芋道源码 2022-03-28

系统日志

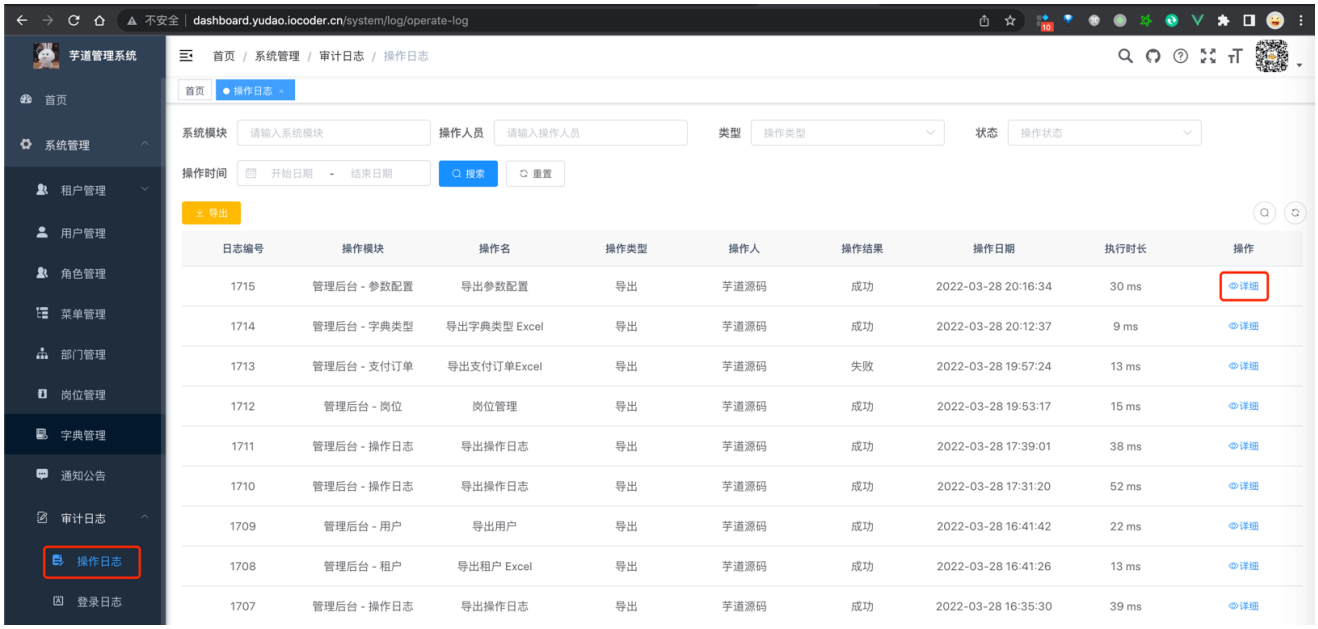
项目提供 2 类 4 种系统日志：

- 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：



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：

```java
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：

```java
@Target({ElementType.METHOD})
@Retention(RetentionPolicy.RUNTIME)
public @interface OperateLog {

    // ========== 模块字段 ==========

    /** 操作模块 ...*/
    String module() default "";
    /** 操作名 ...*/
    String name() default "";
    /** 操作分类 ...*/
    OperateTypeEnum[] type() default {};

    // ========== 开关字段 ==========

    /** 是否记录操作日志 */
    boolean enable() default true;
    /** 是否记录方法参数 */
    boolean logArgs() default true;
    /** 是否记录方法结果的数据 */
    boolean logResultData() default true;

}
```

- `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：

```java
public class DictTypeController {

    @Resource
    private DictTypeService dictTypeService;

    @PostMapping("/create")
    @ApiOperation("创建字典类型")
    @PreAuthorize("@ss.hasPermission('system:dict:create')")
    public CommonResult<Long> createDictType(@Valid @RequestBody DictTypeCreateReqVO reqVO) {
        Long dictTypeId = dictTypeService.createDictType(reqVO);
        return success(dictTypeId);
    }
```

无需添加 @OperateLog 注解

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：

```java
@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：

```java
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，芋艿会引入老友开源的 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！

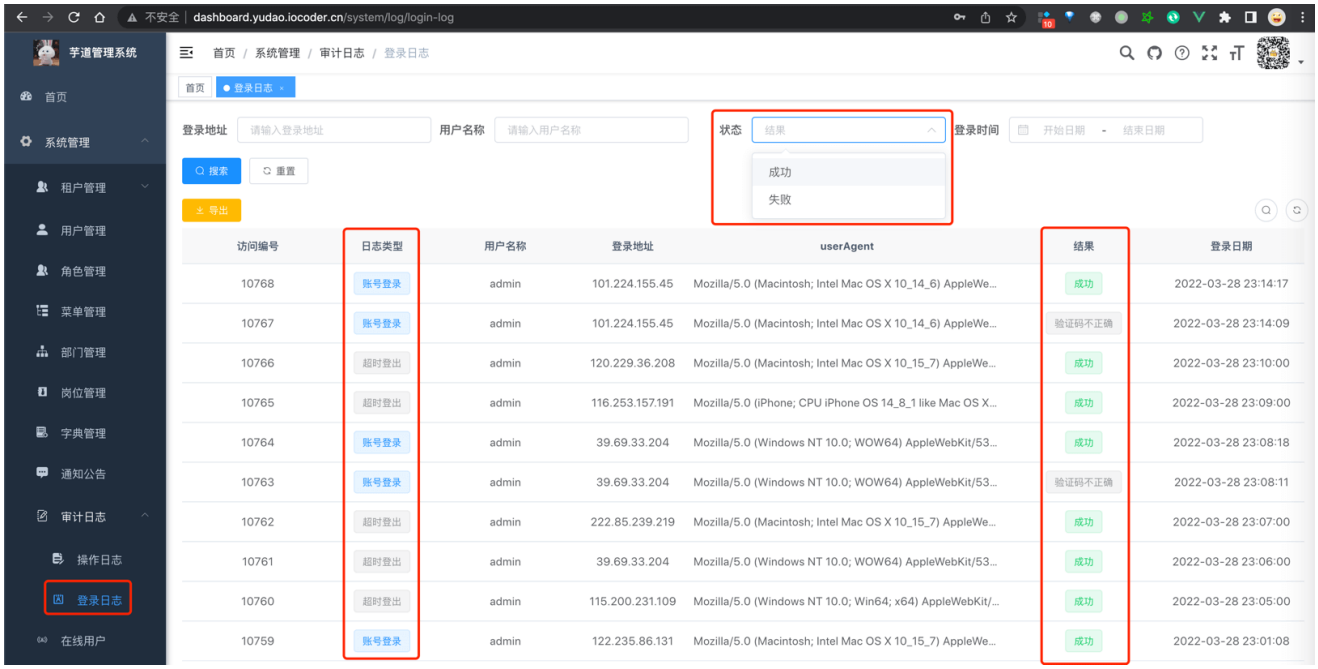
目前，如果要记录具体的操作明细、拓展字段，可以调用 OperateLogUtils 的静态方法，代码如下：

```
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：



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：

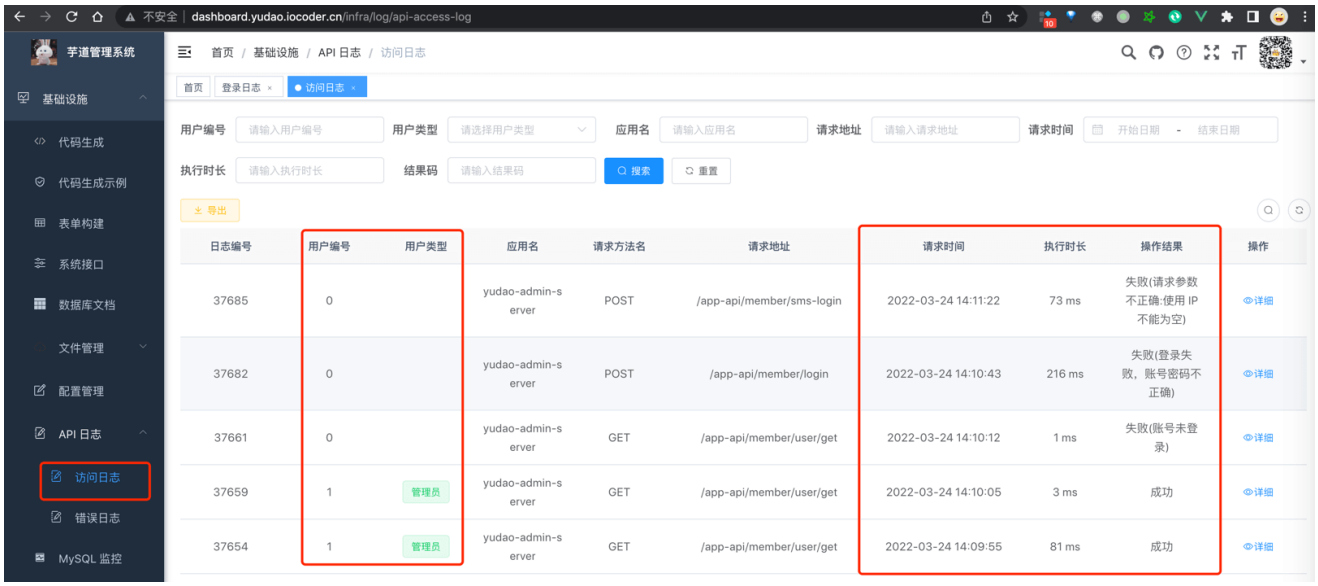
```java
/**
 * 登录日志的类型枚举
 */
@Getter
@AllArgsConstructor
public enum LoginLogTypeEnum {

    LOGIN_USERNAME(100), // 使用账号登录
    LOGIN_SOCIAL(101), // 使用社交登录
    LOGIN_MOCK(102), // 使用 Mock 登录
    LOGIN_MOBILE(103), // 使用手机登陆
    LOGIN_SMS(104), // 使用短信登陆

    LOGOUT_SELF(200),  // 自己主动登出
    LOGOUT_TIMEOUT(201), // 超时登出
    LOGOUT_DELETE(202), // 强制退出
    ;

    /**
     * 日志类型
     */
    private final Integer type;

}
```

```java
/**
 * 登录结果的枚举类
 */
@Getter
@AllArgsConstructor
public enum LoginResultEnum {

    SUCCESS(0), // 成功
    BAD_CREDENTIALS(10), // 账号或密码不正确
    USER_DISABLED(20), // 用户被禁用
    CAPTCHA_NOT_FOUND(30), // 图片验证码不存在
    CAPTCHA_CODE_ERROR(31), // 图片验证码不正确

    UNKNOWN_ERROR(100), // 未知异常
    ;

    /**
     * 结果
     */
    private final Integer result;

}
```

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：



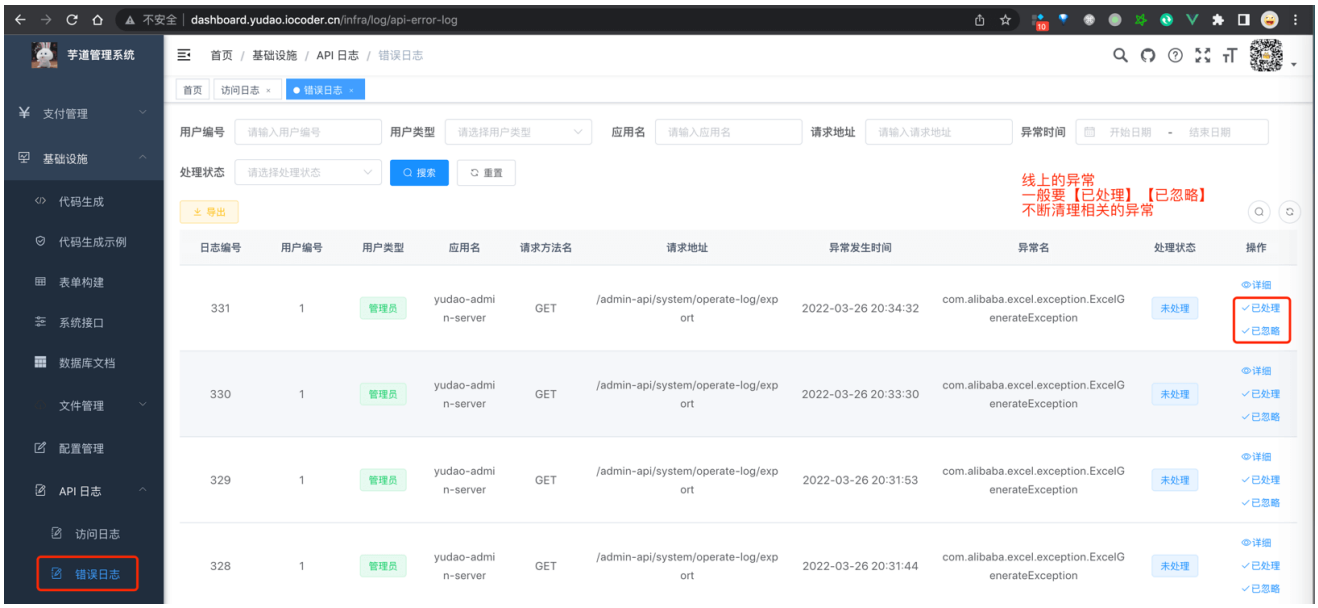
访问日志的记录，由 `yudao-spring-boot-starter-web` 技术组件实现，通过 ApiAccessLogFilter 过滤 RESTful API 请求，**异步**记录日志。

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。

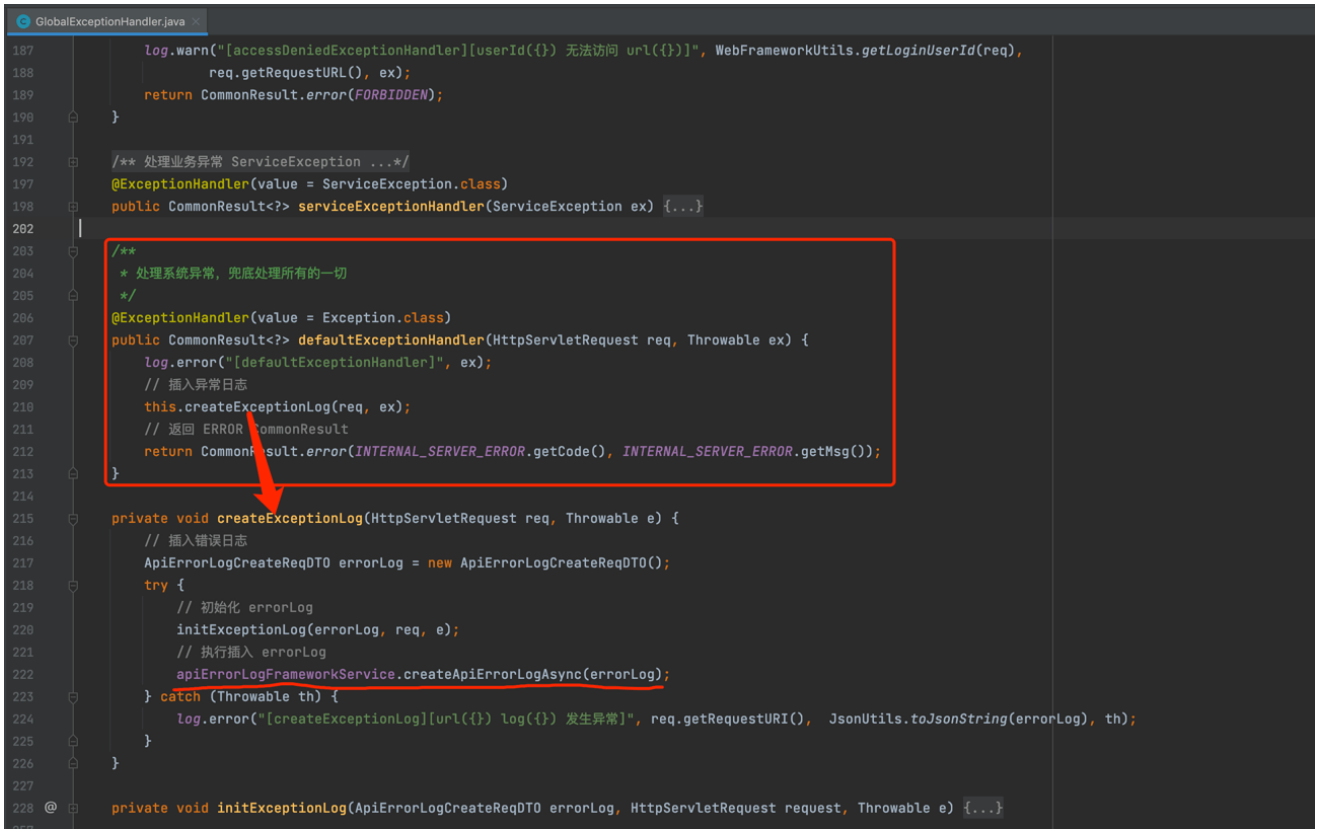
4. API 错误日志

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→