**10-4 限流组件的实现**

自定义注解：

|  |
| --- |
| Java package org.qiyu.live.web.starter.config;  import java.lang.annotation.\*;  /\*\*  \* @Author idea  \* @Date: Created in 14:04 2023/8/5  \* @Description  \*/ @Documented @Target(ElementType.METHOD) @Retention(RetentionPolicy.RUNTIME) public @interface RequestLimit {   /\*\*  \* 允许请求的量  \*  \* @return  \*/  int limit();   /\*\*  \* 指定时间范围，单位秒  \*  \* @return  \*/  int second();   /\*\*  \* 如果出现了拦截，那么就按照msg文案进行提示  \*  \* @return  \*/  String msg() default "请求过于频繁"; } |

核心代码：

|  |
| --- |
| Java package org.qiyu.live.web.starter.context;  import jakarta.annotation.Resource; import jakarta.servlet.http.HttpServletRequest; import jakarta.servlet.http.HttpServletResponse; import org.qiyu.live.web.starter.config.RequestLimit; import org.qiyu.live.web.starter.error.ErrorAssert; import org.qiyu.live.web.starter.error.QiyuBaseError; import org.qiyu.live.web.starter.error.QiyuErrorException; import org.slf4j.Logger; import org.slf4j.LoggerFactory; import org.springframework.beans.factory.annotation.Value; import org.springframework.data.redis.core.RedisTemplate; import org.springframework.web.method.HandlerMethod; import org.springframework.web.servlet.HandlerInterceptor;  import java.util.Optional; import java.util.concurrent.TimeUnit;  /\*\*  \* 对于重复请求，要有专门的拦截器去处理  \*  \* @Author idea  \* @Date: Created in 14:06 2023/8/5  \* @Description  \*/ public class RequestLimitInterceptor implements HandlerInterceptor {   private static final Logger LOGGER = LoggerFactory.getLogger(RequestLimitInterceptor.class);   @Resource  private RedisTemplate<String, Object> redisTemplate;  @Value("${spring.application.name}")  private String applicationName;   @Override  public boolean preHandle(HttpServletRequest request, HttpServletResponse response, Object handler) throws Exception {  HandlerMethod handlerMethod = (HandlerMethod) handler;  boolean hasLimit = handlerMethod.getMethod().isAnnotationPresent(RequestLimit.class);  if (hasLimit) {  //是否需要限制请求  RequestLimit requestLimit = handlerMethod.getMethod().getAnnotation(RequestLimit.class);  Long userId = QiyuRequestContext.getUserId();  if (userId == null) {  return true;  }  //(userId + requestValue),md5,->string,  // /user/login  String requestKey = applicationName + ":" + request.getRequestURI() + ":" + userId;  int limit = requestLimit.limit();  int second = requestLimit.second();  Integer reqTime = (Integer) Optional.ofNullable(redisTemplate.opsForValue().get(requestKey)).orElse(0);  //如果是首次请求  if (reqTime == 0) {  redisTemplate.opsForValue().increment(requestKey, 1);  redisTemplate.expire(requestKey, second, TimeUnit.SECONDS);  return true;  } else if (reqTime < limit) {  redisTemplate.opsForValue().increment(requestKey, 1);  return true;  }  //直接抛出全局异常，让异常捕获器处理  LOGGER.error("[RequestLimitInterceptor] userId is {},req too much", userId);  throw new QiyuErrorException(-1, requestLimit.msg());  } else {  return true;  }  } } |