Android 开发规范以及注意事项

修订历史记录

日期	版本	说明	作者
2014-10-20	1.0	初始建立	曾繁添
2015-05-19	V1.1	基本完善各个章节内容	曾繁添
2015-08-12	V1.2	更新代码混淆注意事项、color 命名	曾繁添

目录

1. 简介

本文档用于指导开发人员在安卓项目开发过程中类名、资源文件名、变量名等开发约定以及命名规范，方便工程的后期维护，提高代码整体质量、可读性。

1.1 目的

统一开发人员代码编写命名规范，提高代码可读性、以及专业程度，方便后期维护管理

1.2 范围

适用于安卓项目开发领域范畴

2. 命名原则

命名尽量简洁、见名思意，禁止出现 a b c 此类低俗、无意义的弱智命名。代码编写规则风格要保持一致

3. 开发规范

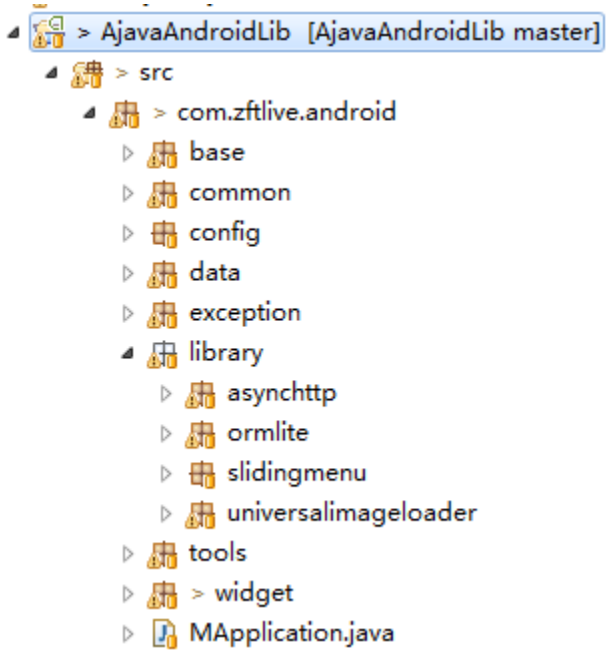
3.1 工程名

工程的命名需要精简、有代表性、符合 Java 命名规范，让别人一看到名称就大概知道该工程是做什么的。不能以特殊符号、下划线、空格、数字、中文开头。名称要见名思意、组成单词首字母大写、或者全部小写。例如：ApiDemos、zftlive

3.2 包名

包命名一概不允许出现大写字母，虽然大写字母不违反 java 标准命名规范，但是任何一个开源框架基本没有出现大写、下划线、特殊符号的包名，必须全部小写、将具有共性、特殊职责处理的类归纳到一个包下，包名称使用具有代表意义的小写英文单词组成或者单词的简称组成。例如：com.zftlive.base com.zftlive.tools com.zftlive.common

例如：



3.3 类文件

类名称必须首字母大写、杜绝出现汉字，完全参照 Java 命名规范

UI 界面对应的类必须加入相应类型的后缀：XxxxActivity、XxxxFragment，XxxxDialog 方便阅读者快速了解实现的 UI 布局

3.4 类属性

类的属性命名参照 Java 命名规范，以小写字母开头，每个连接单词首字母大写、禁止出现随意命名 aa bb _ab123 此类的属性名称。

Boolean 类型的属性推荐 is 或者 has 作为前缀。

3.5 成员变量

类的属性命名参照谷歌推荐写法，以 m 开头+对应的机能名称+实力对象名称（例如：mTopicListview、mListAdapter）；

控件属性命名以控件前缀+功能名称组成（例如：tv_project_name、rl_root_view）

3.6 方法名

方法命名禁止以大写字母开头（.NET 代码例外）、方法名必须具由该方法处理相关业务代表性的动词组成，例如：initView、doBusiness、validateForm 等

3.7 布局文件- layout

1、 自定义控件的布局文件命名→view_控件名称
2、 共通性的、基础的布局分别以 common_、base_作为前缀
3、 Activity、Fragment、Dialog、Popupwindow 的界面布局文件命名必须加上对应的前缀。
4、 activity_功能模块名称、fragment_功能模块名称、dialog_功能模块名称、popup_功能模块名称
为表达清楚功能模块代表含义，多个单词之间以下划线_连接

命名格式:
[view_业务模块简称_功能名称.xml]
[common_业务模块简称_功能名称.xml]
[activity_业务模块简称_功能名称.xml]
[fragment_业务模块简称_功能名称.xml]
[dialog_业务模块简称_功能名称.xml]

示例：
view_pull_refresh_header_horizontal.xml
common_title_bar.xml
activity_zc_project_choose.xml
fragment_v2_main_live_head.xml
dialog_cancel_ok.xml

3.8 资源文件- drawable

→Drawable：存放.9、selector、shape、layer-list、rotate、bitmap 等 xml 写的图片资源文件

命名格式:
[selector_业务模块简称_功能名称.xml]
[shape_业务模块简称_功能名称.xml]
[layer_list_业务模块简称_功能名称.xml]
[rote_业务模块简称_功能名称.xml]

示例：
selector_view_peoject_topic_btn.xml （自定义控件、共通组件的业务模块简称起名要具有代表性）
selector_zc_peoject_topic_btn.xml
shape_zc_guess_like_item.xml

→图片素材切片文件,适配对应的机型放置对应的文件夹下面，特殊机型分辨率，单独适配
~~drawable-ldpi：存放低分辨率的手机素材，基本可以抛弃~~
drawable-mdpi 320*480 中等密度设备素材（1.0）
drawable-hdpi 480*800 分辨率密度代表的设备素材（1.5）
drawable-xhdpi 720*1280 分辨率密度代表的设备素材（2.0）
drawable-xxhdpi 1080*1920 分辨率密度代表的设备素材 （3.0）

命名格式（基本素材都切成 png 格式）：

[业务模块简称前缀_业务功能名称_颜色区分_状态(n/p).png],自定义控件/共通组件以 view 为前缀，

示例：
view_progress_bar_bg.png
zc_project_title_fav_white_n.png
zc_project_title_fav_white_p.png
zc_project_topic_bg.9.png

3.9 动画文件- anmi

自定义控件、共通组件、基类、第三方开源控件涉及的动画、业务模块相关动画要以前缀进行区分，方便以后移植功能模块代码。命名单词之间以下划线_连接

命名格式：
自定义控件/开源控件动画格式：[view_控件名称.xml]
基类涉及相关的动画格式：[base_基类相关命名_功能名称.xml]
业务模块相关的动画格式：[业务模块前缀简称_功能名称.xml]

示例：
view_pull_refresh_slide_in_from_bottom.xml
base_activity_right_in.xml
zc_project_list_item_fade_in.xml

3.10 配置文件- values

所有 values 配置文件必须存在 values 缺省文件夹中，其他适配配置文件按照标准流程走即可(例如：values-800x480、values-960x540、values-1920x1080、values-1280x720、values-sw600dp 等)

→styles
缺省样式、业务模块样式、共通组件样式分别抽取不同的 style 书写

命名格式：
[styles_功能名称.xml]

示例：styles_views.xml / styles_sample.xml / styles.xml

→ strings
缺省字符串、业务模块字符串、共通组件字符串分别抽取不同的 string 书写

命名格式：
[strings_功能名称.xml]
示例：strings_views.xml / strings_sample.xml / strings.xml

→ ids
缺省 id、业务模块 id、共通组件 id 分别抽取不同的 ids 书写

命名格式：
[ids_功能名称.xml]
示例：ids_views.xml / ids_sample.xml / ids.xml

→ dimens
缺省单位、业务模块单位、共通组件单位分别抽取不同的 dimens 书写

命名格式:
[dimens_功能名称.xml]
示例：dimens_views.xml / dimens_sample.xml / dimens.xml

→ colors
缺省颜色、业务模块颜色、共通组件颜色分别抽取不同的 colors 书写

命名格式:
[colors_功能名称.xml]
示例：colors_views.xml / colors_sample.xml / colors.xml
注意：书写每一个颜色的时候，name 命名最好带上颜色值，方便区分，例如：blue_359df5 之类的命名

【<color name="titile_999999">#999999</color>】

→ attrs
缺省属性配置、共通组件属性配置分别抽取不同的 attrs 书写

命名格式:
[attrs_功能名称.xml]
示例：attrs_views.xml / attrs.xml

→ arrays
缺省数组配置、共通组件数组配置、业务需要的数组配置文件分别抽取不同的 arrays 书写

命名格式:
[arrays_功能名称.xml]
示例：arrays_sample.xml

3.11 代码混淆

※SDK 目录自带的混淆配置文件 sample 路径（高版本 ADT，低版本的配置文件是 cfg）
${sdk.dir}/tools/proguard/proguard-android.txt:proguard-project.txt

上架市场之前，商业项目考虑到代码安全防止被人反编译篡改，代码混淆是最基本的防线，其次可以使用第三方的安全加固解决方案，比如爱加密、梆梆安全等。针对代码混淆注意事项，哪些必须保留，哪些可以混淆，总结如下：

1、 Android 四大组件、基本的数据存储不混淆

```
#一些基本的类不进行混淆
-keep public class * extends android.app.Activity
-keep public class * extends android.app.SherlockActivity
-keep public class * extends android.app.Fragment
-keep public class * extends android.support.v4.app.Fragment
-keep public class * extends android.app.Application
-keep public class * extends android.app.Service
-keep public class * extends android.content.BroadcastReceiver
-keep public class * extends android.content.ContentProvider
-keep public class * extends android.app.backup.BackupAgentHelper
-keep public class * extends android.preference.Preference
```

2、 Application、R 文件必须保留

```
#Application不进行混淆、R文件不能混淆
-keep public class * 你的APP包名.JRAuthApplication
-keep public class 你的APP包名.R
-keep class 你的APP包名.R$* {*;}
-keepclassmembers class 你的APP包名.R$* {
    public static <fields>;
}
```

3、 注解、本地 native 方法、枚举、Parcelable、自定义控件、android-support-vX 等必须保留，这个 sdk 目录自带的混淆配置文件就已经自带了

```
-keepattributes *Annotation*
-keep public class com.google.vending.licensing.ILicensingService
-keep public class com.android.vending.licensing.ILicensingService

# For native methods, see
http://proguard.sourceforge.net/manual/examples.html#native
-keepclasseswithmembernames class * {
    native <methods>;
}

# keep setters in Views so that animations can still work.
# see http://proguard.sourceforge.net/manual/examples.html#beans
-keepclassmembers public class * extends android.view.View {
   void set*(***);
   *** get*();
}

# We want to keep methods in Activity that could be used in the XML
attribute onClick
-keepclassmembers class * extends android.app.Activity {
   public void *(android.view.View);
}

# For enumeration classes, see
http://proguard.sourceforge.net/manual/examples.html#enumerations
-keepclassmembers enum * {
    public static **[] values();
    public static ** valueOf(java.lang.String);
}

-keep class * implements android.os.Parcelable {
  public static final android.os.Parcelable$Creator *;
}
```

4、 如果使用了 GSON 之类的实体映射，实体类必须保留，不然无法正常映射数据，使用类继承关系配置即可

```
####################谷歌GSON（开始）#####################
-keepattributes Signature
# For using GSON @Expose annotation
-keepattributes *Annotation*
# Gson specific classes
-keep class sun.misc.Unsafe { *; }
-keep class com.google.gson.stream.** { *; }
# 所有实体类不混淆
-keep public class * extends com.jdjr.library.base.JRBaseBean
####################谷歌 GSON（结束）#####################
```

5、 第三方 jar 或者开源项目不混淆。第三方 jar 有些已经混淆过了，如果再次混淆肯定会出问题，一般情况不混淆，或者按照官方的混淆配置配上。至于开源项目混不混淆都无所谓，可以使用--libraryjars、dontwarn、keep 配合保留

```
####################第三方jar-保留（开始）#####################
-libraryjars libs/zftlive-test.jar

-dontwarn com.zftlive.**
-keep class com.zftlive.** {*;}

####################第三方 jar-保留（结束）#####################
```

6、 单例类的构造方法不能混淆，否则实例化对象会出问题，可以直接指定保留哪个类

3.12 AndroidManifest.xml

1、每一个模块注册的 Activity 加入相应的(开始-结束)注释块、空行，方便后续查找维护

```
<!-- 下拉ScrollView背景回弹效果样例(开始) -->
<!-- 下拉头部背景图片放大界面 -->
<activity
    android:name=".sample.scrollview.PulldownViewActivity"
    android:configChanges="keyboardHidden|orientation|screenSize"
    android:label="@string/PulldownViewActivity"
    android:screenOrientation="portrait"
    android:windowSoftInputMode="stateAlwaysHidden|adjustResize" >
    <intent-filter>
        <action android:name="android.intent.action.MAIN" />

        <category android:name="com.zftlive.android.SAMPLE_CODE" />
    </intent-filter>
</activity>
<!-- 下拉Scrollview回弹效果样例界面 -->
<activity
    android:name=".sample.scrollview.StretchViewActivity"
    android:configChanges="keyboardHidden|orientation|screenSize"
    android:label="@string/PulldownViewActivity"
    android:screenOrientation="portrait"
    android:windowSoftInputMode="stateAlwaysHidden|adjustResize" >
    <intent-filter>
        <action android:name="android.intent.action.MAIN" />

        <category android:name="com.zftlive.android.SAMPLE_CODE" />
    </intent-filter>
</activity>
<!-- 下拉ScrollView背景回弹效果样例(结束) -->


<!-- FadingActionBar官方DEMO -->
<activity
    android:name=".sample.fadingactionbar.HomeActivity"
    android:configChanges="keyboardHidden|orientation|screenSize"
    android:label="@string/FadingActionBar"
    android:screenOrientation="portrait"
    android:windowSoftInputMode="stateAlwaysHidden|adjustResize" >
    <intent-filter>
```

2、注册 activity 的 name 能简写推荐使用简写，因为 APP 的 package 是固定的，所有的 activity 均在该 package 下面
3、activity 几个比较重要的属性要正确合理的配置，如：android:configChanges、android:screenOrientation、android:windowSoftInputMode

4. 内存泄露

A、 Context 静态引用导致无法释放内存问题

```
private static Context instance;

public static Stack<Activity> activitys = new Stack<Activity>();
```

如果你的代码中存在类似上述代码，那么存储的 Context 将无法释放对象，导致内存的泄露，建议使用软引用替代

```
/***寄存整个应用Activity**/
private final Stack<WeakReference<Activity>> activitys = new Stack<WeakReference<Activity>>();
```

如果非要使用全局的 Context，使用 Application 类型的 Context

5. 注意事项

A、避免 context 相关的内存泄露，记住以下几点：

1. 不要让生命周期长的对象引用 activity context，即保证引用 activity 的对象要与 activity 本身生命周期是一样的
2. 对于生命周期长的对象，可以使用 application context （继承类：public class Mpplication extends Application）
3. 尽量使用静态类（全局），避免非静态的内部类，避免生命周期问题，注意内部类对外部对象引用导致的生命周期变化

B、代码格式化问题

布局文件、代码文件一定要保持统一的风格，方便大家阅读查看。每次修改完布局文件/代码文件，记得去除无引用的 import、格式化代码、保存代码，然后再 commit 代码（ctrl+shift+f / ctrl+shift+o / ctrl + s ）

B、 Context 的使用问题

Application 类型的 Context 与 Activity 类型的 Context 是两种不同类型的 Conetext，使用 Activity 类型的 Context 请确保传入引用的地方生命周期与当前 Activity 的生命周期同生同灭。比如：在一个 Activity 创建一个 Dialog 传入的 Context 可以使用 Activity 类型的 Context，因为 Dialog 与当前 Activity 同生同灭，如果不是则用全局的 Application 替代

C、 Fragment 需要保持 inflate 的视图问题

如果需要全局缓存 Fragment 的视图，注意 onCreateView 与 onDestoryView 的代码写法。如果在 onDestoryView 中没有将全局缓存的 View 移除掉，肯定会抛异常，基类回调 onCreateView 的方法时，会将返回的 View 再次 add 到当前界面容器中，由于全局缓存了当前界面渲染的视图 View，同一个 View 重复 add 到一个容器中会发生什么事情，相信做过安卓开发的人都知道，在这里就不详细说明了，具体原因见安卓源代码

```
@Override
public View onCreateView(LayoutInflater inflater, ViewGroup container,
        Bundle savedInstanceState) {
    Log.d(TAG, "BaseFragment-->onCreateView()");
    // 渲染视图View
    if (null == mContextView) {
        mContextView = inflater.inflate(bindLayout(), container, false);
        // 控件初始化
        initView(mContextView);
        // 实例化共通操作
        mBaseOperation = new Operation(getActivity());
        // 业务处理
        doBusiness(getActivity());
    }

    return mContextView;
}
```

```
@Override
public void onDestroyView() {
    super.onDestroyView();
    if (mContextView != null && mContextView.getParent() != null) {
        ((ViewGroup) mContextView.getParent()).removeView(mContextView);
    }
}
```

另外，在当前 Fragment 最好将依附的 Activity 做一个全局的 Context 缓存，防止调用 getActivity 时发生空指针的异常，如果有全局的引用则可以防止依附的 Activity 回收

D、多个类型 type 的 Listview

Listview 使用多个 type 类型的 Item 的时候，在 Adapter 中 getItemViewType 返回的 int 类型一定要从 0 开始的索引连续自然整数，不能跳跃，因为 Listview 回收 View 的时候需要该至当做索引获取缓存在 ArrayList 的 view，具体自行查看 Listview 源码

6. 常见错误

这个需要慢慢积累，后续更新

7. 参考资料

JDK 下载地址：
http://www.oracle.com/technetwork/java/javase/downloads/index.html

JDK 环境变量配置
http://www.cnblogs.com/nicholas_f/articles/1494073.html

Eclipse 下载地址：
http://www.eclipse.org/kepler/

ADT 下载地址整理：
http://blog.csdn.net/xqf222/article/details/9821971
http://www.apkbus.com/android-115125-1-1.html

Android 程序打包及签名
http://www.cnblogs.com/timeng/archive/2012/02/17/2355513.html

android 利用数字证书对程序签名
http://blog.csdn.net/qianfu111/article/details/9113887

Android App 的签名打包（晋级篇）
http://blog.csdn.net/linghu_java/article/details/6701666

8. 备注

如果您有什么好的建议或者平时开发好的习惯欢迎提出你的 IDEA。如果上述文字有什么错误的描述欢迎指正！如果你对安卓开发感兴趣，可以关注作者的开源项目 http://git.oschina.net/zftlive/zftlive

关于作者：

小名：曾繁添
网站：http://www.zftlive.com
博客：http://www.cnblogs.com/fly100/
邮箱：zftlive@163.com
QQ：1260128980