**[#](https://pig4cloud.com/" \l "pom%E4%BE%9D%E8%B5%96) POM依赖**

<!--日志处理-->

<dependency>

<groupId>com.pig4cloud</groupId>

<artifactId>pig-common-log</artifactId>

<version>${log.version}</version>

</dependency>

[**#**](https://pig4cloud.com/#%40syslog-%E6%B3%A8%E8%A7%A3) **@SysLog 注解**

* 接口上使用@SysLog 注释当前接口的作用即可

@SysLog("添加终端")

@PostMapping

@PreAuthorize("@pms.hasPermission('sys\_client\_add')")

public R add(@Valid @RequestBody SysOauthClientDetails sysOauthClientDetails) {

return new R<>(sysOauthClientDetailsService.save(sysOauthClientDetails));

}

[**#**](https://pig4cloud.com/#%E5%8E%9F%E7%90%86%E8%AE%B2%E8%A7%A3) **原理讲解**

* AOP 切面获取当前请求的注解值，并 **异步** 发送时间，减少日志操作的性能损耗

@Aspect

@Slf4j

public class SysLogAspect {

@Around("@annotation(sysLog)")

public Object around(ProceedingJoinPoint point, SysLog sysLog) throws Throwable {

String strClassName = point.getTarget().getClass().getName();

String strMethodName = point.getSignature().getName();

log.debug("[类名]:{},[方法]:{}", strClassName, strMethodName);

SpringContextHolder.publishEvent(new SysLogEvent(logVo));

return obj;

}

}

* 监听器在接收到日志事件后进行调用feign入口处理

@Slf4j

@AllArgsConstructor

public class SysLogListener {

private final RemoteLogService remoteLogService;

@Async

@Order

@EventListener(SysLogEvent.class)

public void saveSysLog(SysLogEvent event) {

SysLog sysLog = (SysLog) event.getSource();

remoteLogService.saveLog(sysLog, SecurityConstants.FROM\_IN);

}

}

[**#**](https://pig4cloud.com/#%E5%BC%82%E6%AD%A5%E6%93%8D%E4%BD%9C%E8%AF%B4%E6%98%8E-%40enableasync) **异步操作说明 @EnableAsync**

@EnableAsync 注解启用了 Spring 异步方法执行功能，在 [Spring Framework API](https://docs.spring.io/spring-framework/docs/current/javadoc-api/org/springframework/scheduling/annotation/EnableAsync.html) 中有详细介绍。