# 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.xoa.controller.email**为例：

第一段com代表公司，org代表组织。

第二段 **xoa**为公司名称项目名称。

第三段 xoa为逻辑层（controller、dao、model、service等）。

第四段 email为应用模块名称。

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。**

示例：EmailController、EmailService、EmailMapper

**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：XxxxxInterface。
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. 逻辑类：XxxxxMapper 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。
5. 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： getHttpMessage() / addUser/deleteUserById/getUserList

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。

【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

普通数据类型变量命名采用驼峰命名规则

### 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 做后缀。
4. 获取统计值的方法用 count 做后缀。
5. 插入的方法用 save
6. 修改的方法用 update 做后缀。
7. **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。

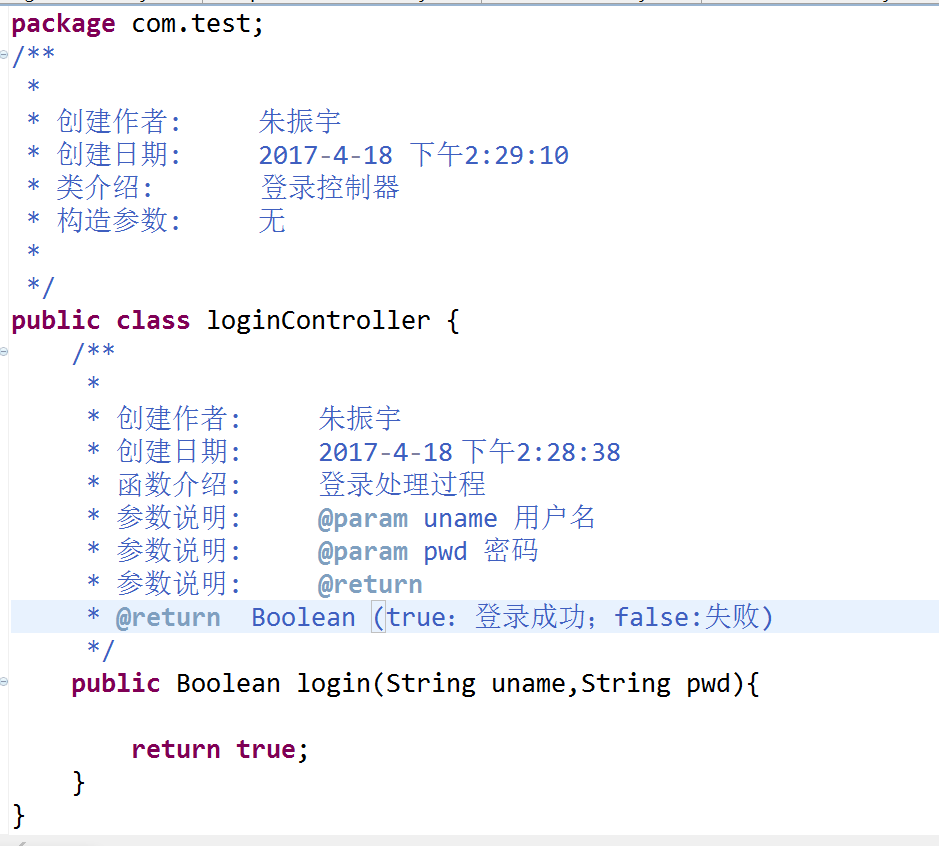
### 2.1.7注释规范：

注释模板: 注释模板V20170418.xml

书写要求：

在IDE中，类或函数前，输入“/\*\*”回车，调出模板，按模板详细介绍类或函数的用途，参数含义，返回值数据及含义。

注释范例：



## 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

4.6.1上传文件接口

4.6.2接口及使用说明

接口前台配置：需在前台上传表单的form标签中配置enctype=*"multipart/form-data"，*

通过post提交 参考<form action=*"？"* method=*"post"* enctype=*"multipart/form-data"*>

定义上传通过input标签上传，定义所有上传文件类型为file，name为file

参考<input type=”file” name=”file”/>

接口url：/upload

规则定义

存储路径：基础路径/公司名/模块名/年月/文件名

文件名：为防止上传文件文件名重复，修改上传文件的文件名，生成规则为当前时间戳和原文件名的拼接中间用“.”分隔

例如：upload/xoa111/email/0412/41213211345.测试.docx

4.1.3参数列表

|  |  |
| --- | --- |
| 参数名 | 参数说明 |
| files | 用于接收前台上传文件的集合 |

4.1.4返回参数列表

因为上传成功后需要根据上传信息向数据库中插入记录，所以定义返回参数如下，

返回值为ToJson<Object[]> 其中object[]数组中封装了四个参数分别为

O[0]=年月编号 ，o[1]=附件id编号，o[2]=附件名称，o[3]=附件所属模块 。

|  |  |
| --- | --- |
| 参数名 | 参数说明 |
| ym | 上传文件时所对应的年月编号 |
| attachId | 附件编号 |
| fileName | 附件名称 |
| module | 附件所属模块 |

4.15上传文件接口示例

/xoa/upload

4.16 返回结果示例

{“flag”:true,”msg”:”有数据”，“obj“：[{“ym”:1704,”attachId”: 780190158,”fileNmae”:cs.txt,”module”:email},{“ym”:1704,”attachId”: 345690158,”fileNmae”:sc.txt,”module”:email}]}

4.2下载文件接口

4.2.1接口及使用说明

接口前台配置：前台需引入jstl标签

实例如下 <%@taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>

前台通过接受后台传递过来的可下载文件集合遍历可下载文件显示在前台页面

示例如下

 <!-- 遍历Map集合 -->

    <c:forEach var="me" items="${fileNameMap}">

        <c:url value="？ " var="downurl">

     <c:param name="filename" value="${me.key}"></c:param>

        </c:url>

        ${me.value}<a href="${downurl}">下载</a>

        <br/>

    </c:forEach>

规则定义：下载url需要拼接该文件对应的附件信息表的主键编号，附件模块编号，附件id编号，附件名称编号

下载url定义为action？AID=6970&MODULE=email&YM=1704

&ATTACHMENT\_ID=-727821006&ATTACHMENT\_NAME=cs.txt

通过解析参数获取文件下载路径和下载文件名

4.2.2参数列表

|  |  |
| --- | --- |
| 参数名 | 参数说明 |
| filePath | 文件下载路径 |
| filename | 下载文件的文件名 |

## 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 e 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]）

## 4.12.1邮件

查询邮件地址显示字段详解

|  |  |
| --- | --- |
| 服务器获取参数 | |
| 参数 | 描述 |
| flag | inbox(收件箱)、drafts(草稿箱)  outbox(发件箱)、recycle(废纸篓)  noRead(未读) |
| page | 当前页码 |
| pageSize | 每页显示条数 |
| useFlag | 是否使用分页插件(true:使用 | false:不适用) |
| userID | 用户user\_id 例:admin |

查询邮件地址显示字段详解

|  |  |
| --- | --- |
| 根据地址返回参数 | |
| 参数 | 描述 |
| BODY\_ID | 邮件自增id，页面中可采取隐藏形式(如 type=”hidden”)，查询单条邮件内容时可用到 |
| SUBJECT | 邮件主题 |
| SEND\_TIME | 发送时间 |
| USER\_NAME | 用户姓名 |

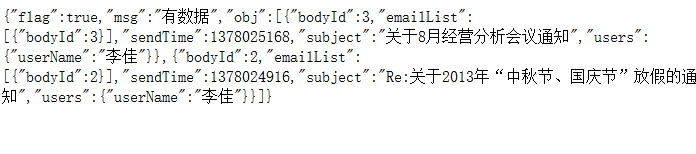
邮件接口地址示例

链接地址具体参数根据实际调整，以下为示例，具体参数可参考2.1表中参数说明调整

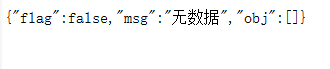
[http://192.168.0.17/xoa/showEmail?flag=inbox&page=1&pageSize=5&useFlag=true&userID=lijia](http://192.168.0.17/xoa/showEmail?flag=inbox&flag=1&pageSize=5&useFlag=true&userID=lijia)

# 返回结果示例

**1、如有相关数据如下显示**



1. **如无相关数据如下显示**



目前接口逐渐完善中，若有调整或增加会补充说明文档