Android_libstagefright框架分析

Last edited by **caoquanli** 1 month ago

Android_libstagefright框架分析

本文主要以MediaScanner作为着入点对stagefright 框架进行分析，这部分以MediaPlayer作为着入点更好一点，但是这部分我没有做过，不是很清楚。

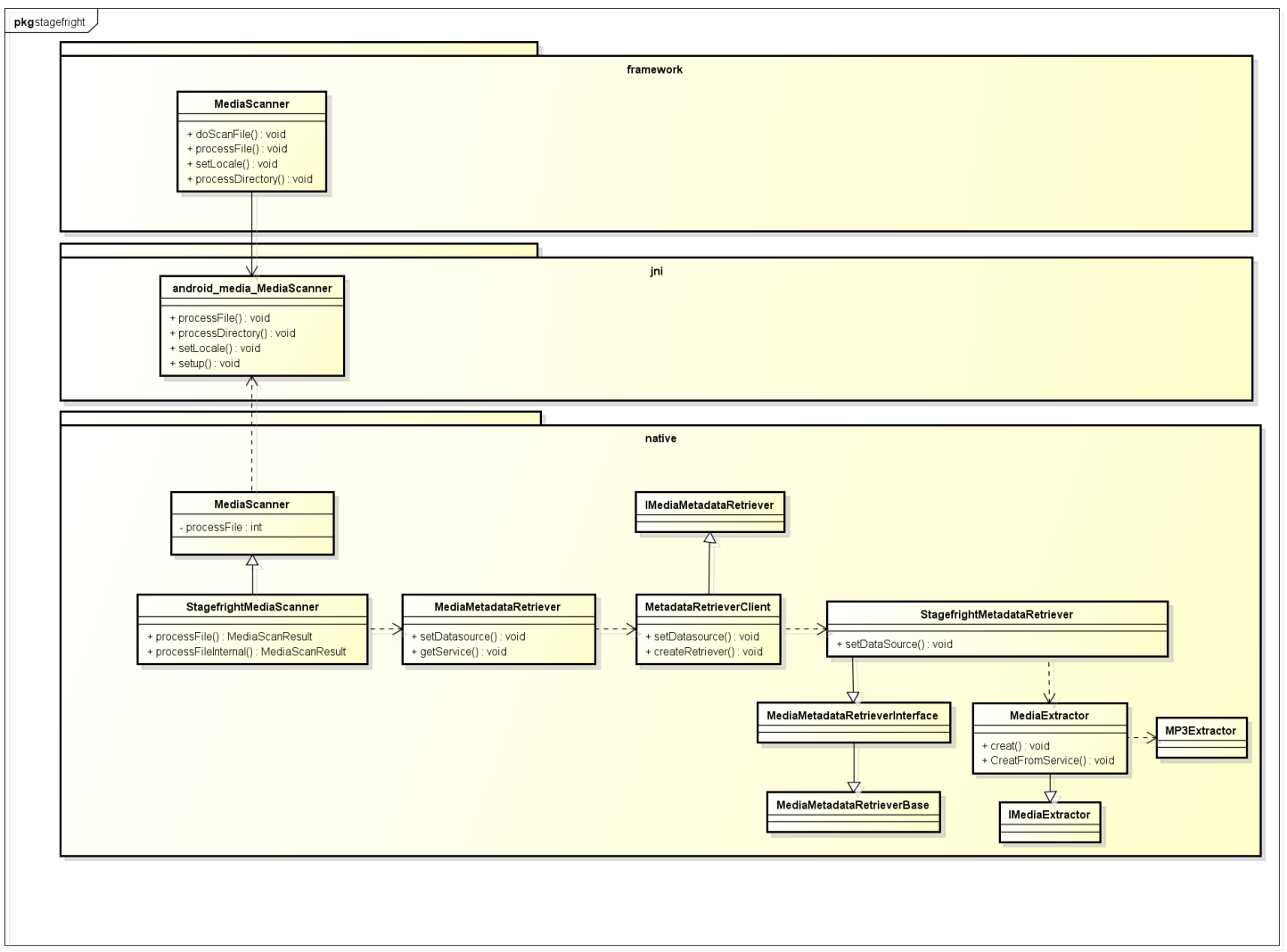
一 libstagefright框架

简单来说，这个框架就是为了实现音视频解码的一套服务流程，我这里是以MediaScanner作为切入点的，在Scan末尾，对于音视频文件，最后会有一次处理过程，这个过程，主要是获取文件的一些信息，比如artist，album，track等等，而stagefrigth就是为了这些服务的，虽然我这脸提取的信息较少，但是MediaPlayer是要获得很多的，不管怎么说，都是通过这个服务框架或得得，下面我就说说，获取信息得流程。

二 MediaScanner获取信息流程

作为切入点，我们直接看MediaScanner.java文件，这里面有一个doScanFile方法，它呢调用了native层一个方法，processFile，在一开始，MediaScanner.java他就加载了一个JNI库。所以呢这个文件的一个主要功能就是与native层进行通信。下面就从这部分开始。

框架类图



1. android_media_MediaScanner.cpp

这里为啥直接列举这个文件的，这个文件是native与java层交互的门户，后面的一系列操作都是从这个文件开始展开的。这里就不细说，JNI是怎么实现的了，直接从函数说起，上面说的java层的processFile这个方法，我们能够找到它本文件中的对应方法，android_media_MediaScanner_processFile()，为啥是这个方法,看下面：

{ "processFile", "(Ljava/lang/String;Ljava/lang/String;Landroid/media/MediaScannerClient;)V", (void *)android_media_MediaScanner_processFile }

和JNI中的方法有关，有兴趣可以自己去了解一下JNI，回到这个方法，看看它到底干了啥，其实也没干啥，就是把自己的事，让StagefrightMediaScanner去干了。现在就真正进入到了分析这个框架的时候了。

2. StagefrightMediaScanner.cpp

在这里不细说，这个对象的具体功能，只研究这部分的框架结构。可以去他的.h文件去看看，其实这个方法是MediaScanner的一个子类，直接去看processFile这个方法，这个也就调用了一下processFileInternal，这个方法做的事情就很多了，关键还是new了一个MediaMetadataRetriever对象，mRetriever，下面就涉及到了一个伴随这个框架

一身的一个方法，setDataSource，这个方法就是连接下面几个对象的枢纽，每次用的都是这个方法。这里有好几个这个方法，具体调用哪个看参数。

3. MediaMetadataRetriever.cpp

看看这个对象的setDataSource，构造函数有兴趣自己看看，这里面好几个重载函数，具体这里直接说明，就是setDataSource(int fd, int64_t offset, int64_t length)，这个方法，这里也有一个mRetriever，这个玩意又是啥呢，去构造函数一看便知，它是一个服务，这个服务从哪里来呢，这是通过binder底层传进来的，其实这地方就是MediaPlayerService，这个服务最终是会被ServiceManager管理，让这个函数去服务，但是呢我这里只是一个简单的扫描请求，看看IMediaMetadataRetriever.cpp

4. IMediaMetadataRetriever.cpp

这里就和binder相关了，我们这个不管这个，只看看它的.h文件，它呢又有个子类BnMediaMetadataRetriever，不用管这个是啥，如果对c++层的binder通信掌握的比较好的话理解起来就比较容易一点，说实话我自己也没掌握清楚，咱也不管这个东西是啥，但是呢他又一个子类就和我们相关了，MetadataRetrieverClient。

5. MetadataRetrieverClient.cpp

C++代码都是从.h文件开始看，看看它的.h文件，好巧，它的父类就是BnMediaMetadataRetriever，这里就说说这个东西到底是干啥的把，他是binder提供的一个对外接口，用来实现这个类，为service服务，现在MetadataRetrieverClient，就是为他们服务的，不看构造函数，从前面我们知道，这里还是看setDataSource这个方法，这里看看这个方法到底干了什么吧，他会new一个Retriver，StagefrightMetadataRetriever。然后呢，很巧。又调用了这个对象的setDatSource方法，再去这个方法看看吧。

6. StagefrightMetadataRetriever.cpp

看这个对象的setDataSource方法就很清楚了，就是创建解析器，对于每一个媒体文件都会有对应得解析器去解析它，进入MediaExtractor。

7. MediaExtractor.cpp

看看它的creat方法，这里还会从binder中获取服务，获取media.extractor这个服务，这个这里就不说，上面的流程是走不到这里的，调用CreateFromService方法，这个方法就会根据数据的mimetype来选择相应的解析器，这里以MP3文件为例，根据mime他会new一个MP3Extractor，这个呢就是解码器了。

8. MP3Extractor.cpp

解码器怎么操作，那得看它的编码方式是怎么搞得，MediaScanner只会取tag信息，这个信息的获取规则请看ID3V2的解码编码规则。这里就会一步步解析出来一些基本TAG信息。具体的返回值，在下一节讲述。

以上就是MediaScanner部分libstagefrigh框架流程。具体细节这里不再赘述。

三 音频编码

很多人都很疑惑，我们手机里面的一个几MB的文件，他是怎么播放的，还有它的歌曲信息都是从哪里来的，这里我就简单说一下，对于视频，在MKV文件调查说明。

首先，想要知道它怎么解码，你得知道它怎么编码，现在都是遵循MPEG(MovingPicture Experts Group)运动图像专家组规则，这个仅仅是音频层的编码部分，所以对视频又另外的方式，根据编码质量和编码的复杂程度，分为三层，即Layer-1、Layer2、Layer3，对应MP1，MP2，MP3，三种格式的文件。前面两种现在已经堪称绝迹了，下面就对MP3文件的编码进行分析。

MP3文件其实是由大量的帧（frame）构成，他是MP3文件的最小组成单位。整个MP3文件包括TAG_V2（ID3V2），数据帧和（TAG_V1）ID3V1，三部分。打开一个文件，用一些专门的软件，会发现，这个文件就是由一堆十六进制构成，所有信息都是从这里面读出来的。

这里先从简单的ID3V1分析，这部分在整个文件的末尾，总共分配给它128Byte，它包含：

```
char Header[3]; /*标签头必须是"TAG"否则认为没有标签*/
char Title[30]; /*标题*/
char Artist[30]; /*作者*/
char Album[30]; /*专集*/
char Year[4]; /*出品年代*/
char Comment[28]; /*备注*/
char reserve[1]; /*保留*/
char track[1]; /*音轨*/
char Genre[1]; /*类型*/
```

以上就是MP3文件的ID3V1内容，后面方括号里面的是字节数。

然后就是ID3V2的信息，他包含的信息更多，这部分在整个文件的首部，由一个标签头和若干个标签帧组成。

标签头为10个字节：

```
char Header[3]; /*必须为"ID3"否则认为标签不存在*/
char Ver[1]; /*版本号ID3V2.3 就记录3*/
char Revision[1]; /*副版本号此版本记录为0*/
char Flag[1]; /*存放标志的字节，这个版本只定义了三位*/
char Size[4]; /*标签大小，包括标签头的10个字节和所有的标签帧的大小*/
```

每个标签帧都有十个字节的帧头和至少一个字节的不固定长度的内容组成：

```
char FrameID[4]; /*用四个字符标识一个帧，说明其内容*/
char Size[4]; /*帧内容的大小，不包括帧头，不得小于1*/
char Flags[2]; /*存放标志，只定义了6 位*/
```

数据帧，个数不固定，由文件大小以及帧大小决定，由帧头，CRC校验和data数据构成。

帧头4个字节：

```
unsigned intsync:11; /*同步信息*/
unsigned intversion:2; /*版本*/
unsigned intlayer:2; /*层*/
unsigned intprotection:1; /*CRC校验*/
unsigned intbitrate:4; /*位率*/
unsigned intfrequency:2; /*采样频率*/
unsigned intpadding:1; /*帧长调节*/
unsigned intprivate:1; /*保留字*/
unsigned intmode:2; /*声道模式*/
unsigned int mode extension:2; /*扩充模式*/
unsigned intcopyright:1; /* 版权*/
unsigned intoriginal:1; /*原版标志*/
unsigned intemphasis:2;  /*强调模式*/
```

虽然就四个字节但是包含的信息很多。

CRC校验两个字节

数据部分，这部分就是数据了，它的长度由帧头部分的位率以及频率决定。

```
帧大小 = ( 每帧采样次数 × 比特率(bit/s) ÷ 8 ÷采样率) + Padding
```

以上简单列举了一下MP3文件的编码规则有兴趣可以自己找找ID3V2的编码规则看看。这里简单说一下中文编码：

```
00 – ISO-8859-1 (LATIN-1, Identical to ASCII for values smaller than 0x80).
01 – UCS-2 encoded Unicode with BOM, in ID3v2.2 and ID3v2.3.
02 – UTF-16BE encoded Unicode without BOM, in ID3v2.4.
03 – UTF-8 encoded Unicode, in ID3v2.4.
```

前面两位在mp3文件中代表编码方式，位置在标签帧的四字节头部，四字节大小后的一位，然后还有在后面跟着两位代表大端还是小端编译。

ID3V1用的编码方式是GB2312（ANSI在简体中文中默认为GB2312）的编码规则。