**开发规范**

开发规范分成目录规范、项目和包名的命名规范、类，方法，变量和常量的命名规范这几种。

**1.目录规范**

目录规范——你在开发中整体文件夹组织结构。 我这里推荐一种文件夹的组织结构。

Requirement——需求文档文件夹

Design——设计文档文件夹

Test——集成测试，系统测试，测试报告，测试清单文件夹

Deployment——发布部署的文件夹

Src——源码文件夹

Help——帮助文档文件夹

**2.项目和报名命名规范**

对于项目和包名命名规范是：包名一律小写, 少用缩写和长名；

采用以下规则：

[基本包].[项目名].[模块名]

包名一般不要超过三级

不得将类直接定义在基本包下，所有项目中的类、接口等都应当定义在各自的项目和模块包中；

例如：

package com.routemap\_information.util;

这样子的规范，能够提高项目组织性，从而便于更好的协同开发。

**3.类，接口,方法，常量和变量命名规则**

**Ⅰ 类和接口的命名**

类或接口名是一个名词，采用大小写混合的方式，每个单词的首字母大写。尽量让类名简洁而富于描述。尽量使用完整单词，避免用缩写词(除非该缩写词被更广泛使用，像URL，HTML)。例如：

class Raster;

class ImageSprite;

interface RasterDelegate;

interface Storing;

命名采用单词组合取名，单词首字母为大写，单词之间可采用“\_”下划线进行区分，也可不采用。

根据定义类型首字母加以区分：

Interface：命名首字母加大写的“I”;

Abstract class：命名首字母加大写“A”;

Class：无需加

根据功能类型结尾加上功能描述字符串：

页面类：“Page”，例如“LoginPage”

处理类：“Handle”，例如“LogicHandle”

工厂实现类：“Impl”，例如“FactoryImpl”

动作事件定义类：“Action”，例如“LoginAction”

网络事件定义类：“Net”，例如“LoginNet”

数据定义类：“Data”，例如“LoginData”

消息处理类：“Msg”，例如“LoginRequestMsg”

资源管理类：“Manager”，例如“ImageManager”

缓存类：“Cache”，例如“UserCache”

参数传递类：“Param”，例如“LoginParam”

功能提供类：“Util”，例如“MathUtil”

数据输入输出类：“Steam”，例如“CacheOutStream”

类命名不能使用中文字符，尽量不要在命名字符串中出现“0-9”的数值描述和除下划线以外的其他字符描述，命名的字母组合尽量能够在本身的文字意义上初步了解类的大体功能。

**采用驼峰的方式，第一个单词的首字母小写，其后单词的首字母大写；**

**Ⅱ 变量命名方法**

变量名不应以下划线或美元符号开头；

尽量避免单个字符的变量名，除非是一次性的临时变量。临时变量通常被取名为i，j，k，m和n，它们一般用于整型；c，d，e，它们一般用于字符型；

对不易清楚识别出该变量类型的变量，应使用类型名或类型名缩写作其后缀，例如：

Thread animationThread;

String responseStr;

组件或部件变量使用其类型名或类型名缩写作其后缀，例如：

Command backCommand;

Image barImage；

TextField passwordField;

Player dogSoundPlayer;

集合类型变量，例如数组和矢量，应采用复数命名或使用表示该集合的名词做后缀：

Image[] images;

Vector requestQueue;

**Ⅲ 常量命名**

全部采用大写，单词间用下划线隔开：

static final int MIN\_WIDTH = 4;

static final int MAX\_WIDTH = 999;

static final int GET\_THE\_CPU = 1;

**Ⅳ 方法命名**

方法名是一个动词，采用大小写混合的方式，第一个单词的首字母小写，其后单词的首字母大写；取值类可使用get前缀，设值类可使用set前缀，判断类可使用is(has)前缀。

getName(); setSarry(); isLogon();

对于方法中尽量要加上适当的非空判断，与try catch 语句等等程序健壮性的判断。

**4.Android编码规范**

1.layout中的id命名

命名模式为：view缩写\_模块名称\_view的逻辑名称。view的缩写详情如下：

LayoutView：lv

RelativeView:rv

TextView:tv

ImageView:iv

ImageButton:im

Button:btn

2.activity中的view变量命名

命名模式为：逻辑名称+view缩写

建议：如果layout文件很复杂，建议将layout分成多个模块，每个模块定义一个moduleViewHolder，其成员变量包含所属view

3.strings.xml中的id命名

命名模式：activity名称\_功能模块名称\_逻辑名称/activity名称\_逻辑名称/common\_逻辑名称

strings.xml中，使用activity名称注释，将文件内容区分开来

4.drawable中的图片命名

命名模式：activity名称\_逻辑名称/common\_逻辑名称

5.styles.xml：将layout中不断重现的style提炼出通用的style通用组件，放到styles.xml中

6.服务端可以实现的，就不要放在客户端

7.注意图片的.9处理

8.做好单元测试（逻辑测试、界面测试）

9.不要重用父类的handler，对应一个类的handler也不应该让其子类用到，否则会导致message.what冲突

10.activity中在一个View.OnClickListener中处理所有的逻辑

11.strings.xml中使用%1$s实现字符串的通配

12.如果多个Activity中包含共同的UI处理，那么可以提炼一个CommonActivity，把通用部分交由它来处理，其他activity只要继承它即可

13.如果所开发的为通用组件，为避免冲突，将drawable/layout/menu/values目录下的文件名增加前缀

14.数据一定要效验，例如:

电话号码的数字校验、位数校验；

字符型转数字型，如果转换失败一定要有缺省值；

服务端响应数据是否有效判断；

15.有的按钮要避免重复点击

**5.注释**

原则——对已经不推荐使用的类和方法需要注明@Deprecated，并说明替代的类或者方法。

**Ⅰ 文件注释**

所有的源文件都应该在开头有一个注释，其中列出文件的版权声明、文件名、功能描述以及创建、修改记录：

/\*

\* Copyright (C) 2009-2014 ROUTEMAP Inc.All Rights Reserved.

\* FileName：KQQMIDlet.java

\* Description：简要描述本文件的内容

\* History：

\* 版本号 作者 日期 简要介绍相关操作

\* 1.0 wuhui 2009-01-21 Create

\* 1.1 wuhui 2009-01-30 Add Logo Screen

\*\*/

**Ⅱ 类或接口注释**

采用JavaDoc文档注释，在类、接口定义之前应当对其进行注释，包括类、接口的描述、最新修改者、版本号、参考链接等：

/\*\*

\* 描述

\* @author 作者（最新修改者）

\* @version 版本号（最新版本号）

\*/

class Window extends BaseWindow

{

...

}

注：JavaDoc文档注释：描述Java的类、接口、构造方法、方法、以及字段。每个文档注释都会被置于注释定界符/\*\*...\*/之中，一个注释对应一个类、接口或成员。该注释应位于声明之前。文档注释的第一行(/\*\*)不需缩进，随后的文档注释每行都缩进1格(使星号纵向对齐)。

**Ⅲ 方法注释**

采用JavaDoc文档注释，在方法定义之前当对其进行注释，包括方法的描述、输入、输出及返回值说明、抛出异常说明、参考链接等：

/\*\*

\* 描述

\* @param 参数说明：每个参数一行，注明其取值范围等

\* @return 返回值：注释出失败、错误、异常时的返回情况

\* @exception 异常：注释出什么条件下会引发什么样的异常

\*/

public char charAt(int index)

{

...

}

**Ⅳ 其它注释(非JavaDoc文档注释)**

1单行代码注释一律使用注释界定符"//"，例如：

// explain what this means

if(bar > 1)

{

……

}

int isShow = 0;// 是否显示

2.多行注释使用注释界定符"/\*...\*/"，例如：

/\*

\* Here is a block comment with

\* multiple lines for text comments.

\*/