Atitit 解析m4a文件的元数据标签音乐名，歌手 专辑 年代等信息 java版本

目录

[1.1. 自己解析mp4 m4a结构 1](#_Toc5449)

[1.2. 格式返回 1](#_Toc31210)

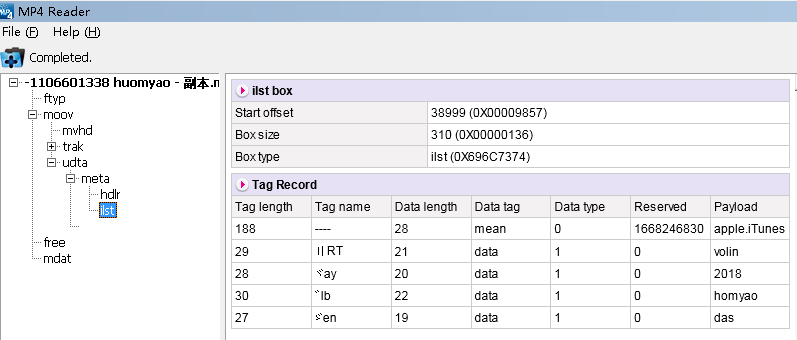
[1.3. /bookmarksHtmlEverythingIndexPrj/src/apkg/audioM4aParser.java 2](#_Toc22400)

[1.4. 2、分割MP4文件 10](#_Toc14466)

## 自己解析mp4 m4a结构

当然最好的方法是使用cli类库ffmpeg获取了，自己写麻烦。。但更加的底层。。

使用图形化分析工具MP4Reader分析其ATOM/BOX结构



可以看到标签路径是 moov/udta/meta/ilst

二、MP4文件结构

MP4是由Atom嵌套来存放媒体信息。Atom的基本结构是：

[4bytes atom length] [4bytes atom name] [8bytes largesize, if size ==1] [contents of the atom, if any]

## 格式返回

{"offset":0,"atomBox":"ftyp","atomBoxSize":28}

{"offset":28,"atomBox":"moov","atomBoxSize":39289}

{"offset":36,"atomBox":"mvhd","atomBoxSize":108}

{"offset":144,"atomBox":"trak","atomBoxSize":38801}

{"offset":38945,"atomBox":"udta","atomBoxSize":372}

{"offset":38953,"atomBox":"meta","atomBoxSize":356}

{"offset":38965,"atomBox":"hdlr","atomBoxSize":34}

{"offset":38999,"atomBox":"ilst","atomBoxSize":310}

{"offset":39007,"atomBox":"----","atomBoxSize":188}

{"offset":39195,"atomBox":"�ART","atomBoxSize":29}

{"offset":39195,"atomBox":"�ART","atomBoxSize":29,"ati�ART":"volin"}

{"offset":39237,"atomBox":"�day","atomBoxSize":28}

{"offset":39237,"atomBox":"�day","atomBoxSize":28,"ati�day":"2018"}

{"offset":39278,"atomBox":"�alb","atomBoxSize":30}

{"offset":39278,"atomBox":"�alb","atomBoxSize":30,"ati�alb":"homyao"}

{"offset":39321,"atomBox":"�gen","atomBoxSize":27}

{"offset":39321,"atomBox":"�gen","atomBoxSize":27,"ati�gen":"das"}

{"offset":39361,"atomBox":"Xtra","atomBoxSize":8}

{"offset":39369,"atomBox":"free","atomBoxSize":18141}

{"offset":57510,"atomBox":"mdat","atomBoxSize":3347320}

## /bookmarksHtmlEverythingIndexPrj/src/apkg/audioM4aParser.java

**package** apkg;

**import** java.io.EOFException;

**import** java.io.FileNotFoundException;

**import** java.io.IOException;

**import** java.io.RandomAccessFile;

**import** java.util.Arrays;

**import** java.util.List;

**import** java.util.Map;

**import** org.apache.log4j.Logger;

**import** com.alibaba.fastjson.JSON;

**import** com.google.common.collect.Lists;

**import** com.google.common.collect.Maps;

**public** **class** audioM4aParser {

**private** **static** **final** **int** ***CMP4TAGATOM\_ERROR*** = 0; // 初始化值

**private** **static** **final** **int** ***CMP4TAGATOM\_ALBUM*** = 1; // 专辑

**private** **static** **final** **int** ***CMP4TAGATOM\_ARTIST*** = 2; // 艺术家

**private** **static** **final** **int** ***CMP4TAGATOM\_NAME*** = 3; // 名称

**private** **static** **final** **int** ***CMP4TAGATOM\_DATE*** = 4; // 日期

**private** **static** **final** **int** ***CMP4TAGATOM\_GENRE*** = 5; // 流派

**private** **static** **final** **int** ***CMP4TAGATOM\_COVER*** = 6; // 封面

**static** **int** *offset*;

**public** **static** **void** main(String[] args) **throws** Exception {

String m4afile = "C:\\Users\\Administrator\\Music\\-1106601338 huomyao - 副本.m4a";

RandomAccessFile RandomAccessFile1 = **new** RandomAccessFile(m4afile, "r");

*foreachTag*(RandomAccessFile1);

// System.out.println(JSON.toJSONString(getTagInfo(m4afile), true));

// ;

}

// 递归好处：代码更简洁清晰，可读性更好

**private** **static** **void** foreachTag(RandomAccessFile RandomAccessFile1) **throws** IOException {

**int** startOffset=*offset*;

Map m = Maps.*newLinkedHashMap*();

m.put("offset", startOffset);

// 每个Atom的起首为四个字节的数据长度（Big Endian）和四个字节的类型标识

// read atom/box length,read 4bytes for to a int num

**int** atomBoxSize = RandomAccessFile1.readInt();

*offset* = *offset* + 4;

String atomBox = *readFullyAtomboxName4bytes*(RandomAccessFile1);

m.put("atomBox", atomBox);

m.put("atomBoxSize", atomBoxSize);

System.***out***.println(JSON.*toJSONString*(m));

**if** (*hasChildSubTag*(atomBox)) { //multiChildNodes

// set next offset

*SkipNextTagOffset*(RandomAccessFile1, atomBox);

*foreachTag*(RandomAccessFile1);

}

**if**(*musicTag*(atomBox))

{

*processMusicTag*(atomBox,atomBoxSize,RandomAccessFile1,m);

*foreachTag*(RandomAccessFile1);

}

// termnal node

**int** atomBodyLength = atomBoxSize - 8;// reduce head 8 byte

RandomAccessFile1.skipBytes(atomBodyLength);

*offset* = *offset* + atomBodyLength;

*foreachTag*(RandomAccessFile1);

}

**private** **static** **void** processMusicTag(String atomBox, **int** atomBoxSize, RandomAccessFile RandomAccessFile1, Map m) **throws** IOException {

// if( atomBox.equals( String.valueOf((char)65533)+"ART" ))

// {

**int** atomBodyLength = atomBoxSize - 8;// reduce head 8 byte

*offset* = *offset* + atomBodyLength;

RandomAccessFile1.skipBytes(8);//skip data lenth and type byte 8ge

*offset* = *offset* +8;

**int** dataver=RandomAccessFile1.readInt();

*offset* = *offset* +4;

**int** reserved=RandomAccessFile1.readByte();//

*offset* = *offset* +1;

**int** datalen=atomBodyLength-8-4-1;

**byte**[] ba=**new** **byte**[datalen];

RandomAccessFile1.readFully(ba);

m.put("ati"+atomBox, **new** String(ba).trim());

System.***out***.println(JSON.*toJSONString*(m));

// }

}

**private** **static** **boolean** musicTag(String atomBox) {

**return** atomBox.charAt(0)== (**char**)65533;

}

**private** **static** **void** SkipNextTagOffset(RandomAccessFile randomAccessFile1, String atomBox) **throws** IOException {

**int** steop = 0;

**if** (atomBox.equals("meta"))

steop = 4;

randomAccessFile1.skipBytes(steop);

*offset* = *offset* + steop;

}

**private** **static** **boolean** hasChildSubTag(String atomBox) {

String as = "udta,meta,moov,ilst";

**if** (as.contains(atomBox))

**return** **true**;

**else**

**return** **false**;

}

**private** **static** String readFullyAtomboxName4bytes(RandomAccessFile randomAccessFile1) **throws** IOException {

**byte**[] AtomBoxTypeReadBuf = **new** **byte**[4];

randomAccessFile1.readFully(AtomBoxTypeReadBuf);

String atomBox = **new** String(AtomBoxTypeReadBuf);

*offset* = *offset* + 4;

**return** atomBox;

}

**static** org.apache.log4j.Logger *logger* = Logger.*getLogger*(PicInZipThumb.**class**);

}

## ****2、分割MP4文件****

在视频点播服务中，需要将MP4文件分割为多个分片，此时需要获取关键帧、切割时间轴、更新moov和生成各个分片文件。

附录: ISO/IEC 14496 MPEG的协议标准

ISO/IEC 14496是MPEG专家组制定的MPEG-4标准于1998年10月公布第1版，1999年1月成为国际标准，1999年12月公布了第2版，2000年初成为国际标准。

全文分为27个部分：

（1）ISO/IEC 14496-1系统部分，描述视频和音频数据流的控制、同步以及混合方式（即混流 Multiplexing，简写为MUX）。

（2）ISO/IEC 14496-2视频部分，定义了一个对各种视觉信息（包括自然视频、静止纹理、计算机合成图形等等）的编解码器。（例如XviD编码就属于MPEG-4 Part 2）。

（3）ISO/IEC 14496-3音频部分，定义了一个对各种音频信号进行编码的编解码器的集合。包括高级音频编码（Advanced Audio Coding，缩写为AAC）的若干变形和其他一些音频/语音编码工具。

（4）ISO/IEC 14496-4一致性部分，定义了比特流和设备的一致性条件，用来测试MPEG-4的实现。

（5）ISO/IEC 14496-5参考软件，提供了用于演示功能和说明本标准其他部分功能的软件。

（6）ISO/IEC 14496-6多媒体传送整体框架DMIF，这是MPEG-4应用层与传输网络的接口，定义了通信协议，使MPEG-4系统的数据流能进入各种传输网络。还包含一个存储格式MP4，用于存储编码的场景。

（7) ISO/IEC 14496-7优化的参考软件，提供了对实现进行优化的例子(这里的实现指的是第五部分)。

（8）ISO/IEC 14496-8在IP网络上传输，定义了在IP网络上传输MPEG-4内容的方式。

（9）ISO/IEC 14496-9参考硬件描述，提供了用于演示怎样在硬件上实现本标准其他部分功能的硬件设计方案

（10）ISO/IEC 14496-10高级视频编码AVC，定义了一个视频编解码器（codec）。AVC和XviD都属于MPEG-4编码，但由于AVC属于MPEG-4 Part 10，在技术特性上比属于MPEG-4 Part2的XviD要先进。另外，它和ITU-T H.264标准是一致的，故又称为H.264。

（11）ISO/IEC 14496-11场景描述和应用引擎。

（12）ISO/IEC 14496-12ISO媒体文件格式，定义了一个存储媒体内容的文件格式。

（13）ISO/IEC 14496-13知识产权管理和保护（IPMP：Intellectual Property Management and Protection）扩展。

（14）ISO/IEC 14496-14MP4文件格式，定义了基于第十二部分的用于存储MPEG-4内容的容器文件格式。

（15）ISO/IEC 14496-15AVC文件格式，定义了基于第十二部分的用于存储第十部分的视频内容的文件格式。

（16）ISO/IEC 14496-16动画框架扩展AFX（Animation Framework eXtension）。

（17）ISO/IEC 14496-17同步文本字幕格式。

（18）ISO/IEC 14496-18字体压缩和流式传输(针对公开字体格式)。

（19）ISO/IEC 14496-19合成材质流（Synthesized Texture Stream）。

（20）ISO/IEC 14496-20简单场景表示（LASeR for Lightweight Scene Representation）。

（21）ISO/IEC 14496-21用于描绘(Rendering)的MPEG-J拓展。

（22）ISO/IEC 14496-22开放字体格式（Open Font Format）。

（23）ISO/IEC 14496-2符号化音乐表示（Symbolic Music Representation）。

（24）ISO/IEC 14496-24音频与系统交互作用（Audio and systems interaction）。

（25）ISO/IEC 14496-253D图形压缩模型（3D Graphics Compression Model）。

（26）ISO/IEC 14496-26音频一致性检查：定义了测试音频数据与ISO/IEC 14496-3是否一致的方法（Audio conformance）。

（27）ISO/IEC 14496-273D图形一致性检查：定义了测试3D图形数据与ISO/IEC 14496-11:2005, ISO/IEC 14496-16:2006, ISO/IEC 14496-21:2006, 和 ISO/IEC 14496-25:2009是否一致的方法（3D Graphics conformance）。

---------------------