图片处理
大图的加载 out of memory(OOM) 内存溢出 memory leak()内存泄露
分辨率高
ARGB.8888 一个像素 4byte
BitmapFactory.Options.inSampleSize >1 被压缩
根据屏幕的分辨率 getWindowManager.getDefaultDisplay.getSize(Point());
获取图片宽度高度
BitmapFactory.Options.inJustDecodeBounds = true;
Options.outWidth outHeight

不断试验inSampleSize

图片副本
Bitmap (mutable 可以修改的) Bitmap.createBitmap(); 创建空白的图片
Canvas drawbitmap(Bitmap, Matrix,Paint);
setScale postScale
translate
rotate

setOnTouchListenser(OnTouchListenser)
getX() getY();

audio video
MediaPlayer new MediaPlayer Idle
setDataSource();
prepare(); prepared
start(); seekTo(); pause();
getDuration(); getCurrentPostion();
isPlaying();
stop;

prepareAsync();
setOnpreparedListener
prepareCompleted

混合方式开启
播放暂停
进度展示/ 控制播放进度
SeekBar
setMax()

setProgress()

onBind() 1次
onServiceConnected(,IBinder)

视频的播放
SurfaceView MediaPlayer setDisplay(SurfaceHolder)

VideoView
setVideoPath
set

vitamio　VLC

1 fragment入门
通过xml文件声明fragment
0 创建Fragment的布局文件 xml
①创建一个类继承Fragment
②重写onCreateView方法 把fragment对应的xml布局文件加载进来 在这个方法中创建fragment的界面

```
public class FirstFragment extends Fragment {

    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container,
            Bundle savedInstanceState) {
        //把fragment 对应的xml布局文件文件 转换为View对象 加载到界面上
        View view = inflater.inflate(R.layout.fragment_first, null);
        return view;
    }
}
```

③ 在activity的布局文件中 声明一个fragment节点（fragment要小写 一定要声明一个id）

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="horizontal"
    android:layout_width="match_parent"
    android:layout_height="match_parent">
    <fragment android:name="com.itheima.fragmentdemo.FirstFragment"
            android:id="@+id/list"
            android:layout_weight="1"
            android:layout_width="0dp"
            android:layout_height="match_parent" />
```

```
    <fragment android:name="com.itheima.fragmentdemo.SecondFragment"
            android:id="@+id/viewer"
            android:layout_weight="2"
            android:layout_width="0dp"
            android:layout_height="match_parent" />
</LinearLayout>
```

2 动态替换fragment

android.R.id.content 每一个应用 android系统都会创建一个FrameLayout id是content 自己的界面是放到这个FrameLayout里面的

```
public class MainActivity extends Activity {

    @SuppressWarnings("deprecation")
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        int width = getWindowManager().getDefaultDisplay().getWidth();
        int height = getWindowManager().getDefaultDisplay().getHeight();
        //①获取fragmentmanager
        FragmentManager manager = getFragmentManager();
        //②开启fragment事务
        FragmentTransaction transaction = manager.beginTransaction();

        if(width>height){
            //横屏
            //③把fragment对象 替换到 viewgroup节点下
            //第一个参数 用来放置fragment的viewgroup的id
            //第二个参数 要显示的fragment对象
            transaction.replace(R.id.fragment_container, new SecondFragment());
        }else{
            //竖屏
            transaction.replace(R.id.fragment_container, new FirstFragment());
        }
        //④设置完对应的fragment一定要调用commit提交事务
        transaction.commit();
    }
}
```

3 使用fragment创建一个选项卡页面

4 使用fragment兼容低版本的写法

跟Fragement相关的api都要调用support.v4包中的

如果是新的项目 可以直接使用android.app.Fragment 没有必要考虑低版本

如果项目中之前用的就是support包的 为了保持一致 还是要使用兼容低版本的写法

```java
import android.os.Bundle;
import android.support.v4.app.FragmentActivity;
import android.support.v4.app.FragmentManager;
import android.support.v4.app.FragmentTransaction;

public class MainActivity extends FragmentActivity {
    //要动态加载v4包中的fragment 必须继承FragmentActivity

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        //获取fragmentManager
//        FragmentManager fragmentManager = getFragmentManager();
        //FragmentManager 也要使用v4包中的  v4包中 获取 FragmentManager 要用 getSupportFragmentManager
        // getSupportFragmentManager这个方法 实在v4包的 FragmentActivity中才可以使用
        FragmentManager supportFragmentManager = getSupportFragmentManager();
        //开始fragment事务
        FragmentTransaction transaction = supportFragmentManager.beginTransaction();
        //替换fragment
        transaction.replace(R.id.fragment_container, new DemoFragment());
        //提交事务
        transaction.commit();
    }
}
```

5 fragment的生命周期

6 fragment之间的通信

动态加载fragment的时候 使用三个参数的方法

```java
 //第三个参数 给fragment添加一个标签
  transaction.replace(R.id.ll_left, new LeftFragment(),"left");
```

通过这个标签就可以找到 对应的fragment

```java
//①获取activity  getActivity
//② 通过activity 获取fragmentManager
//③ 通过fragmentManager 调用 findFragmentByTag方法 找到对应的fragment对象
RightFragment right = (RightFragment) getActivity().getFragmentManager().findFragmentByTag("right");
//RightFragment rightFragment = new RightFragment();
right.changeText("hello");
```

7 menu菜单

```
    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        // Inflate the menu; this adds items to the action bar if it is present.
        getMenuInflater().inflate(R.menu.main, menu);
        return true;
    }

    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        //MenuItem就是被点击的菜单项 对象
        //可以获取菜单项的id  通过id判断哪个条目被点击了
        int id = item.getItemId();
        switch (id) {
        case R.id.action_settings:
            Toast.makeText(this, "最下面的条目被点击了", Toast.LENGTH_SHORT).show(
);
            break;
        case R.id.action_settings1:
            Toast.makeText(this, "中间的条目被点击了", Toast.LENGTH_SHORT).show();
            break;
        case R.id.action_settings2:
            Toast.makeText(this, "最上面的条目被点击了", Toast.LENGTH_SHORT).show(
);
            break;

        default:
            break;
        }
        return super.onOptionsItemSelected(item);
    }

    @Override
    public boolean onMenuOpened(int featureId, Menu menu) {
        //当用户点击菜单按钮就会执行这个方法
        //如果返回值是true 那么会走系统默认的实现 执行onCreateOptionsMenu
        //如果想实现一个自定义菜单效果 可以在这个方法中处理 并且让这个方法返回fals
e
        AlertDialog.Builder builder = new Builder(this);
        builder.setTitle("这是一个菜单");
        builder.setMessage("菜单的内容");
        builder.setPositiveButton("确定", new OnClickListener() {

            @Override
            public void onClick(DialogInterface dialog, int which) {
                Toast.makeText(getApplicationContext(), "菜单确定", Toast.LENGTH_S
HORT).show();
            }
        });
        builder.show();
        return false;
    }
```

8 AutoCompleteTextView控件的使用

```
public class MainActivity extends Activity {
    private String[] names = {"laowang","laozhang","laoli","xiaowang","xiaozhang","xiaoli"};
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
      AutoCompleteTextView actv = (AutoCompleteTextView) findViewById(R.id.actv_text);
      //给自动补全的textview设置一个数据适配器 这个适配器用来提供显示下拉列表的内容
      ArrayAdapter<String> adapter = new ArrayAdapter<String>(getApplicationContext(),R.layout.item, names);
      //设置适配器
      actv.setAdapter(adapter);
    }
```

布局文件中声明对应节点

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    tools:context=".MainActivity" >

    <AutoCompleteTextView
        android:id="@+id/actv_text"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:completionThreshold="1"
        android:hint="请输入内容" />

</RelativeLayout>
```

completionThreshold 通过这个属性来指定 输入多少个字符会出现提示

12 帧动画

①在res目录下创建一个drawable文件夹
要展示的动画的图片资源都放到这个目录下
在drawable目录下声明一个xml文件

```xml
<?xml version="1.0" encoding="utf-8"?>
<animation-list xmlns:android="http://schemas.android.com/apk/res/android"
    android:oneshot="false">
    <item android:drawable="@drawable/girl_1" android:duration="200" />
    <item android:drawable="@drawable/girl_2" android:duration="200" />
    <item android:drawable="@drawable/girl_3" android:duration="200" />
    <item android:drawable="@drawable/girl_4" android:duration="200" />
    <item android:drawable="@drawable/girl_5" android:duration="200" />
    <item android:drawable="@drawable/girl_6" android:duration="400" />
    <item android:drawable="@drawable/girl_5" android:duration="200" />
    <item android:drawable="@drawable/girl_6" android:duration="400" />
    <item android:drawable="@drawable/girl_5" android:duration="200" />
    <item android:drawable="@drawable/girl_6" android:duration="400" />
    <item android:drawable="@drawable/girl_7" android:duration="400" />
    <item android:drawable="@drawable/girl_8" android:duration="400" />
    <item android:drawable="@drawable/girl_9" android:duration="200" />
    <item android:drawable="@drawable/girl_10" android:duration="200" />
    <item android:drawable="@drawable/girl_11" android:duration="200" />
</animation-list>
```

根元素 animation-list 可以声明一个属性 oneshot 如果设置为true 动画只执行一次 如果是false会重复执行 (默认是false)
里面每一个item 对应一帧动画的资源 drawable指定图片资源 duration 指定每一帧动画播放的时长

用一个ImageView 可以把动画设置为imageview的background

```java
public class MainActivity extends Activity {

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

        //找到图片背景对应的Drawable对象 强制转换为AnimationDrawable
        AnimationDrawable animation = (AnimationDrawable) iv_image.getBackground()
;
        //调用start方法开始动画
        animation.start();
    }
```

9补间动画

10使用xml方式定义补间动画
res目录下 创建一个目录 anim

```xml
<?xml version="1.0" encoding="utf-8"?>
<alpha
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:fromAlpha="1"
    android:toAlpha="0"
    android:duration="1000"
    android:repeatMode="reverse"
    android:repeatCount="1"
    >
</alpha>
```

旋转 如果pivotX 是针对父容器 `android:pivotX="50%p"` 如果没有P 就是针对自己的

```xml
<rotate
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:fromDegrees="0"
    android:toDegrees="180"
    android:pivotX="50%"
    android:pivotY="50%"
    android:duration="1000"
    android:repeatMode="restart"
    android:repeatCount="1"
    android:fillAfter="true">
</rotate>
```

缩放

```xml
<?xml version="1.0" encoding="utf-8"?>
<scale
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:fromXScale="1"
    android:toXScale="2"
    android:fromYScale="1"
    android:toYScale="2"
    android:pivotX="50%"
    android:pivotY="50%"
    android:duration="1000"
    android:fillAfter="true">


</scale>
```

平移

```xml
<?xml version="1.0" encoding="utf-8"?>
<translate
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:fromXDelta="10"
    android:toXDelta="100"
    android:fromYDelta="10"
    android:toYDelta="100"
    android:duration="1000"
```

```
    android:interpolator="@android:anim/bounce_interpolator"
    android:fillAfter="true"
    >
</translate>
```

集合

```
<?xml version="1.0" encoding="utf-8"?>
<set
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:shareInterpolator="false">
    <scale
        android:interpolator="@android:anim/accelerate_decelerate_interpolator"
        android:fromXScale="1.0"
        android:toXScale="1.4"
        android:fromYScale="1.0"
        android:toYScale="0.6"
        android:pivotX="50%"
        android:pivotY="50%"
        android:fillAfter="false"
        android:duration="700" />
    <set android:interpolator="@android:anim/decelerate_interpolator">
        <scale
           android:fromXScale="1.4"
           android:toXScale="0.0"
           android:fromYScale="0.6"
           android:toYScale="0.0"
           android:pivotX="50%"
           android:pivotY="50%"
           android:startOffset="700"
           android:duration="400"
           android:fillBefore="false" />
        <rotate
           android:fromDegrees="0"
           android:toDegrees="-45"
           android:toYScale="0.0"
           android:pivotX="50%"
           android:pivotY="50%"
           android:startOffset="700"
           android:duration="400" />
    </set>

</set>
```

加载xml文件形式定义的动画

AnimationUtils.LoadAnimation方法

11 属性动画

xml文件声明属性动画

res目录下 创建一个animator目录

```
<?xml version="1.0" encoding="utf-8"?>
<animator xmlns:android="http://schemas.android.com/apk/res/android" >
    <objectAnimator
        android:propertyName="translationX"
        android:duration="2000"
        android:valueFrom="10"
        android:valueTo="100"
        ></objectAnimator>

</animator>
```

属性动画(在3.0之后加入的api)和View动画区别
view动画 不会改变View的属性
属性动画 会改变view的属性

属性动画用到的Api

```
ObjectAnimator oa = ObjectAnimator.ofFloat(iv, "translationX", 10, 50,20,100,150);
```

```
//位移动画
    public void translate(View v){
        //创建属性动画
        /**
         * target 执行的目标
         * propertyName 属性名字  The name of the property being animated.
         * float... values 可变参数
         */
        ObjectAnimator oa = ObjectAnimator.ofFloat(iv, "translationX", 10, 50,20,100,150);
        oa.setDuration(2000);
        oa.start(); //开始动画

    }
    //缩放动画
    public void scale(View v){
        ObjectAnimator oa = ObjectAnimator.ofFloat(iv, "scaleY", 0.1f, 2, 1, 2);
        oa.setDuration(2000);
        oa.start();
    }

    //实现透明的效果
    public void alpha(View v){
        ObjectAnimator oa = ObjectAnimator.ofFloat(iv, "alpha", 0, 0.5f, 0, 1,0,1);
        oa.setDuration(2000);
        oa.start();
    }

```

```
    //实现旋转的效果
    public void rotate(View v){
        ObjectAnimator oa = ObjectAnimator.ofFloat(iv, "rotation", 0, 180, 90, 360);
//      ObjectAnimator oa = ObjectAnimator.ofFloat(iv, "rotationY", 0, 180, 90, 360);
        oa.setDuration(2000);
        oa.start();
    }


    //一起飞
    public void fly(View v){
        AnimatorSet as = new AnimatorSet();
        ObjectAnimator oa = ObjectAnimator.ofFloat(iv, "translationX", 10, 50, 20, 100);
        ObjectAnimator oa2 = ObjectAnimator.ofFloat(iv, "scaleY", 0.1f, 2, 1, 2);
        ObjectAnimator oa3 = ObjectAnimator.ofFloat(iv, "alpha", 0, 0.5f, 0, 1);
        ObjectAnimator oa4 = ObjectAnimator.ofFloat(iv, "rotationY", 0, 180, 90, 360);
        as.setDuration(2000);
        as.setTarget(iv);
        //往集合中添加动画
        //挨个飞
        //as.playSequentially(oa, oa2, oa3, oa4);
        //一起飞
        as.playTogether(oa, oa2, oa3, oa4);
        as.start();
    }
```

13 通知栏介绍

Notification 通知

Notification.Builder 设置通知的内容使用builder

NotificationMananger 通过这个api发送一个通知

开发步骤①获取Notification.Builder

② 通过Notification.Builder 设置通知相关参数

③Notification.Builder.build()创建一个notification对象

④ 获取NotificationMananger 对象 使用NotificationMananger 发送通知 notify

```
 public void send(View v){
        //通过Notification.Builder 创建一个notification
        Notification.Builder builder = new Builder(this);
        //通知第一次收到的时候 会在状态栏中显示文字 这个文字就是通过setTicker设置的
        builder.setTicker("账户异动通知:您的账户 涉嫌洗钱操作,已经被检方冻结,解冻事宜请与王警官联系,电话号码13987654321");
        //设置当前的通知 用户点击之后就消失掉
        builder.setAutoCancel(true);
```

```
        //设置在通知栏中显示的大标题
        builder.setContentTitle("账户异动通知");
        //设置在通知栏中显示的小的文字内容
        builder.setContentText("您的账户 涉嫌洗钱操作,已经被检方冻结,解冻事宜请与王警官联系,电话号码13987654321");
        //设置在状态栏显示的小图标
        builder.setSmallIcon(R.drawable.ic_launcher);
        //builder.setOngoing(true);
        Intent intent2 = new Intent(this,MainActivity.class);
        //pendingIntent 延迟执行的意图 当通知被点击的时候 就会执行intent
        //如果想操作activity getActivity
        //如果想操作service getService
        //第二个参数 请求码  用来区分不同的pendingIntent
        //第三个参数 意图
        //第四个参数  传FLAG_UPDATE_CURRENT 更新pendingintent
        PendingIntent intent = PendingIntent.getActivity(getApplicationContext(), 1, intent2, PendingIntent.FLAG_UPDATE_CURRENT);
        builder.setContentIntent(intent);
        Notification notification = builder.build();
        //获取notificationManager
        NotificationManager mananger = (NotificationManager) getSystemService(NOTIFICATION_SERVICE);
        //调用这个方法 通知会发送到通知栏中
        //第一个参数 通知的id
        mananger.notify(1, notification);
        mananger.cancel(1);
    }
```

Fragment ☆☆☆☆☆

帧动画 view动画 属性动画
xml文件定义动画 ☆☆☆☆

notification ☆☆☆☆

menu/autocompletetextview ☆☆☆

14 应用反编译