软件编程规范说明书

1. **java编码规范**

**1、命名规范**

定义这个规范的目的是让项目中所有的文档都看起来像一个人写的，增加可读性，减少项目组中因为换人而带来的损失。（这些规范并不是一定要绝对遵守，但是一定要让程序有良好的可读性）

较短的单词可通过去掉“元音”形成缩写

较长的单词可取单词的头，并用括号明确表达式的操作顺序，避免使用默认优先级

**Package的命名**

Package的名字应该都是由一个小写单词组成。

**Class的命名**

Class的名字必须由大写字母开头而其他字母都小写的单词组成，对于所有标识符，其中包含的所有单词都应紧靠在一起，而且大写中间单词的首字母。

**Class变量的命名**

变量的名字必须用一个小写字母开头，后面的单词用大写字母开头。

Static Final变量的命名

Static Final变量的名字应该都大写，并且指出完整含义。

**参数的命名**

参数的名字必须和变量的命名规范一致。

**数组的命名**

数组应该总是用下面的方式来命名：

Byte[ ]buffer;

而不是：byte buffer[ ]

**方法的参数**

使用有意义的参数命名，如果可能的话，使用和要赋值的字段一样的名字。

**2、变量定义规范**

去掉没必要的公共变量。

构造仅有一个模块或函数可以修改、创建，而其余有关模块或函数只访问的公共变量，防止多个不同模块或函数可以修改、创建同一个公共变量的现象。

仔细定义并明确公共变量的含义、作用、取值范围及公共变量间的关系。

明确公共变量与操作此公共变量的函数或过程的关系，如访问、修改及创建等。

当向公共变量传递数据时，要十分小心，防止赋予不合理的值或越界等现象发生。

防止局部变量与公共变量重名。

仔细设计结构中元素的布局与排列顺序，使结构容易理解、节省占用空间，并减少引起误用现象。

结构的设计要尽量考虑向前兼容和以后的版本升级，并为某些未来可能的应用保留余地（如预留一些空间等）。

留心具体语言及编译器处理不同数据类型的原则及有关细节。

严禁使用未经初始化的变量，声明变量的同时对变量进行初始化。

编程时，要注意数据类型的强制转换。

**3、代码编写格式**

**{ }对**

{ }中语句应该单独作为一行。

**括号**

左括号和后一个字符之间不应该出现空格，同样，右括号和前一个字符之间也不应该出现空格。

**4、注释规范**

定义这个规范的目的是让项目中所有额文档都看起来像一个人写的，增加可读性，减少项目中因为换人而带来的损失。

注释要简单明了。

边写代码边注释，修改代码同时修改相应的注释，以保证注释与代码的一致性。

在必要的地方注释，注释量要适中，注释的内容要清楚、明了、含义准确，防止注释二义性，保持注释与其描述的代码相邻，即注释的就近原则。

对代码的注释应放在其上方相邻位置，不可放在下面。对数据结构的注释应放在其上方相邻位置，不可放在下面；对结构中的每个域的注释应放在此域的右方；同一结构中不同域的注释要对齐。

变量、常量的注释应放在其上方相邻位置或右方。全局变量要有较详细的注释，包括对其功能、取值范围、哪些函数或过程存取以及存取时注意事项等的说明；在每个源文件的头部要有必要的注释信息，包括：文件名；版本号；作者；生成日期；模块功能描述；主要函数或过程清单及文本历史修改记录等。

在每个函数或过程的前面要有必要的注释信息，包括：函数或过程名称；功能描述；输入、输出及返回值说明；调用关系及被调用关系说明等。

**5、函数、过程**

函数的规模尽量限制在200行以内。

一个函数最好仅完成一件功能。

为简单功能编写函数。

函数的功能应该是可以预测的，也就是只要输入数据相同就应产生同样的输出。

尽量不要编写依赖其他函数内部实现的函数。

避免设计多参数函数，不使用的参数从接口中去掉。

用注释详细说明每个参数的作用、取值范围及参数间的关系。

检查函数所有参数输入的有效性。

检查函数的所有非参数输入的有效性，如数据文件、公共变量等。

函数名应准确描述函数的功能。

避免使用无意义或含义不清的动词为函数命名。

编写可重入函数时，若使用全局变量，则应通过关中断、信号量等手段对其加以保护。

**6、程序编写规范**

**exit( )**

exit除了在main中可以被调用外，其他的地方不应该调用。因为这样做不给任何代码机会来截获退出。一个类似后台服务的程序不应该因为其他某一个库模块决定了要退出就退出。

**异常**

申明的错误应该抛出一个RuntimeException或者派生的异常。

顶层的main（）函数应该截获所有的异常，并且打印在屏幕上啊。

**垃圾收集**

Java使用成熟的后天垃圾收集技术来替代引用计数。但是这样会导致一个问题：必须在使用完对象的实例后进行清场工作。

**Final类**

绝对不要因为性能的原因将类定义为final。

如果一个类还没有准备好被继承，最好在类文档中注明，而不要将他定义为final的，这是因为没有人可以保证会不会由于什么原因需要继承他。

1. **排版规范**

关键词和操作符之间加适当的空格。

相对独立的程序块与块之间加空行。

较长的语句、表达式等要分成多行书写。

划分出的新行要进行适应的缩进，使排版整齐，语句可读。

长表达式要在低优先级操作符出划分新行，操作符放在新行之首。

循环、判断等语句中若有较长的表达式或语句，则要进行适应的划分。

若函数或过程中的参数较长，则要进行适当的划分。

不允许把多个短语写在一行中，即一行只写一条语句。

函数或过程的开始、结构的定义及循环、判断等语句中的代码都要采用缩进分格。

1. **Java的文件格式**

**Package/Imports**

Package行要在imports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行中包含了同一包中不同子目录，则应该用\*来处理。

**Class Fields**

Public的成员变量必须生成文档。Proceted、private和package定义的成员变量如果名字含义明确的话，可以没有注释。

**二、JSP编码规范**

整个jsp/jsp bean 表示层应当尽可能的瘦和简单化。

牢记大多数的jsp都应当是只读的视图，而由页面bean来提供模型。

应当一起设计jsp和jsp bean.

在尽可能合理的情况下，把业务逻辑从jsp中移走。

尽量把条件逻辑放在控制器中而不是视图中。

为jsp、包含的文件、jsp bean和实现扩展标记的类使用遵循标准的命名惯例。

应当在jsp中避免使用页面引入指令。

Jsp不应该访问属性文件，或者使用bean可以访问属性。

如果jsp bean具有的所有的属性不能够从页面请求中被映射到，就要尽力在<jsp:usebean>标记中设置属性。

应当避免设计即显示表单又处理结果的页面。

在jsp中避免代码重复，把重复的功能放在一个包含jsp、bean或标记扩展中，使得它能够被重用。

Jsp bean应当永远不要去产生HTML。

Jsp层不应该直接访问数据，这包括JDBC数据库和EJB访问。

在长度上，scriptlests的代码最好不要超过五行。

除了jsp bean之外，jsp不应当去实例化复杂的可读写的对象。如果这样的话，就有可能在jsp中去执行不适当的业务逻辑。

定制标记应当适当把逻辑从jsp中移走的地方。

应当使用隐藏的注释来组织输出HTML过大。

在jsp文件中都应当使用一个错误页面来处理不能够从中恢复的异常。

在jsp错误页面中，使用HTML注释来显示传递到该页面中的异常跟踪信息。

只有在能够获得性能上的好处时，才使用jspInin()方法和jspDestroy()方法。获取和放弃资源是jsp beans和标记处理器的事，而不是由jsp来负责的。

应当谨慎的使用<jsp：forward>标记，在jsp中它是一个等价的goto。

如果没有充分的理由，就不要再jsp中定义方法。

1. **Struts编码规范**

**Action、Form、Bean命名规则**

Action:命名以Action结尾，

例：添加新闻的Action，命名为AddNewAction

修改新闻的Action，命名为UpNewAction

删除新闻的Action，命名为DelNewAction

查看新闻的Action，命名为ViewNewAction

**Form:命名以Form结尾**

操作新闻的Form.命名为NewForm

**Bean:命名以Bean结尾**

操作新闻的bean，命名为NewBean