Android 兼容篇

工程师 李洪江

1. mipmap 和 drawable 的区别

mipmap是android 4.2（17）支持的纹理映射技术。google 建议大家只把 app 的启动图标放在 mipmap 目录中，其他图片资源仍然放在 drawable 下面。

1. fragment技术

Fragment是Android3.0（11）后引入的一个新的API，他出现的初衷是为了适应大屏幕的平板电脑， 当然现在他仍然是平板APP UI设计的宠儿，而且我们普通手机开发也会加入这个Fragment， 我们可以把他看成一个小型的Activity，又称Activity片段！想想，如果一个很大的界面，我们 就一个布局，写起界面来会有多麻烦，而且如果组件多的话是管理起来也很麻烦！而使用Fragment 我们可以把屏幕划分成几块，然后进行分组，进行一个模块化的管理！从而可以更加方便的在 运行过程中动态地更新Activity的用户界面！另外Fragment并不能单独使用，他需要嵌套在Activity 中使用，尽管他拥有自己的生命周期，但是还是会受到宿主Activity的生命周期的影响，比如Activity 被destory销毁了，他也会跟着销毁！

1. RTL支持

要实现RTL（从右到左）的布局镜面反射，仅仅需要遵循下列步骤就可以做到：

1. 在你的应用程序声明文件（manifest）里声明开启RTL mirroring的支持。具体做法是：在manifest.xml声明文件的<application>元素中，添加 android:supportsRtl=”true”

2. 修改应用程序中所有的“left/right”布局属性，改为对应的”start/end”布局

1）如果你的应用程序是针对Android 4.2目标平台（应用的targetSdkVersion或者minSdkVersion是17或者更高), 那么你就应当用“start”和“end”替换原来的“left”和“right”。例如，android:paddingLeft应当被替换为android:paddingStart。

2) 如果你想让你的应用程序与Android 4.2之前的版本保持兼容（也就是与targetSdkVersion或者minSdkVersion为16或者更早的版本），那么你应当既加上“start”和“end”，又加上“left”和“right”。例如，你应当同时写上：adnroid:paddingLeft和android:paddingStart。

1. Android 6.0 （APi23）新增了权限控制。

<http://www.jianshu.com/p/0273443e0d52>

1 Android 6.0 （APi23）新增了权限控制，但是如果targetSdkVersion ＝ 21, 我们在代码中不需要权限检查，因为targetSdkVersion=21很明确的说了只兼容到Android5.0，app可以正常打开运行。在23的设备不需要动态申请。

2 如果目标是23，使用新的动态权限检查机制。低于23 的设备不配置一样没权限。

1. sdk理解

实际开发中一般把这个值设置为android:compileSdkVersion一样,这样看来我们开发的app兼容范围就是：minSdkVersion至targetSdkVersion， 那么这三种配置理想情况应该是：

minSdkVersion(最低兼容版本) <= targetSdkVersion== compileSdkVersion (最新SDK)

1. APK Signature Scheme V2 是在Android 7.0 中引入的新的Android APK签名机制。

先给出一个关于Scheme V2的总结：

1. 用来验证APK可靠性的加密签名现在位于压缩文件的Central Directory之前。

2. 新的签名是基于整个APK文件的二进制内容生成的。与基于压缩包中每个解压文件的内容的V1版本不同。

3. 一个APK可以同时使用V1，V2签名，可以避免兼容性问题。

为什么要引入新的签名机制呢？

首先，新的签名机制有更高的安全性和可扩展性。===对比===

其次，新签名机制不需要解压缩后对每个文件内容做判断，性能更高，能提高应用验证速度，缩短安装时间。

使用Scheme V2需要注意的事情：

因为V2 与 V1不同，是基于整个APK的二进制文件内容做签名。所以，不能在签名后再对APK做一些操作。 比如V1 签名后，可以对APK做 ZipAlign操作，可以通过添加空文件的形式做渠道标记。 使用V2 签名后不能在签名后做类似操作。

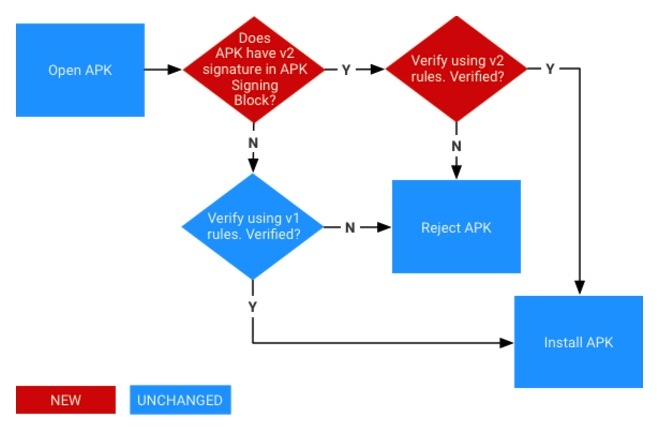
如果你要手动做Align和Sign，那么要使用Android SDK新提供的apksigner，它支持V1和V2签名。 需要注意，之前的签名工具jarsigner不支持V2签名。

如果你要禁用V1或者V2签名，可以在build.gradle 的signingConfig部分设置：

v1SigningEnabled false

v2SigningEnabled false

在Android Gradle Plugin 2.2上，默认两种签名都开启，优先使用V2。



在 Android 7.0 以上版本的设备上，APK 可以根据Full Apk Signature（v2 方案） 或者 JAR-signed（ v1方案）进行验证；  
而对于7.0以下版本的设备其会忽略 v2 版本的签名，只验证 v1 签名。

7.0之上非强制V2，必须要V1.

1. android 中view的setTranslationX(float translationX)方法在API 11及之后才能使用。
2. 为了提高私有文件的安全性，面向 Android 7.0 或更高版本的应用私有目录被限制访问。

参考： <http://m.blog.csdn.net/DJY1992/article/details/72533310>

FileProvider，是Android 7.0新增的一个类，该类位于v4包下的android.support.v4.content.FileProvider，使用方法和ContentProvider类似，操作步骤如下：

1. 在资源文件夹res/xml下新建file\_provider.xml文件，文件声明权限请求的路径。
2. AndroidManifest.xml 添加组件 provider 相关信息，类似组件 activity ，指定resource属性引用上一步创建的xml文件（后面会详细介绍各个属性的用法）。
3. 代码上做动态权限申请，使用getUriForFile()和grantUriPermission()。