**API开发指南之约定与陷阱**

没有人可以保证他的系统不存在缺陷或漏洞。

因为系统设计上的潜在约定以及某些缺陷，在API开发中难免会遇到令人疑惑或发狂的问题，如果无视系统约定或者不知晓系统设计的缺陷，对开发者来说就是个噩梦，因为你不知道什么时候会落入陷阱里，后果就是影响开发速度，同时也为系统贴上了“垃圾”的标签。

所以这篇文档旨在明确系统的潜在约定以及设计上的某些缺陷，以便API开发者遵守这些约定并避免落入陷阱中。

# 一 约定

在API开发指南中提到了API开发者的职责，就以此来逐个描述其中的约定。

## 1 编写API

* ORM层

**约定1**：按模块分包；

**约定2**：PO放在相应模块包下的model包中，类名以Po结尾，以注解方式配置对象关系映射，实现Serializable接口；

**约定3**：DTO放在相应模块包下的dto包中，类名以Dto结尾，继承BaseDto，Dto可以有多个，尽量保证其中的属性有用且最少。比如：Dto中不放angencyCode、extend1等属性。

* Dao层

**约定1**：DAO层接口放在相应模块包下的dao包中，类名以Dao结尾，继承BaseDao接口；

**约定2**：DAO层接口实现类放在子模块工程下的dao.impl包下，以DaoImpl结尾，继承BaseDaoImpl类，实现相应的DAO接口，并使用注解的方式注册。

* Service层

**约定1**：Service层接口放在framework工程中的servcie包下按应用模块划分的每个模块所在的包中，以Service结尾，继承BaseService接口；

**约定2**：Service层接口实现类放在子模块工程下的servce.impl包下，以ServiceImpl结尾，继承BaseServiceImpl类，实现相应的Service接口，并使用注解的方式注册；

**约定3**：日志Service实现类放在子模块工程下的service.impl包下，以LogServiceImpl结尾，实现BaseLogService接口，并使用注解的方式注册；

**约定4**：SQL语句配置文件一般位于对应模块的src/main/resources下，并且需要在framework模块下的src/main/resources包下的tk-config.properties文件中进行注册；

**约定5**：所有保存操作需以add、save、create、update、touch、apply或refresh开头；所有的删除操作需以delete开头；所有的获取操作需以select、find、get、query或pageGet开头。

* Resource层

**约定1**：所有Resource均位于apis模块下的com.tk.apis包下，按应用模块划分，继承BaseResources类，使用RestController注解注册，并使用RequestMapping注解为其设置匹配的URI，以”/版本号“开始；

**约定2**：Resource类必须以【ResourceV+版本号】结尾，并且除去【ResourceV+版本号】的前半部分必须与操作日志实现类除去【LogServiceImpl】的前半部分相同。例如：用户资源类命名为UserResourceV1，则用户资源的操作日志实现类必须命名为UserLogServiceImpl。 因为操作日志的处理依赖于Resource类和操作日志实现类的命名，务必记住这个命名规则。

**约定3**：Resource类中的方法至少添加三个注解：@ResourceDescription，@ApiOperation，@RequestMapping，所有的方法返回值均为ResponseModel，所有的异常均需正确抛出。

## 2 实现操作日志

编写完API之后要实现该API对应的操作日志，参考已实现的日志实现类。注意上面提到的命名规则。

## 3 编写单元测试

编写完API之后需要针对API编写单元测试，单元测试位于apis模块下的src/test/java包下，按应用模块划分。注意：已经提供了一个测试基类，如果满足测试条件的话直接继承这个测试基类就可以了。

# 二 陷阱

在API开发中陷阱最深的莫过于ORM层，而这之中陷阱最深的莫过于查询文件中SQL的编写。因为实现机制不完善外加没有参考的文档，几乎所有的开发者都曾折戟于此，这里介绍其中最常见的陷阱供参考，如果有更好的实现那就再好不过了。

在参考下面的陷阱之前，首先，确保你的数据源各项属性配置正确。

## 1 查询文件陷阱

* 查询名称

查询名称是指查询文件中<sql-query>或<query>节点的name属性对应的值。注意两点：

① 查询名称在所有查询配置文件中要保持唯一；

② Java代码中调用Dao封装的 \*CustomSQLName 方法时指定的查询名称必须存在。

* 属性映射

注意三点：

① 查询文件中的<return-scalar>节点的column及type属性要与查询返回结果类型中的属性和类型匹配，且不能超过实体类的属性范围，类型可以使用hibernate指定类的简写也可以是全类名；

② 在<![CDATA[ ]]节点中书写的SQL语句中查询返回列的别名要与<return-scalar>节点的column的值匹配；

③ Java代码中传递的参数不能超过SQL语句中指定的可传参数，及以“:”或者“#”开头的参数。

* 关键字

实体类中的属性及查询文件中的<return-scalar>节点的column的值避免使用SQL关键字。

* 符号规则

如果上述提到的都检查OK，SQL没有问题，在SQL工具下可以通过，但是在项目中仍然出错，那几乎可以确定的是：符号用错了。没错，符号的错误使用绝对是掉入查询陷阱的重要因素。

① " "：排在第一位的当属空格，这个符号在查询文件中频繁使用，是最重要也最容易被忽视的一个符号，因为肉眼看不见，极其隐蔽，所以出错首先排查空格。典型的空格使用：

|  |
| --- |
|  |

注意：所有标红的位置都需要添加空格。

② "{"和"}"：标识位于"{" 和 "}"中SQL语句中有来自Java代码传递的参数需要匹配，如果匹配到则会去掉"{"和"}"。

③ ":" : 传递的参数只有一个值时使用，与Statement类似的参数传递符号，注意":"后面要紧跟参数名；

④ "#"：传递的参数有多个值时使用，如：Collection或者Array，典型的#使用：

|  |
| --- |
|  |

注意：所有标红的位置都需要添加空格。

* 分页排序

要使用分页排序，请使用封装的分页查询方法，在查询文件中使用 order by并不能实现排序。