**Android编码规范建议稿**

目录

[**一、** **Android Studio配置和工具** 2](#_Toc469493643)

[**1.** **code style** 2](#_Toc469493644)

[**2.** **encoding** 2](#_Toc469493645)

[**3.** **new line** 3](#_Toc469493646)

[**4.** **lint** 3](#_Toc469493647)

[**二、** **Java代码编码规范** 3](#_Toc469493648)

[**1.** **注释** 4](#_Toc469493649)

[**2.** **驼峰式命名** 4](#_Toc469493650)

[**3.** **Javadoc** 4](#_Toc469493651)

[**4.** **其他** 5](#_Toc469493652)

[**三、** **资源文件编码规范** 5](#_Toc469493653)

[**1.** **命名遵循前缀表明类型的习惯** 5](#_Toc469493654)

[**2.** **关闭标签/>单独起一行** 5](#_Toc469493655)

[**3.** **考虑使用Designtime attributes 设计时布局属性** 5](#_Toc469493656)

[**4.** **使用styles** 5](#_Toc469493657)

[**5.** **Colors.xml是一个调色板** 5](#_Toc469493658)

[**6.** **Strings.xml中字符串命名** 5](#_Toc469493659)

[**四、** **commit message编写规范** 5](#_Toc469493660)

[为什么要关注提交信息 6](#_Toc469493661)

[基本要求 6](#_Toc469493662)

[**五、** **总结** 6](#_Toc469493663)

[参考资料 6](#_Toc469493664)

1. **Android Studio配置和工具**
2. **code style**

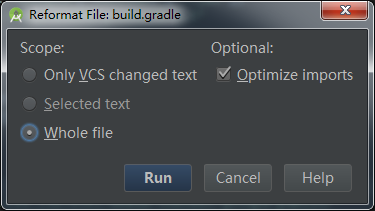
一般常用的code style有如下两种：

a) square android code style: <https://github.com/square/java-code-styles>

b) google java code style: <https://github.com/google/styleguide/blob/gh-pages/intellij-java-google-style.xml> (个人建议)

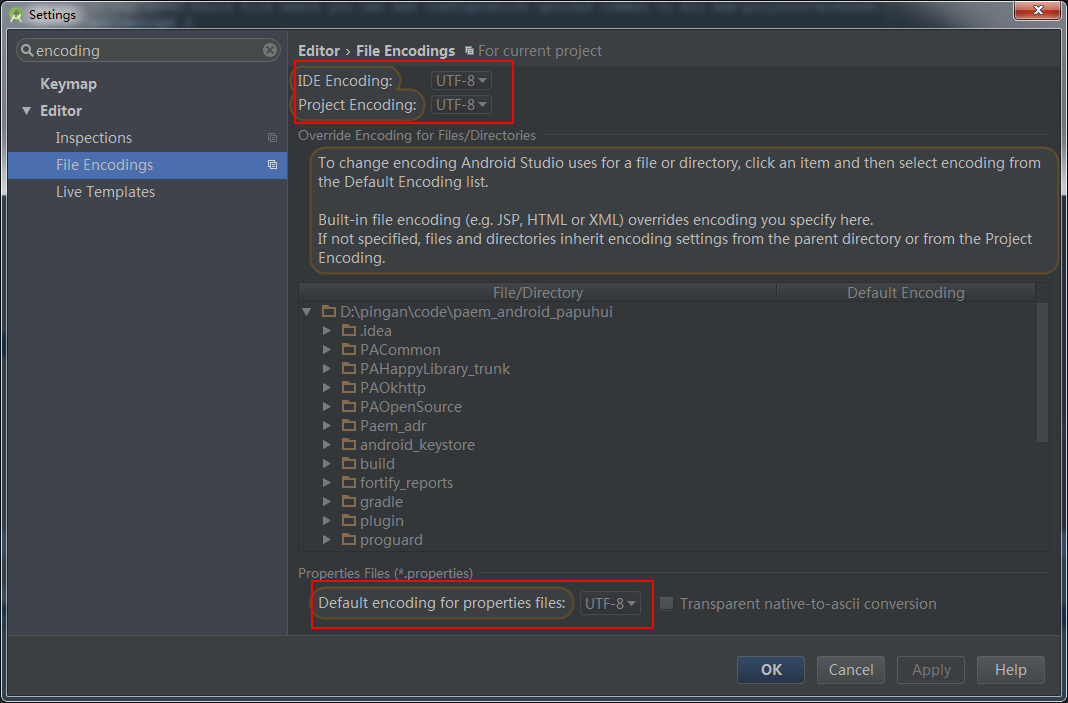
**在每次提交代码前需要对Version Control – Local Changes 中的文件进行format，解决warnnings**

format设置如下:



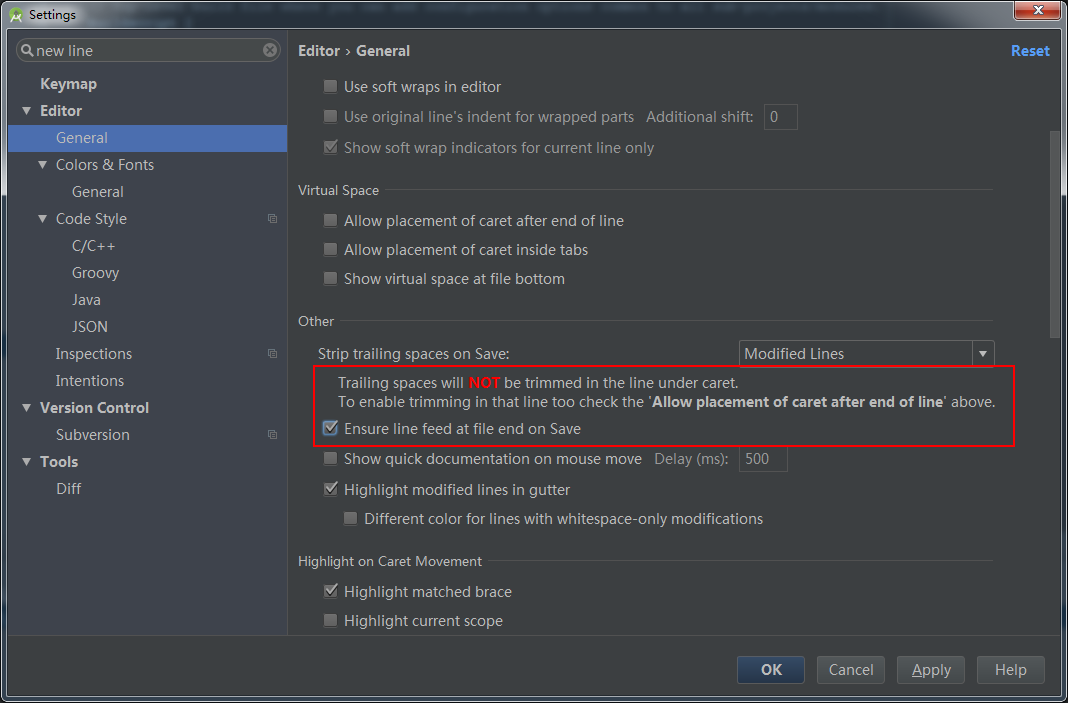
1. **encoding**

在Android Studio的配置中统一设置编码为utf-8(一般windows上的需要修改)



1. **new line**

确保文件最后一行有一个空换行(有一些工具会对最后一行的new line 进行检查)



1. **lint**

经常使用lint检查项目代码，对不符合规范的地方进行修改，如warnning, 未使用的资源，重复的资源，可能存在的内存泄露，对性能有影响的代码段等；详细内容可自行查阅相关资料，这里不做展开。

1. **Java代码编码规范**

Java编码规范可参考[google java style](https://google.github.io/styleguide/javaguide.html), [中文链接](http://www.hawstein.com/posts/google-java-style.html)

1. **注释**

块注释与其周围的代码在同一缩进级别。它们可以是/\* ... \*/风格，也可以是// ...风格(注意：//后要空一格)。对于多行的/\* ... \*/注释，后续行必须从\*开始， 并且与前一行的\*对齐。

1. **驼峰式命名**

驼峰式命名发分为大驼峰式命名法(UpperCamelCase)和小驼峰式命名法(lowerCamelCase)。

转换方案，名字从prose form开始:

1. 把短语转换为纯ASCII码，并且移除任何单引号。例如：”Mueller’s algorithm”将变成”Muellers algorithm”。
2. 把这个结果切分成单词，在空格或其它标点符号(通常是连字符)处分割开。

* 推荐：如果某个单词已经有了常用的驼峰表示形式，按它的组成将它分割开(如”AdWords”将分割成”ad words”)。 需要注意的是”iOS”并不是一个真正的驼峰表示形式，因此该推荐对它并不适用。

1. 现在将所有字母都小写(包括缩写)，然后将单词的第一个字母大写：

* 每一个单词的第一个字母都大写，来得到大驼峰式命名。
* 除了第一个单词，每个单词的第一个字母都大写，来得到小驼峰式命名。

1. 最后将所有的单词连接起来得到一个标识符。

示例：

Prose form Correct Incorrect

---------------------------------------------------------------------------------------------

"XML HTTP request" xmlHttpRequest XMLHTTPRequest

"new customer ID" newCustomerId newCustomerID

"inner stopwatch" innerStopwatch innerStopWatch

"supports IPv6 on iOS?" supportsIpv6OnIos supportsIPv6OnIOS

"YouTube importer" youTubeImporter

**类名：**使用大驼峰命名，以类型做后缀，如BaseActivity, StringUtils

**方法名：**使用小驼峰命名，通常是动词或动词短语，如isEmpty(), getXXX(), initXXX()

**常量名：**全部字母大写，以下划线分割单词,如

**非常量字段名：**小驼峰命名，以前缀表明类型，如:btnLogin

**scope：范围**（对于范围不强制要求，因为一般IDE中都有颜色区分）

非公有，非静态字段命名以m开头。

静态字段命名以s开头。

公有非静态字段命名以p开头。

公有静态字段（全局变量）命名以g开头。

public static final 字段(常量) 全部大写，并用下划线连起来。

1. **Javadoc**

至少在每个public类及它的每个public和protected成员处使用Javadoc，以下是一些例外：1.不言自明的方法，2.重载

1. **其他**

* 字段名的定义位置或顺序最要有一定的逻辑性或相关性
* Java类中有一定生命周期或调用顺序的方法最好按序排列
* 尽量面向接口编程，减少模块间的依赖关系，降低耦合

1. **资源文件编码规范**

资源文件及其中的字段命名全部使用小写字母加下划线分割的方式命名

1. **命名遵循前缀表明类型的习惯**

类型\_模块\_功能, 如:activity\_home, item\_city

1. **关闭标签/>单独起一行**

有助于调整和添加新的属性

1. **考虑使用[Designtime attributes 设计时布局属性](http://tools.android.com/tips/layout-designtime-attributes)**

Android Studio已经提供支持，而不是硬编码android:text等

1. **使用styles**

Style的命名可参考Android系统源码中的命名方式，应用中需要一些通用的style，并由此子类化一些style。

作为一个经验法则,android:layout\_\*\*\*\*属性应该在 layout XML 中定义,同时其它属性android:\*\*\*\* 应放在 styler XML中。此规则也有例外，不过大体工作 的很好。这个思想整体是保持layout属性(positioning, margin, sizing) 和content属性在布局文件中，同时将所有的外观细节属性（colors, padding, font）放 在style文件中。

1. **Colors.xml是一个调色板**

在colors.xml中应该只是映射颜色的名称一个ARGB值，而没有其他的，不要使用它为不同的业务定义ARGB值，这样能避免出现重复的ARGB值定义，并且方便修改或重构，会使一共使用了多少种颜色变得非常清晰，如果需要配合业务抽象一些colors，最好放到styles中一起定义

**像对待colors.xml一样对待dimes.xml**

1. **Strings.xml中字符串命名**

最好根据内容进行直译，并按字母顺序排序，这样可以避免很多的重复字符串，减少包的体积，并且在做国际化时会方便很多。宁外对于不同的功能模块可以以前缀开头，如title\_\*\*\*, head\_\*\*\*, toast\_\*\*\*, 尽量减少带业务模块的前缀

1. **commit message编写规范**

## 为什么要关注提交信息

* 加快Reviewing Code的过程
* 帮助我们写好Release Note
* 几年之后帮你快速想起来某branch,tag或者commit增加了什么功能，改变了什么代码
* 让其他开发者在运行git blame的时候想跪谢

## 基本要求

* 第一行标题字数在50个字符内，不要用句号结束标题行
* 在标题使用祈使语气，即动词开头，如Fix issue #832; Add … feature; Change …;
* 用一个空行隔开标题和正文
* 正文在72个字符处换行
* 使用正文解释是什么和为什么，而不是如何做

1. **总结**

对于以上的Android编程规范只是列出了一个大纲，根据个人工作的一样经验和网上的一些资料，并结合当前的项目代码情况给出的一个最简化的建议，更详细的资料可阅读参考资料。相信在今后的code review中相互学习指正很快能尽量统一编码风格，提高代码质量。

最后总结一下，最重要的三个基础：格式化代码，命名规范，注释。

**重要的事情：**

**在每次提交代码前需要对Version Control – Local Changes 中的文件进行format，尽量解决能解决的warnning**

# 参考资料

* Google Java style: <https://google.github.io/styleguide/javaguide.html>
* Code Style for Contributors：<https://source.android.com/source/code-style.html#java-language-rules>
* Android编码规范：<http://www.jianshu.com/p/0a984f999592>
* Best practices in Android development: <https://github.com/futurice/android-best-practices>
* Git 写出好的 commit message: <https://ruby-china.org/topics/15737>
* Information in commit messages: <https://wiki.openstack.org/wiki/GitCommitMessages#Information_in_commit_messages>