# JAVA开发规范

## 1.数据库表设计命名规范

1. **表名命名**：全字母小写（）、模块名\_业务类对象名 如：sys\_user、sys\_role

系统表：sys\_\*

业务表：大业务模块简写字母或项目业务名为主\_\* 如：sf\_\*

提示：不需要骆驼峰命名、或下划线分割如：sys\_depart\_ment；

直接全小写sys\_department

只有中间表才可以\_分割：如sys\_role\_resources、sys\_user\_role

2）**字段命名**：全字母小写、骆驼峰命名方式、字段一一对应属性名，

如：属性名为userName 字段为userName

3）**主键命名**：mysql下，统一bigint\varchar(20)

## 2.Java类、方法、属性、常量规范

### 1.1.Java编程基础规范

提示：类、方法、属性、常量等都遵循java编程基本规范

1）包的命名：

* java包的名字都是由小写单词组成。
* 一般采用域名的反写, 如：com.baidu.www.demo

2）类的命名（帕斯卡命名法）

* 类名必须由大写字母开头，一个单词中的其他字母均为小写字母
* 若多单词组成。则建议每个单词的首字母均大写，如TestDemo
* 若类名中包含单词缩写，则建议将这个词的每个字母均为大写，如XMLDemotional
* 类名尽量用名词

3）方法的命名（骆驼峰命名法）

* 方法的名字第一个单词应以小写字母开头，后面的单词首字母要大写
* 方法名字尽量用动词， 如getPicture().

4）常量的命名

* 常量的名字应该都是用大写字母。并且指出该常量完整含义。CUREE
* 如果一个常量名称由多个单词组成，则建议用下划线分割这些单词  MIN\_VALUE

5）参数的命名

* 参数命名和方法的命名规范相同，参数的命名尽可能的明确

6）java注释

统一使用template.xml等代码注释模板

另外：属性注释 请使用行注释 如 //注释

### 1.2.项目开发编程基础规范

1）包命名：

实体类（entity）

持久层（dao\dao.impl）

服务层（service\service.impl）

web层（controller）

查询对象（query）

工具包（utils）

提示：命名可以根据项目模块或业务大模块进行再次分分层

2）类命名：

尽可使用名词，保证和数据库表名一致，帕斯卡命名法

实体类：类命名必须保证和数据库表一致(不管模块名) 如表sys\_user ---> 类User

DAO层：\_接口类I开头，Dao结尾 如：IUserDao

\_实现类DaoImpl结尾，

Service层：\_接口类I开头，Service结尾 如：IUserService

\_实现类ServiceImpl结尾 如UserServiceImpl

web层：以Controller结尾 如：UserController

查询对象：以query结尾 如UserQuery

1. Mbatis的Mapper映射文件

骆驼峰命名法，实体类名（首字母小写）+ 下划线 + mapper结尾

如：user\_mapper.xml

1. IOC依赖注入

在做IOC依赖注入对象时，对象属性声明必须全写拼完不能简写

如：服务层注入Dao持久化对象

@Autowired

**Private** IUserDao userDao

如：Web层注入Service对象

@Autowired

**Private** IUserService userService

1. SpringMVC请求注解servlet的mapping映射

Web层：

* 类：映射尽可使用实体类的全小写作为请求映射名

如：@RequestMapping("/user")

**public** **class** UserController {

* 方法：映射尽可保证和方法一致，全小写作为请求映射名

如：@RequestMapping("/delete/")

**public** **void** delete {

1. Shiro权限认证规则key定义

不管是注解注册权限认证规则，还是url配置的方式，

为了shiro配合业务功能模块的权限认证及可读性和后期维护性，

则权限Key = controller映射名+方法映射名

如：

（1）注解方式：

@RequiresPermission("user\_delete")

**public** **void** delete {

@RequiresPermission("user\_add")

**public** **void** add {

@RequiresPermission("user\_add")

**public** **void** addUI {

1. url配置方式：

/user/delete = perms[user\_delete]

/user/addUI = perms[user\_add]

/user/add = perms[user\_add]

说明:当你请求地址为 项目名:8080/user/delete时，必须要权限认证key为user\_delete，则这样业务模块和方法更加清晰，后期维护更简单点

## 3.Mybatis的mapper映射规范

**Xml命名规范：**

骆驼峰命名法，实体类名（首字母小写）+ 下划线 + mapper结尾

如：user\_mapper.xml

**Mapper持久化方法规范：**

SQL唯一key对应的持久化Dao方法名映射规范

插入：insert(默认)

修改：update(默认ID)

删除：delete（默认ID）deleteBy\*（条件）

一般查询：get（默认ID）getByWhere（条件）getBy\*（特定条件）

分页查询：query（结果集）count（条数）

## 4.MVC层方法名定义规范及Base默认实现

**Mvc层方法定义规范：**

|  |  |  |  |
| --- | --- | --- | --- |
| **操作** | **持久层Mapper/DAO** | **服务层** | **控制层** |
| 添加 | insert | save | add/addUI |
| 修改 | update | update | edit/editUI |
| 删除 | delete | delete | delete |
| 查询（ID） | get | getById | 业务而定 |
| 查询（条件） | getByWhere | getByWhere | 业务而定 |
| 查询（特定条件） | getBy\* | getBy\* | 业务而定 |
| 查询（分页） | query/count | pageList/pageCount | pageList/pageCount |

**Base默认方法声明和方法实现：**

为了减少工作量，提供开发效率,会让base去默认声明或实现CRUD基本方法

1. 持久层：BaseDao默认接口方法声明

insert、update、delete、get、getByWhere、delete、query/count

1. 服务层：BaseServiceImpl默认方法实现

save、update、delete、getById、getByWhere、pageList/pageCount

提示：1）满足业务需求的，controller层直接调用即可

2）不完全满足业务需求的，只需要@override对应Base服务层的方法即可

（针对：基本满足需求，但还需要在此基础上添加业务逻辑的或拓展的）

3）对于特定或特殊的业务需求的，没办法只有自己编码实现