## 开发规范

## 1、JAVA开发规范

### 1.1、总体编码原则

* 模块是有层次的，模块之间的依赖一定是单向的，上层依赖下层。
* 不要写超过一个屏幕的大方法。
* 原则上不要自己编写工具类。能采用Spring Utils中提供的工具类全部采用他们的工具类。
* 在 jsp 中尽量不要写 java 代码，jsp 只负责展示，业务逻辑的部分放到相关的

逻辑处理类中。

* + 在类中一律不准使用System.out.print()，需要在控制台打印信息时全部使用SLF4J（Log4j）来实现。
  + 不能有多余的Import
  + 该采用泛型处理的一律采用泛型处理，不能采用泛型处理的需要注释清楚，让方法使用者了解
  + 所有的功能模块都是无状态的，不能有session或上下文的信息
  + 开发完成后要生成javadoc，所以类和接口要写详细的注释信息。

### 1.2、包命名规范

包名一律小写，并遵循“Java 程序包结构”中的规则定义。包的命名遵循“基本包名+子系统名+模块名+层级名”的命名规则。

* + com. ariadnethread.<SubAppName>.<ModuleName>
  + controller 存放Spring MVC的Controller类
  + service 存放业务逻辑实现类
  + domain 存放POJO类
  + dao.mybatis 存放DAO接口类和mybatis配置文件类
  + rest 存放restful实现类
  + pages存放jsp、html等UI文件
  + 包名是全小写的名词

### 1.3、类命名规范

Class 的名字必须由大写字母开头，采用驼峰式的命名方式，即每个单词首字

母大写而其他字母都小写的。类名不可包含下划线。

* controller类：<ModuleName>Controller
* service类：<ModuleName>Service
* dao类：<ModuleName>Dao
* mybatis文件：<ModuleName>Mapper
* rest类：<ModuleName>RestController

### 1.4、方法命名规范

第一个词汇全部小写，多个词汇时，采取驼峰式的规则。方法名中一般不建

议出现下划线。比如 findUsers。

* 新增方法：save<ModuleName>
* 删除方法：delete<ModuleName>
* 修改方法：update<ModuleName>
* 批量操作：batch +操作+<ModuleName>，如批量删除用户：batchDeleteUser
* 查询方法：方法名以 find 开头。比如 findUser，findAll，findById 等
* 判断方法：方法名以 is 开头，返回true或false.
* 分页查询：方法名以 find 开头，以Page结束，比如findUserListPage
* 页面重定向：以to开头，主要用在controller类中。比如：toUserAdd

### 1.5、变量命名规范

* 变量的名字必须是小写字母开头。后面尽量采用驼峰式的命名方式。比如

createDate，userName，password 等

* static final 变量名字应该全部大写，单词之间使用下划线。比如：

USER\_CACHE\_KEY；

* 如果变量类型是数组，则总是按照下面的规则来命名：String[] users。而不是

String users[]。

### 1.6、注释规范

* 类注释：标明作者，时间，类用途，如果是接口，要在接口上写注释。
* 方法注释：方法描述，参数，返回值，异常信息。比如

/\*\*

\* 方法描述（如果逻辑复杂，则需详细说明）

\*

\* @param 参数1：参数1描述

\* @param 参数n：参数n描述

\* @return 返回值：返回值描述

\* @throws 抛出的异常：抛出的异常描述

\*/

### 1.7、Domain开发规范

* Domain类必须继承DomainBase
* Domain类必须加注解@MyBatisDomain
* 使用基于Hibernate Validator的注解验证

在POJO类的属性getXXX方法上增加验证的注解，用来保证业务逻辑的一致性，可用的注解有：

@NotNull 该字段不能为空

@Max(value) 设置该字段的值不能大于设定的值

@Min(value) 该字段的值不能小于设定的值

@Size(min, max) 字段长度必须在设定值之间

@Future 检查给定的日期是否比现在晚.

参考资料：Java Validation Framework API：Package javax.validation.constraints

Hibernate Validation：Hibernate Validation说明文档

* 为了满足多租户和数据交互的需求，建议每个POJO类都有下面6个自动写入的值的字段:（非必须，产品经理自行决定）

**private** java.util.Date lastUpdateDate; // 最后修改时间

**private** java.lang.String lastUpdatedBy; // 最后修改人

**private** java.util.Date creationDate; // 创建时间

**private** java.lang.String createdBy; // 创建人

**private** java.lang.String tenantId; //租户ID

**private** java.lang.Long version; // 版本，乐观锁控制用

* 关于Blob和Clob字段。这两个字段必须使用java.sql.Blob和java.sql.Clob来定义，不能使用特定的数据库的定义方式。

### 1.8、Service开发规范

* Service类必须加注解@ Service
* 引用其他Bean时采用@Autowired注解
* 调用其他模块的方法。在Service层调用其他模块时，只能调用该模块Service层中和该模块的DAO层方法，一般情况下，不允许调用其他模块的controller层方法、rest层方法等。
* 遵守事务处理的方法命名原则。在平台中，Spring事务处理器拦截的方法命名规范是save\*、delete\*、update\*、do\*、batch\* 、find\*、get\*，其他命名方式都将采用只读事务的处理方式，在给业务操作方法命名时，需要遵守上面的命名原则。
* 禁止在Service层中侵入session、上下文等状态信息。

### 1.9、Controller开发规范

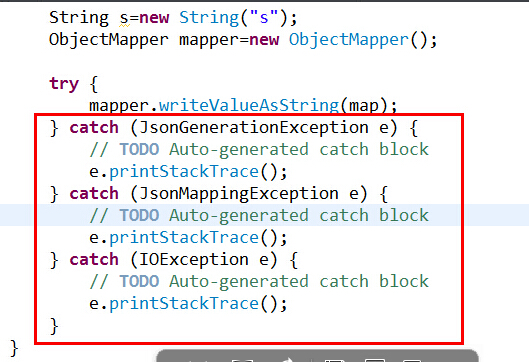
* Controller类必须加注解@ Controller
* Controller类的所有操作均为restful风格，具体参考springmvc 官方文档。

### 1.10、DAO层的说明

* Dao接口类必须加注解@ MyBatisRepository
* 方法名称必须与xxxxMapper.xml中保持一致
* Dao接口类和mybatis配置文件xxxxMapper.xml必须放到同级目录下
* Dao接口中save方法主键默认为32位uuid

### 1.11、异常处理规范

* DAO层不显示声明异常，默认由mybaits抛出运行时异常；
* Service层不要声明和捕获RuntimeException，如有CheckedException，要显式声明抛出，且尽量不要声明多个异常，可以把多个异常归并成一个自定义异常。比如下面的写法就欠妥：



* Controller层（restful接口）要把CheckedException抛出给restful sdk,同时注意，该层方法不能返回null（否则，httpclient调用序列化会出异常），而是要返回空对象，比如new ArrayList();
* Restful sdk层：该层捕获到服务端异常后要继续抛出给业务调用方法；

### 1.12、工具类规范

* 原则上不要自己编写工具类。能采用Spring Utils中提供的工具类全部采用他们的工具类。具体请baidu.

Spring工具类包名：

org.springframework.util.\*

比如：

**字符串工具类 org.springframework.util.StringUtils**

**对象序列化与反序列化 org.springframework.util.SerializationUtils**

**数字处理 org.springframework.util.NumberUtils**

**文件复制 org.springframework.util.FileCopyUtils**

**Web工具类 org.springframework.web.bind.ServletRequestUtils**

### 1.13、单元测试规范

* 每个模块必须有一个单元测试类，要针对service接口测试
* 测试使用JUnit4 + Spring Test实现，测试类命名为xxxServiceTest，包名为test.原包名。

### 1.14、Effective Java（强烈建议看此书）

以下是此书中，针对代码开发的几条建议，可以提高代码的质量，减少bug：

* 避免创建不必要的对象

极端例子：

String s=**new** String("s");

此类代码不要出现。

如果某个对象初始化代价非常昂贵，尽量提前初始化

* 消除过期的对象引用

java 也存在内存泄露（这类泄露成为“无意识的对象保持”）。例如保存在map中的过期对象引用。

* 覆盖equals时需要覆盖hashcode
* 始终覆盖tostring

覆盖此方法，返回对象中所有值得关心的信息，有助于诊断程序。

* 接口优于抽象类

基于接口构建产品的骨架，同时与抽象类的优点结合起来。Spring就是基于此思想做的架构。

* 优先考虑泛型

泛型提供了更加安全的类型检查，减少了代码出错的概率。

* 返回零长度的数组或集合，而不是null

返回零长度的数组或集合，更加符合语义，减少了代码出错的概率。

## 2、数据库设计规范

### 表及字段规范

#### 2.1.1、表命名约定

表名规则：业务系统简写 + ‘\_’+ 英文，如UC\_USER表示用户中心下的USER表，英文如果是两个单词及以上，单词之间尽量用下划线分割。如果英文有公共熟知的缩写，可以使用缩写。如CC\_DEPT。

#### 2.1.2、表字段约定

1、为了满足多租户和数据交互需求，建议每个表中包含如下7个字段：（非必须，产品经理自行决定）

|  |  |  |
| --- | --- | --- |
| 字段名称及作用 | 字段代码 | 类型 |
| 表ID，作为表主键 | ID | VARCHAR(50) |
| 创建人 | CREATED\_BY | VARCHAR(50) |
| 创建时间 | CREATION\_DATE | Date |
| 最后修改人 | LAST\_UPDATED\_BY | VARCHAR(50) |
| 最后修改时间 | LAST\_UPDATE\_DATE | Date |
| 租户ID | TENANT\_ID | VARCHAR(50) |
| 版本 | VERSION | Number(16,0) |

2、主键字段：每个表必须以ID作为主键，不要设置复合主键，通过建立Unique Key来实现实际的主键组合。如部门表中，将部门代码建立为Unique Key

3、外键字段：系统不显式建立外键，表中的字段如果需要引用其他表的主键，尽量以其他表的表名+ ‘\_ID’ 作为此字段名，并在注释中显式说明，如：员工表中引用部门表的主键的字段，命名为HR\_Dept\_ID；如果表中有两个及以上的字段需要引用同一个表的主键，需要在此规则的基础上，前面添加用途，如两个字段都引用部门表主键，制造部门命名为：Manufact\_HR\_Dept\_ID，使用部门命名为：User\_HR\_Dept\_ID。

4、字段注释：每个字段必须有注释，说明该字段的含义，值的来源。对于标识，状态之类的字段，必须注明其每个可能的取值，及每个取值的含义。如对于标识字段，注释需要标明：Y-是，N-否，说明此字段的取值为Y和N，含义分别为是和否。

### 2.2、视图规范

视图命名规范为：

给表建立视图：表名 + ‘\_V’

如果在视图的基础上再建立视图，命名规范为：视图名 +‘V’

### 2.3、临时表、索引命名规范

1、临时表命名为前缀“tmp\_”加表名 tmp\_item\_info

2、普通索引命名前缀“ind\_”加字段名ind\_id,组合索引则为ind\_col1\_col2如果名字过长可以适当缩写

3、唯一性索引命名前缀“uk\_”加字段名uk\_id,组合索引则为uk\_col1\_col2名过长时可以适当的缩写

4、主键命名前缀“pk\_”加字段名pk\_id,如果是默认的情况则为字段名，建议显示指定

5、外键命名为前缀“fk\_“加字段名fk\_item\_id

## 3、WEB UI规范

* 目录结构

保持目录清晰，为后期编写和扩展程序提供良好的开发条件。

## 

* static 目录

前端文件的集合目录，便于维护升级，为将来的静态缓存做准备。

* Extends 目录

产品扩展目录，存放和产品相关的css、images和js文件，应尽量按照产品模块再次划 分目录，特别是js目录，便于后期维护。

* lib 目录

存放第三方库、组件或皮肤等相关文件。

css和images目录存放与组件相关的公共文件。

js目录按照版本号存放第三方库和组件及其扩展文件，便于后期维护升级。

skin存放产品多套皮肤，便于切换风格。

* Jsp 文件规范

1. 申明字符编码为UTF-8，例如：<%@ page contentType="text/html;charset=UTF-8" %>
2. 申明c标签，例如：<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>

* Html文件规范

1. 文件名必须由小写字母、数字、下划线组成。
2. 文件编码默认使用UTF-8。
3. css 和 js 文件通过外链方式引入。
4. 页面代码头部增加“doctype”标签来告知浏览器使用哪种 HTML 或 XHTML 规 范，doctype完整生命语句为：

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml">

所有标签必须小写，除头部doctype声明外

5. 各属性的值必须加英文双引号。

* JavaScript 文件规范

1. 文件编码统一为utf-8

2. 书写过程过, 每行代码结束必须有分号;

3. 变量命名: 以下划线链接每个单词，类型\_变量名

s：表示字符串。

n：表示数字。

b：表示逻辑。

a：表示数组。

r：表示正则表达式。

f：表示函数。

o：表示以上未涉及到的其他对象，

4. 变量在作用域的顶部申明

5. 常量：全部大写，以下划线分隔 如网站的URL，域, 图片目录路径

6. 类命名: 首字母大写, 驼峰式命名. 如 Comment;

7. 函数命名: 首字母小写驼峰式命名. 如getUserName();

8. 类中私有方法，以下划线+方法名标注,私有方法不能在类外被调用。

8. 命名语义化, 尽可能利用英文单词或其缩写;

9. 代码结构明了化, 加适量注释. 提高函数重用率;

1) 大功能区块的功能描述(类，函数)

/\*

功能描述

参数描述

返回值

\*/

2) 单行的注释(放在行末尾)

//注释

10. 将javascript代码外部化，建立独立的js文件管理

11. 把加载dom时不需要的JavaScript文件放在HTML底部,</body>的前面

12. 为了避免混乱，我们建议在HTML中使用双引号，在JavaScript中使用单引号。

* 图片文件规范

名称分为头尾两两部分，用下划线隔开。头部分表示此图片的大类性质，例如广告、标 志、菜单、按钮等等。

一般来说:

1. 放置在页面顶部的广告、装饰图案等长方形的图片为banner\_filename

2. 标志性的图片为logo\_filename

3. 在页面上位置不固定并且带有链接的小图片为button\_filename

4. 在页面上某一个位置连续出现，性质相同的链接栏目的图片为menu\_filename

5. 装饰用的照片为pic\_filename

6. 不带链接表示标题的图片为title\_filename