGV[2])
end
-- 如果次数还没有过期，并且还在规定的次数内，说明还在请求同一接口
if limitCount > tonumber(ARGV[1]) then
    return false
end

return true
```

07、Lua限流脚本配置类

lua配置类主要是去加载lua文件的内容，到时内存中。方便redis去读取和控制。

```java
package com.kuangstudy.config;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import org.springframework.core.io.ClassPathResource;
import org.springframework.data.redis.core.script.DefaultRedisScript;
import org.springframework.scripting.support.ResourceScriptSource;

/**
 * @author 飞哥
 * @Title: 学相伴出品
 * @Description: 我们有一个学习网站：https://www.kuangstudy.com
 * @date 2021/5/21 12:01
 */
@Configuration
public class LuaConfiguration {

    /**
     * 将lua脚本的内容加载出来放入到DefaultRedisScript
     * @return
     */
    @Bean
    public DefaultRedisScript<Boolean> ipLimitLua() {
        DefaultRedisScript<Boolean> defaultRedisScript = new DefaultRedisScript<>();
        defaultRedisScript.setScriptSource(new ResourceScriptSource(new ClassPathResource("lua/iplimiter.lua")));
        defaultRedisScript.setResultType(Boolean.class);
        return defaultRedisScript;
    }

    /**
     * 将lua脚本的内容加载出来放入到DefaultRedisScript
     * @return
     */
    @Bean
    public DefaultRedisScript<Boolean> ipLimiterLuaScript() {
        DefaultRedisScript<Boolean> defaultRedisScript = new DefaultRedisScript<>();
        defaultRedisScript.setScriptSource(new ResourceScriptSource(new ClassPathResource("lua/iplimiter2.lua")));
        defaultRedisScript.setResultType(Boolean.class);
        return defaultRedisScript;
    }

}
```

08、限流注解

```java
package com.kuangstudy.aop;

import java.lang.annotation.*;

/**
```

```java
 * @author 飞哥
 * @Title: 学相伴出品
 * @Description: 飞哥B站地址:
https://space.bilibili.com/490711252
 * 记得关注和三连哦!
 * @Description: 我们有一个学习网站: https://www.kuangstudy.com
 * @date 2021/12/22 23:03
 */
@Target(ElementType.METHOD)
@Retention(RetentionPolicy.RUNTIME)
@Documented
public @interface AcessLimter {
    // 每timeout限制请求的个数
    int limit() default 10;

    // 时间, 单位默认是秒
    int timeout() default 1;
}

```

09、请求获取用户IP工具类

```java
package com.kuangstudy.aop;

import javax.servlet.http.HttpServletRequest;

/**
 * @author 飞哥
 * @Title: 学相伴出品
 * @Description: 飞哥B站地址:
https://space.bilibili.com/490711252
 * 记得关注和三连哦!
 * @Description: 我们有一个学习网站: https://www.kuangstudy.com
 * @date 2021/12/22 23:18
 */
public class RequestUtils {

    public static String getIpAddr(HttpServletRequest request)
    {
        if (request == null)
        {
            return "unknown";
        }
        String ip = request.getHeader("x-forwarded-for");
        if (ip == null || ip.length() == 0 || "unknown".equalsIgnoreCase(ip))
        {
            ip = request.getHeader("Proxy-Client-IP");
        }
        if (ip == null || ip.length() == 0 || "unknown".equalsIgnoreCase(ip))
        {
            ip = request.getHeader("X-Forwarded-For");
        }
        if (ip == null || ip.length() == 0 || "unknown".equalsIgnoreCase(ip))
        {
            ip = request.getHeader("WL-Proxy-Client-IP");
        }
        if (ip == null || ip.length() == 0 || "unknown".equalsIgnoreCase(ip))
        {
            ip = request.getHeader("X-Real-IP");
        }

        if (ip == null || ip.length() == 0 || "unknown".equalsIgnoreCase(ip))
        {
            ip = request.getRemoteAddr();
        }

        return "0:0:0:0:0:0:0:1".equals(ip) ? "127.0.0.1" : ip;
    }
```

```java
}

```

10、限流AOP切面类

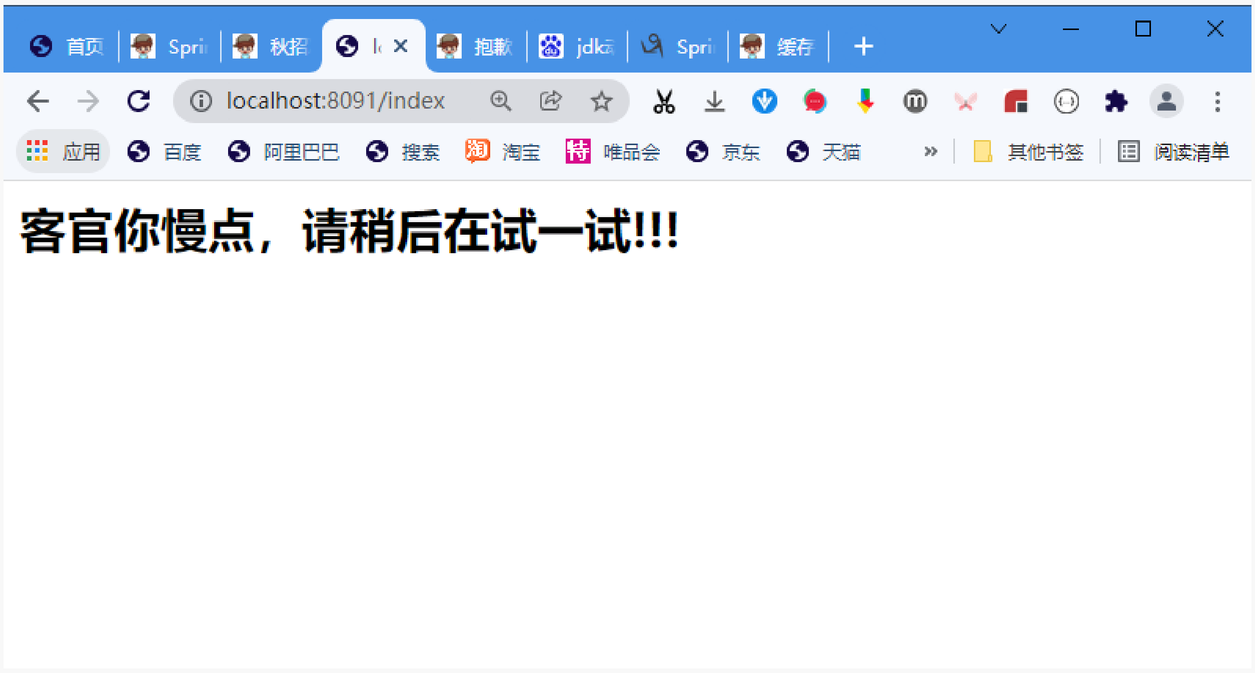
```java
package com.kuangstudy.aop;

import com.google.common.collect.Lists;
import com.sun.org.apache.xpath.internal.operations.Bool;
import lombok.extern.slf4j.Slf4j;
import org.aspectj.lang.JoinPoint;
import org.aspectj.lang.annotation.Aspect;
import org.aspectj.lang.annotation.Before;
import org.aspectj.lang.annotation.Pointcut;
import org.aspectj.lang.reflect.MethodSignature;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.data.redis.core.StringRedisTemplate;
import org.springframework.data.redis.core.script.DefaultRedisScript;
import org.springframework.stereotype.Component;
import org.springframework.web.context.request.RequestContextHolder;
import org.springframework.web.context.request.ServletRequestAttributes;

import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import java.io.PrintWriter;
import java.lang.reflect.Method;

/**
 * @author 飞哥
 * @Title: 学相伴出品
 * @Description: 飞哥B站地址：https://space.bilibili.com/490711252
 * 记得关注和三连哦！
 * @Description: 我们有一个学习网站：https://www.kuangstudy.com
 * @date 2021/12/22 23:05
 */
@Component
@Aspect
@Slf4j
public class LimiterAspect {

    @Autowired
    private StringRedisTemplate stringRedisTemplate;
    @Autowired
    private DefaultRedisScript<Boolean> ipLimiterLuaScript;
    @Autowired
    private DefaultRedisScript<Boolean> ipLimitLua;

    // 1: 切入点
    @Pointcut("@annotation(com.kuangstudy.aop.AcessLimter)")
    public void limiterPonicut() {
    }

    @Before("limiterPonicut()")
    public void limiter(JoinPoint joinPoint) {
        log.info("限流进来了.......");
        // 1：获取方法的签名作为key
        MethodSignature methodSignature = (MethodSignature) joinPoint.getSignature();
        Method method = methodSignature.getMethod();
        String classname = methodSignature.getMethod().getDeclaringClass().getName();
        String packageName = methodSignature.getMethod().getDeclaringClass().getPackage().getName();
        log.info("classname:{},packageName:{}",classname,packageName);
        // 4: 读取方法的注解信息获取限流参数
        AcessLimter annotation = method.getAnnotation(AcessLimter.class);
        // 5: 获取注解方法名
        String methodNameKey = method.getName();
        // 6: 获取服务请求的对象
        ServletRequestAttributes requestAttributes = (ServletRequestAttributes) RequestContextHolder.getRequestAttributes();
        HttpServletRequest request = requestAttributes.getRequest();
        HttpServletResponse response = requestAttributes.getResponse();
        String userIp = RequestUtils.getIpAddr(request);
        log.info("用户IP是: .......{}", userIp);
        // 7: 通过方法反射获取注解的参数
        Integer limit = annotation.limit();
        Integer timeout = annotation.timeout();
        String redisKey = method + ":" + userIp;
        // 8: 请求lua脚本
        Boolean acquired =  stringRedisTemplate.execute(ipLimitLua, Lists.newArrayList(redisKey), limit.toString(), timeout.toString());
        // 如果超过限流限制
        if (!acquired) {
            // 抛出异常，然后让全局异常去处理
            response.setCharacterEncoding("UTF-8");
            response.setContentType("text/html;charset=UTF-8");
            try (PrintWriter writer = response.getWriter();) {
                response.getWriter().print("<h1>客官你慢点，请稍后在试一试!!!</h1>");
            } catch (Exception ex) {
                throw new RuntimeException("客官你慢点，请稍后在试一试!!!");
            }
        }
```

```
    }
}
```

11、限流测试Controller

```java
package com.kuangstudy.controller;

import com.kuangstudy.aop.AcessLimter;
import lombok.extern.java.Log;
import
org.springframework.beans.factory.annotation.Autowired;
import org.springframework.data.redis.core.RedisTemplate;
import org.springframework.web.bind.annotation.GetMapping;
import
org.springframework.web.bind.annotation.RestController;

/**
 * @author 飞哥
 * @Title: 学相伴出品
 * @Description: 飞哥B站地址:
https://space.bilibili.com/490711252
 * 记得关注和三连哦!
 * @Description: 我们有一个学习网站: https://www.kuangstudy.com
 * @date 2021/12/22 22:45
 */
@RestController
public class UserController {

    @GetMapping("/index")
    @AcessLimter(timeout = 1,limit = 5)
    public String index() {
        // 分布锁
        return "success";
    }

    @GetMapping("/index2")
    public String index2() {
        return "success";
    }

}

```

访问刷新：http://localhost:8091/index



客官你慢点，请稍后在试一试!!!

12、限流的核心代码

- 获取请求对象

```java
// 3：获取服务请求的对象
ServletRequestAttributes requestAttributes =
(ServletRequestAttributes)
RequestContextHolder.getRequestAttributes();
HttpServletRequest request =
requestAttributes.getRequest();
HttpServletResponse response =
requestAttributes.getResponse();
```

- key唯一性

  考虑：包名 +类名+方法名 + userIp

```java
// 1：获取方法的签名作为key
MethodSignature methodSignature = (MethodSignature)
joinPoint.getSignature();
Method method = methodSignature.getMethod();
String classname = methodSignature.getClass().getName();
String packageName =
methodSignature.getClass().getPackage().getName();
log.info("classname:{},packageName:
{}",classname,packageName);
```

- 反射获取方法注解的信息

```java
// 4: 读取方法的注解信息获取限流参数
AcessLimter annotation =
method.getAnnotation(AcessLimter.class);
// 注意这个代码，要加下判断，防止没加注解的方法乱入的问题
if (annotation == null) {
    return;
}
```

- 限流核心

```java
// 4: 请求lua脚本
Boolean acquired = 
 stringRedisTemplate.execute(ipLimiterLuaScript,
Lists.newArrayList(redisKey), limit.toString(),
timeout.toString());
// 如果超过限流限制
if (!acquired) {
    // 抛出异常，然后让全局异常去处理
    response.setCharacterEncoding("UTF-8");
    response.setContentType("text/html;charset=UTF-8");
    try (PrintWriter writer = response.getWriter();) {
        response.getWriter().print("<h1>客官你慢点，请稍后在试一试!!!</h1>");
    } catch (Exception ex) {
        throw new RuntimeException("客官你慢点，请稍后在试一试!!!");
    }
}
```

- 获取Ip的时候

```java
package com.kuangstudy.aop;

import javax.servlet.http.HttpServletRequest;

/**
 * @author 飞哥
 * @Title: 学相伴出品
 * @Description: 飞哥B站地址：https://space.bilibili.com/490711252
 * 记得关注和三连哦!
 * @Description: 我们有一个学习网站：https://www.kuangstudy.com
 * @date 2021/12/22 23:18
 */
public class RequestUtils {

    public static String getIpAddr(HttpServletRequest request)
    {
        if (request == null)
        {
            return "unknown";
        }
        String ip = request.getHeader("x-forwarded-for");
        if (ip == null || ip.length() == 0 || "unknown".equalsIgnoreCase(ip))
        {
            ip = request.getHeader("Proxy-Client-IP");
        }
        if (ip == null || ip.length() == 0 || "unknown".equalsIgnoreCase(ip))
        {
            ip = request.getHeader("X-Forwarded-For");
        }
        if (ip == null || ip.length() == 0 || "unknown".equalsIgnoreCase(ip))
        {
            ip = request.getHeader("WL-Proxy-Client-IP");
        }
        if (ip == null || ip.length() == 0 || "unknown".equalsIgnoreCase(ip))
        {
            ip = request.getHeader("X-Real-IP");
        }

        if (ip == null || ip.length() == 0 || "unknown".equalsIgnoreCase(ip))
        {
            ip = request.getRemoteAddr();
        }

        return "0:0:0:0:0:0:0:1".equals(ip) ? "127.0.0.1" : ip;
    }
```

```
}

```

- Nginx代理拦截IP问题？

  在本机返回的都是：127.0.0.1，但是在服务器肯定要获取真实用户ip。但是还是返回127.0.0.1为为什么：nginx的反向代理的问题。把目标tomcat服务器request对象做了反向代理。所有你获取不真实的用户IP.

```


#以下属性中，以ssl开头的属性表示与证书配置有关。
server {
    listen 443 ssl;
    #配置HTTPS的默认访问端口为443。
    #如果未在此处配置HTTPS的默认访问端口，可能会造成Nginx无法启动。
    #如果您使用Nginx 1.15.0及以上版本，请使用listen 443 ssl代替listen 443和ssl on。
    server_name www.itbooking.net; #需要将yourdomain.com替换成证书绑定的域名。
    root html;
    index index.html index.htm;
    ssl_certificate cert/6179501_www.itbooking.net.pem;  #需要将cert-file-name.pem替换成已上传的证书文件的名称。
    ssl_certificate_key cert/6179501_www.itbooking.net.key; #需要将cert-file-name.key替换成已上传的证书密钥文件的名称。
    ssl_session_timeout 5m;
    ssl_ciphers ECDHE-RSA-AES128-GCM-SHA256:ECDHE:ECDH:AES:HIGH:!NULL:!aNULL:!MD5:!ADH:!RC4;
    #表示使用的加密套件的类型。
    ssl_protocols TLSv1 TLSv1.1 TLSv1.2; #表示使用的TLS协议的类型。
    ssl_prefer_server_ciphers on;
    location / {
        # 让程序能够正常的获取到用户的IP
        proxy_set_header Host $http_host;
```

```
            proxy_set_header X-Real-IP $remote_addr;
            proxy_set_header X-Forwarded-For $remote_addr;
            proxy_pass http://tomcatservers;
        }
}

```