# 开发环境配置

## 1、安装统一版本JDK

Java 8允许我们给接口添加一个非抽象的方法实现，只需要使用 default关键字即可，这个特征又叫做扩展方法。

## 2、安装Eclipse4.6 和必要插件

通过Help-- >Install New Software安装插件

设置工作空间编码：UTF-8

修改Eclipse的JDK设置：设置为自己安装的JDK（在首选项的Java/Installed JRE中设置）-而不是使用JRE

注意：由于网速慢，在安装插件时，需要计算依赖关系非常缓慢，请耐心等待（40mins）；

## 3、安装maven

3.1使用eclipse4X自带的maven插件，配置本地的maven仓库，在本地的用户目录下新建setting.xml配置文件

3.2修改Eclipse的首选项—Maven---user Settings; 第一项配置为系统用户目录下的settings.xml，例如：C:\Users\Franky\.m2\settings.xml

3.3 maven项目基本操作：右键项目

1） Maven-update project

1. run as – maven install
2. run as – maven build

3.4 发布项目到私服

操作命令：Run as……..Maven build （第七项），在goals的文本框中输入deploy运行即可；

发布项目打包到Nexus私服需要的配置：

<distributionManagement>

<repository>

<id>releases</id>

<url>http://xxxxx:8081/nexus/content/repositories/releases/</url>

</repository>

<snapshotRepository>

<id>snapshots</id>

<url>http://xxxxx:8081/nexus/content/repositories/snapshots/</url>

</snapshotRepository>

</distributionManagement>

## 3、项目命名

命名方式：项目英文/中文拼音名称\_开发组编号\_序列号

## 4、jsp/html命名规则

2.1文件名全部用小写，多个单词之间用\_分隔

例如：new\_message.html或my\_file\_list.jsp

## 5、数据库

### 5.1 表命名方式

采用google标准格式：模块名称\_结构名称\_存储信息名词。

例如某张表： base\_t\_user，某视图 base\_v\_user\_role。

用`t`表达表，用`v`表达视图，特殊地，`t\_j`表明这是一张关联表，`j`代表join，`t\_i`表明这是一张接口表，`i`代表interface。

### 5.2 字段命名方式

字段要避免与数据库关键字和保留字冲突。

主键统一命名为`id`。

所有布尔类型的字段，统一用`is\_`开头。例如`is\_deleted`。

所有枚举类型的字段，统一用`enum\_`开头。例如`enum\_state`。

所有json类型的字段，统一用`\_json`结尾。

所有时间戳类型的字段，统一用`\_time`结尾。

所有日期类型的字段，统一用`\_date`结尾。

所有表达“总数”的字段，统一用`\_total`结尾。

所有外键关联的字段，统一用`\_id`结尾。

所有varchar类型字段，根据实际需要给定范围。

能给默认值的字段，一定要给默认值。原则上非空或者不能重复的字段，一定要配置好。

## 6 java后台

### 6.1 基本命名规则

命名时首先考虑语义化，即是否能只通过读名字就知道某些信息，无论在系统的哪个部分，都避免取无意义的，或者容易发生混淆的名称。

其次考虑简明，在满足语义化的前提下，尽量缩短名字的长度，使用容易理解的缩写，以及基本的常用英文单词。避免使用生僻词和过长的词。

程序中如果有缩写的话，全大写，比如OK不要写成Ok。

\*\*严格禁止使用拼音命名！\*\*

DAO层方法中，如果参数达到四个或者四个以上，就封装成dto，反之则使用`@Param`注解。

在程序中的命名使用驼峰命名法，某些缩写使用全大写字母，比如`JSON`。

方法使用动词开头。尽量避免使用`get`和`set`开头，因为这样容易与属性访问器混淆。

一个方法只做一件事情，如果命名时候出现“做某件事and某件事”，就说明方法需要拆分。

查询语句中，比较常用的有三类：返回一条记录，返回多条记录，返回一个count值。分别命名：

返回一条：以`queryOne`开头。

返回多条：以`queryList`开头。

返回全部：以`queryAll`开头。

返回count：以`count`开头。

新增：`save`/`insert`

修改：`update`

假删除：`delete`

真删除：`distory`

在DAO层中保持以上命名规则，在命名时，更多地使用SQL语言中的动词如update等，以清楚地表明该方法直接操作了数据库，相对的，在业务层中避免使用以上的方法开头，也避免使用SQL语言中的关键字，而应该倾向于体现业务逻辑。

一般情况下，方法应该尽量有一个返回值而不使用void。

update，insert，delete语句的返回值使用int，不要用void。

在程序中，如果需要在远端获取数据，比如调用api返回等，使用`pull`开头。

### 6.2.常量类和工具类

常量类要将构造方法私有化，并加`final`前缀。内部的常量字段全部加`public static final`前缀。

工具类分两种，一种是以`tool`结尾，这种工具类更接近业务逻辑，依赖除JDK以外的包。一种是以`util`结尾，这种工具类更接近底层算法，除了JDK没有其他任何依赖。

6.3.配置管理类

系统上线运行后经常要修改一些常用的变量，比如给用户的一些帮助信息、异常提示等，我们可以将其放到\*.properties文件中保存，变量即可随时修改而无需重启web服务器。

## 7.REST url设计

在保证语义化的基础上，尽量简化单词，隐藏原有意义，避免暴露逻辑。

模块自身的业务接口统一以`/m`开头;

公用的API数据接口统一以`/a`开头;

后续逐级细分功能，如

/m/a/book

表明是一个模块，是模块当中的图书漂流，是图书漂流中的图书管理功能;

但以上url应该继续修正为：`/m/c/b`或者`/m/b\_c/b`这种形式，这样能避免用户从url中猜出业务逻辑。

REST风格要求在url上不体现操作动作，而全部使用名词来指定资源，在请求中指定操作动作，但实际应用中达不到这样的标准。所以建议统一用CRUD来代表增删改查四种操作，对于开发人员有足够的语义，对于用户也能起到足够的混淆。比如这样：

/m/c/b/r/{bookId}

即代表根据bookId读取一条记录。

同样是查询，可能根据不同需求有多种实现，这时可以用数字划分，比如：

/m/c/b/r/1/{groupId}

即根据套系查询书籍。

尽量避免在url上提供参数，url上能提供的参数只应该有id，且只能有一个，即\*\*通过get请求，并将参数拼接在url上返回的场景，只能是单主键查询\*\*。其他的操作统一使用post请求。这样可以保持url的短小简洁。

## .代码提交

代码提交前使用Ctrl+Shift+F格式化，以减少因个人习惯不同造成的代码更新冲突，

与自己开发环境相关配置的修改，一定不要提交到SVN（在提交代码时，不要直接

在项目根目录上点击提交，只提交代码、配置文件的修改即可）

## .代码注释

使用Maven生成站点报告和JavaDoc文档，因此要为方法添加详细的注释，并且要符合JavaDoc的规范；

某些接口文件的方法相当多、且有些方法含义接近，因此详细的注释是必须的：查询条件、是否分页等；

方法的用途-----保证阅读通顺、容易理解、无歧义;

 @param-----参数意义

 @return-----返回值是什么、可能情况

 @author信息----白盒测试也便于找责任人