# 统一管理平台代码开发标准

## 阅读指南

* 进行代码开发之前，先阅读项目结构文件，对项目结构有个大概的了解；
* 主要熟悉common模块中的工具类，避免重复造轮子，减轻工作负担。

## 命名指南

* 变量命名

采用驼峰命名，名称要有意义，使人一目了然，没有歧义；例如：filePath----文件路径、userName----用户名等等；

尽量避免无意义命名，例如：int a=1、String aa=”xxxx”

* 方法命名

采用驼峰命名，名称要有意义，根据功能编写方法名，一般是以动词+名词的形式进行命名，例如：

增加：insert...

修改：update...

删除：delete...

查询：select...

备注：方法名可以根据方法进行的功能自行命名，保证有意义

* 项目注释

类必须要有注释，说明当前类的功能

方法必须要有注释，说明方法的功能以及相关参数的说明

实体类中的属性需要有相关说明

类和方法的注释必须包含以下几个部分：

Author：作者

Description：描述

createDate：创建时间

## 开发流程指南

项目代码分为三层：mvc层（controller）、service层、

dao层（Mapper）：

### Controller：

* 项目使用前后端分离的开发模式，不需要返回视图，所有controller注解一律使用RestController
* 类和方法都需要设置RequestMapping

类的RequestMapping为类名首字母小写，去除Controller后缀

方法的RequestMapping的值和方法名保持一致

* 只调用service的方法和项目中返回值，不做任何多余处理

代码：

*/\*\*  
 \** ***@Description:*** *java类作用描述  
 \** ***@Author: xxx*** *\** ***@CreateDate:*** *2019/5/23$ 19:13$  
 \*/*

@RestController  
@RequestMapping(**"/infoResource"**)  
**public class** InfoResourceController {  
  
 @Autowired  
 **private** InfoResourceService **infoResourceService**;  
  
 */\*\*  
 \* 查询所有信息资源  
 \** ***@return*** *\*/* @RequestMapping(**"/selectAllInfoResource"**)  
 **public** Result selectAllInfoResource() {  
 List<xxx> list = **infoResourceService**.selectAllInfoResource();

**return** Result.*success*(list);//有返回值的情况

**return** Result.*success*();//没有返回值的情况

}

}

### Service：

* **接口**

*/\*\*  
 \** ***@Description:*** *java类作用描述  
 \** ***@Author: xxx*** *\** ***@CreateDate:*** *2019/5/23$ 19:13$  
 \*/***public interface** InfoResourceService {  
 */\*\*  
 \** ***@Description:*** *\** ***@Author:*** *张乾  
 \** ***@return*** *\*/*

List<xxx>selectAllInfoResource();

}

* **实现类**

*/\*\*  
 \** ***@Description:*** *java类作用描述  
 \** ***@Author: xxx*** *\** ***@CreateDate:*** *2019/5/23$ 19:13$  
 \*/***public class** InfoResourceService {  
  
 */\*\*  
 \** ***@Description:*** *\** ***@Author:*** *张乾  
 \** ***@return*** *\*/*

List<xxx>selectAllInfoResource() {

List<XXX> list =...;

**return** list**;**

}

}

### dao层

逆向功能生成的几个方法可以不写注释，实现业务功能的方法需要写清楚说明

Xml文件中sql的写法没有强制的要求，尽量统一自己的开发风格

# 开发要求

* 一个controller只能注入一个service接口
* 一个service接口实现类可以注入多个mapper，也可以注入多个service相互调用实现复杂业务
* 所有异常不进行处理，不进行trycatch捕获，除非是需要捕获异常进行业务处理，可以不用抛出异常