Java Web项目编程规范

[一 目的 2](#_Toc393701699)

[二 环境搭建 2](#_Toc393701700)

[2.1.1 环境规则 2](#_Toc393701701)

[三 命名及开发规范 3](#_Toc393701702)

[3.1 html页面元素命名及开发规范 3](#_Toc393701703)

[3.1.1 页面元素命名规范 3](#_Toc393701704)

[3.1.2 页面元素开发规范 3](#_Toc393701705)

[3.2 脚本变量的命名和开发规范 4](#_Toc393701706)

[3.2.1 脚本变量命名规范 4](#_Toc393701707)

[3.2.2 脚本函数以及子过程命名规范 5](#_Toc393701708)

[四 WEB面页开发规范 5](#_Toc393701709)

[4.1 客户端程序部分 5](#_Toc393701710)

[4.2 服务器端程序部分 6](#_Toc393701711)

[五 Java程序及JSP代码的风格 7](#_Toc393701712)

[5.1 Java命名 7](#_Toc393701713)

[5.2 总体风格 7](#_Toc393701714)

[5.3 类文件风格 9](#_Toc393701715)

[六 开发测试环境的规范 11](#_Toc393701716)

[6.1 数据库环境 11](#_Toc393701717)

# 目的

为了保证编写出的程序都符合相同的规范，而且便捷，保证一致性、统一性更符合构件化的要求而建立的程序编码规范，要使程序易懂。

# 环境搭建

### 环境规则

|  |  |
| --- | --- |
| 开发工具 | Myelipse10(或2013) |
| 数据库软件 | mysql5.0（或以上） |
| 托管工具 | Github |
| 字符编码 | UTF-8 |
| 开发运行环境 | Tomcat6 |
| 单元测试 | Junit |
| 框架 | SSH |
| Spring | 3.1 |
| hibernate | 4.1 |
| Struts2 | 2.3.1 |
| Java环境 | JDK 1.6 |

使用ssh 还是只用struts2 待定。。。

# 命名及开发规范

所有命名（类、函数、变量..）均要求意义明确易于理解，尽量使用有实际意义的英文单词或英文单词的缩写，避免在代码中直接使用数字等不确定意义的词，更切忌使用中文拼音的首字母。如这样的名称是不提倡的：Value1，Value2，Value3，Value4 …。

## html页面元素命名及开发规范

### 页面元素命名规范

|  |  |  |
| --- | --- | --- |
| **Html元素** | **缩写** | **例如** |
| text输入框 | txt | txtReason |
| button按钮 | btn | btnCheck |
| select下拉选择框 | sel | selBeginTime |
| option项 | opt | optDate |
| Checkbox多选项 | chk | chkProduceNumber |
| form表单 | frm | frmContractQuery |
| frame框架 | fra | fraWorkProcess |
| hidden表单项 | hdn | hdnSteelNumber |
| div标记 | div | divResult |
| span标记 | span |  |
| 对话框对象 | dlg |  |
| 窗口对象 | win |  |

### 页面元素开发规范

* img元素

alt：所有展示类图片都要具有能简要描述图片内容的文字说明。

* Input元素

maxlength：所有INPUT控件都需要制定maxlength属性，默认值为数据库中对应的字段的长度。readonly：所有不可更改的信息都要使用readonly属性。

* Form元素

action：所有Form都要指定action，如果提交给本身就指定action=""

method：执行不可逆动作使用POST，可逆动作使用GET

onsubmit：所有form都要指定提交前需要的检查程序。

* button元素

onclick：form中用于提交的button不容许使用此方法，所有数据检查通过form的onsubmit激活。

* title属性

所有页面都要具有和本页标题相同的title。

* 语言设置

所有中文页面都要加上如下语句：

<meta http-equiv="Content-Language"content="zh-cn">

<meta http-equiv="Content-Type"content="text/html;charset=gb2312">

* 元素属性赋值

所有元素的属性值都要使用双引号包括起来。

* 元素缩进

所有子元素比父元素缩进两个空格。

## 脚本变量的命名和开发规范

### 脚本变量命名规范

* 常量以及全局变量名必须全部使用大写字母。
* 变量名除首字母小写外，其他单词首字符必须大写
* 所有命名（类、函数、变量..）均要求意义明确易于理解，尽量使用有实际意义的英文单词或英文单词的缩写，避免在代码中直接使用数字等不确定意义的词，更切忌使用中文拼音的首字母。如这样的名称是不提倡的：Value1，Value2，Value3，Value4 …。
* 变量的使用尽量缩小到小的作用域。如循环使用

for(var i = 0; i < 12; i++){

}

　　而不是：

　　var i;

for(i = 0; I < 12; i++){

}

### 脚本函数以及子过程命名规范

* 函数命名必须能够体现函数的功能
* 函数命名第一个单词的首字母小写，后面每一个单词的首字母大写
* Dao 层命名如下：
* getByXXX 表示根据XXX条件获取单个对象 不要用geUsertByXXX
* findByXXX 表示根据XXX条件获取对象集合
* save 表示保存对象
* delete 表示删除对象
* update 表示更新
* 如果要删除多个,使用方法重载

# WEB面页开发规范

## 客户端程序部分

* 错误提示信息的处理

所有错误信息全部使用中文提示错误信息，标点使用中文半角符号，格式如下：

"错误："+提示信息+"！"

* 成功提示信息的处理

所有成功信息全部使用中文提示成功信息，标点使用中文半角符号，格式如下：

"成功："+提示信息+"！"

* 建立错误代码和错误页面机制，客户端显示错误页

服务端异常时，如：应用服务器页面不存在异常 （错误代码：404）

应用服务器页面出现错误代码 （错误代码：500）

在web.xml文件中对异常进行声明，如下所示：

<error-page>

<error-code>404</error-code>

<location>/页面不存在错误页.html</location>

</error-page>

<error-page>

<error-code>500</error-code>

<location>/服务端内部错误页.html</location>

</error-page>

<error-page>

<exception-type>java.lang.NumberFormatException</exception-type>

<location>/运行异常错误页.html</location>

</error-page>

* 页面的返回

所有需要返回上一页的时候使用history.back();不使用history.go(-1);

需要返回前n页（n>1）时使用history.go(-n)；

* 提交前数据的判断

保证提交前的数据都会通过后台进行数据类型以及长度的判断；

长度判断：长度要判断去掉前后空格后的实际长度；

为空判断：所有不容许为空的输入字段都要在去掉前后空格后进行判断，同时如果该字段为查询条件则必须不能为空；

如果判断条件发现数据错误，则提示错误信息；

所有涉及删除的操作，在用户选定以后都要再进行一次确认操作。

* 页面中session、application的使用规范

统筹定义JSP内建对象session、application的属性（提高模块内相关功能间对象属性使用率）

* 在页面中使用Java Bean、等时注意声明其作用范围：scope。如果用STRUTS架构，一般将DTO放置在session中，页面上没有设置，但在Action中注意session的清空，否则占用系统内存。

## 服务器端程序部分

* 数据的取得

只通过Post方式获取数据，传递过来的数据在使用前都要通过trim去掉数据前后的空格。

* 对象的关闭

所有数据库和文件对象都要在使用后尽可能早的close，同时赋null。 一般在用数据连接后都要对其进行捕捉错误，在finally中再次清空数据连接。

* 提示信息

所有错误提示信息使用JavaScript提示，保证使用者看不到任何内部错误信息。（如1-1）

涉及数据库Update，Del，Insert的操作成功都要提示。（如1-2）

* 服务端的监听和过滤

针对服务程序中重点属性信息的修改，在服务端设置并启用监听程序，记录操作过程，并形成文本形式的日志文件；

页面字符乱码问题使用服务编码过滤器处理；

针对部分保密级高的目录使用服务访问认证过滤处理

# Java程序及JSP代码的风格

## Java命名

## 总体风格

* 缩进

缩进建议以2个空格为单位。语句块的“{”放到语句的末尾，“}”与其前一行对齐，便于匹对。

* 空格

原则上变量、类、常量数据和函数在其类型，修饰名称之间适当空格并据情况对齐。如：

String strRange = "0"; 而不是String strRange = "0";

关键字原则上空一格，如：if （ ... ） 等。运算符的空格规定如下： "["、"]"、"++"、"--"、"!"、"+"、"-"（指正负号）等几个运算符两边不加空格（其中单目运算符系指与操作数相连的一边），其它运算符（包括大多数二目运算符和三目运算符"?："）两边均加一空格。

* 对齐

原则上关系密切的行应对齐，对齐包括类型、修饰、名称、参数等各部分对齐。另每一行的长度不应超过屏幕太多，必要时适当换行，换行时尽可能在"，"处或运算符处，换行后最好以运算符打头，并且以下各行均以该语句首行缩进，但该语句仍以首行的缩进为准。

* 空行

不得存在无规则的空行，比如说连续十个空行。程序文件结构各部分之间空两行，若不必要也可只空一行，各函数实现之间一般空两行，由于每个函数还要有函数说明注释，故通常只需空一行或不空，但对于没有函数说明的情况至少应再空一行。对自己写的函数，建议也加上“//------”做分隔。函数内部数据与代码之间应空至少一行，代码中适当处应以空行空开，建议在代码中出现变量声明时，在其前空一行。类中四个“p”之间至少空一行，在其中的数据与函数之间也应空行。

* 注释

注释是软件可读性的具体体现。程序注释量一般占程序编码量的20%，软件工程要求不少于30%。程序注释不能用抽象的语言，类似于"处理"、"循环"这样的计算机抽象语言，要精确表达出程序的处理说明。例如："计算净需求"、"计算第一道工序的加工工时"等。避免每行程序都使用注释，可以在一段程序的前面加一段注释，具有明确的处理逻辑。

注释必不可少，但也不应过多，不要被动的为写注释而写注释。以下是四种必要的注释：

A. 标题、附加说明。

B. 函数、类等的说明。对几乎每个函数都应有适当的说明，通常加在函数实现之前，在没有函数实现部分的情况下则加在函数原型前，其内容主要是函数的功能、目的、算法等说明，参数说明、返回值说明等，必要时还要有一些如特别的软硬件要求等说明。公用函数、公用类的声明必须由注解说明其使用方法和设计思路。如：

/\*\*

\*@author name time

\*@version version-number

\*@param parameter-name description

\*@return description

\*/

C. 在代码不明晰或不可移植处必须有一定的说明。

D. 及少量的其它注释，如自定义变量的注释、代码书写时间等。

E．修改、增加或注释掉的代码行需要标注修改/增加人、时间，必要时写明原因或实现的功能。

建议以如下格式说明：修改-------- //changed by yangjingping 2006-4-26

增加--------- //added by yangjingping 2006-4-26

注释掉------ //concealed by yangjingping 2006-4-26

* 代码长度

对于每一个方法建议尽可能控制其业务代码长度30行左右，超过30行的代码要重新考虑将其拆分为两个或两个以上的函数。函数拆分规则应该一不破坏原有算法为基础，同时拆分出来的部分应该是可以重复利用的。对于在多个模块或者窗体中都要用到的重复性代码，完全可以将起独立成为一个，放置于一个公用模块中。

* 页宽

页宽应该设置为80字符。源代码一般不会超过这个宽度， 并导致无法完整显示， 但这一设置也可以灵活调整. 在任何情况下， 超长的语句应该在一个逗号或者一个操作符后折行. 一条语句折行后， 应该比原来的语句再缩进2个字符。

* 行数

把完成比较独立功能的程序块抽出，单独成为一个函数。把完成相同或相近功能的程序块抽出，独立为一个子函数。可以发现，越是上层的函数越简单，就是调用几个子函数，越是底层的函数完成的越是具体的工作。这是好程序的一个标志。这样，我们就可以在较上层函数里容易控制整个程序的逻辑，而在底层的函数里专注于某方面的功能的实现了。

## 类文件风格

所有的 Java(\*.java) 文件都必须遵守如下的样式规则：

* package/import

package 行要在 import 行之前，import 中标准的包名要在本地的包名之前，而且按照字母顺序排列。如果 import 行中包含了同一个包中的不同子目录，则应该用 \* 来处理。

package hotlava.net.stats;

import java.io.\*;

import java.util.Observable;

import hotlava.util.Application;

这里 java.io.\* 使用来代替InputStream and OutputStream 的。

系统类引用和自定义类引用用空行分开

* class类的注释

/\*\*

\* @Description

\* @author

\* @date 2014-7-21 上午10:20:29

\* @version V1.0

\*/

* 构造函数

接下来是构造函数，它应该用递增的方式写（比如：参数多的写在后面）。

public Material (int p\_intSize){

  this.size = p\_intSize;

}

* 类方法

下面开始写类的方法：

/\*\*

\*@author liandti

\*@version V1.3

\*@param key需要加到MaterialCounters中的物料主键。

\*@return int MaterialCounters

\*/

public MaterialCounters addMaterial(String key){

……

}

* 神秘的数

首先要说什么是神秘的数。我们在程序里经常会用到一些量，它是有特定的含义的。例如，现在我们写一个薪金统计程序，公司员工有50人，我们在程序里就会用50这个数去进行各种各样的运算。在这里，50就是"神秘的数"。为什么称它为神秘呢？因为别的程序员在程序里看到50这个数，不知道它的含义，只能靠猜了。

在程序里出现"神秘的数"会降低程序的可读性，应该尽量避免。避免的方法是把神秘的数定义为一个常量。注意这个常量的命名应该能表达该数的意义，并且应该全部大写，以与对应于变量的标识符区别开来。例如上面50这个数，我们可以定义为一个名为NUMOFEMPLOYEES的常量来代替。这样，别的程序员在读程序的时候就可以容易理解了。

# 开发测试环境的规范

## 数据库环境

服务器：59.70.149.46

用户：root

密码：itstudio\_zzuli

* 附： 所有代码修改前必须先从服务器同步代码。
* 代码提交前 必须“格式化”