Android 开发规范与应用(二)

开发规范

资源文件 Resources
命名 遵循前缀表明类型的习惯，形如type_foo_bar.xml。例如：
fragment_contact_details.xml,
view_primary_button.xml,
activity_main.xml.
组织布局文件 若果你不确定如何排版一个布局文件，遵循一下规则可能会有帮助。

每一个属性一行，缩进4个空格
android:id 总是作为第一个属性
android:layout_**** 属性在上边
style 属性在底部
关闭标签/>单独起一行，有助于调整和添加新的属性
考虑使用Designtime attributes 设计时布局属性，Android Studio已经提供支持，而不是硬编码android:text

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    >

    <TextView
        android:id="@+id/name"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_alignParentRight="true"
        android:text="@string/name"
        style="@style/FancyText"
        />

    <include layout="@layout/reusable_part" />

</LinearLayout>
```

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。

例外有以下这些:

android:id 明显应该在layout文件中
layout文件中android:orientation对于一个LinearLayout布局通常更有意义
android:text 由于是定义内容，应该放在layout文件中
有时候将android:layout_width 和 android:layout_height属性放到一个style中作为一个通用的风格中更有意义，但是默认情况下这些应该放到layout文件中。

- **使用styles** 几乎每个项目都需要适当的使用style文件，因为对于一个视图来说有一个重复的外观是很常见的。
  在应用中对于大多数文本内容，最起码你应该有一个通用的style文件，例如：

```
<style name="ContentText">
    <item name="android:textSize">@dimen/font_normal</item>
    <item name="android:textColor">@color/basic_black</item>
</style>
```

应用到TextView 中:

```
<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="@string/price"
    style="@style/ContentText"
    />
```

你或许需要为按钮控件做同样的事情，不要停止在那里。将一组相关的和重复android:**的属性放到一个通用的style中。

将一个大的style文件分割成多个文件 你可以有多个styles.xml 文件。Android SDK支持其它文件，styles这个文件名称并没有作用，起作用的是在文件里xml的 style标签。因此你可以有多个style文件styles.xml,style_home.xml,style_item_details.xml,styles_forms.xml。
不用于资源文件路径需要为系统构建起的有意义，在res/values目录下的文件可以任意命名。

colors.xml是一个调色板 在你的colors.xml文件中应该只是映射颜色的名称一个RGBA值，而没有其它的。不要使用它为不同的按钮来定义RGBA值。

不要这样做

```
<resources>
    <color name="button_foreground">#FFFFFF</color>
    <color name="button_background">#2A91BD</color>
    <color name="comment_background_inactive">#5F5F5F</color>
    <color name="comment_background_active">#939393</color>
    <color name="comment_foreground">#FFFFFF</color>
    <color name="comment_foreground_important">#FF9D2F</color>
    <color name="comment_shadow">#323232</color>
```

使用这种格式，你会非常容易的开始重复定义RGBA值，这使如果需要改变基本色变的很复杂。同时，这些定义是跟一些环境关联起来的，如button或者comment,
应该放到一个按钮风格中，而不是在color.xml文件中。

相反，这样做:

```
<resources>

    <!-- grayscale -->
    <color name="white"     >#FFFFFF</color>
    <color name="gray_light">#DBDBDB</color>
    <color name="gray"      >#939393</color>
    <color name="gray_dark" >#5F5F5F</color>
    <color name="black"     >#323232</color>

    <!-- basic colors -->
    <color name="green">#27D34D</color>
    <color name="blue">#2A91BD</color>
    <color name="orange">#FF9D2F</color>
    <color name="red">#FF432F</color>

</resources>
```

向应用设计者那里要这个调色板，名称不需要跟”green”, “blue”, 等等相同。
“brand_primary”, “brand_secondary”, “brand_negative” 这样的名字也是完全可以接受的。
像这样规范的颜色很容易修改或重构，会使应用一共使用了多少种不同的颜色变得非常清晰。
通常一个具有审美价值的UI来说，减少使用颜色的种类是非常重要的。

像对待colors.xml一样对待dimens.xml文件 与定义颜色调色板一样，你同时也应该定义一个空隙间隔和字体大小的“调色板”。
一个好的例子，如下所示：

```
<resources>

    <!-- font sizes -->
    <dimen name="font_larger">22sp</dimen>
    <dimen name="font_large">18sp</dimen>
    <dimen name="font_normal">15sp</dimen>
    <dimen name="font_small">12sp</dimen>

    <!-- typical spacing between two views -->
    <dimen name="spacing_huge">40dp</dimen>
    <dimen name="spacing_large">24dp</dimen>
    <dimen name="spacing_normal">14dp</dimen>
    <dimen name="spacing_small">10dp</dimen>
    <dimen name="spacing_tiny">4dp</dimen>

    <!-- typical sizes of views -->
    <dimen name="button_height_tall">60dp</dimen>
    <dimen name="button_height_normal">40dp</dimen>
    <dimen name="button_height_short">32dp</dimen>

</resources>
```

布局时在写 margins 和 paddings 时，你应该使用spacing_****尺寸格式来布局，而不是像对待String字符串一样直接写值。
这样写会非常有感觉，会使组织和改变风格或布局是非常容易。

避免深层次的视图结构 有时候为了摆放一个视图，你可能尝试添加另一个LinearLayout。你可能使用这种方法解决：

```
<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    >

    <RelativeLayout
        ...
        >

        <LinearLayout
            ...
            >

            <LinearLayout
                ...
                >

                <LinearLayout
                    ...
                    >
                </LinearLayout>

            </LinearLayout>

        </LinearLayout>

    </RelativeLayout>

</LinearLayout>
```

即使你没有非常明确的在一个layout布局文件中这样使用，如果你在Java文件中从一个view inflate（这个inflate翻译不过去，大家理解就行） 到其他views当中，也是可能会发生的。

可能会导致一系列的问题。你可能会遇到性能问题，因为处理起需要处理一个复杂的UI树结构。
还可能会导致以下更严重的问题StackOverflowError.

因此尽量保持你的视图tree：学习如何使用RelativeLayout,
如何 optimize 你的布局 和如何使用
标签.

小心关于WebViews的问题. 如果你必须显示一个web视图，
比如说对于一个新闻文章，避免做客户端处理HTML的工作，
最好让后端工程师协助，让他返回一个 “纯” HTML。
WebViews 也能导致内存泄露
当保持引他们的Activity，而不是被绑定到ApplicationContext中的时候。
当使用简单的文字或按钮时，避免使用WebView，这时使用TextView或Buttons更好。

测试框架
Android SDK的测试框架还处于初级阶段，特别是关于UI测试方面。Android Gradle
目前实现了一个叫connectedAndroidTest的测试，
它使用一个JUnit 为Android提供的扩展插件 extension of JUnit with helpers for Android.可以跑你生成的JUnit测试，

- **只当做单元测试时使用 Robolectric ，views 不用**
  它是一个最求提供”不连接设备的”为了加速开发的测试，
  非常时候做 models 和 view models 的单元测试。
  然而，使用Robolectric测试时不精确的，也不完全对UI测试。
  当你对有关动画的UI元素、对话框等，测试时会有问题，
  这主要是因为你是在 “在黑暗中工作”（在没有可控的界面情况下测试）

- **Robotium 使写UI测试非常简单。** 对于UI测试你不需 Robotium 跑与设备连接的测试。
  但它可能会对你有益，是因为它有许多来帮助类的获得和分析视图，控制屏幕。
  测试用例看起来像这样简单：

```
solo.sendKey(Solo.MENU);
solo.clickOnText("More"); // searches for the first occurence of
"More" and clicks on it
solo.clickOnText("Preferences");
solo.clickOnText("Edit File Extensions");
Assert.assertTrue(solo.searchText("rtf"));
```

模拟器
如果你全职开发Android App,那么买一个Genymotion emulatorlicense吧。
Genymotion 模拟器运行更快的秒帧的速度，比起典型的AVD模拟器。他有演示你APP的工具，高质量的模拟网络连接，GPS位置，等等。它同时还有理想的连接测试。
你若涉及适配使用很多不同的设备，买一个Genymotion 版权是比你买很多真设备便宜多的。

注意：Genymotion模拟器没有装载所有的Google服务，如Google Play Store和Maps。你也可能需
要测试Samsung指定的API，若这样的话你还是需要购买一个真实的Samsung设备。

混淆配置
ProGuard 是一个在Android项目中广泛使用的压缩和混淆打包的源码的工具。

你是否使用ProGuard取决你项目的配置，当你构建一个release版本的apk时，通常你应该配置gradle文件。

```
buildTypes {
    debug {
        minifyEnabled false
    }
    release {
        signingConfig signingConfigs.release
        minifyEnabled true
        proguardFiles 'proguard-rules.pro'
    }
}
```

为了决定哪些代码应该被保留，哪些代码应该被混淆，你不得不指定一个或多个实体类在你的代码中。
这些实体应该是指定的类包含main方法，applets，midlets，activities，等等。
Android framework 使用一个默认的配置文件，可以在
SDK_HOME/tools/proguard/proguard-android.txt
目录下找到。自定义的工程指定的 project-specific 混淆规则，如在my-project/app/proguard-rules.pro中定义，
会被添加到默认的配置中。

关于 ProGuard 一个普遍的问题，是看应用程序是否崩溃并报ClassNotFoundException 或者 NoSuchFieldException 或类似的异常，
即使编译是没有警告并运行成功。
这意味着以下两种可能：

1.ProGuard 已经移除了类，枚举，方法，成员变量或注解，考虑是否是必要的。
2.ProGuard 混淆了类，枚举，成员变量的名称，但是这些名字又被拿原始名称使用了，比如通过Java的反射。
检查app/build/outputs/proguard/release/usage.txt文件看有问题的对象是否被移除了。
检查 app/build/outputs/proguard/release/mapping.txt 文件看有问题的对象是否被混淆了。

In order to prevent ProGuard from stripping away needed classes or class members, add a keep options to your proguard config:
以防 ProGuard 剥离 需要的类和类成员，添加一个 keep选项在你的 proguard 配置文件中：

```
-keep class com.futurice.project.MyClass { *; }
```

防止 ProGuard 混淆 一些类和成员，添加 keepnames:

```
-keepnames class com.futurice.project.MyClass { *; }
```

- **在构建项目之初，发布一个版本** 来检查ProGuard规则是否正确的保持了重要的部分。
  同时无论何时你添加了新的类库，做一个发布版本，同时apk在设备上跑起来测试一下。
  不要等到你的app要发布 “1.0”版本了才做版本发布，那时候你可能会碰到好多意想不到的异常，需要一些时间去修复他们。

- **Tips**每次发布新版本都要写 mapping.txt。每发布一个版本，如果用户遇到一个bug，同时提交了一个混淆过的堆栈跟踪。
  通过保留mapping.txt文件，来确定你可以调试的问题。

- **DexGuard** 若果你需要核心工具来优化，和专门混淆的发布代码，考虑使用DexGuard,一个商业软件，ProGuard 也是有他们团队开发的。它会很容易将Dex文件分割，来解决65K个方法限制问题。

附录（北京试行）

- **Android编码规范**
  1. java代码中不出现中文，最多注释中可以出现中文.xml代码中注释
  2. 成员变量，局部变量、静态成员变量命名、常量(宏)命名
     1). 成员变量: activity中的成员变量以m开头，后面的单词首字母大写（如Button mBackButton; String mName）；实体类和自定义View的成员变量可以不以m开头（如ImageView imageView， String name），
     2）. 局部变量命名：只能包含字母，组合变量单词首字母出第一个外，都为大写，其他字母都为小写
     3）. 常量(宏)命名: 只能包含字母和，字母全部大写，单词之间用隔开UMENG_APP_KEY
  3. Application命名
     项目名称+App， 如SlimApp,里面可以存放全局变量，但是杜绝存放过大的实体对象
  4. activity和其中的view变量命名
     activity命名模式为：逻辑名称+Activity
     view命名模式为：逻辑名称+View
     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
  5. layout及其id命名规则
     layout命名模式：activity_逻辑名称，或者把对应的activity的名字用“_”把单词分开。
     命名模式为：view缩写_模块名称_view的逻辑名称, 用单词首字母进行缩写
     view的缩写详情如下
     LayoutView：lv
     RelativeView:rv

TextView:tv
ImageView:iv
ImageButton:ib
Button:btn

6. strings.xml中的
   1）. id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
   2). strings.xml中使用%1$s实现字符串的通配,合起来写
7. drawable中的图片命名
   命名模式：activity名称逻辑名称/*common*逻辑名称/ic_逻辑名称
   (逻辑名称: 这是一个什么样的图片，展示功能是什么)
8. 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；
9. 使用layer-list和selector，主要是View onCclick onTouch等事件界面反映
10. 切图需求
    1). 多图片组合的，尽量分拆成多个可重用的图片，避免一个图片太大.
    2). 尽量使用.9图片，自动适配适配界面。
    11.服务端可以实现的，就不要放在客户端
    1). 图片过大时，服务端压缩图片后返回来可减少很多问题出现
    2). 及时更新的数据，尽管本地有缓存
11. 引用第三方库要慎重，避免应用大容量的第三方库，导致客户端包非常大，或者其他非本应用的信息会出现，影响体验。
12. 处理应用全局异常和错误，将错误以邮件的形式发送给服务端
13. 使用静态变量方式实现界面间共享要慎重
14. Log(TAG，详细描述), 加开关，打包时关掉log，提高运行速度。如
    if (SlimConf.DEBUG) {
    Log.d(TAG, “plan_init_enabled= ” + plan_init_enabled);
    }
15. 单元测试（逻辑测试、界面测试），避免次生问题随着解决问题增加。
16. 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msg.what冲突，方法是声明父类handler为private
17. activity中的Listener
    Activity只用一个View.OnClickListener， View.OnTouchListener等Listener中处理所有控件的逻辑，即，XXActivity impliments OnClickListener方式实现接口，绑定周期，减少接口实例数量
    注意：尽量不要给每个点击控件设置一个View.OnClickListener实例
18. 如果多个Activity中包含共同的UI处理，那么可以提炼一个AbsXXActivity.java或者XXActivityBase.java，可以是一个抽象类，把通用部分叫由它来处理，其他activity只要继承它即可，可以是一个抽象activity，子类定义具体需求的函数（重载或者重写）
19. 使用button+activitgroup实现tab效果时，使用Button.setSelected(true)，确保按钮处于选择状态，并使activitygroup的当前activity与该button对应
20. 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
22.数据一定要效验，例如
字符型转数字型，如果转换失败一定要有缺省值；if（object instanceof User）服务端响应数据是否有效判断；
23.有的按钮要避免重复点击，避免重复发送几个http请求或耗时任务等。
方法：点击之后setEnable（false），一段时间之后setEnable（true），

21. 复杂循环体超过8行时，需要在外部加注释；
22. 方法，函数名字尽量体现其功能含义（函数名能完全体现功能是可以不要注释；函数中的注释一定按照缩进格式注释。函数内部单行注释写在该行上面）；
23. 函数行数尽量不能超过100（if，for语句中的行数不能超过50，超过时，提出来作为函数使用（Switch除外））。
24. 同一个类文件中不能出现雷同的code段，相同的部分提出来作为函数使用，或者使用函数重载，用参数的数量和类型来控制其功能区别。

- **Android性能优化**
  1. http可以用gzip压缩，设置连接超时时间和响应超时时间
     http请求按照业务需求，分为是否可以缓存和不可缓存，那么在无网络的环境中，仍然通过缓存的httpresponse浏览部分数据，实现离线阅读。可以存在数据库，下次继续使用
  2. listview 性能优化
     1). Adapter 的getView中复用convertView
     在getView中，判断convertView是否为空，如果不为空，可复用。如果couvertview中的view需要添加listerner，代码一定要在if(convertView==null){}之外。不要在里面轻易地new对象，其数量随着item数量增加
     2). 异步加载图片
     item中如果包含有webimage，那么最好异步加载
     3). 快速滑动时不显示图片
     当快速滑动列表时（SCROLL_STATE_FLING），item中的图片或获取需要消耗资源的view，可以不显示出来；而处于其他两种状态
     （SCROLL_STATE_IDLE 和SCROLL_STATE_TOUCH_SCROLL），则将那些view显示出来
     4). list中异步加载的图片，当不在可视范围内，按照一定的算法及时回收（如在当前可视范围的上下10条item以外的图片进行回收，或者将图片进行缓存，设置一个大小，按照最近最少使用原则超过部分进行回收）
     5). BaseAdapter避免内存溢出
     如果BaseAdapter的实体类有属性非常消耗内存，可以将保存到文件；为提高性能，可以进行缓存，并限制缓存大小，退出界面后释放资源。
  3. 使用线程池，分为核心线程池和普通线程池，下载图片等耗时任务放置在普通线程池，避免耗时任务阻塞线程池后，导致所有异步任务都必须等待
  4. 异步任务，分为核心任务和普通任务，只有核心任务中出现的系统级错误才会报错，异步任务的ui操作需要判断原activity是否处于激活状态，否则，跳出run函数
     1). 主线程不要进行网络处理；

2). 主线程不要进行数据库处理；
3). 主线程不要进行文件处理；
new Thread 处理完成后， send message to Handler notify UI thread to update UI

5. 尽量避免static成员变量引用资源耗费过多的实例,比如Context,多以引用的的形式出现。
1 ）不要在project Application的子类中使用过多或者过大的实体变量
2 ）需要使用较大的全局变量时，使用单例instance来实现，保持项目中只有一份实例
6 .使用WeakReference或SoftReference代替强引用，弱引用可以让您保持对对象的引用，同时允许GC在必要时释放对象，回收内存。对于那些创建便宜但耗费大量内存的对象，即希望保持该对象，又要在应用程序需要时使用，同时希望GC必要时回收时，可以考虑使用弱引用。如：

```
A a = new A();
a.str = "Hello, reference";
WeakReference<A> weak = new WeakReference<A>(a);
```

1. 占内存的Bitmap
及时的销毁(Activity的onDestroy时将bitmap回收，在被UI组件使用后马上进行回收会抛RuntimeException: Canvas: trying to use a recycled bitmap android.graphics.Bitmap)
设置一定的采样率(有开发者提供的图片无需进行采样，对于有用户上传或第三方的大小不可控图片，可进行采样减少图片所占的内存)，从服务端返回图片，建议同时反馈图片的size
巧妙的运用软引用
drawable对应resid的资源，bitmap对应其他资源
任何类型的图片，如果获取不到（例如文件不存在，或者读取文件时跑OutOfMemory异常），应该有对应的默认图片（默认图片放在在apk中，通过resid获取）；
2. Drawable目录下的图片
ui组件需要用到的图片是apk包自带的drawable-目录下的图片，那么一律用setImageResource或者setBackgroundResource，而不要根据resourceid，无形中构造了对象
注意：get(getResources(), R.drawable.btn_achievement_normal)该方法通过resid转换为drawable，需要考虑回收的问题，如果drawable是对象私有对象，在对象销毁前是肯定不会释放内存的。
3. 保证Cursor 占用的内存被及时的释放掉，而不是等待GC来处理。并且 Android明显是倾向于编 程者手动的将Cursor close掉（但是不要翻来覆去的打开和关闭数据库，耗时）
4. 线程也是造成内存泄露的一个重要的源头。线程产生内存泄露的主要原因在于线程 生命周期的不可控，设置开关boolean flag，退出界面，或者应用是跳出线程。避免莫名其妙的更新UI，造成错误
5. 如果ImageView的图片是来自网络，进行异步加载，注意，尽量缓存，减少请求次数

6. 应用开发中自定义View的时候，交互部分，千万不要写成线程不断刷新界面显示（如onResume()函数中刷新），而是根据TouchListener事件主动触发界面的更新
7. 复用、回收Activity对象
   临时的activity及时手动finish
   主界面设置为singleTask
   一般界面设置为singleTop
   14.位置信息
   获取用户的地理位置信息时，在需要获取数据的时候打开GPS，之后及时关闭掉
   15.在onResume时设置该界面的电源管理，在onPause时取消设置
   1. 模块之间的交互使用接口来定义，上下层次关系，数据层与控制层之间外部调用只关注接口函数名和参数，内部实现不考虑，外人调用出现问题时，负责人修改内部实现，调用者不要轻易修改，免得引发次生问题，（仅限于以上关系，其他另外考虑）

8. 工具类的函数名要根据功能来起名，尽量加上注释，方便以后调用，修改。
   1）类似功能的要用重载，用参数来控制功能区别
   2) 同一个文件（甚至项目中）不要出现相同的代码段，可以提出来当函数公用.

```
/**
     * download image bitmap from net, and add to cache ,files
     * @param context
     * @param url
     * @param handler
     * @param isLoadPrompt
     * @param isNoNetPrompt
     */
public void downloadNetBitmap(final Context context, final String
url, final Handler handler, boolean isLoadPrompt, boolean
isNoNetPrompt) {
   ...
}
```

AndroidUI优化

1. layout组件化，尽量使用merge及include复用，复杂布局使用RelativeLayout，避免其他手机运行后看不见空间的情况。
2. 使用styles，复用样式定义，降低xml的臃肿，风格统一，修改时一劳永逸。
3. 软键盘的弹出控制，不要让其覆盖输入框
4. 数字、字母和汉字混排占位问题：将数字和字母全角化。由于现在大多数情况下我们的输入都是半角，所以 字母和数字的占位无法确定，但是一旦全角化之后，数字、字母的占位就和一个汉字的占位相同了，这样就可以避免由于占位导致的排版问题。
5. 英文文档排版：textview自动换行时要保持单词的完整性，解决方案是计算字符串长度，然后手动设定每一行显示多少个字母并加上‘\n‘
6. textvie单行需求时，设置单行滚动走马灯样式
   7.自适应屏幕，使用dp或dip替代px，文字用sp

8.使用android:layout_weight或者TableLayout制作等分布局
9.使用animation-list制作动画效果，能带来不一样的效果。但是，动画会占用资源，逻辑复杂时尽量不用

打包，版本更新

同一个客户端如果要放在不同的市场，而且要统计各个市场下载及使用数据时针对不同的客户端打不同的包，唯一的区别是slimcoach_211_Official_Umeng.apk文件名为slimcoach_211_Official_Umeng.apk
在升级时,自己服务器情况，要将自己的versionCode和versionName一并传给服务端，如果需要升级，则下载versionName相对应的apk，其他服务器，及时更新apk
关于是否要强制升级：
1). 不管何种情况都强制升级
2). 判断用户的版本和当前最新版本，如果兼容则强制升级，否则可选；
3). 检查渠道号，appkey及其他要求