一、App 资源编译打包

创建ResoureTable 资源表结构

Pacakge: 描述资源包

一个应用程序至少有两个资源包, 一个是 app 自身的资源

另一个是引用的 framwork-res.apk 中的资源

AaptAssets: 描述资源文件夹

资源的编译

收集 AaptAssets 中资源文件夹的信息到 ResoureTable 的 Package 中

编译 Xml 文件为二进制 Xml 文件

写入 Xml 头信息

写入 Xml 中采集到的字符串

写入 Xml 中采集到的 ID 值

把所有的字符串替换成采集后的索引值

资源的打包

遍历 Package 中的资源项, 为其分配 ID

资源项 ID = PacakgeId + TypeId + 资源序号

如 0x7f 01 0001

生成资源索引表 resources.arsc

收集数据

生成 Package 数据块

写入 resources.arsc

生成其他文件

编译 AndroidManifest 文件为二进制 Xml

将 resources.arsc 的 ID 导出, 生成 R.java 文件

打包为 APK 文件

二、资源管理器

构建 Application 的 ContextImpl 中通过 ResourcesManager 获取 Resources 的步骤如下：

构建 ResourcesKey

根据 ResourcesKey 从缓存 mResourceImpls 中获取 ResourcesImpl 对象

缓存不存在则构建 ResourcesImpl 对象, 创建流程如下

获取 AssetManager

构建 ApkAsset

创建 Native 的 ApkAsset, 解析 apk 中的resources.arsc 数据到内存

保存 resources.arsc 中的资源项值的字符串资源池 StringBlock

构建 AssetManager

合并 framework-res.apk 的系统资源 ApkAsset 集合

构建 Native 层的 AssetManager2

将合并后的资源集合保存在 AssetManager2 中

保存资源集合

重新构建引用表

重新构建缓存集合

刷新缓存

创建 ResourcesImpl 对象

内部持有 AssetManager

优先从 mResourceReferences 缓存中获取 Resources 对象

不存在则创建新的 Resources 对象

将 ClassLoader 和 ResourcesImpl 注入

当 Application 创建完成之后, 会通过 rewriteRValues 来重新复写 R 文件中的 ID 为运行时 ID。

1. 资源的查找

查找资源

从资源表 resource.arsc 中查找资源

String 类型的资源需要到 Java 的 StringBlock 中二次查找

从 ApkAsset 的 StringBlock 字符串资源池获取资源路径

优先从 Java 缓存池中获取

若缓存不存在, 则构建缓存

到 Native 层根据 idx 查找数据

投放到 Java 缓存池

构建解析器

有了资源的路径, 从 base.apk 压缩包中将我们要查找的资源提取出来

AssetManager 打开资源文件

获取 XML 文件资源的 Asset 对象

从 .apk 压缩包中查找我们的资源文件的数据项 ZipEntry

从 .apk 中将我们的所需的文件资源提取出来

对于压缩文件

通过 FileMap 将资源文件 mmap 到内存

通过 Asset::createFromCompressedMap 创建 Asset 对象

对于未压缩文件

通过 FileMap 将资源文件 mmap 到内存

通过 Asset::createFromUncompressedMap 创建 Asset 对象

将 Asset 数据注入 ResXMLTree

保存二进制 XML 数据到 mOwnedData

获取 XML 的 String 池

获取 XML 的 ID 池

将 ResXMLTree 保存到 XmlBlock 中

构建 XmlResourceParser

1. AAPT/AAPT2

Android 为我们提供了 R 这个类，指定了一个资源的索引（id），然后我们只需要告诉系统在不同的业务场景下，使用对应的资源就好了，至于具体是指定资源里面的哪一个具体文件，由系统根据开发者的配置决定。

在这种场景下，假设我们给定的 id 是 x 值，那么当下业务需要使用这个资源的时候，手机的状态就是 y 值，有了(x,y)，在一个表里面就能迅速的定位到资源文件的具体路径了。这个表就是 resources.arsc，它是从 aapt 编译出来的。

AssetManager 在我们调用 R 相关的 id 的时候，就会在这个表里面找到对应的文件，读取出来。

aapt2 主要分两步，一步叫 compile，一步叫 link。