# 心通达OA 项目

# 软件开发规范

版本V1.00

文件版本控制

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **版本号** | **状态** | **类型** | **修改人** | **审核人** | **日期** | **描述** |
| V1.0.0 | 完结 | 初稿 |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |

# 1．前言

## 1.1编写目的

为规范项目组的应用开发、解决开发中的代码规范问题和命名规范问题、促进开发工作顺利有序地进行，特制定本开发规范手册。

## 1.2范围

适用于项目开发组所有人员。

# 2．Java程序代码规范

## 2.1命名规范

### 2.1.1包名命名规范

包名必须均为小写。包名需要使用英文说明包内代码功能，不能采用拼音。具体示例如下：

以**com.gsubo.xoa.web**为例：

第一段com代表公司，org代表组织。

第二段 **xxx** 为公司名称 **xxx**(gsubo)。

第三段 xoa为项目名称。

第四段 web为模块名称（分层）。

对于与各项目之间的通用包如util包、 db包直接隶属于xxx.apimanager包下。通用包的内的代码必须不依赖于任何项目，可独立编译。

### 2.1.2文件名命名规范

1. **Java文件名称必须以大写字母开头，可以使用2-4个英文单词（尽量不要用缩写）组成每个单词的首字母大写。文件名称必须能说明文件内代码功能。**

【强制】代码中的命名均不能以下划线或美元符号开始，也不能以下划 线或美元 符号结束。

反例： \_name / \_\_name / $Object / name\_ / name$ / Object$

【强制】 代码中的命名严禁使用拼音与英文混合的方式，更不允许直接 使用中文的方式。说明：正确的英文拼写和语法可以让阅读者易于理解， 避免歧义。注意，即使纯拼音命名方式也要避免采用。

反例：DaZhePromotion [打折] / getPingfenByName() [评分] / int 某变量= 3

正例：alibaba / taobao / youku / hangzhou 等国际通用的名称，可视同英文。

**2.对java文件大致分为接口，实现类，模型，静态类型声明，逻辑类，工具类。**

1. 接口类命名：IxxxxInterFace。
2. 实现类命名：XxxxxImpl。 实现类的名称前段必须与接口一致说明是哪个接口的实现。
3. 模型类命名：XxxxxModel或XxxxxManager。模型类的名称必须与数据库表对应。
4. 静态类型声明：XxxxxType。
5. 逻辑类：XxxxxDAO 逻辑类为执行sql并返回结果的Model加工工厂。逻辑类返回XxxxxModel或由XxxxxModel组成的ArrayList，List。等对象。
6. 工具类：如StringUtil，DateUtil等。

### 2.1.3方法名命名规范

基本规范如下：

1. 方法名应该能够简单描述出该方法实现的功能，如deleteUsers；
2. 方法名尽量不使用缩写，除非它是**众所周知**的，如可以使用db表示database；
3. 方法名可以有两个或三个单词组成，但**尽量**不要多于三个以上；
4. 方法名中由多个单词合成，第一个单词通常为动词，首字母小写，中间的每个单词的首字母都要大写，如删除用户：deleteUsers；

【强制】方法名、参数名、成员变量、局部变量都统一使用 lowerCamelCase 风格，必须遵从驼峰形式。

正例： localValue / getHttpMessage() / inputUserId

1. 方法名中不要使用下划线字符。

### 2.1.4常量名命名规范

常量名也应当有一定的意义，常量名均为大写。如果是由多个单词构成，可以用下划线隔开，如果是对象类型的常量，则是大小写混合，由大写字母把单词隔开。

如: private const int PI=3.1415926；

int YEAR,

String TITLE = “XXXX”；

int WEEK\_OF\_MONTH；

【强制】常量命名全部大写，单词间用下划线隔开，力求语义表达完整清楚，不要 嫌名字长。

正例： MAX\_STOCK\_COUNT

反例： MAX\_COUNT

### 2.1.5变量名命名规范

在程序中变量的名称要求能够体现出就是的功能,如果是单个单词那么变量名用小写，例如：email。 如果变量名是多个单词组成那么首字母小写，中间的每个单词的首字母都要大写。例如：isSuperUser;userLogin。

为了和普通变量区别，如果变量是参数，那么应在变量前加P\_。例如：P\_UserPassword。

如果是循环变量,那么变量名必须是一个小写字母,如:i,j,k。

变量名也尽量不使用缩写，且一定要有含义，不能使用没有表达意义的变量名，如：a1;a2;mmm。

### 2.1.6各层命名规约：

1. **Service/DAO 层方法命名规约**
2. 获取单个对象的方法用 get 做前缀。
3. 获取多个对象的方法用 list 做前缀。
4. 获取统计值的方法用 count 做前缀。
5. 插入的方法用 save（推荐）或insert做前缀。
6. 插入的方法用 save（推荐）或删除的方法用 remove（推荐）或 delete 做前缀。
7. 修改的方法用 update 做前缀。
8. **领域模型命名规约**

1） 数据对象：xxxDO，xxx 即为数据表名。

2） 数据传输对象：xxxDTO，xxx 为业务领域相关的名称。

3） 展示对象：xxxVO，xxx 一般为网页名称。

4） POJO 是 DO/DTO/BO/VO 的统称，禁止命名成 xxxPOJO。

## 2.2代码规范

### 2.2.1类引用规范

**import**时使用星号“**\***”会消耗更多的应用资源，因此要**避免使用**，在当前类 如果引用其它类的方法，那么就明确的import该方法所属的类文件。 此外要避免 import的类没有被使用或者重复。

### 2.2.2避免使用内部赋值语句

不允许这种书写方式：String s = Integer.toString(**i = 2**);

### 2.2.3避免魔法数

也叫 MagicNumber，会让程序不可读。

说明：所谓魔法数值，是指在代码中直接出现的数值，而只有在这个数值记述的那部分代码中才能明确了解其含义。

魔法数值使代码的可读性大大下降。而且，如果同样的数值多次出现时，到底这些数值是不是带有同样的含义呢，谁也说不清楚。另一方面，如果本来应该使用相同数值的地方，一旦用错了，也很难发现。因此，需要注意以下几点，极力避免使用魔法数值。

比如：

sex = 0

表示的什么意思？所以，这里的"0" 就是一个魔法数。建议改用下面的书写方式：

public static final int MALE= 0;

sex = MALE;

这样我们就知道，原来是把 sex 变量的值初始化为 “男性”。

### 2.2.4字符串(String)的比较

检查字符串的比较时没有使用 == 或!=。

正确的写法应该是使用equals方法，字符串在前，变量在后，避免空指针异常。如：

if ("something".equals(x))

### 2.2.5嵌套的try 层次控制

避免try-catch 的嵌套的层次过多而降低代码可读性。建议最大的嵌套深度不超过3层。

### 2.2.6控制return 语句的数量

如果return 语句太多，说明某个方法需要实现的功能太多，而且很难阅读，这个时候就需要重构。建议每个方法最多有5个return 语句。

# 3．系统要求

## 3.1API与Server的交互过程

用户终端

2、逻辑处理

1、用户发送上行

服务器

数据库

5.数据组合

3、查询数据库

4、返回数据

MVC逻辑处理

5.数据组合

6.json数据

## 3.2程序包说明

## 3.3JDK版本

使用的jdk版本：jdk 1.7及以上。

# 4．功能模块说明

## 4.1多企业

系统需要支持多企业，企业之间采用多数据库进行数据隔离，实现企业saas化，用户登陆通过主库进行登陆验证，获取对应的企业ID，系统自动切换到对应的企业库，用户信息（用户名，用户ID，电话，企业id）保存搭配session中

平台库：xoa

企业表：enterprises（id、eid、ename）

企业库：xoa0001、xoa0002

说明：与平台库的差异是没有企业表，其余都一样

开发获取全局数据

获得eid：request.getSession().getAttribute（“eid”）登陆时候选择组 织确定eid

获得uname：request.getSession().getAttribute（“uname”）

获得企业库名：request.getSession().getAttribute（“ename”）

## 4.2组织架构

创建一个公共类，定义组织的操作方法

1、单位管理

2、部门管理

3、用户管理

**1)选部门控件**

**2)选用户控件**

**3)选角色控件**

## 4.3系统工具

公共js调用规范，如:函数名

公共函数封装到底层类中

## 4.4多语言

1、需要考虑cookie值存储

2、一种语言一个语言包，命名采取国家和语言文件标识符，如：xoa.config.i18properties.message\_zh\_CN.properties

xoa.config.i18properties.message\_en\_US.properties；

3、语言包中按模块划分，如：email.send

4、程序中或系统自带数据采用多语言

5、国家化jsp页面需引入

<%@taglib prefix=*"mvc"* uri=*"http://www.springframework.org/tags/form"* %>

<%@taglib prefix=*"fmt"* uri=*"http://www.springframework.org/tags"* %>

需国际化的属性(语言配置文件中的key：headQuarters)：

<fmt:message code=*"**headQuarters"* />

## 4.5角色权限控制

1、角色管理

1. 角色增删改查
2. 角色授权

2、权限控制

1） url路径的控制

2） 框架菜单的展示

3) 每个url需要验证登陆

## 4.6文件

1、存储路径位置，需添加properties文件读取路径。

2、文件类库规范化，代码位置和js公共位置

3、文件存入服务器中，数据库存对应路径，并可考虑编码转换

4、后台功能文件类的方法：上传、下载

## 4.7数据库操作:Mybatis

多数据库兼容性，规范数据库操作代码

增：

<insert id=*"save"* parameterType=*"com.xoa.model.email.EmailBody"*>

insert into email\_body (BODY\_ID) values (#{bodyId})

</insert>

删：

<delete id=*"delete"* parameterType=*"com.xoa.model.email.EmailBody""*>

delete from email\_body where BODY\_ID = #{bodyId,javaType=INTEGER}

</delete>

改：

<update id=*"update* parameterType=*"**com.xoa.model.email.EmailBody"*>

update email\_body FROM\_ID = #{fromId} where BODY\_ID = #{bodyId}

</update>

查：

<select id=*"select"* parameterType=*"Map"* resultMap=*"BaseResultMap"*>

select USER\_NAME from email\_body where e.READ\_FLAG=#{readFlag} AND e.SIGN=#{sign} group by BODY\_ID

</select>

## 4.8移动接口

Json

移动和PC接口在具体类中区分

## 4.9即时通讯

Web程序实现

## 4.10第三方接口

单点登陆，组织同步，用户同步，WebService接口

## 4.11 定时任务

在特定时间执行事先安排好的任务

## 4.12范例模块展示

新闻、邮件和组织管理（数据列表、数据录入表单、编辑表单、查询表单、按钮[css 的Class]）