

# 📖 系统日志

项目提供 2 类 4 种系统日志：

- 审计日志：用户的操作日志、登录日志
- API 日志：RESTful API 的访问日志、错误日志

## # 1. 操作日志

操作日志，记录「谁」在「什么时间」对「什么对象」做了「什么事情」。

打开 [系统管理 -> 审计日志 -> 操作日志] 菜单，可以看到对应的列表，如下图所示：

日志编号	操作模块	操作名	操作类型	操作人	操作结果	操作日期	执行时长	操作
1715	管理后台 - 参数配置	导出参数配置	导出	芋道源码	成功	2022-03-28 20:16:34	30 ms	详情
1714	管理后台 - 字典类型	导出字典类型 Excel	导出	芋道源码	成功	2022-03-28 20:12:37	9 ms	详情
1713	管理后台 - 支付订单	导出支付订单 Excel	导出	芋道源码	失败	2022-03-28 19:57:24	13 ms	详情
1712	管理后台 - 岗位	岗位管理	导出	芋道源码	成功	2022-03-28 19:53:17	15 ms	详情
1711	管理后台 - 操作日志	导出操作日志	导出	芋道源码	成功	2022-03-28 17:39:01	38 ms	详情
1710	管理后台 - 操作日志	导出操作日志	导出	芋道源码	成功	2022-03-28 17:31:20	52 ms	详情
1709	管理后台 - 用户	导出用户	导出	芋道源码	成功	2022-03-28 16:41:42	22 ms	详情
1708	管理后台 - 租户	导出租户 Excel	导出	芋道源码	成功	2022-03-28 16:41:26	13 ms	详情
1707	管理后台 - 操作日志	导出操作日志	导出	芋道源码	成功	2022-03-28 16:35:30	39 ms	详情

操作日志的记录，由 [yudao-spring-boot-starter-biz-operatelog](#) 技术组件实现，

[OperateLogAspect](#) 通过 Spring AOP 拦声明了 [@OperateLog](#) 注解的方法，异步记录日志。使用示例如下：

```
public class ConfigController {  
  
    @Resource  
    private ConfigService configService;  
  
    @GetMapping("/export")  
    @ApiOperation("导出参数配置")  
    @PreAuthorize("@ss.hasPermission('infra:config:export')")  
    @OperateLog(type = EXPORT)  
    public void exportSysConfig(@Valid ConfigExportReqVO reqVO,  
                                HttpServletResponse response) throws IOException {  
        List<ConfigDO> list = configService.getConfigList(reqVO);  
        // 拼接数据  
        List<ConfigExcelVO> datas = ConfigConvert.INSTANCE.convertList(list);  
        // 输出  
        ExcelUtils.write(response, filename: "参数配置.xls", sheetName: "数据", ConfigExcelVO.class, datas);  
    }  
}
```

操作日志的存储，由 `yudao-module-system` 的 `OperateLog` 模块实现，记录到数据库的 `system_operate_log` 表。

## 1.1 @OperateLog 注解

`@OperateLog` 注解，一共有 6 个属性，如下图所示：

```
17  @Target({ElementType.METHOD})
18  @Retention(RetentionPolicy.RUNTIME)
19  public @interface OperateLog {
20
21      // ===== 模块字段 =====
22
23      /** 操作模块 ...*/
24      String module() default "";
25      /** 操作名 ...*/
26      String name() default "";
27      /** 操作分类 ...*/
28      OperateTypeEnum[] type() default {};
29
30      // ===== 开关字段 =====
31
32      /** 是否记录操作日志 */
33      boolean enable() default true;
34      /** 是否记录方法参数 */
35      boolean logArgs() default true;
36      /** 是否记录方法结果的数据 */
37      boolean logResultData() default true;
38  }
```

- `module` 属性：操作模块，例如说：用户、岗位、部门等等。为空时，默认会读取类上的 Swagger `@Tag` 注解的 `name` 属性。
- `name` 属性：操作名，例如说：新增用户、修改用户等等。为空时，默认会读取方法的 Swagger `@Operation` 注解的 `summary` 属性。
- `type` 属性：操作类型，在 `OperateTypeEnum` 枚举。目前有 `GET` 查询、`CREATE` 新增、`UPDATE` 修改、`DELETE` 删除、`EXPORT` 导出、`IMPORT` 导入、`OTHER` 其它，可进行自定义。

## 1.2 自动记录

操作日志往往记录的是针对某个对象的写操作，所以针对 `POST`、`PUT`、`DELETE` 等写请求，`yudao-spring-boot-starter-biz-operatelog` 组件会自动记录操作日志。

- 基于请求方法，转换出对应的 `type` 操作方法：`POST` 对应 `CREATE` 类型，`PUT` 对应 `UPDATE` 类型，`DELETE` 对应 `DELETE` 类型，其它对应 `OTHER` 类型。
- 基于 Swagger 注解，转换出对应的 `module` 操作模块、`name` 操作名。

因此，绝大多数 RESTful API 对应的方法，无需添加 `@OperateLog` 注解。例如说：

```

31 public class DictTypeController {
32
33     @Resource
34     private DictTypeService dictTypeService;
35
36     @PostMapping("/create")
37     @ApiOperation("创建字典类型")
38     @PreAuthorize("@ss.hasPermission('system:dict:create')") 无需添加 @OperateLog 注解
39     public CommonResult<Long> createDictType(@Valid @RequestBody DictTypeCreateReqVO reqVO) {
40         Long dictTypeId = dictTypeService.createDictType(reqVO);
41         return success(dictTypeId);
42     }

```

一般来说，只有两种场景需要添加 `@OperateLog` 注解。

① 场景一：需要自定义 `@OperateLog` 注解的属性。例如说：

```

@ApiOperation("导出数据类型")
@GetMapping("/export")
@PreAuthorize("@ss.hasPermission('system:dict:query')")
@OperateLog(type = EXPORT) GET 请求，但是希望操作类型是 EXPORT 导出
public void export(HttpServletResponse response, @Valid DictTypeExportReqVO reqVO) throws IOException {
    List<DictTypeDO> list = dictTypeService.getDictTypeList(reqVO);
    List<DictTypeExcelVO> data = DictTypeConvert.INSTANCE.convertList02(list);
    // 输出
    ExcelUtils.write(response, filename: "字典类型.xls", sheetName: "类型列表", DictTypeExcelVO.class, data);
}

```

② 场景二：不想自动记录操作日志。例如说：

```

public class AppAuthController {

    @Resource
    private MemberAuthService authService;

    @PostMapping("/login")
    @ApiOperation("使用手机 + 密码登录") 登录请求是 POST，但是不想记录操作日志
    @OperateLog(enable = false) // 避免 Post 请求被记录操作日志
    public CommonResult<AppAuthLoginRespVO> login(@RequestBody @Valid AppAuthLoginReqVO reqVO) {
        String token = authService.login(reqVO, getClientIP(), getUserAgent());
        // 返回结果
        return success(AppAuthLoginRespVO.builder().token(token).build());
    }
}

```

## 1.3 后续优化

`yudao-spring-boot-starter-biz-operatelog` 组件目前提供的是轻量级的操作日志的解决方案，暂时未提供很好的记录操作对应、操作明细、拓展字段的能力。例如说：

- 小明

增加 自定义属性【分类-交互优化，稳定性/bug】

2021-01-28 21:37:31
  - 小明

优先级由 为空 变更为 中

2021-01-28 21:37:27
  - 小明

更新了 描述 至V1版本（上一版本快照：V0）

2021-01-28 21:37:02
- 【新增】2021-09-16 10:00 订单创建，订单号：NO.11089999，其中涉及变量订单号“NO.11089999”
  - 【修改】2021-09-16 10:00 用户小明修改了订单的配送地址：从“金灿灿小区”修改到“银盏盏小区”

未来，芳芳会引入老友开源的 <https://github.com/mouzt/mzt-biz-log> 操作日志组件，优化项目的操作日志功能。大家记得给个 Star 哟！

目前，如果要记录具体的操作明细、拓展字段，可以调用 `OperateLogUtils` 的静态方法，代码如下：

```
public class OperateLogUtils {

    public static void setContent(String content) { OperateLogAspect.setContent(content); } 【操作明细】

    public static void addExt(String key, Object value) { OperateLogAspect.addExt(key, value); } 【拓展字段】
}
```

## 2. 登录日志

登录日志，记录用户的登录、登出行为，包括成功的、失败的。

打开 [系统管理 -> 审计日志 -> 登录日志] 菜单，可以看对应的列表，如下图所示：

访问编号	日志类型	用户名称	登录地址	userAgent	结果	登录日期
10768	账号登录	admin	101.224.155.45	Mozilla/5.0 (Macintosh; Intel Mac OS X 10_14_6) AppleWe...	成功	2022-03-28 23:14:17
10767	账号登录	admin	101.224.155.45	Mozilla/5.0 (Macintosh; Intel Mac OS X 10_14_6) AppleWe...	验证码不正确	2022-03-28 23:14:09
10766	超时登出	admin	120.229.36.208	Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7) AppleWe...	成功	2022-03-28 23:10:00
10765	超时登出	admin	116.253.157.191	Mozilla/5.0 (iPhone; CPU iPhone OS 14_8_1 like Mac OS X...	成功	2022-03-28 23:09:00
10764	账号登录	admin	39.69.33.204	Mozilla/5.0 (Windows NT 10.0; WOW64) AppleWebKit/53...	成功	2022-03-28 23:08:18
10763	账号登录	admin	39.69.33.204	Mozilla/5.0 (Windows NT 10.0; WOW64) AppleWebKit/53...	验证码不正确	2022-03-28 23:08:11
10762	超时登出	admin	222.85.239.219	Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7) AppleWe...	成功	2022-03-28 23:07:00
10761	超时登出	admin	39.69.33.204	Mozilla/5.0 (Windows NT 10.0; WOW64) AppleWebKit/53...	成功	2022-03-28 23:06:00
10760	超时登出	admin	115.200.231.109	Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/...	成功	2022-03-28 23:05:00
10759	账号登录	admin	122.235.86.131	Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7) AppleWe...	成功	2022-03-28 23:01:08

登录日志的存储，由 `yudao-module-system` 的 `LoginLog` 模块实现，记录到数据库的 `system_login_log` 表。

登录类型通过 `LoginLogTypeEnum` 枚举，登录结果通过 `LoginResultEnum` 枚举，都可以自定义。代码如下：

```
6  /**
7  * 登录日志的类型枚举
8  */
9  @Getter
10 @AllArgsConstructor
11 public enum LoginLogTypeEnum {
12
13     LOGIN_USERNAME(100), // 使用账号登录
14     LOGIN_SOCIAL(101), // 使用社交登录
15     LOGIN MOCK(102), // 使用 Mock 登录
16     LOGIN_MOBILE(103), // 使用手机登陆
17     LOGIN_SMS(104), // 使用短信登陆
18
19     LOGOUT_SELF(200), // 自己主动登出
20     LOGOUT_TIMEOUT(201), // 超时登出
21     LOGOUT_DELETE(202), // 强制退出
22
23 ;
24
25 /**
26 * 日志类型
27 */
28 private final Integer type;
29 }
```

```
6  /**
7  * 登录结果的枚举类
8  */
9  @Getter
10 @AllArgsConstructor
11 public enum LoginResultEnum {
12
13     SUCCESS(0), // 成功
14     BAD_CREDENTIALS(10), // 账号或密码不正确
15     USER_DISABLED(20), // 用户被禁用
16     CAPTCHA_NOT_FOUND(30), // 图片验证码不存在
17     CAPTCHA_CODE_ERROR(31), // 图片验证码不正确
18
19     UNKNOWN_ERROR(100), // 未知异常
20
21 ;
22
23 /**
24 * 结果
25 */
26 private final Integer result;
27
28 }
```

### 3. API 访问日志

API 访问日志，记录 API 的每次调用，包括 HTTP 请求、用户、开始时间、时长等等信息。打开 [基础设施 -> API 日志 -> 访问日志] 菜单，可以看对应的列表，如下图所示：

dashboard.yudao.iocoder.cn/infra/log/api-access-log

李道管理系统

基础设施

代码生成

代码生成示例

表单构建

系统接口

数据库文档

文件管理

配置管理

API 日志

访问日志

错误日志

MySQL 监控

首页 / 基础设施 / API 日志 / 访问日志

首页 登录日志 访问日志

用户编号 请输入用户编号 用户类型 请选择用户类型 应用名 请输入应用名 请求地址 请输入请求地址 请求时间 开始日期 - 结束日期

执行时长 请输入执行时长 结果码 请输入结果码 搜索 重置

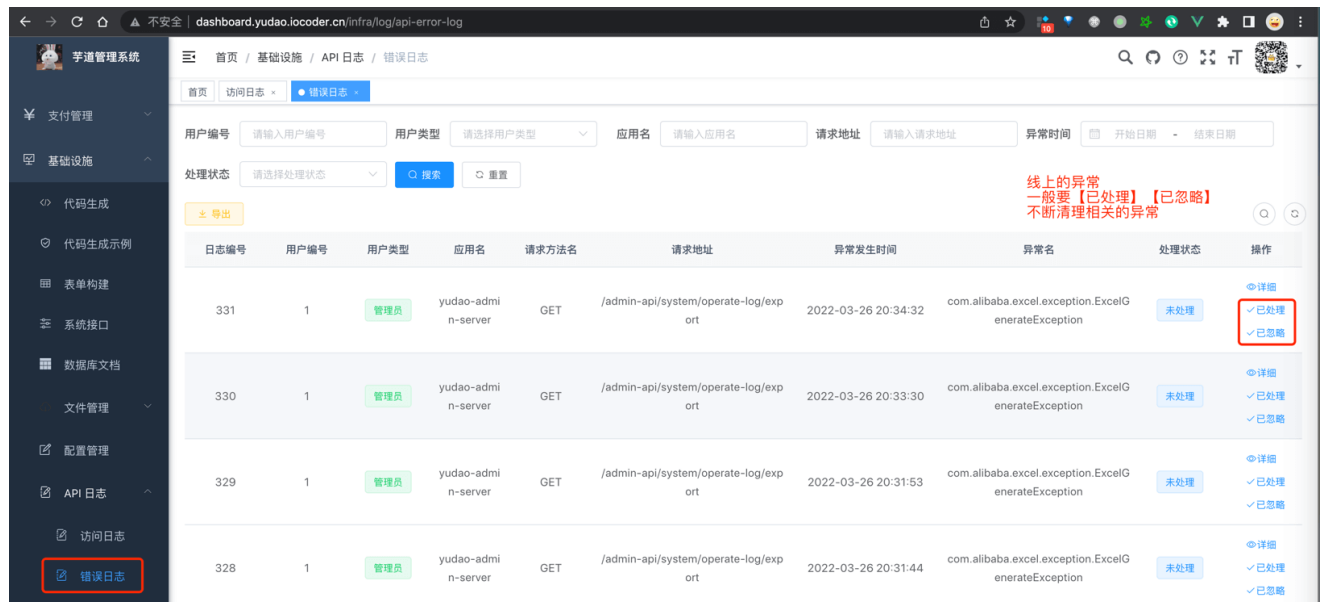
导出

日志编号	用户编号	用户类型	应用名	请求方法名	请求地址	请求时间	执行时长	操作结果	操作
37685	0		yudao-admin-server	POST	/app-api/member/sms-login	2022-03-24 14:11:22	73 ms	失败(请求参数不正确:使用 IP 不能为空)	详情
37682	0		yudao-admin-server	POST	/app-api/member/login	2022-03-24 14:10:43	216 ms	失败(登录失败, 账号密码不正确)	详情
37661	0		yudao-admin-server	GET	/app-api/member/user/get	2022-03-24 14:10:12	1 ms	失败(账号未登录)	详情
37659	1	管理员	yudao-admin-server	GET	/app-api/member/user/get	2022-03-24 14:10:05	3 ms	成功	详情
37654	1	管理员	yudao-admin-server	GET	/app-api/member/user/get	2022-03-24 14:09:55	81 ms	成功	详情

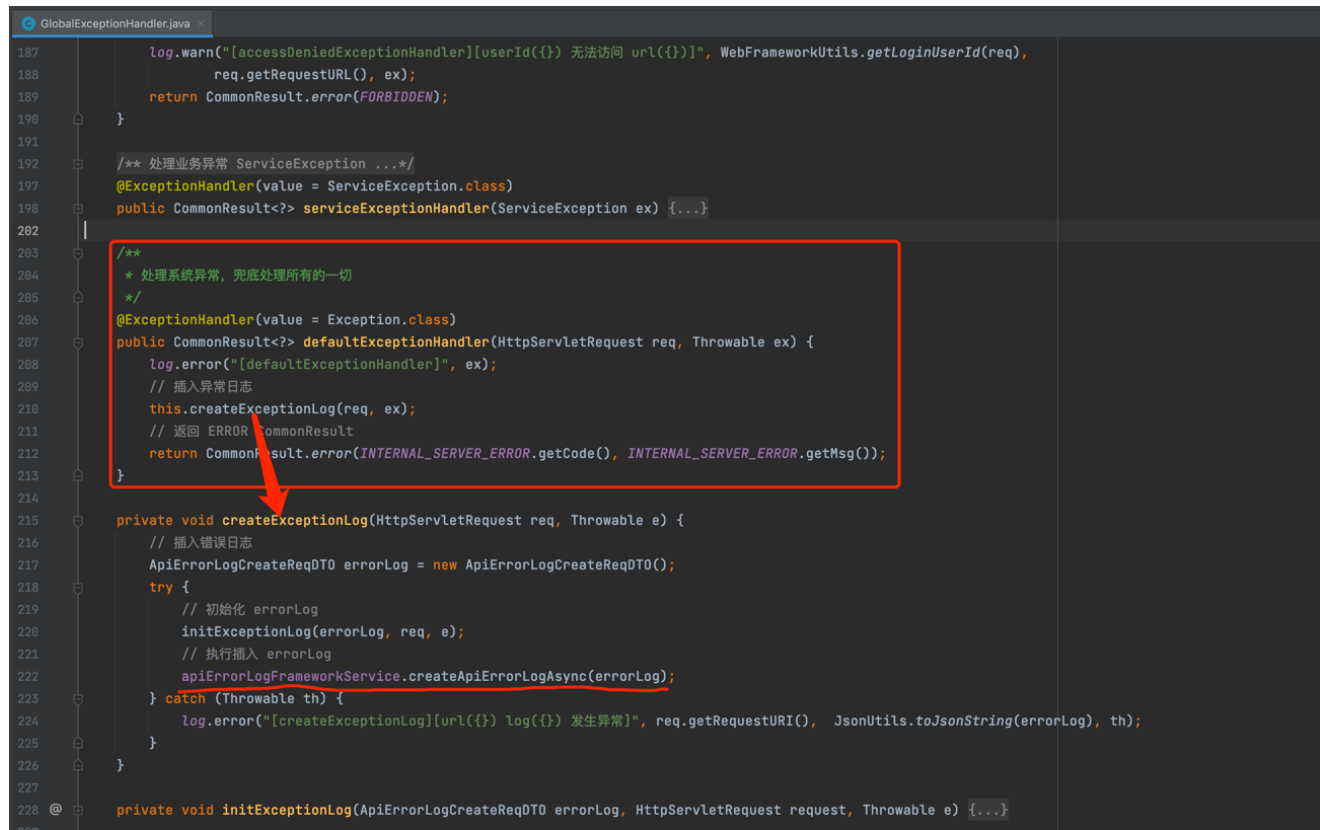
访问日志的记录，由 `yudao-spring-boot-starter-web` 技术组件实现，通过 `ApiAccessLogFilter` 过滤 RESTful API 请求，异步记录日志。访问日志的存储，由 `yudao-module-infra` 的 `AccessLog` 模块实现，记录到数据库的 `infra_api_access_log` 表。

### 4. API 错误日志

API 错误日志，记录每次 API 的异常调用，包括 HTTP 请求、用户、异常的堆栈等等信息。打开 [基础设施 -> API 日志 -> 错误日志] 菜单，可以看对应的列表，如下图所示：



错误日志的记录，由 `yudao-spring-boot-starter-web` 技术组件实现，通过 `GlobalExceptionHandler` 拦截每次 RESTful API 的系统异常，异步记录日志。



错误日志的存储，由 `yudao-module-infra` 的 `ErrorLog` 模块实现，记录到数据库的 `infra_api_error_log` 表。

← [Excel 导入导出](#)

[MyBatis 数据库](#) →



Theme by **Vdoing** | Copyright © 2019-2023 芋道源码 | MIT License