1 计算机表示图形的几种方式

gif 质量比较低 网络电邮
jpg 质量良好 计算机 网络 电子邮件
png 高质量 计算机 网络 (android)
bmp 高质量

bit 位 一位二进制数 100Mb /8 12.5MB/s
Byte 8位二进制数
KB
MB
GB
TB
PB

单色位图 1位2进制数表示 一个像素 1byte 8位二进制数 1个像素占用1/8个byte 5000byte
16色位图 16位的二进制数 表示一个像素 16种颜色表示一个像素 1111 1111 1/2byte
256色位图 1个像素用256种颜色表示 2^8=256 1个byte对应一个像素 39k
24位位图 24位 位代表一位2进制数 24位二进制数 3byte 117k

argb8888 android默认情况 每个像素占用4byte内存
alpha 8 透明度 #66000000
r red 8
g green 8
b blue 8

2 大图加载

核心思路 获取压缩比例 inSampleSize

```
1.  public class MainActivity extends Activity {
2.      private String path = "mnt/sdcard/big.jpg";
3.      private ImageView iv_image;
4.      private int width;
5.      private int height;
6.
7.      @SuppressLint("NewApi") @SuppressWarnings("deprecation")
8.      @Override
9.      protected void onCreate(Bundle savedInstanceState) {
10.         super.onCreate(savedInstanceState);
```

```
11.          setContentView(R.layout.activity_main);
12.        iv_image = (ImageView) findViewById(R.id.iv_image);
13.        //获取屏幕的宽度和高度
14.        width = getWindowManager().getDefaultDisplay().getWidth();
15.        height = getWindowManager().getDefaultDisplay().getHeight();
16.
17.  //      Point outSize = new Point();
18.  //  getWindowManager().getDefaultDisplay().getSize(outSize);
19.  //  int screenWidth = outSize.x;
20.  //  int screenHeight = outSize.y;
21.  //
22.  //  System.out.println("width"+width+"==="+screenWidth);
23.  //  System.out.println("height"+height+"==="+screenHeight);
24.      }
25.     public void loadpic(View v){
26.      loadpic3();
27.     }
28.
29.     public void loadpic1(){
30.         //用图片的宽高 跟屏幕的宽高比较
31.         //如果屏幕宽高比较小 把图片压缩到跟屏幕宽高差不多就可以了
32.         BitmapFactory.Options opts = new Options();
33.         //inJustDecodeBounds 这个参数设置为true 再调用BitmapFactory.decodeXXX方法
34.         //并不会把图片真正加载到内存 而是把图片的大小读出来 设置到opts.outHeight 和
     opts.outWidth属性中
35.         opts.inJustDecodeBounds = true;
36.         Bitmap bitmap = BitmapFactory.decodeFile(path, opts);
37.         //获取图片的宽度和图片的高度
38.         int picHeight = opts.outHeight;
39.         int picWidth = opts.outWidth;
40.  //      if(bitmap == null){
41.  //          System.out.println("picheight"+picHeight+"picwide"+picWidth);
42.  //      }
43.         opts.inSampleSize = 1;
44.         if(picHeight>height||picWidth>width){
45.             //为了结果跟准确 把宽高转换成float类型 然后用Math.round对结果进行四舍五
     入
46.            int heightIndex = Math.round((float)picHeight/(float)height);
47.            int widthIndex = Math.round((float)picWidth/(float)width);
48.            //取计算出来的两个数的最大值
49.            opts.inSampleSize = Math.max(heightIndex, widthIndex);
50.         }
51.
52.         //把inJustDecodeBounds 改为false 这样可以用算出的inSampleSize去加载图片到内
     存
53.         opts.inJustDecodeBounds = false;
54.         System.out.println("opts.inSampleSize="+opts.inSampleSize);
55.         bitmap = BitmapFactory.decodeFile(path, opts);
56.         iv_image.setImageBitmap(bitmap);
57.     }
58.
59.     public void loadpic2(){
60.         BitmapFactory.Options opts = new Options();
61.         //inSampleSize 可以告诉BitmapFactory 在解析图片的时候压缩图片的大小 用来节
```

```
省内存
62.         //inSampleSize的值 指的是对图片的宽度 和高度 做相同大小的压缩处理  也就是说 inSampleSize = 2
63.         //创建出来的位图文件 宽度是原图的1/2 高度 大小也是1/2 最终占用内存是不压缩情况下的1/4
64.         opts.inSampleSize =  2;
65.         //加载磁盘上的图片文件 转换成bitmap对象
66.         Bitmap bitmap = BitmapFactory.decodeFile(path, opts);
67.         //把bitmap位图文件展示到imageview上
68.         iv_image.setImageBitmap(bitmap);
69.     }
70.
71.     public void loadpic3(){
72.         //第二种计算inSampleSize 压缩比例的思路 不断去试验 如果有内存溢出 就给它抓住 提高压缩比例 继续试验
73.         //知道能够正确加载图片位置
74.         BitmapFactory.Options opts = new Options();
75.         int i = 1;
76.
77.         Bitmap bitmap = null;
78.         for(;;){
79.             try {
80.                 opts.inSampleSize = i;
81.                 //如果当前的压缩比例能够争取的加载图片 循环就会退出
82.                 bitmap = BitmapFactory.decodeFile(path, opts);
83.                 break;
84.          } catch (Error e) {
85.              //如果走到异常了 就提高压缩比例 继续试验
86.              //inSampleSize 这个值要取2的幂指数 如果不是2的多少次幂
87.              //系统也会把这个数转化为最近的那个2的多少次幂的值
88.              i*=2;
89.              System.out.println("i = "+i);
90.          }
91.         }
92.         iv_image.setImageBitmap(bitmap);
93.     }
94. }
```

3.创建图片副本

```
1.  public class MainActivity extends Activity {
2.
3.      private ImageView iv_image;
4.      private ImageView iv_image2;
5.
6.      @Override
7.      protected void onCreate(Bundle savedInstanceState) {
8.          super.onCreate(savedInstanceState);
9.          setContentView(R.layout.activity_main);
10.         iv_image = (ImageView) findViewById(R.id.iv_image);
11.         iv_image2 = (ImageView) findViewById(R.id.iv_image2);
12.         Bitmap bitmap = BitmapFactory.decodeResource(getResources(), R.drawable.to
```

```
mcat);
//        bitmap.setPixel(30, 30, Color.RED);
        iv_image.setImageBitmap(bitmap);
        //①创建一张空的图片
        Bitmap copy_bitmap = Bitmap.createBitmap(bitmap.getWidth(), bitmap.getHeight(), bitmap.getConfig());
        //②用创建好的空的图片创建一个canvas对象  画板
        Canvas canvas = new Canvas(copy_bitmap);
        //矩阵 用来处理图片的旋转 缩放 平移效果
        Matrix matrix = new Matrix();
        //画笔
        Paint paint = new Paint();
        //③向空的图片上画对应的信息 这个方法执行之后 创建的空的图片就有相关的图片信息了
        //第一个参数 原图  第二个参数 矩阵 第三个参数 画笔
        canvas.drawBitmap(bitmap, matrix, paint);
        for(int i = 0;i<30;i++){
            copy_bitmap.setPixel(20+i, 20+i, Color.RED);
        }
        iv_image2.setImageBitmap(copy_bitmap);
    }
}
```

4 图形处理的api

Matrix rotate旋转 translate平移 scale缩放

```
public class MainActivity extends Activity {

    private ImageView iv_image;
    private ImageView iv_image2;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        iv_image = (ImageView) findViewById(R.id.iv_image);
        iv_image2 = (ImageView) findViewById(R.id.iv_image2);

        Bitmap bitmap = BitmapFactory.decodeResource(getResources(), R.drawable.tomcat);
        iv_image.setImageBitmap(bitmap);
        //用要复制的图片的相关信息创建一张新的空图片
        //第一个参数图片的宽度  第二个参数图片的高度 第三个参数 图片的配置信息  都从原图中获取
        Bitmap copy_bm = Bitmap.createBitmap(bitmap.getWidth(), bitmap.getHeight(), bitmap.getConfig());
        //创建一个画板对象 Canvas
        Canvas canvas = new Canvas(copy_bm);
        //通过canvas画图片
        Matrix matrix = new Matrix();
```

```
        //rotate 旋转  tranlate 平移  scale 缩放
        //setRotate 旋转图片 第一个参数 旋转的度数 第二个参数 第三个参数 旋转时 旋转的中心坐标
        //matrix.setRotate(90f,copy_bm.getWidth()/2,copy_bm.getHeight()/2);
       //setTranslate 平移  第一个参数 x方向平移的距离  第二个参数 y方向平移的距离
       // matrix.setTranslate(30, 0);
//        matrix.setScale(-1f, 1f);
//        matrix.postTranslate(copy_bm.getWidth(), 0);
        //如果setScale传了负数 就会有镜像的效果
        matrix.setScale(1f, -1f);
        //多次设置 大小 平移和旋转 需要使用post 每一次setXXX都会清空上一句set的结果
        matrix.postTranslate(0, copy_bm.getHeight());
        Paint paint = new Paint();
        canvas.drawBitmap(bitmap, matrix, paint);
        iv_image2.setImageBitmap(copy_bm);
    }
}
```

5 画画板小案例

setonTouchListener 设置触摸事件

```
public class MainActivity extends Activity {

    private Canvas canvas;
    private Bitmap copy_bm;
    private Paint paint;
    private ImageView iv_image;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        iv_image = (ImageView) findViewById(R.id.iv_image);
        Bitmap bitmap = BitmapFactory.decodeResource(getResources(),
                R.drawable.bg);
        //①创建空图片
        copy_bm = Bitmap.createBitmap(bitmap.getWidth(),
                bitmap.getHeight(), bitmap.getConfig());
        //用空的图片准备canvas 画板
        canvas = new Canvas(copy_bm);
        //用来做图片处理的 矩阵对象
        Matrix matrix = new Matrix();
        //画笔
        paint = new Paint();
        //在canvas上把要花的图片 画到创建的空的图片对象上
        canvas.drawBitmap(bitmap, matrix, paint);
        iv_image.setImageBitmap(copy_bm);

        //给imagview设置了一个触摸监听
        iv_image.setOnTouchListener(new OnTouchListener() {
```

```
30.
31.             private float startX;
32.             private float startY;
33.
34.             @Override
35.             public boolean onTouch(View v, MotionEvent event) {
36.                 // MotionEvent 动作事件 通过MotionEvent 可以获取当前用户的触摸动作
    是哪种类型
37.                 int action = event.getAction();
38.                 switch (action) {
39.                 case MotionEvent.ACTION_DOWN:
40.                     System.out.println("按下");
41.                     //手指按下 记录起始的位置
42.                     startX = event.getX();
43.                     startY = event.getY();
44.                     break;
45.                 case MotionEvent.ACTION_MOVE:
46. //                  System.out.println("移动");
47.                     float stopX = event.getX();
48.                     float stopY = event.getY();
49. //                  System.out.println("x ==="+x+"y=="+y);
50.                     //画线
51.                     canvas.drawLine(startX, startY, stopX, stopY, paint);
52.                     //更新起始坐标
53.                     startX = stopX;
54.                     startY = stopY;
55.                     //把画线后的图片展示到imageview上
56.                     iv_image.setImageBitmap(copy_bm);
57.                     break;
58.                 case MotionEvent.ACTION_UP:
59.                     System.out.println("抬起");
60.
61.                     break;
62.
63.                 }
64.                 //返回值是true说明当前控件 消费了这个触摸事件 不需要交给其他控件处
    理
65.                 return true;
66.             }
67.         });
68.
69.     }
70.     /**
71.      * 把画笔的颜色改变为红色
72.      * @param v
73.      */
74.     public void changecolor(View v){
75.         paint.setColor(Color.RED);
76.     }
77.
78.     public void bold(View v){
79.         //设置画笔的宽度
80.         paint.setStrokeWidth(5);
81.     }
```

```
82.
83.     public void save(View v){
84.         try {
85.             //compress方法 压缩保存图片
86.             //第一个参数 保存的格式 CompressFormat是一个枚举类 只能取固定的枚举值
     CompressFormat.PNG CompressFormat.JPEG
87.             //第二个参数 保存图片的质量 0 压缩的最厉害 图片体积最小 100质量最高 体
    积最大
88.             //第三个参数 保存文件用到的输出流
89.             copy_bm.compress(CompressFormat.PNG, 100, openFileOutput("pic.png", MO
    DE_PRIVATE));
90.         } catch (FileNotFoundException e) {
91.             // TODO Auto-generated catch block
92.             e.printStackTrace();
93.         }
94.     }
95. }
```

6 撕衣服小案例

7 mediaplayer播放音频

8 音乐播放器完成

Activity

```
1.  public class MainActivity extends Activity {
2.
3.      public static final int GET_DURATION = 0;
4.      public static final int UPDATE_PROGRESS = 1;
5.      private ImageView iv_player;
6.      private String path = "mnt/sdcard/xpg.mp3";
7.      private MyConn conn;
8.      private static MyBinder binder;
9.      private static SeekBar sb_progress;
10.     public static Handler handler = new Handler(){
11.         public void handleMessage(android.os.Message msg) {
12.             switch (msg.what) {
13.             case GET_DURATION:
14.                 sb_progress.setMax((Integer)msg.obj);
15.                 break;
16.             case UPDATE_PROGRESS:
17.                 updateProgress();
18.                 break;
19.             default:
20.                 break;
21.             }
22.         };
23.     };
24.
25.     @Override
```

```
26.     protected void onCreate(Bundle savedInstanceState) {
27.         super.onCreate(savedInstanceState);
28.         setContentView(R.layout.activity_main);
29.         iv_player = (ImageView) findViewById(R.id.iv_play);
30.         sb_progress = (SeekBar) findViewById(R.id.sb_progress);
31.         //需要设置进度条的最大值
32.         sb_progress.setOnSeekBarChangeListener(new OnSeekBarChangeListener() {
33.             @Override
34.             public void onStopTrackingTouch(SeekBar seekBar) {
35.                 //停止操作进度条
36.             }
37. 
38.             @Override
39.             public void onStartTrackingTouch(SeekBar seekBar) {
40.                 //开始操作进度条(拖动)
41.             }
42. 
43.             @Override
44.             public void onProgressChanged(SeekBar seekBar, int progress,
45.                     boolean fromUser) {
46.                 //fromUser= true 说明是用户在手动拖动进度条改变进度
47.                 //fromUser = false 说明是通过代码的方式setProgress改变进度
48.                 if(fromUser){
49.                 //只处理用户手动改变进度的情况
50.                     binder.seekTo(progress);
51.                 }
52.             }
53.         });
54.         iv_player.setOnClickListener(new OnClickListener() {
55. 
56.             @Override
57.             public void onClick(View v) {
58.                 //点击按钮开始播放
59.                 binder.playPause();
60.                 upDatePlayIcon();
61.             }
62. 
63. 
64.         });
65. 
66.         Intent service = new Intent(this,MusicService.class);
67.         //混合方式开启service
68.         startService(service);
69.         conn = new MyConn();
70.         bindService(service, conn, BIND_AUTO_CREATE);
71. 
72.     }
73. 
74.     protected static void updateProgress() {
75.         //获取当前播放进度设置到进度条上
76.         sb_progress.setProgress(binder.getCurrentPostion());
77.         handler.sendEmptyMessageDelayed(UPDATE_PROGRESS, 500);
78.     }
79. 
```

```
    private class MyConn implements ServiceConnection{

        @Override
        public void onServiceConnected(ComponentName name, IBinder service) {
            binder = (MyBinder) service;
            //当调用bindservice activity和service建立连接之后 就会走这个方法
            //在onServiceConnected 做初始化图标状态的操作
            upDatePlayIcon();
            //初始化进度条状态 初始化最大值
            sb_progress.setMax(binder.getDuration());
            //发送消息开始跟新进度条状态
            //if(binder.isPlaying()){
                handler.sendEmptyMessage(UPDATE_PROGRESS);
            //}
        }
        @Override
        public void onServiceDisconnected(ComponentName name) {
        }

    }

    @Override
    protected void onStop() {
        super.onStop();
        //界面不可见
        //把handler未执行的排队消息清空
        handler.removeCallbacksAndMessages(null);
    }

    @Override
    protected void onDestroy() {
        super.onDestroy();
        //页面退出 跟service断开连接 避免connection泄露
        unbindService(conn);

    }
    private void upDatePlayIcon() {
        if(binder.isPlaying()){
            //如果是正在播放的状态 那么显示暂停的图标
            iv_player.setImageResource(R.drawable.btn_audio_pause);
        }else{
            //如果处于暂停的状态 那么显示播放的图标
            iv_player.setImageResource(R.drawable.btn_audio_play);
        }
    }
}
```

Service

```
public class MusicService extends Service {
    private String path = "mnt/sdcard/xpg.mp3";
    private MediaPlayer player;

    @Override
```

```
6.      public IBinder onBind(Intent arg0) {
7.          return new MyBinder();
8.      }
9.
10.     @Override
11.     public void onCreate() {
12.         super.onCreate();
13.         player = new MediaPlayer();
14.         try {
15.             //给媒体播放器设置数据源
16.             player.setDataSource(path);
17.             //媒体播放器开始准备 把准备变成异步准备并且准备好的监听器 就可以播放网
    络/本地音乐
18.             player.prepare();
19.             //获取音乐播放的总时长
20.             //获取消息对象
21.             Message msg = Message.obtain();
22.             //把总时长 通过msg.obj 封装到message里
23.             msg.obj = player.getDuration();
24.             //设置msg.what 用于区分不同的消息
25.             msg.what = MainActivity.GET_DURATION;
26.             //通过handler发送消息
27.             MainActivity.handler.sendMessage(msg);
28.
29.         } catch (Exception e) {
30.             // TODO Auto-generated catch block
31.             e.printStackTrace();
32.         }
33.     }
34.
35.     public class MyBinder extends Binder{
36.         /**
37.          * 控制音乐的播放和暂停 如果暂停那么开始播 如果正在播放 那么暂停
38.          */
39.         public void playPause(){
40.             if(player!=null){
41.                 if(player.isPlaying()){
42.                     //isPlaying 获取播放的状态 如果正在播放会返回true
43.                     //pause暂停
44.                     player.pause();
45.                 }else{
46.                     //正处于暂停的状态 应该让它继续播放
47.                     player.start();
48.                     //通知handler开始更新进度
49.                     MainActivity.handler.sendEmptyMessage(MainActivity.UPDATE_PROG
    RESS);
50.                 }
51.
52.             }
53.         }
54.
55.         /**
56.          * 获取当前播放的状态
57.          * @return 如果正在播放返回true
```

```
58.          */
59.         public boolean isPlaying(){
60.             if(player != null){
61.                 return player.isPlaying();
62.             }else{
63.                 return false;
64.             }
65.         }
66.
67.         /**
68.          * 获取当前播放的进度
69.          * @return
70.          */
71.         public int getCurrentPostion(){
72.             if(player!=null){
73.                 return player.getCurrentPosition();
74.             }else{
75.                 return 0;
76.             }
77.         }
78.
79.         /**
80.          * 获取播放总时长
81.          * @return
82.          */
83.         public int getDuration(){
84.             if(player!=null){
85.                 return player.getDuration();
86.             }else{
87.                 return 0;
88.             }
89.         }
90.
91.         /**
92.          * 更新进度
93.          * @param msec 播放的毫秒值
94.          */
95.         public void seekTo(int msec){
96.             if(player != null)
97.             player.seekTo(msec);
98.         }
99.     }
100. }
```

9 mediaplayer生命周期(状态机)

prepare()/prepareAsync()

prepare()同步的准备(在当前线程(主线程)进行准备工作)

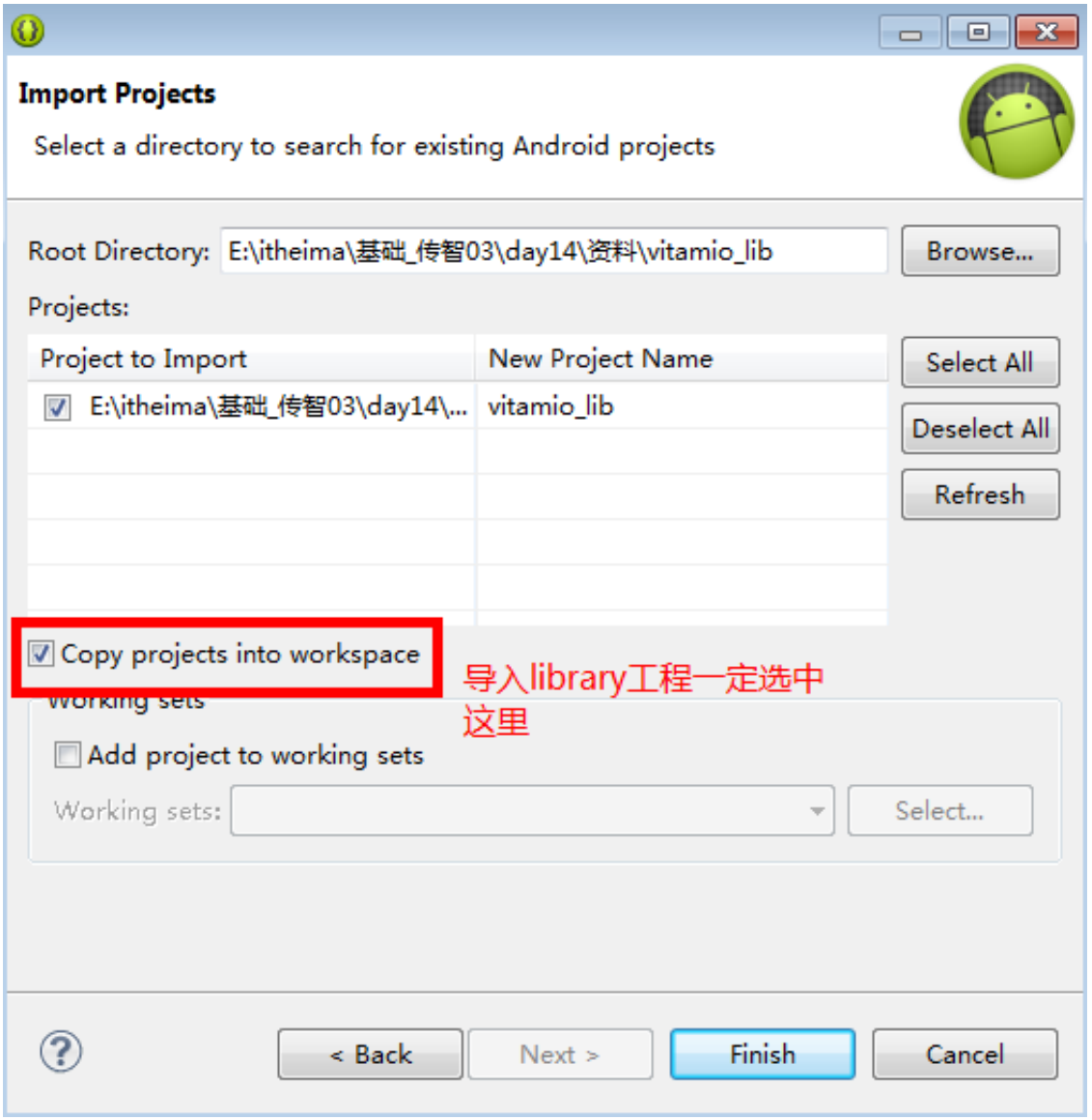
prepareAsync()异步准备 开启子线程 如果是访问网络的音频那么要使用异步的准备

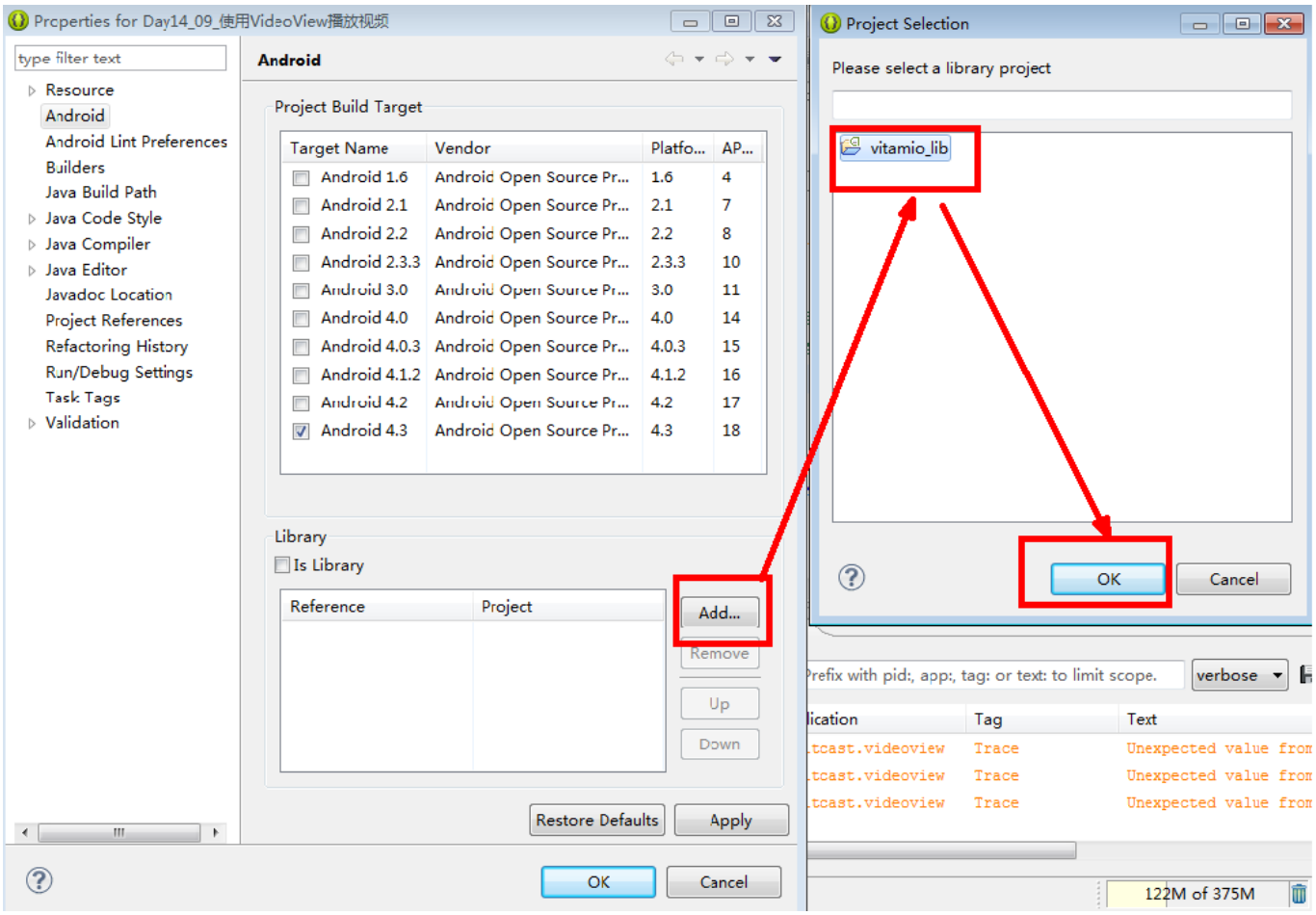
start() 和 pause()之间可以相互转换
stop() 一旦调用了stop就要重新准备(prepare()/prepareAsync())

创建MediaPlayer -> setDataSource()->prepare()/prepareAsync()-> start()/pause()/seekTo() ->stop->prepare()/prepareAsync()

10surfaceview介绍
通过mediaplayer和serfaceView配合使用可以实现播放视频
mediaplayer用来处理视频文件的解码和声音的播放 surfaceView用来展示图像
surfaceView是一个重量级的控件 加载需要一点儿时间
surfaceview内部维护两个线程 一个用来加载数据 一个用来展示界面

11videoview控件介绍
12vitamio 三方音视频播放框架 VLC android也有VLC的分支
ffmpeg c/c++实现的开源媒体解码框架 支持95%以上的媒体格式
vitamio的api跟VideoView十分类似 只有几个方法接收的数据类型不同 多了几个方法





今日重点

① 大图加载

② 图片操作(移动 旋转 缩放)

③ 画画板

④ 音乐播放器 ****

⑤ VideoView播放视频(通过Vitamio) 导入library工程