# 项目编码规范

# **第1章 概述**

## 1.1 前言

代码之于程序员,就像零件之于机械工,庄稼之于农民,它是软件的基石,一行行代码都是程序员的心血经过日日夜夜凝结成的。做为一个程序员, 应该像母亲呵护孩子一样呵护自己的代码,它不仅仅是一行一行的文字,它是一个程序员的尊严和价值所在;它是活的,你甚至能感受到它的心跳。编码规范只是大家达成一致的约定,这样大家的代码就可以互相看懂,维护起 来更加容易,思想更畅快的交流,经验更快的得到传播。代码规范不是束缚程序员的桎梏,应该知道,不遵守规范的个性的代码并不代表程序员的性格,并不能张扬个性。个性应该体现在用更简单、更优雅、更易读、更易理解以及算法实现效率更高等方面。

可读性,可理解性是代码的重要方面,本规范主要围绕如何去产生规范易读的代码。另外,它也保证了大家有共同的先验知识。

## 1.2 术语

Pascal case - 所有单词第一个字母大写,其它字母小写。 例: OrderDetail, OilTank。

Camel case - 除了第一个单词,所有单词第一个字母大写,其他字 母小写。

在后面的描述中我们使用 Pascal 代表第一种表示方式, Camel 代表第二种表示方式。 1.3 约束

没有一个规范可以到处适用,也不可能无所不包。

1. 本规范并非完全强制性规范,对于任何违背本规范但能提高代码可读性的措施,都可以采用。

2. 本规范第 3,4,5 章为强制性规范,其它章节为建议规范。如违背本条,请参考第一条。

# 第2章 一般规则

1) 所有包,类,接口,方法,属性,变量,参数均使用英文单词进行命名,具 体细节请参见命名规范一章。

2) 命名包,类,接口,方法以及变量时,尽量使用贴近问题域的表意丰富的名称。

3) 修改源代码时,应尽量保持与所修改系统的编码风格保持一致。

4) 所有包名使用必须使用 com.yao前缀,所有项目使用 com.yao.[project name], project name 是项目的缩写。

# 第3章 格式规范

5)包的导入应该按照相关性进行分组。

import java.io.IOException;

import java.net.URL;

import java.rmi.RmiServer;

import java.rmi.server.Server;

import javax.swing.JPanel;

import javax.swing.event.ActionEvent;

import org.linux.apache.server.SoapServer;

6) 只倒入明确需要的类,禁止通配导入。这样只要看导入列表,就可以知道该类依赖于哪些类和接口,保证可读性。

import java.util.List; // 避免: import java.util.\*

import java.util.Arraylist;

import java.util.HashSet;

7) 类和接口中元素的布局顺序。

1. 类和接口的文档描述

2. 类和接口的声明

3. 类的静态变量,按照 public,protected,package,private 的顺序。

4. 实例变量,按照 public,protected,package,private 的顺序。

5. 类的方法,无固定顺序。

8) 方法修饰关键字定义顺序。

public static double square(double a); //避免: static public double square(double a);

9) 变量声明,采用 Camel 表示法不要在一行声明多个变量。

10) 保证明确的类型转换,不要默认进行隐式类型转换

intValue = (int) floadValue; //避免 intValue = floatValue

11) 数组指示符紧跟类型变量 int[] a = new int[20]; // 避免: int a[] = new int[20]

一个变量要代表独立的意思,不要在其生命周期赋予它不同的概念。

int tempValue;

tempValue = maxValue;

tempValue 在生命周期内表示了各种各样的意图,增加理解代码的难度。 应该为每个独立概念定义单独的变量:

int tempMaxValue;

int tempMinValue;

int tempAnotherValue;

12) 仅仅循环控制变量才能出现在 for()循环中

sum = 0;

for (i = 0; i < 100; i++) {

sum += value[i];

}

14) 循环变量应靠近循环体初始化

isDone = false

while(!isDone){

}

//避免

isDone = false;

while(!isDone){

}

1. 避免长的布尔表达式,应换成多个更容易理解的表达式。
2. bool isFinished = (elementNo < 0) || (elementNo > maxElement);

bool isRepeatedEntry = elementNo == lastElement;

if (isFinished || isRepeatedEntry) {

}

16) 不要在条件语句中执行方法,以提高可读性

InputStream stream = File.open(fileName, "w");

if (stream != null) {

}

17) 代码缩进,应该使用 4 个空格为一个单位进行缩进。 （tab）

public String invoke() throws Exception {

....String profileKey = "invoke: ";

try {

....UtilTimerStack.push(profileKey);

if (executed) {

....test = true;

}

catch{ }

}

18) 条件语句的主要形式,即使单条语句,也要使用括号括起来。

if (condition) {

statements;

}

if (condition) {

statements;

} else {

statements;

}

if (condition) {

statements;

} else if (condition) {

statements;

} else {

statements;

}

20) switch 语句的使用格式

switch (condition) {

case ABC :

statements; //穿透,一定要做出注释

}

24) 逻辑上紧密相关的代码块应该用一个空行分开。

// Create a new identity matrix

Matrix4x4 matrix = new Matrix4x4();

// Precompute angles for efficiency

double cosAngle = Math.cos(angle); double sinAngle = Math.sin(angle);

// Specify matrix as a rotation transformation

matrix.setElement(1, 1, cosAngle);

matrix.setElement(1, 2, sinAngle);

matrix.setElement(2, 1, -sinAngle);

matrix.setElement(2, 2, cosAngle);

// Apply rotation

transformation.multiply(matrix);

28) 三元条件运算符,条件要用括号括起来。

alpha = (aLongBooleanExpression) ? beta : gamma;

29) 代码每次提交前，需要格式化一下

# 第4章 命名规范

## 4.1 一般命名规范

29) 包名应该用小写字母,不要出现下划线等符号,名词用有意义的缩写或者英文单词。

//推荐

com.esse.business

java.lang.util

30) 所有类命名使用 Pascal 表示方式,使用名词组合。

UserManager, ClassLoader, HttpHeaderResult

31) 接口命名Pascal 形式的表示方式,前面不使用I。

Query,DataAccess,ReportBuilder,

32) (可选)使用名词组合或形容词去命名一个接口,接口声明了一个对象能提供的 服务,也描述了一个对象的能力。一般以“able”和“ible”作为后缀, 代表了一种能力。

public interface Runnable{ public void run();

}

public interface Accessible{ public Context getContext();

}

33) 变量名和参数名使用 Camel 表示方式。 userName, objectFactory, entrys, list

34) 对于常量名,使用大写字母,并使用下划线做间隔。 MAX\_TIMES, DEFAULT\_NAME

这样做的好处是,当因需要修改实际的数字时,比如修改 25 为 30,只需要修改一处。

35) 方法名应该使用动词开头,使用 Camel 表示方式,一般由动词+名词组 成。

getName, initialize, addParameter, deleteUser

36) 缩写字母也应该保持首字母大写

exportHtmlSource(); // 避免: exportHTMLSource();

openDvdPlayer(); // 避免: openDVDPlayer();

37) 参数的名字应该和类型贴切，禁止单个字符的名称

void setTopic(Topic topic)

当同时定义多个属于同一个类的变量时,把类型作为实例的后缀,如:

Point startPoint;Point centerPoint;

38) 根据变量的作用范围,作用范围大的应该使用长名称,作用范围大,表 明变量的生命周期比较长,为了有助于理解,应尽量用长名称以表达变量的

## 4.2 特殊命名规范

39)  使用 get/set 对类属性进行访问,这是 Java 社区的核心编码规范。

40)  使用 is 前缀表示一个布尔变量和方法。

isUsed, isEmpty,isVisible,isFinished

有时也可以使用 has,can,should:

boolean hasLicense();

boolean canEvaluate();

boolean shouldAbort = false;

41) 在Dao查询方法中应使用 find,query 作为前缀

vertex.findNearestVertex();

matrix.findSmallestElement();

node.findShortestPath(Node destinationNode)

42) 在Dao更新方法中应使用update,delete等作为前缀

42) 使用 initialize 做为对象初始化的方法前缀,也可以简写为 init

initializeFiles();init();initFontSet();

43) 对于对象集合, 变量名称应使用复数。 Collection<Point> points;int[] values;

44) 对于抽象类应该使用 Abstract 前缀。

45) 对于表示编号的变量,应加 No 后缀。 tableNo, userNo,employeeNo

46) 常在一起使用的对称词汇,这些词汇一起使用,方法的表达意图自然可以互相推测和演绎。

47) 避免使用否定布尔变量

bool isError; // 避免: isNoError

bool isFound; // 避免: isNotFound

48) 异常类应该使用 Exception 做为后缀。

AccessException, RuntimeException

49) 缺省接口实现应该使用 Default 前缀

class DefaultTableCellRenderer

implements TableCellRenderer {

}

50) 对于单例类(Singleton),应该使用 getInstance 方法得到单例。

class UnitManager {

private final static UnitManager instance = new UnitManager();

private UnitManager() {

}

public static UnitManager getInstance(){ return instance\_;

}

51) 对于工厂类,进行创建对象的方法,应该使用 new 前缀

class PointFactory { public Point newPoint(...) {

}

51) 接口的实现类后缀需要Impl,在Service和Dao层

# 第5章 注释规范

## 5.1 概述

代码中为什么要包含注释?

除了自己,别人要阅读和维护你的代码。为了使代码更容易维护,首先要使代码更易于理解,才能在理解的基础上进行维护。对这些代码进行文档描述,将使这个过程变得更加容易。 对代码进行注释,是在代码可读性的基础上,使用自然语言对代码所表达的意思进行阐述。并不是说代码可以写的很烂,注释写的很详细,这不是好的方式。对于类,主要着重要描述它的职责,即它能干什么,对于复杂的算法实现,应该使用内部实现注释,说明算法的主要思路,对于长方法,要让 阅读代码的人比较容易的明白方法实现的主要流程。反之,对于一看就懂的方法,则不需要进行注释,比如 get/set 方法。

## 5.2 一般原则

代码应该和注释保持同步,如果代码和注释不同步,则阅读代码的人会 想,“到底是代码准确,还是注释准确啊”。注释尽量简洁,尺度没有准确的定义,大部分人能明白即可,可以将自 己的代码给同事看看。太简单的方法就不要注释了,比如上面提到的 get/set 方法。

## 5.3 注释内容

package com.opensymphony.xwork2;

import com.opensymphony.xwork2.interceptor.PreResultListener;

Result getResult(String name) throws Exception{

String name = this.getName();

}

52) (可选)尽可能在类描述中加入代码调用示例,使用<pre></pre>标记,提示 JavaDoc 工具不要改变格式.

/\*\*

\* DateFormat is an abstract class for date/time formatting

\* formats and parses dates or time in a language-independent manner.

\*

\* <pre>

\* myString = DateFormat.getDateInstance().format(myDate);

53) 使用@deprecated废弃方法,不要删掉它。

/\* Get a default date/time formatter that uses the SHORT date and the time. public final static DateFormat getInstance() { return getDateTimeInstance(SHORT, SHORT);

}

\*/

包含代码调用示

54) (可选)使用行末注释对深层嵌套代码进行注释

for(i...){

for(j...){

while(...){

if(...){

switch(...){

}//end for j

55)对于需要加入代码，但是还未完成的，加入//TODO标记

# 第6章 编程实践

55) 对于静态方法,应该使用类名去使用,不应该用实例去引用,主要是为 了体现更多的语义。Thread.sleep(1000);

//避免,无法体现 sleep 是静态方法还是实例方法 thread.sleep(1000);

56) 对一些基本数据类型和不会变化的变量还有不太可能通过继承进行扩展的类,应声明为 final,提高效率,减少出错。

57) 类和方法的粒度保持适中,保持类的规模尽量短小,职责单一。小类有 很多好处,易于设计,易于测试,易于理解。同样方法也要尽量的小,每个方法尽量不要超出 25 行。

58) 开闭原则,软件应该对扩展开放,对修改关闭。也就是说,应该在不修改以前源代码的基础上,改变程序的行为以适应新的需求。

59) 里氏代换原则:假设有两个类,一个是基类 Base,一个是派生类 Derived,如果一个方法可以接受基类对象 b 的话:method1(Base b), 同样,这个方法也应该接受派生类 Derived 的对象 d,而不影响方法的行为。里氏代换原则是继承复用的基石。

60) 抽象依赖原则(稳定依赖原则)。应该依赖于抽象而不依赖与具体类,抽象的类和接口是稳定的,而具体类是易变的,如果依赖于具体类,代码就会非常脆弱,失去了灵活性。

61) 接口隔离原则,一个类对另外一个类的依赖应该建立在最小的接口之上的。

62) 单一职责原则,如果一个类有多于一种的职责,当需求变化时,类的职责就要发生变化,而因此就会引起引用该类的代码发生改变,职责越多,这 个类就容易跟更多的类产生耦合关系,而且改变一个职责,可能会影响到另 外一个职责的履行。

63) 编写代码前,先编写注释(可以认为是伪代码),先想后写。

/\*\*

\* 报表构建器,主要职责:

\* 1.创建拷贝报表模版

\* 2.填充报表数据

\* 3.构建报表

\* 4.画图处理

\*/

# 第七章 工程规范

## 7.1 工程布局

64)工程需要分模块开发

建议模块

[project.name]-common 通用模块(通用的工具类，一些静态变量)

[project.name]-business 接口消息对像模块(接口声明和传递的消息对像)

[project.name]-business-impl 接口实现模块(业务实现)

[project.name]-service 远程服务模块(dubbo服务启动和服务导出,thrift服务启动和服务导出，如有必要，也可单独拆分出thrift模块)

[project.name]-admin 后台管理控制(项目相关管理控制页面，一般会集成在manage.111.com.cn)

[project.name]-front 前端展示(有顾客界面的模块)

[project.name]-cache 缓存(常用的redis配置及与实现无关的缓存模块)

[project.name]- persistence 持久化模块(如果有需要分离的话)

其它模块有必要的话，自行增加.

65)必需以maven做为项目管理工具

66)maven配置文件jar包依赖关系，需要配置在使用这个jar包最近的地方

67)父模块和business模块禁止依赖任何jar包。父模块中只可以声明jar包，但是不能真正的依赖。

## 7.2 企业开发

68) Action,Service,Dao三层原则

Action中只处理请求、响应和页面显示的相关逻辑。

所有业务逻辑必须包含在Service 层

69) 事务必需是注解式配置

70) 无需事务的地方禁止添加事务，并且要区分readonly

71) 所有资源使用完必需保证有释放

72) 工程中需要有自己的异常体系，并为每种异常分配一个唯一的异常编码，并且需要正确处理异常

73) 关键业务流程需要增加业务日志，建议区分不同的日志记录器及每天一个日志文件

74) 所有新增接口需要增加系统级参数，以便区分逻辑和定位问题

1.来源:PC,H5,APP(Android,iOs)

75) 对于无法做到幂等性的接口调用，必需在消费者端将dubbo接口重试次数修改为0

76) 所有的接口禁止返回null

因为返回null没有明确的目的和意义。当需要返回空时，如果是集合，可以返回大小为0的空集合.。如果是对像，可以使用nullObject模式。