回顾多媒体
加载图片
①大图加载
OutOfMemory 内存溢出 Error OOM
图片分辨率比较高 那么要压缩加载
BitmapFactory.Options
Options.inSampleSize 压缩的比例(主要思路 计算合适的压缩比例)
Options.inJustDecodeBounds 只把图片的边界解码出来

跟手机屏幕的宽高进行比较
1.1获取屏幕的宽高
WindowManager getWindowManager.getDefaultDisplay.getwidth
getWindowManager().getDefaultDisplay().getSize(Point);
1.2获取图片宽高
Options.inJustDecodeBounds = true;
BitmapFactory.decodeXXXX(图片对应的路径/位置 , options);
options.outWidth();
1.3确定inSampleSize 把Options.inJustDecodeBounds = false;
再加载图片

试验inSampleSize
try
catch(Error)
② 创建图片副本并且修改
2.1 创建一个空的bitmap对象 bitmap.createBitmap()
2.2 利用空的bitmap创建一个canvas
2.3 通过canvas.drawbitmap(原图,Matrix,Paint);
2.4 通过imageview展示图片的副本

如果想对图片进行 rotate 旋转 scale 缩放 translate平移 处理
Matrix setRotate setScale setTranslate
        postRotate postScale postTranslate

画画板 撕衣服
setOnTouchListener
MotionEvent.Action_down Action_move Action_up
返回值true

音频播放 Mediaplayer
复习混合方式开启服务 并且跟activity配合
handler.sendEmptyMessageDelayed
handler.sendEmptyMessageAtTime(what,millisec)

startservice
bindservice

onserviceconneted 可以获取service内部类的Binder对象 就可获取音乐播放的状态

MusicItem
public string name;

public string artist;
public string path;

setonItemclicklistener{
Intent intent = new Intent(this,Service.class);
intent.putextra("name",)
startService(intent);
}

onbind(Intent intent)
oncreate()
onstartcommand(Intent intent)
Mediaplayer.reset(); //重新设置Mediaplayer
MedisPlayer.setDataSouce(intent.getStringExtra)
Mediaplayer.prepare;
Mediaplayer.start();

视频播放 VideoView (Vitamio)
继承了SurfaceView 封装了MediaPlayer

1 fragment入门
①创建一个java类 继承Fragment重写oncreateView方法

```
public class FirstFragment extends Fragment {

    //当activity创建的时候 需要把界面展示出来 就会走onCreateView方法
    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container,
            Bundle savedInstanceState) {
        //通过传进来的LayoutInflater 把一个xml文件转换成view对象 做为onCreateView方法的返回值
        View view = inflater.inflate(R.layout.fragment_first, null);
        TextView tv_text = (TextView) view.findViewById(R.id.tv_text);
        return view;
    }
}
```

fragment 也会对应一个布局文件 这个布局文件可以通过传入的LayoutInflater加载到界面上 作为oncreateview的返回值

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="horizontal"
    android:layout_width="match_parent"
    android:layout_height="match_parent">
   <TextView
       android:id="@+id/tv_text"
       android:layout_width="wrap_content"
       android:layout_height="wrap_content"
       android:text="第一个fragment"/>
</LinearLayout>
```

②在布局文件中声明相应的fragment节点

```
<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="cn.itcast.fragmenthello.FirstFragment"
            android:id="@+id/list"
            android:layout_weight="1"
            android:layout_width="0dp"
            android:layout_height="match_parent" />
    <fragment android:name="cn.itcast.fragmenthello.SecondFragment"
            android:id="@+id/viewer"
            android:layout_weight="1"
            android:layout_width="0dp"
            android:layout_height="match_parent" />
</LinearLayout>
```

注意在声明fragment节点时 fragment首字母小写 name 必须制定 内容是要显示的Fragment的全类名

2 动态替换fragment

2.1写一个类继承fragment 重写oncreateView方法 把布局文件加载进来
2.2在activity当中 获取fragmentManager 通过 fragmentManager 开启fragmentTransaction
通过fragmentTransaction 调用replace(替换的意思)方法 替换fragment
replace方法 第一个参数 viewGroup(LinearLayout Relativelayout Framelayout)的id 就是fragment要显示到哪一个布局之中 传入这个布局的id
addview
removeAllView
getChildAt()
getChildCount()
第一个参数 可以使用android.R.id.content(android系统在创建布局的时候 先加入进来的根布局id 每一个应用都有)
也可以使用自己应用中声明的布局id
第二个参数 fragment对象
2.3 调用fragmentTransaction.commit() 提交 只有commit()之后 fragment才会真正的显示出来

3 使用fragment创建一个选项卡页面

模拟微信界面

```
public class MainActivity extends Activity implements OnClickListener {

    private ImageView iv_weixin;
    private ImageView iv_contact;
    private ImageView iv_find;
    private ImageView iv_me;

    private WeixinFragment weixinFragment;
    private ContactFragment contactFragment;
    private FindFragment findFragment;
    private MeFragment meFragment;

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

        iv_weixin.setOnClickListener(this);
```

```
        iv_contact.setOnClickListener(this);
        iv_find.setOnClickListener(this);
        iv_me.setOnClickListener(this);

        //执行点击weixin图标
        iv_weixin.performClick();

    }

    @Override
    public void onClick(View v) {
        //清除所有按钮的点击状态恢复到没有点击的情况
        clearImageState();
        //获取fragmentManager
        FragmentManager manager = getFragmentManager();
        //开启fragmentTransaction
        FragmentTransaction transaction = manager.beginTransaction();
        switch (v.getId()) {
        case R.id.iv_weixin:
            if(weixinFragment == null){
                weixinFragment= new WeixinFragment();
            }
            transaction.replace(R.id.rl_container, weixinFragment);
            //处理按钮的效果
            iv_weixin.setImageResource(R.drawable.weixin_pressed);
            break;
        case R.id.iv_contact:
            if(contactFragment == null){
                contactFragment= new ContactFragment();
            }
            transaction.replace(R.id.rl_container, contactFragment);
            //处理按钮的效果
            iv_contact.setImageResource(R.drawable.contact_list_pressed);
            break;

        case R.id.iv_find:
            if(findFragment == null){
                findFragment = new FindFragment();
            }
            transaction.replace(R.id.rl_container, findFragment);
            iv_find.setImageResource(R.drawable.find_pressed);
            break;
        case R.id.iv_me:
            if(meFragment == null){
                meFragment = new MeFragment();
            }
            transaction.replace(R.id.rl_container, meFragment);
            iv_me.setImageResource(R.drawable.profile_pressed);
            break;
        }
        //提交fragment事务
        transaction.commit();
    }

    public void clearImageState(){
        iv_contact.setImageResource(R.drawable.contact_list_normal);
        iv_find.setImageResource(R.drawable.find_normal);
        iv_me.setImageResource(R.drawable.profile_normal);
        iv_weixin.setImageResource(R.drawable.weixin_normal);
    }
}
```

Fragment

```
public class WeixinFragment extends Fragment {

    @Override
```

```
    public View onCreateView(LayoutInflater inflater, ViewGroup container,
            Bundle savedInstanceState) {
        View view = inflater.inflate(R.layout.fragment_weixin, null);
        getActivity();
        return view;
    }
}
```

activity布局

```
<LinearLayout 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:orientation="vertical"
    tools:context=".MainActivity" >

    <RelativeLayout
        android:id="@+id/rl_container"
        android:layout_width="match_parent"
        android:layout_height="0dp"
        android:layout_weight="1"
         ></RelativeLayout>
<LinearLayout
    android:orientation="horizontal"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:gravity="center_vertical"
    android:background="@drawable/menu_bg">
    <ImageView
        android:id="@+id/iv_weixin"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_weight="1"
        android:src="@drawable/weixin_pressed"/>
     <ImageView
        android:id="@+id/iv_contact"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_weight="1"
        android:src="@drawable/contact_list_normal"/>
      <ImageView
        android:id="@+id/iv_find"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_weight="1"
        android:src="@drawable/find_normal"/>
       <ImageView
        android:id="@+id/iv_me"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_weight="1"
        android:src="@drawable/profile_normal"/>
</LinearLayout>
</LinearLayout>
```

4 使用fragment兼容低版本的写法

所有的跟fragement相关的api

Fragment

FragmentManager v4包中获取FragmentManager 要使用`getSupportFragmentManager` 这个方法在`FragmantActivity`中才有

FragmentTransaction
都要使用android.support.v4包中的相关类

5 fragment的生命周期

oncreate
当fragment被创建的时候 调用oncreate
oncreateView
在这个方法当中加载fragment的布局文件
onpause
onStop
处理界面需要显示的东西 状态的保存

ondestory
资源的释放

6 fragment之间的通信

一个fragment中如何获取另外一个fragment对象
每个fragment都是加载在activity当中的
getActivity() 可以获取到这个公共的activity
activity可以获取到fragmentManager
fragmentManager有一个方法 findFragmentByTag(String tag);
transaction.replace(要显示fragment的ViewGroup的ID, fragment对象, tag);
通过第三个参数可以给fragment起一个名字
Activity代码

```
1.  public class MainActivity extends Activity {
2.
3.      @Override
4.      protected void onCreate(Bundle savedInstanceState) {
5.          super.onCreate(savedInstanceState);
6.          setContentView(R.layout.activity_main);
7.          //获取fragmentmanger
8.          FragmentManager manager = getFragmentManager();
9.          //开启事务
10.         FragmentTransaction transaction = manager.beginTransaction();
11.         //通过事务替换fragmetn
12.         transaction.replace(R.id.ll_left, new FirstFragment(),"first");
13.         transaction.replace(R.id.ll_right, new SecondFragment(),"second");
14.         transaction.commit();
15.     }
16. }
```

Fragment

```
1.  public class FirstFragment extends Fragment {
2.
3.      //当activity创建的时候 需要把界面展示出来 就会走onCreateView方法
4.      @Override
5.      public View onCreateView(LayoutInflater inflater, ViewGroup container,
6.              Bundle savedInstanceState) {
7.          //通过传进来的LayoutInflater 把一个xml文件转换成view对象 做为onCreateView方法的返回值
8.          View view = inflater.inflate(R.layout.fragment_first, null);
9.          Button btn_test = (Button) view.findViewById(R.id.btn_test);
10.         btn_test.setOnClickListener(new OnClickListener() {
11.
12.             @Override
```

```
            public void onClick(View v) {
                SecondFragment second = (SecondFragment) getActivity().getFragmentManager().findFragmentBy
Tag("second");
                //SecondFragment fragment = new SecondFragment();
                second.changeText("hahah");

            }
        });
        return view;
    }
}
```

```
public class SecondFragment extends Fragment {

    private TextView tv_text;

    //当activity创建的时候 需要把界面展示出来 就会走onCreateView方法
    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container,
            Bundle savedInstanceState) {
        //通过传进来的LayoutInflater 把一个xml文件转换成view对象 做为onCreateView方法的返回值
        View view = inflater.inflate(R.layout.fragment_second, null);
        tv_text = (TextView) view.findViewById(R.id.tv_text2);
        return view;
    }

    public void changeText(String s){
        tv_text.setText(s);
    }
}
```

7 menu菜单

```
@Override
    public boolean onCreateOptionsMenu(Menu menu) {
        getMenuInflater().inflate(R.menu.main, menu);
        return true;
    }

    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        int id = item.getItemId();
        switch (id) {
        case R.id.action_settings:
            Toast.makeText(getApplicationContext(), "setting", Toast.LENGTH_SHORT).show();
            break;
        case R.id.action_settings1:
            Toast.makeText(getApplicationContext(), "选项被点中", Toast.LENGTH_SHORT).show();
            break;
        case R.id.action_settings2:
            Toast.makeText(getApplicationContext(), "选项1被点中", Toast.LENGTH_SHORT).show();
            break;

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

    @Override
    public boolean onMenuOpened(int featureId, Menu menu) {
        AlertDialog.Builder builder= new Builder(this);
        builder.setTitle("警告");
```

```
        builder.setMessage("message");
        builder.setPositiveButton("确定", new OnClickListener() {

            @Override
            public void onClick(DialogInterface dialog, int which) {
                System.out.println("确定");
            }
        });
        builder.show();
        return false;
    }
```

`onCreateOptionsMenu` 创建系统默认的菜单时会走这个方法

`onOptionsItemSelected` 具体某个菜单项被选中

`onMenuOpened` 当按下menu键会先走这个方法 如果这个方法返回true就会调用系统默认的菜单 如果返回false不会调用系统默认的菜单 可以在onMenuOpened方法里弹出自定义的菜单

8 AutoCompleteTextView控件的使用

在布局文件中声明AutoCompleteTextView

```
<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="请输入姓名" />

</RelativeLayout>
```

completionThreshold 这个属性 指定了用户输入几个字母才会弹出提示

在java代码中找到AutoCompleteTextView 给他设置一个适配器, 通过适配器用下拉框的形式展示要显示的提示信息

```
public class MainActivity extends Activity {
    private String[] names = {"laozhang","laoli","laowang","xiaoqiang","xiaomao"};
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        AutoCompleteTextView actv = (AutoCompleteTextView) findViewById(R.id.actv_text);
        actv.setAdapter(new ArrayAdapter<String>(getApplicationContext(), android.R.layout.simple_list_ite
m_1, names));
    }


}
```

andriod中动画

PropertyAnimation 属性动画

ViewAnimation view动画/补间动画

DrawableAnimation 帧动画

9补间动画

View动画 只是在视觉上看到view的位置变化了 但是View的实际属性没有任何改变

```
public class MainActivity extends Activity {

    private ImageView iv_icon;

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

            @Override
            public void onClick(View v) {
                Toast.makeText(getApplicationContext(), "点中了", Toast.LENGTH_SHORT).show();
            }
        });
    }

    public void alpha(View v) {
        //AlphaAnimation 透明度动画 创建透明度动画传递两个参数
        //第一个参数 动画开始的时候的透明度值
        //第二个参数 动画结束的时候的透明度值   注意 取值范围 0~1.0f 0代表完全透明 1 代表完全不透明
//      AlphaAnimation animation = new AlphaAnimation(1.0f, 0.0f);
//      //指定动画执行的时间
//      animation.setDuration(1000);
//      //设置重复的次数
//      animation.setRepeatCount(2);
//      //设置重复的模式 restart重新开始  reverse反着来一次 只有设置了重复的次数>=1时重复的模式才有效果
////        animation.setRepeatMode(Animation.REVERSE);
//      //这个参数设置为true 动画结束后会停在结束状态 否则会回到初始状态
//      animation.setFillAfter(true);
        Animation animation = AnimationUtils.loadAnimation(getApplicationContext(), R.anim.alpha_animation
);
        //用imageview执行动画
        iv_icon.startAnimation(animation);


    }

    public void rotate(View v) {
        //创建RotateAnimation 第一个参数 开始旋转的度数 第二个参数 旋转到的度数
//      RotateAnimation animation = new RotateAnimation(0, 360);
//      //第一个参数 开始旋转的度数 第二个参数 旋转到的度数
//      //第三个参数 旋转中心点x坐标的类型  Animation.RELATIVE_TO_PARENT 相对父容器  Animation.RELATIVE_TO
_SELF 相对自己
//      //第四个参数 旋转中心点x坐标的值 0~1.0
//      //第五个参数 旋转中心点y坐标的类型  Animation.RELATIVE_TO_PARENT 相对父容器  Animation.RELATIVE_TO
_SELF 相对自己
//      //第六个参数 旋转中心点y坐标的值 0~1.0
//      animation = new RotateAnimation(0, 360, Animation.RELATIVE_TO_SELF, 0.5f, Animation.RELATIVE_TO_SE
LF, 0.5f);
//      //设置动画时长
//      animation.setDuration(1000);
//      //设置重复次数
//      animation.setRepeatCount(1);
//      //设置重复模式
//      animation.setRepeatMode(Animation.REVERSE);
//      //设置是否停在结束状态
//      animation.setFillAfter(true);
        Animation animation = AnimationUtils.loadAnimation(getApplicationContext(), R.anim.rotate);
        //开始动画
        iv_icon.startAnimation(animation);

    }

```

```
62.     public void scale(View v) {
63.         //第一个参数 动画开始时x方向大小
64.         //第二个参数 动画结束时x方向大小
65.         //第三个参数 动画开始时y方向大小
66.         //第四个参数 动画结束时y方向大小
67.         ScaleAnimation animation = new ScaleAnimation(1.0f, 0.2f, 1.0f, 0.2f);
68.         //第一个参数 动画开始时x方向大小
69.         //第二个参数 动画结束时x方向大小
70.         //第三个参数 动画开始时y方向大小
71.         //第四个参数 动画结束时y方向大小
72.         //后四个参数指定了缩放中心点的坐标
73.         //第五个参数 中心点x坐标类型
74.         //第六个参数 中心点x坐标具体的值
75.         //第七个参数 中心点y坐标类型
76.         //第八个参数 中心点y坐标具体的值
77.         animation = new ScaleAnimation(1.0f, 2.0f, 1.0f, 2.0f, Animation.RELATIVE_TO_PARENT, 0.5f, Animati
    on.RELATIVE_TO_PARENT, 0.5f);
78.         animation.setDuration(2000);
79.         animation.setRepeatCount(1);
80.         animation.setFillAfter(true);
81.
82.         Animation animation2 = AnimationUtils.loadAnimation(getApplicationContext(), R.anim.scale);
83.         iv_icon.startAnimation(animation2);
84.
85.     }
86.
87.     public void translate(View v) {
88.         //前两个参数 指定 X方向平移开始位置和结束位置
89.         //后两个参数 指定Y方向平移开始位置和结束位置
90.         TranslateAnimation animation = new TranslateAnimation(0f, 100f, 0f, 100f);
91.         //前四个参数 指定 X方向平移开始位置和结束位置   类型 开始值 类型 结束值
92.         //后四个参数  指定Y方向平移开始位置和结束位置 类型 开始值 类型 结束值
93.         animation = new TranslateAnimation(Animation.RELATIVE_TO_PARENT, -0.5f, Animation.RELATIVE_TO_PARE
    NT, 0.5f, Animation.RELATIVE_TO_PARENT, -0.5f, Animation.RELATIVE_TO_PARENT, 0.5f);
94.         animation.setDuration(2000);
95.         animation.setRepeatCount(1);
96.         animation.setRepeatMode(Animation.REVERSE);
97.         //添加动画的插入器 指定一个特效
98.         animation.setInterpolator(new AccelerateDecelerateInterpolator());
99.         Animation animation2 = AnimationUtils.loadAnimation(getApplicationContext(), R.anim.translate);
100.        iv_icon.startAnimation(animation2);
101.
102.    }
103.
104.    public void combine(View v) {
105.        //创建一个动画集合对象
106.        AnimationSet set = new AnimationSet(false);
107.
108.        AlphaAnimation alpha = new AlphaAnimation(1.0f, 0.0f);
109.        //指定动画执行的时间
110.        alpha.setDuration(1000);
111.        //设置重复的次数
112.        alpha.setRepeatCount(2);
113.        //设置重复的模式 restart重新开始  reverse反着来一次 只有设置了重复的次数>=1时重复的模式才有效果
114.//      animation.setRepeatMode(Animation.REVERSE);
115.        //这个参数设置为true 动画结束后会停在结束状态 否则会回到初始状态
116.        alpha.setFillAfter(true);
117.        //创建一个动画就把动画添加到动画集合
118.        set.addAnimation(alpha);
119.
120.        RotateAnimation rotate = new RotateAnimation(0, 360);
121.        //第一个参数 开始旋转的度数 第二个参数 旋转到的度数
122.        //第三个参数 旋转中心点x坐标的类型  Animation.RELATIVE_TO_PARENT 相对父容器  Animation.RELATIVE_TO
    _SELF 相对自己
123.        //第四个参数 旋转中心点x坐标的值 0~1.0
124.        //第五个参数 旋转中心点y坐标的类型  Animation.RELATIVE_TO_PARENT 相对父容器  Animation.RELATIVE_TO
    _SELF 相对自己
125.        //第六个参数 旋转中心点y坐标的值 0~1.0
```

```
        rotate = new RotateAnimation(0, 360, Animation.RELATIVE_TO_SELF, 0.5f, Animation.RELATIVE_TO_SELF, 0.5f);
        //设置动画时长
        rotate.setDuration(1000);
        //设置重复次数
        rotate.setRepeatCount(1);
        //设置重复模式
        rotate.setRepeatMode(Animation.REVERSE);
        //设置是否停在结束状态
        rotate.setFillAfter(true);
        //创建一个动画就把动画添加到动画集合
        set.addAnimation(rotate);

        ScaleAnimation scale = new ScaleAnimation(1.0f, 2.0f, 1.0f, 2.0f, Animation.RELATIVE_TO_PARENT, 0.5f, Animation.RELATIVE_TO_PARENT, 0.5f);
        scale.setDuration(2000);
        scale.setRepeatCount(1);
        scale.setFillAfter(true);
        //创建一个动画就把动画添加到动画集合
        set.addAnimation(scale);

        TranslateAnimation animation = new TranslateAnimation(0f, 100f, 0f, 100f);
        //前四个参数 指定 X方向平移开始位置和结束位置    类型 开始值 类型 结束值
        //后四个参数  指定Y方向平移开始位置和结束位置 类型 开始值 类型 结束值
        animation = new TranslateAnimation(Animation.RELATIVE_TO_PARENT, -0.5f, Animation.RELATIVE_TO_PARENT, 0.5f, Animation.RELATIVE_TO_PARENT, -0.5f, Animation.RELATIVE_TO_PARENT, 0.5f);
        animation.setDuration(2000);
        animation.setRepeatCount(1);
        animation.setRepeatMode(Animation.REVERSE);
        //添加动画的插入器 指定一个特效
        animation.setInterpolator(new AccelerateDecelerateInterpolator());
        //创建一个动画就把动画添加到动画集合
        set.addAnimation(animation);
        Animation animation2 = AnimationUtils.loadAnimation(getApplicationContext(), R.anim.animationset);
        //让imageview指定动画集合 就会把动画组合执行起来
        iv_icon.startAnimation(animation2);

    }

}
```

10使用xml方式定义补间动画

alpha

```
<?xml version="1.0" encoding="utf-8"?>
<alpha
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:fromAlpha="1.0"
    android:toAlpha="0.0"
    android:fillAfter="true"
    android:duration="1000"
    android:repeatCount="1"
    android:repeatMode="reverse">
</alpha>
```

rotate

```
<?xml version="1.0" encoding="utf-8"?>
<rotate
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:fromDegrees="0"
    android:toDegrees="360"
    android:pivotX="50%p"
```

```
    android:pivotY="50%p"
    android:duration="2000"
    android:repeatCount="1"
    android:repeatMode="reverse">
</rotate>
```

scale

```
<?xml version="1.0" encoding="utf-8"?>
<scale
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:fromXScale="1.0"
    android:toXScale="2.0"
    android:fromYScale="1.0"
    android:toYScale="2.0"
    android:pivotX="50%"
    android:pivotY="50%"
    android:repeatCount="1"
    android:duration="1000">
</scale>
```

translate

```
<?xml version="1.0" encoding="utf-8"?>
<translate
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:fromXDelta="0"
    android:toXDelta="30"
    android:fromYDelta="0"
    android:toYDelta="30"

    android:fillAfter="true"
    android:duration="1000"
    android:repeatCount="1">
</translate>
```

组合动画

```
<?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>
```

11 属性动画objectAnimator

属性动画会修改view的具体属性 经过属性动画的操作 view的属性确实发生改变

12 帧动画

在res目录下创建一个drawable目录 在drawable目录中创建xml文件 要把显示的动画图片也放到drawable目录中
在xml文件中声明<animation-list>节点

```
<?xml version="1.0" encoding="utf-8"?>
<animation-list xmlns:android="http://schemas.android.com/apk/res/android"
    android:oneshot="true" >
     <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="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="200" />
      <item android:drawable="@drawable/girl_4" android:duration="800" />
    <item android:drawable="@drawable/girl_5" android:duration="800" />
     <item android:drawable="@drawable/girl_6" android:duration="800" />
    <item android:drawable="@drawable/girl_7" android:duration="200" />
     <item android:drawable="@drawable/girl_8" android:duration="200" />
    <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>
```

<item>节点 对应一张图片 drawable属性 指定具体的资源id duration 某一张图片显示多长时间(单位毫秒)
<animation-list>可以指定onshot属性 如果设置为true 那么动画只执行一次 默认值是false 会反复执行

使用animation-list 要在布局文件中声明ImageView节点 在ImageView中加载对应的动画xml

```
<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" >

    <ImageView
        android:id="@+id/iv_animation"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:background="@drawable/animation" />

</RelativeLayout>
```

在java代码中 找到imageview 找到imageview加载的动画对象 AnimationDrawable 调用start方法开启动画

```
public class MainActivity extends Activity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        ImageView image = (ImageView) findViewById(R.id.iv_animation);
        AnimationDrawable animation = (AnimationDrawable) image.getBackground();
        animation.start();
    }

}
```

13 通知栏介绍
14 应用反编译

四大组件 应用的基石
Activity 生命周期方法 Fragment oncreate onstart onresume onpause onstop ondestroy
Service oncreate onbind/onstartcommand ondestroy Activity onserviceConnected onserviceDisConnected
BroadCastReceiver onReceive
ContentProvider onCreate

View相关 android中所有控件的父类
View Button TextView EditText ImageView CheckBox SeekBar ProgressBar
ViewGroup LinearLayout RelativeLayout FrameLayout
setOnclickListener
setonTouchListener
ListView Adatper getcount getItem getItemId getView
setOnItemclickListener

文件存储
sharedPreferences
Context.openFileInput
context.openFileOutPut
Enviroment.getExternalStorageDirectory();

存到数据库中
SQLiteOpenHelper
SQliteDatabase

访问网络 TCP/IP HTTP协议 提交7种 常用两种 get /post 响应码 200 206 302 304 404 500 range头
HttpUrlConnetion
联网必须开线程
子线程更新UI要使用Handler
多线程下载的原理 服务端支持range头 可以把数据分成多个部分返回 客户端 RandomAccessFile seek()

加载图片处理多媒体

加载大图压缩
创建图片的副本
修改图片的内容
图片相关的动画

今日重点 上午的Fragment ①微信demo ② 两个fragment调用方法
　　下午 通过xml方式定义动画