



安卓端时光笔记APP

代码规范

|  |  |  |  |
| --- | --- | --- | --- |
| 文件状态： | 文件标识： | | SE-2018春-G01 |
| [ ] 草稿 | 当前版本： | | 0.2.0 |
| [ ] 正式发布 | 作 | 者： | 郦哲聪、陈铉文、刘晓倩 |
| [√] 正在修改 | 完成日期： | | 2018-06-16 |

**版本历史**

|  |  |  |  |
| --- | --- | --- | --- |
| 版本/状态 | 参与者 | 起止日期 | 备注 |
| 0.1.0 | 郦哲聪、陈铉文、刘晓倩 | 2018-06-02至2018-06-07 | 代码规范初稿 |
| 0.2.0 | 郦哲聪、陈铉文、刘晓倩 | 2018-06-13至2018-06-16 | 文档随项目进度进一步增减 |

目录

[1.目的 5](#_Toc517216416)

[2.范围 5](#_Toc517216417)

[3.注释规范 5](#_Toc517216418)

[3.1概述 5](#_Toc517216419)

[3.2模块（类）注释 5](#_Toc517216420)

[3.3方法注释 6](#_Toc517216421)

[3.4代码间注释 6](#_Toc517216422)

[4.命名总体规则 6](#_Toc517216423)

[5.命名规范 7](#_Toc517216424)

[5.1变量（Variable）命名 7](#_Toc517216425)

[5.2常量命名 7](#_Toc517216426)

[5.3类（Class）命名 7](#_Toc517216427)

[5.4接口（Interface）命名 8](#_Toc517216428)

[5.5方法（Method）命名 8](#_Toc517216429)

[5.6名称空间Namespace）命名 9](#_Toc517216430)

[6.编码规则 9](#_Toc517216431)

[6.1错误检查规则 9](#_Toc517216432)

[6.2大括号规则 9](#_Toc517216433)

[6.3缩进规则 9](#_Toc517216434)

[6.4小括号规则 10](#_Toc517216435)

[6.5 If Then Else规则 10](#_Toc517216436)

[6.6 Case规则 10](#_Toc517216437)

[6.7对齐规则 10](#_Toc517216438)

[6.8单一功能规则 11](#_Toc517216439)

[6.9简单功能规则 11](#_Toc517216440)

[6.10选用FALSE规则 11](#_Toc517216441)

[6.11独立赋值规则 11](#_Toc517216442)

[6.12定义常量规则 11](#_Toc517216443)

[6.13模块化规则 11](#_Toc517216444)

[6.14交流规则 11](#_Toc517216445)

[7.编程准则 12](#_Toc517216446)

[7.1变量使用 12](#_Toc517216447)

[7.2数据库操作 12](#_Toc517216448)

[7.3对象使用 12](#_Toc517216449)

[7.4模块设计原则 12](#_Toc517216450)

[7.5结构化要求 12](#_Toc517216451)

[7.6函数返回值原则 13](#_Toc517216452)

[8.代码包规范 13](#_Toc517216453)

[9.代码的控制 13](#_Toc517216454)

[9.1代码库/目录的建立 13](#_Toc517216455)

[9.2代码归档 13](#_Toc517216456)

[10. 输入控制校验规则 14](#_Toc517216457)

[10.1登陆控制 14](#_Toc517216458)

[10.2数据录入控制 14](#_Toc517216459)

[11.数据库命名规范 14](#_Toc517216460)

# 1.目的

本规范为一套编写高效可靠的Java代码的标准、约定和指南。 它以安全可靠的软件工程原则为基础, 使代码易于理解、维护和增强，提高生产效率。同时，将带来更大的一致性，使软件开发团队的效率明显提高。

# 2.范围

本规范适用于开发组全体人员，作用于软件项目开发的代码编写阶段和后期维护阶段。

# 3.注释规范

## 3.1概述

a) 注释要求英文及英文的标点符号

b) 注释中，应标明对象的完整的名称及其用途，但应避免对代码过于详细的描述

c) 每行注释的最大长度为100个字符，为便于阅读代码，每行代码的长度应少于100个字符

d) 将注释与注释分隔符用一个空格分开

e) 不允许给注释加外框

f) 编码的同时书写注释

g) 重要变量必须有注释

h) 变量注释和变量在同一行，所有注释必须对齐。

i) 典型算法必须有注释

j) 在循环和逻辑分支地方的上行必须就近书写注释

k) 程序段或语句的注释在程序段或语句的上一行

l) 在代码交付之前，必须删掉临时的或无关的注释

## 3.2模块（类）注释

模块开始必须以以下形式书写模块注释：  
///<summary>  
///</summary>

## 3.3方法注释

在类的方法声明前必须以以下格式编写注释  
             /// <summary>  
             /// </summary>  
             /// <param name="<参数名称>"><参数说明></param>  
                 /// <returns>  
                 ///<对方法返回值的说明，该说明必须明确说明返回的值代表什么含义>  
             /// </returns>

## 3.4代码间注释

代码间注释分为单行注释和多行注释：  
//<单行注释>  
           /\*多行注释1  
             多行注释2  
             多行注释3\*/  
代码中遇到语句块时必须添加注释（if,for,foreach,……）,添加的注释必须能够说明此语句块的作用和实现手段（所用算法等等）。

# 4.命名总体规则

1. 名字应该能够标识事物的特性
2. 名字一律使用英文单词，而不能为拼音
3. 名字尽量不使用缩写，除非它是众所周知的
4. 名字可以有两个或三个单词组成，但不应多于三个，控制在3至30个字母以内
5. 在名字中，多个单词用大写第一个字母（其它字母小写）来分隔。例如：IsSuperUser
6. 名字尽量使用前缀而不是后缀
7. 名字中的单词尽量使用名词，如有动词，也尽量放在后面。例如：FunctionUserDelete（而不是FunctionDeleteUser）

# 5.命名规范

## 5.1变量（Variable）命名

a)    程序中变量名称 = 变量的前缀 +代表变量含意的英文单词或单词缩写。

鉴于大多数名称都是通过连接若干单词构造的，请使用大小写混合的格式以简化它们的阅读。每个单词的第一个字母都是大写  
1、即使对于可能仅出现在几个代码行中的生存期很短的变量，仍然使用有意义的名称。仅对于短循环索引使用单字母变量名，如 i 或 j

2、在变量名中使用互补对，如 min/max、begin/end 和 open/close

3、不要使用原义数字或原义字符串，如 For (i = 1;i <= 7;i++)。而是使用命名常数，如 For (i = 1;i <= NUM\_DAYS\_IN\_WEEK;i++) 以便于维护和理解。  
b)    控件命名  
控件命名 = 控件名称前二到三个字母 + 名称，如Labl控件（labUserName）

## 5.2常量命名

常量名也应当有一定的意义，格式为 NOUN 或 NOUN\_VERB。常量名均为大写，字之间用下划线分隔。  
例：  
        private const bool   WEB\_ENABLEPAGECACHE\_DEFAULT           = true;  
        private const int    WEB\_PAGECACHEEXPIRESINSECONDS\_DEFAULT = 3600;  
        private const bool   WEB\_ENABLESSL\_DEFAULT                 = false;  
注：  
变量名和常量名最多可以包含 255 个字符，但是，超过 25 到 30 个字符的名称比较笨拙。

## 5.3类（Class）命名

a) 名字应该能够标识事物的特性  
b)     名字尽量不使用缩写，除非它是众所周知的  
c)     名字可以有两个或三个单词组成，但通常不应多于三个  
d)     在名字中，所有单词第一个字母大写。例如  IsSuperUser，包含ID的，ID全部大写，如CustomerID  
e)     使用名词或名词短语命名类   
f)     少用缩写   
g) 不要使用下划线字符 (\_)

例： public class FileStream  
public class Button  
public class String

1. 类名往往用不同的后缀表达额外的意思

|  |  |  |
| --- | --- | --- |
| 后缀名 | 意义 | 举例 |
| Inter | 这个类是一个接口 | LifeCycleInter |
| Dao | 这个类封装了数据访问方法 | PaymentOrderDao |
| Action | 直接处理页面请求，管理页面逻辑了类 | UpdateOrderListAction |
| Bean | 这是一个POJO | MenuStateBean |

## 5.4接口（Interface）命名

和类命名规范相同，唯一区别是接口在名字前加上“i”前缀

例：interface IDBCommand;  
 interface IButton;

## 5.5方法（Method）命名

和类命名规范相同, 动词前缀往往表达特定的含义

|  |  |  |
| --- | --- | --- |
| 前缀名 | 意义 | 举例 |
| create | 创建 | createOrder() |
| delete | 删除 | deleteOrder() |
| add | 创建，暗示新创建的对象属于某个集合 | addPaidOrder() |
| Init或者initialize | 初始化，暗示会做些诸如获取资源等特殊动作 | initializeObjectPool |
| write | 写入 | writeUserName() |
| get | 获得 | getName() |
| modity | 修改 | modifyActualTotalAmount() |
| start | 开始 | startOrderProcessing() |
| find | 查找对象 | findNewSupplier() |
| update | 更新对象 | updateCommission() |

## 5.6名称空间Namespace）命名

和类命名规范相同

# 6.编码规则

## 6.1错误检查规则

a) 编程中要考虑函数的各种执行情况，尽可能处理所有流程情况。  
b)     检查所有的系统调用的错误信息，除非要忽略错误。  
c)     将函数分两类：一类为与屏幕的显示无关， 另一类与屏幕的显示有关。对于与屏幕显示无关的函数，函数通过返回值来报告错误。对于与屏幕显示有关的函数，函数要负责向用户发出警告，并进行错误处理。  
d)     错误处理代码一般放在函数末尾。  
e)     对于通用的错误处理，可建立通用的错误处理函数，处理常见的通用的错误。

## 6.2大括号规则

将大括号放置在关键词下方的同列处，例如：  
if ($condition)     while ($condition)  
{                  {  
      ...                   ...  
}                  }

## 6.3缩进规则

使用一个“Tab”为每层次缩进。例如：  
function func()  
   {  
      if (something bad)  
      {  
         if (another thing bad)  
         {  
            while (more input)  
            {  
            }  
         }  
      }  
   }

## 6.4小括号规则

a)     不要把小括号和关键词（if 、while等）紧贴在一起，要用空格隔开它们。  
b)     不要把小括号和函数名紧贴在一起。  
c)     除非必要，不要在Return返回语句中使用小括号。因为关键字不是函数，如果小括号紧贴着函数名和关键字，二者很容易被看成是一体的。

## 6.5 If Then Else规则

如果你有用到else if 语句的话，通常最好有一个else块以用于处理未处理到的其他情况。可以的话放一个记录信息注释在else处，即使在else没有任何的动作。其格式为：  
if (条件1)                 // 注释  
   {  
   }  
   else if (条件2)            // 注释  
   {  
   }  
   else                           // 注释  
   {  
   }  
注：if 和循环的嵌套最多允许4层

## 6.6 Case规则

default case总应该存在，如果不允许到达，则应该保证：若到达了就会触发一个错误。Case的选择条件最好使用int或string类型。

## 6.7对齐规则

变量的申明和初始化都应对齐。例如：  
   int       m\_iCount;  
   int       i,j;  
   float     m\_fIncome,m\_fPay;  
   
   m\_iCount  = 0;  
   i         = 1;  
   m\_fIncome = 0.3;

## 6.8单一功能规则

原则上，一个程序单元（函数、例程、方法）只完成一项功能

## 6.9简单功能规则

原则上，一个程序单元的代码应该限制在一页内（25~30行）

## 6.10选用FALSE规则

大部分函数在错误时返回FALSE、0或NO之类的值，但在正确时返回值就不定了（不能用一个固定的TRUE、1或YES来代表），因此检测一个布尔值时应该用  FALSE、0、NO之类的不等式来代替。例如：使用“if  ( FALSE !=  f( ) )” 而不用“if  (TRUE ==  f( ) )”

## 6.11独立赋值规则

嵌入式赋值不利于理解程序，同时可能回造成意想不到的副作用，应尽量编写独立的赋值语句。例如：使用“a = b + c ;  e = a + d;”而不用“e = ( a = b + c ) + d ”

## 6.12定义常量规则

对于代码中引用的常量（尤其是数字），应该define成一个大写的名字，在代码中引用名字而不直接引用值。

## 6.13模块化规则

某一功能，如果重复实现一遍以上，即应考虑模块化，将它写成通用函数。并向小组成员发布。同时要尽可能利用其它人的现成模块

## 6.14交流规则

共享别人的工作成果，向别人提供自己的工作成果。  
在具体任务开发中，如果有其它的编码规则，则在相应的软件开发计划中予   以明确定义

# 7.编程准则

## 7.1变量使用

a) 不允许随意定义全局变量。  
b) 一个变量只能有一个用途；变量的用途必须和变量的名称保持一致。  
c) 所有变量都必须在类和函数最前面定义，并分类排列

## 7.2数据库操作

a)      查找数据库表或视图时，只能取出确实需要的那些字段。  
b)      使用无关联子查询，而不要使用关联子查询。  
c)      清楚明白地使用列名，而不能使用列的序号。  
d)      用事务保证数据的完整性。

## 7.3对象使用

尽可能晚地创建对象，并且尽可能早地释放它

## 7.4模块设计原则

a)      不允许随意定义公用的函数和类。  
b)      函数功能单一，不允许一个函数实现两个及两个以上的功能。  
c)      不能在函数内部使用全局变量，如要使用全局变量，应转化为局部变量。  
d)      函数与函数之间只允许存在包含关系，而不允许存在交叉关系。即两者之间只存在单方向的调用与被调用，不存在双向的调用与被调用

## 7.5结构化要求

a)     禁止出现两条等价的支路。  
例如：if (a == 2)  
       else if (a== 3)  
             //  
       else if (a == 2)  
             //  
       else  
           //  
b)     避免使用GOTO语句  
c)     用 IF 语句来强调只执行两组语句中的一组。禁止 ELSE GOTO 和 ELSE RETURN。  
d)     用 CASE 实现多路分支  
e)     避免从循环引出多个出口。  
f)     函数只有一个出口。  
g)     不使用条件赋值语句。  
h)     避免不必要的分支。  
i)     不要轻易用条件分支去替换逻辑表达式

## 7.6函数返回值原则

避免使用结构体等复杂类型  
使用bool类型：该函数只需要获得成功或者失败的返回信息时候  
使用int类型：错误代码用负数表示，成功返回0

# 8.代码包规范

项目中，代码包的版本号由圆点隔开的两个数字组成，第一个数字表示发行号，第二个数字表示该版的修改号。具体用法如下：  
1)     当代码包初版时，版本号为 V1.00；  
2)     当代码包被局部修改或bug修正时，发行号不变，修改号第二个数字增1。例如，对初版代码包作了第一次修订，则版本号为 V1.01；  
3)     当代码包在原有的基础上增加部分功能，发行号不变，修改号第一个数字增1，例如，对V1.12版的基础上增加部分功能，则新版本号为 V1.20；  
4)     当代码包有重要修改或局部修订累积较多导致代码包发生全局变化时，发行号增1。例如，在 V1.15 版的基础上作了一次全面修改，则新版本号为 V2.00

# 9.代码的控制

## 9.1代码库/目录的建立

项目负责人在VSS中建立项目的文档库目录，即为“Software”目录，以便快速查询

## 9.2代码归档

所有代码在完成一个稳定的版本后，项目负责人都应打包后，存放于VSS中该目的“Software”目录下，并且依据代码包的命名规范为代码包分配一个唯一名称

# 10. 输入控制校验规则

## 10.1登陆控制

用户登陆ID和登陆密码，要限定输入长度范围，必须检查输入合法性

## 10.2数据录入控制

a)     TextBox输入  
1、  要保持用户输入和数据库接收的长度一致  
2、  必须进行输入合法性校验  
如：E\_mail格式   [XXX@XXX.XXX](mailto:XXX@XXX.XXX)...  
        电话格式     020-12345678     (020)12345678  
        邮政编码是六位  
b) 除CheckBox、RadioButton外，禁止在DataGrid内嵌入其他编辑控件，用以添加编辑数据

# 11.数据库命名规范

使用本系统遵循以下命名规范：   
1、表命名：用一个或三个以下英文单词组成，单词首字母大写，如：DepartmentUsers；   
2、表主键名称为：表名+ID，如Document表的主键名为：DocumentID   
3、存储过程命名：表名+方法,如：p\_my\_NewsAdd,p\_my\_NewsUpdate;   
4、视图命名：View\_表名，如：ViewNews;   
5、Status为表中状态的列名，默认值为0，在表中删除操作将会改变Status的值而不真实删除该记录；  
6、Checkintime为记录添加时间列，默认值为系统时间；  
7、表、存储过程、视图等对象的所有都为dbo，不要使用数据库用户名，这样会影响数据库用户的更改