视频播放用的三方库

用到技术：

核心类：VideoView - - - 〉最终播放由　MediaPlayer 完成

定义了很多监听回调的方法：

视频缓冲结束的回调处理（可以在里面开始播放，更新播放进度，更新播放时间）

视频缓冲进度发生改变时监听处理（设置缓冲的进度）

视频播放出错的回调处理（可以给一些提示，或者其它处理）

视频播放完成的回调处理（ ）

视频播放状态的回调处理（ ）

技术核心方法及思想：调用setVideoPath(视频播放路径) - - - > 这个方法内部就做调用了setVideoURI（ ）方法将地址转为资源 - - - - > 然后调用了openVideo();这个方法对传入的资源进行了判断是否为空并定义了回调方法 - - - - >然后调用start();方法系统其实调用了MediaPlayer的start()方法 - - - - - - >调用了本地方法通过JNI调用底层驱动开始播放

万能播放器

Vitamio 使用了 FFmpeg 做为媒体解析器和最主要的解码器，同时开发了针对不同移动平台的硬解码方案，能够完美支持 H.264/AVC、H.263、 MPEG4 等常见的视频编码

网络请求用到的三方库

用到的技术：**Volley，Retrofit，OkHttp**

技术的核心方法及思想：

Volley：

1. 扩展性强
   基于接口设计。
2. 一定程度上符合http规范。
   返回包括ResponseCode 的处理，请求头的处理，缓存机制的支持。
3. 重试以及优先级的定义。
4. 2.3以上基于HttpUrlconnection。
   2.3以下是Httpclient（没有这号机子了吧）
5. 提供简单的图片加载工具。

Retrofit：

Retrofit 基于注解，提供JSON to POJO(Plain Ordinary Java Object,简单Java对象)，POJO to JSON，网络请求(POST，GET,PUT，DELETE等)封装。

区别：

我们知道在 Android 开发中是可以直接使用现成的 api 进行网络请求的，就是使用 HttpClient、HttpUrlConnection 进行操作，目前 HttpClient 已经被废弃，而 android-async-http 是基于 HttpClient 的，我想可能也是因为这个原因作者放弃维护。

而 OkHttp 是 Square 公司开源的针对 Java 和 Android 程序，封装的一个高性能 http 请求库，所以它的职责跟 HttpUrlConnection 是一样的，支持 spdy、http 2.0、websocket ，支持同步、异步，而且 OkHttp 又封装了线程池，封装了数据转换，封装了参数使用、错误处理等，api 使用起来更加方便。可以把它理解成是一个封装之后的类似 HttpUrlConnection 的一个东西，但是你在使用的时候仍然需要自己再做一层封装，这样才能像使用一个框架一样更加顺手。

Volley 是 Google 官方出的一套小而巧的异步请求库，该框架封装的扩展性很强，支持 HttpClient、HttpUrlConnection，甚至支持 OkHttp
而且 Volley 里面也封装了 ImageLoader ，所以如果你愿意你甚至不需要使用图片加载框架，不过这块功能没有一些专门的图片加载框架强大，对于简单的需求可以使用，对于稍复杂点的需求还是需要用到专门的图片加载框架。Volley 也有缺陷，比如不支持 post 大数据，所以不适合上传文件。
Retrofit 是 Square 公司出品的默认基于 OkHttp 封装的一套 RESTful 网络请求框架，可以使用不同 Json Converter 来序列化数据，同时提供对 RxJava 的支持，使用 Retrofit + OkHttp + RxJava + Dagger2 可以说是目前比较潮的一套框架

Volley VS OkHttp

毫无疑问 Volley 的优势在于封装的更好，而使用 OkHttp 你需要有足够的能力再进行一次封装。而 OkHttp 的优势在于性能更高，因为 OkHttp 基于 NIO 和 Okio ，所以性能上要比 Volley更快。

xUtils 支持大文件上传

- 目前xUtils主要有四大模块：
  - DbUtils模块：数据库

    - android中的orm框架，一行代码就可以进行增删改查；
    - 支持事务，默认关闭；
    - 可通过注解自定义表名，列名，外键，唯一性约束，NOT NULL约束，CHECK约束等（需要混淆的时候请注解表名和列名）；
    - 支持绑定外键，保存实体时外键关联实体自动保存或更新；
    - 自动加载外键关联实体，支持延时加载；
    - 支持链式表达查询，更直观的查询语义，参考下面的介绍或sample中的例子。

  - ViewUtils模块：完全注解方式就可以进行UI绑定和事件绑定

- android中的ioc框架，完全注解方式就可以进行UI，资源和事件绑定；
- 新的事件绑定方式，使用混淆工具混淆后仍可正常工作；
- 目前支持常用的20种事件绑定，参见ViewCommonEventListener类和包com.lidroid.xutils.view.annotation.event。

- HttpUtils模块：网络请求下载等 HTTP实现UrlConnection

- 支持同步，异步方式的请求；
- 支持大文件上传，上传大文件不会oom；
- 支持断点续传，随时停止下载任务，开始任务
- 支持GET，POST，PUT，MOVE，COPY，DELETE，HEAD，OPTIONS，TRACE，CONNECT请求；
- 下载支持301/302重定向，支持设置是否根据Content-Disposition重命名下载的文件；
- 返回文本内容的请求(默认只启用了GET请求)支持缓存，可设置默认过期时间和针对当前请求的过期时间。

- BitmapUtils模块：加载网络图片

- 加载bitmap的时候无需考虑bitmap加载过程中出现的oom和android容器快速滑动时候出现的图片错位等现象；
- 支持加载网络图片和本地图片；
- 内存管理使用lru算法，更好的管理bitmap内存；
- 可配置线程加载线程数量，缓存大小，缓存路径，加载显示动画等...

数据解析

用到的技术：
核心方法及思想：

视图注解与findViewById()

用到的技术：
核心思想：

获取手机视频，音频，图片的方法及思想

手机中的这些资源信息都存放在一个数据库中，通过读取数据库中的相应的信息可以获取相应的数据（data\data\media\external.db数据哭的fies表中） 通过内容提供者共享数据

用到的技术：

获取数据可以通过内容提供者读取数据

ContentResolver 同步查询

AsyncQueryHandler 异步查询

同法： 对像的构造方法需要一个ContentResolver对象 可以通过上下获取一个context.getContentResolver() 对象

1、AsyncQueryHandler aqh = new AsyncQueryHandler(){重写相应的方法（提供了增删改查完成后的回调方法）}

操作完成的结果数据都在回调方法中，查询会返回一个游标cursor 其他返回一个结果result

游标的作用：

通过游标查询获取相应的数据，设置给JAVAbean

```
AudioBean bean = new AudioBean();
bean.id = cursor.getInt(cursor.getColumnIndex(MediaStore.Video.Media._ID));
bean.displayName =
cursor.getString(cursor.getColumnIndex(MediaStore.Video.Media.DISPLAY_NAME));
bean.size =
cursor.getLong(cursor.getColumnIndex(MediaStore.Video.Media.SIZE));
bean.duration =
cursor.getLong(cursor.getColumnIndex(MediaStore.Video.Media.DURATION));
bean.path =
cursor.getString(cursor.getColumnIndex(MediaStore.Video.Media.DATA));
```

通过位置返回一个游标

```
Cursor cursor = (Cursor) parent.getItemAtPosition(position);
```

2、通过AsyncQueryHandler对象发起相应的操作

（如查询：queryHandler.startQuery(token, cookie, uri,projection, selection, selectionArgs, orderBy);）

```
queryHandler.startDelete();   删
queryHandler.startInsert();   增
queryHandler.startUpdate();   改
```

方法参数说明：

token,　　相当于Message的what
cookie,　　相当于Message的obj
uri,　　要查询数据的uri
projection,　要查询的列
selection,　　查询条件 ″name = ?″
selectionArgs,　查询条件参数
orderBy　　排序条件　降序″ DESC ″　　升序″ ASC ″

关于查询视频的字URI及字段 可以通过数据库获取相应的字段

Uri uri = MediaStore.Video.Media.EXTERNAL_CONTENT_URI;

MediaStore.Audio.Media._ID,　　// 视频的_id　" AS _id"
MediaStore.Audio.Media.TITLE,　　// 视频的名称
MediaStore.Audio.Media.SIZE,　　// 视频的大小
MediaStore.Audio.Media.DURATION,　　// 视频的播放时长
MediaStore.Audio.Media.DATA,　　// 视频保存的路径

关于查询音频的字URI及字段
Uri uri = MediaStore.Audio.Media.EXTERNAL_CONTENT_URI;
关于查询图片的字URI及字段
Uri uri = MediaStore.Images.Media.EXTERNAL_CONTENT_URI;

服务开启与停止的区别

用服务的地方：

三级缓存的实现及应用

线程池原理

事件分发机制原理

屏幕适配方案：

sp
dp

px

广播的作用

Eventbus与Handler的区别及Eventbus的原理

屏幕滑动触摸事件监听

GestureDetector mGestureDetector = new GestureDetector(this, mOnGestureListener);
mOnGestureListener = new GestureDetector.SimpleOnGestureListener(){}
选择重写方法：
滑动时
e1：用户按时的点
e2: 用户滑动结束松开手时点
onScroll(MotionEvent e1, MotionEvent e2, float distanceX, float distanceY)

按下时
onDown(MotionEvent e)

单击时
onSingleTapUp(MotionEvent e)

双击
onDoubleTap(MotionEvent e)

长按
onLongPress(MotionEvent e)

将事件交给触摸事件处理触摸生效
public boolean onTouchEvent(MotionEvent event)
    mGestureDetector.onTouchEvent(event);

手机电池电量获取的方法

1、动态注册广播接收系统电池变化
IntentFilter filter = new IntentFilter();
filter.addAction(Intent.ACTION_BATTERY_CHANGED);

```
registerReceiver(mBatteryReceiver, filter);
2、接收广播监听到的电池信息
private BroadcastReceiver mBatteryReceiver = new BroadcastReceiver() {
 @Override
public void onReceive(Context context, Intent intent) {
  if (Intent.ACTION_BATTERY_CHANGED.equals(intent.getAction())) {
  Bundle extras = intent.getExtras();
            /**获取所有电池的相关信息
             * Set<String> keys = extras.keySet();
             *  for (String key:keys) {
             *  Object o = extras.get(key);
             *  Global.logs(""+key+"    "+o); }   */
 int level = extras.getInt("level");  //获取系统电量
```

格式化时间

DateFormat: kk:mm:ss

```
String format ="kk:mm:ss";
DateFormat.format(format, duration).toString();
```

SimpleDateFormat: hh:mm:ss

SimpleDateFormat 是一个以国别敏感的方式格式化和分析数据的具体类。 它允许格式化 (date -> text)、语法分析 (text -> date)和标准化。

```
            String str ="2017-03-08 09:22:35";   指定的日期
            SimpleDateFormat sdf = new SimpleDateFormat("yyyy-MM-dd hh:mm:ss");
            Date date = sdf.parse(str);//默认的系统时间格式
            long time = date.getTime(); //将一个日期转为毫秒值
            SimpleDateFormat sdf1 = new SimpleDateFormat("yyyy年MM月dd日");
            String str1 = sdf1.format(date);   //将指定的日期转为指定的格式
```

格式化数据为相应MB

内存存储

```
File romDir = Environment.getDataDirectory();
```

获取内存总的大小
long totalRomSpace = romDir.getTotalSpace();// bits

获取内存剩余的大小
long freeRomSpace = romDir.getFreeSpace();

将获取到的内存数值long类型转换为MB/GB/KB等
Formatter.formatFileSize(this, freeRomSpace);

外部存储

File sdDir = Environment.getExternalStorageDirectory();

获取总的大小
long totalSDSpace = sdDir.getTotalSpace();// bits

获取剩余的大小
long freeSDSpace = sdDir.getFreeSpace();

将获取到的内存数值long类型转换为MB/GB/KB等
Formatter.formatFileSize(this, usedSDSpace)