

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

setOnTouchListener(OnTouchListener)

getX() getY();

audio video

MediaPlayer new MediaPlayer Idle

setDataSource();

prepare(); prepared

start(); seekTo(); pause();

getDuration(); getCurrentPosition();

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

```
1. public class FirstFragment extends Fragment {  
2.  
3.     @Override  
4.     public View onCreateView(LayoutInflater inflater, ViewGroup container,  
5.         Bundle savedInstanceState) {  
6.         //把fragment 对应的xml布局文件文件 转换为View对象 加载到界面上  
7.         View view = inflater.inflate(R.layout.fragment_first, null);  
8.         return view;  
9.     }  
10. }
```

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)

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

```

10.     <fragment android:name="com.itheima.fragmentdemo.SecondFragment"
11.             android:id="@+id/viewer"
12.             android:layout_weight="2"
13.             android:layout_width="0dp"
14.             android:layout_height="match_parent" />
15. </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

```

1.  public class MainActivity extends Activity {
2.
3.     @SuppressWarnings("deprecation")
4.     @Override
5.     protected void onCreate(Bundle savedInstanceState) {
6.         super.onCreate(savedInstanceState);
7.         setContentView(R.layout.activity_main);
8.         int width = getWindowManager().getDefaultDisplay().getWidth();
9.         int height = getWindowManager().getDefaultDisplay().getHeight();
10.        //获取fragmentmanager
11.        FragmentManager manager = getFragmentManager();
12.        //开启fragment事务
13.        FragmentTransaction transaction = manager.beginTransaction();
14.
15.        if(width>height){
16.            //横屏
17.            //把fragment对象 替换到 viewgroup节点下
18.            //第一个参数 用来放置fragment的viewgroup的id
19.            //第二个参数 要显示的fragment对象
20.            transaction.replace(R.id.fragment_container, new SecondFragment());
21.        }else{
22.            //竖屏
23.            transaction.replace(R.id.fragment_container, new FirstFragment());
24.        }
25.        //设置完对应的fragment一定要调用commit提交事务
26.        transaction.commit();
27.    }
28. }

```

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

```

1. import android.os.Bundle;
2. import android.support.v4.app.FragmentActivity;
3. import android.support.v4.app.FragmentManager;
4. import android.support.v4.app.FragmentTransaction;
5.
6. public class MainActivity extends FragmentActivity {
7.     //要动态加载v4包中的fragment 必须继承FragmentActivity
8.
9.     @Override
10.    protected void onCreate(Bundle savedInstanceState) {
11.        super.onCreate(savedInstanceState);
12.        setContentView(R.layout.activity_main);
13.        //获取FragmentManager
14.        // FragmentManager fragmentManager = getFragmentManager();
15.        //FragmentManager 也要使用v4包中的 v4包中 获取 FragmentManager 要用 getSupportFragmentManager
16.        // getSupportFragmentManager这个方法 实在v4包的 FragmentActivity中才可以使用
17.        FragmentManager supportFragmentManager = getSupportFragmentManager();
18.        //开始fragment事务
19.        FragmentTransaction transaction = supportFragmentManager.beginTransaction(
20.        );
21.        //替换fragment
22.        transaction.replace(R.id.fragment_container, new DemoFragment());
23.        //提交事务
24.        transaction.commit();
25.    }
26. }

```

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

```

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

```

通过这个标签就可以找到 对应的fragment

```

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

```

7 menu菜单

```
1.     @Override
2.     public boolean onCreateOptionsMenu(Menu menu) {
3.         // Inflate the menu; this adds items to the action bar if it is present.
4.         getMenuInflater().inflate(R.menu.main, menu);
5.         return true;
6.     }
7.
8.     @Override
9.     public boolean onOptionsItemSelected(MenuItem item) {
10.        //MenuItem就是被点击的菜单项 对象
11.        //可以获取菜单项的id 通过id判断哪个条目被点击了
12.        int id = item.getItemId();
13.        switch (id) {
14.            case R.id.action_settings:
15.                Toast.makeText(this, "最下面的条目被点击了", Toast.LENGTH_SHORT).show(
16.                );
17.                break;
18.            case R.id.action_settings1:
19.                Toast.makeText(this, "中间的条目被点击了", Toast.LENGTH_SHORT).show();
20.                break;
21.            case R.id.action_settings2:
22.                Toast.makeText(this, "最上面的条目被点击了", Toast.LENGTH_SHORT).show(
23.                );
24.                break;
25.            default:
26.                break;
27.        }
28.        return super.onOptionsItemSelected(item);
29.    }
30.
31.    @Override
32.    public boolean onMenuOpened(int featureId, Menu menu) {
33.        //当用户点击菜单按钮就会执行这个方法
34.        //如果返回值是true 那么会走系统默认的实现 执行onCreateOptionsMenu
35.        //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false
36.
37.        AlertDialog.Builder builder = new Builder(this);
38.        builder.setTitle("这是一个菜单");
39.        builder.setMessage("菜单的内容");
40.        builder.setPositiveButton("确定", new OnClickListener() {
41.            @Override
42.            public void onClick(DialogInterface dialog, int which) {
43.                Toast.makeText(getApplicationContext(), "菜单确定", Toast.LENGTH_S
44.                HORT).show();
45.            }
46.        });
47.        builder.show();
48.        return false;
49.    }
```

8 AutoCompleteTextView控件的使用

```
1. public class MainActivity extends Activity {
2.     private String[] names = {"laowang", "laozhang", "laoli", "xiaowang", "xiao Zhang",
3.     "xiaoli"};
4.     @Override
5.     protected void onCreate(Bundle savedInstanceState) {
6.         super.onCreate(savedInstanceState);
7.         setContentView(R.layout.activity_main);
8.         AutoCompleteTextView actv = (AutoCompleteTextView) findViewById(R.id.actv_text);
9.         //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
10.        ArrayAdapter<String> adapter = new ArrayAdapter<String>(getApplicationContext(), R.layout.item, names);
11.        //设置适配器
12.        actv.setAdapter(adapter);
13.    }
```

布局文件中声明对应节点

```
1. <RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
2.     xmlns:tools="http://schemas.android.com/tools"
3.     android:layout_width="match_parent"
4.     android:layout_height="match_parent"
5.     android:paddingBottom="@dimen/activity_vertical_margin"
6.     android:paddingLeft="@dimen/activity_horizontal_margin"
7.     android:paddingRight="@dimen/activity_horizontal_margin"
8.     android:paddingTop="@dimen/activity_vertical_margin"
9.     tools:context=".MainActivity" >
10.
11.     <AutoCompleteTextView
12.         android:id="@+id/actv_text"
13.         android:layout_width="match_parent"
14.         android:layout_height="wrap_content"
15.         android:completionThreshold="1"
16.         android:hint="请输入内容" />
17.
18. </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

```

1. <?xml version="1.0" encoding="utf-8"?>
2. <animation-list xmlns:android="http://schemas.android.com/apk/res/android"
3.     android:oneshot="false">
4.     <item android:drawable="@drawable/girl_1" android:duration="200" />
5.     <item android:drawable="@drawable/girl_2" android:duration="200" />
6.     <item android:drawable="@drawable/girl_3" android:duration="200" />
7.     <item android:drawable="@drawable/girl_4" android:duration="200" />
8.     <item android:drawable="@drawable/girl_5" android:duration="200" />
9.     <item android:drawable="@drawable/girl_6" android:duration="400" />
10.    <item android:drawable="@drawable/girl_5" android:duration="200" />
11.    <item android:drawable="@drawable/girl_6" android:duration="400" />
12.    <item android:drawable="@drawable/girl_5" android:duration="200" />
13.    <item android:drawable="@drawable/girl_6" android:duration="400" />
14.    <item android:drawable="@drawable/girl_7" android:duration="400" />
15.    <item android:drawable="@drawable/girl_8" android:duration="400" />
16.    <item android:drawable="@drawable/girl_9" android:duration="200" />
17.    <item android:drawable="@drawable/girl_10" android:duration="200" />
18.    <item android:drawable="@drawable/girl_11" android:duration="200" />
19. </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

```

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.         ImageView iv_image = (ImageView) findViewById(R.id.iv_image);
8.
9.         //找到图片背景对应的Drawable对象 强制转换为AnimationDrawable
10.        AnimationDrawable animation = (AnimationDrawable) iv_image.getBackground();
11.        ;
12.        //调用start方法开始动画
13.        animation.start();
14.    }
15. }

```

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

```
1. <?xml version="1.0" encoding="utf-8"?>
2. <alpha
3.     xmlns:android="http://schemas.android.com/apk/res/android"
4.     android:fromAlpha="1"
5.     android:toAlpha="0"
6.     android:duration="1000"
7.     android:repeatMode="reverse"
8.     android:repeatCount="1"
9. >
10. </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

```
1. <rotate
2.     xmlns:android="http://schemas.android.com/apk/res/android"
3.     android:fromDegrees="0"
4.     android:toDegrees="180"
5.     android:pivotX="50%"
6.     android:pivotY="50%"
7.     android:duration="1000"
8.     android:repeatMode="restart"
9.     android:repeatCount="1"
10.    android:fillAfter="true">
11. </rotate>
```

缩放

```
1. <?xml version="1.0" encoding="utf-8"?>
2. <scale
3.     xmlns:android="http://schemas.android.com/apk/res/android"
4.     android:fromXScale="1"
5.     android:toXScale="2"
6.     android:fromYScale="1"
7.     android:toYScale="2"
8.     android:pivotX="50%"
9.     android:pivotY="50%"
10.    android:duration="1000"
11.    android:fillAfter="true">
12.
13.
14. </scale>
```

平移

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



```
9.     android:interpolator="@android:anim/bounce_interpolator"
10.     android:fillAfter="true"
11.     >
12. </translate>
```

集合

```
1. <?xml version="1.0" encoding="utf-8"?>
2. <set
3.     xmlns:android="http://schemas.android.com/apk/res/android"
4.     android:shareInterpolator="false">
5.     <scale
6.         android:interpolator="@android:anim/accelerate_decelerate_interpolator"
7.         android:fromXScale="1.0"
8.         android:toXScale="1.4"
9.         android:fromYScale="1.0"
10.        android:toYScale="0.6"
11.        android:pivotX="50%"
12.        android:pivotY="50%"
13.        android:fillAfter="false"
14.        android:duration="700" />
15.     <set android:interpolator="@android:anim/decelerate_interpolator">
16.         <scale
17.             android:fromXScale="1.4"
18.             android:toXScale="0.0"
19.             android:fromYScale="0.6"
20.             android:toYScale="0.0"
21.             android:pivotX="50%"
22.             android:pivotY="50%"
23.             android:startOffset="700"
24.             android:duration="400"
25.             android:fillBefore="false" />
26.         <rotate
27.             android:fromDegrees="0"
28.             android:toDegrees="-45"
29.             android:toYScale="0.0"
30.             android:pivotX="50%"
31.             android:pivotY="50%"
32.             android:startOffset="700"
33.             android:duration="400" />
34.     </set>
35.
36. </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

```
1. <?xml version="1.0" encoding="utf-8"?>
2. <animator xmlns:android="http://schemas.android.com/apk/res/android" >
3.     <objectAnimator
4.         android:propertyName="translationX"
5.         android:duration="2000"
6.         android:valueFrom="10"
7.         android:valueTo="100"
8.     ></objectAnimator>
9.
10. </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);
```

```
1. //位移动画
2. public void translate(View v){
3.     //创建属性动画
4.     /**
5.      * target 执行的目标
6.      * propertyName 属性名字 The name of the property being animated.
7.      * float... values 可变参数
8.      */
9.     ObjectAnimator oa = ObjectAnimator.ofFloat(iv, "translationX", 10, 50,20,100,150);
10.     oa.setDuration(2000);
11.     oa.start(); //开始动画
12.
13. }
14. //缩放动画
15. public void scale(View v){
16.     ObjectAnimator oa = ObjectAnimator.ofFloat(iv, "scaleY", 0.1f, 2, 1, 2);
17.     oa.setDuration(2000);
18.     oa.start();
19. }
20.
21. //实现透明的效果
22. public void alpha(View v){
23.     ObjectAnimator oa = ObjectAnimator.ofFloat(iv, "alpha", 0, 0.5f, 0, 1,0,1)
24. ;
25.     oa.setDuration(2000);
26.     oa.start();
27. }
```

```

28. //实现旋转的效果
29. public void rotate(View v){
30.     ObjectAnimator oa = ObjectAnimator.ofFloat(iv, "rotation", 0, 180, 90, 360
    );
31. //     ObjectAnimator oa = ObjectAnimator.ofFloat(iv, "rotationY", 0, 180, 90, 360
    0);
32.     oa.setDuration(2000);
33.     oa.start();
34. }
35.
36.
37. //一起飞
38. public void fly(View v){
39.     AnimatorSet as = new AnimatorSet();
40.     ObjectAnimator oa = ObjectAnimator.ofFloat(iv, "translationX", 10, 50, 20,
    100);
41.     ObjectAnimator oa2 = ObjectAnimator.ofFloat(iv, "scaleY", 0.1f, 2, 1, 2);
42.     ObjectAnimator oa3 = ObjectAnimator.ofFloat(iv, "alpha", 0, 0.5f, 0, 1);
43.     ObjectAnimator oa4 = ObjectAnimator.ofFloat(iv, "rotationY", 0, 180, 90, 3
    60);
44.     as.setDuration(2000);
45.     as.setTarget(iv);
46.     //往集合中添加动画
47.     //挨个飞
48.     //as.playSequentially(oa, oa2, oa3, oa4);
49.     //一起飞
50.     as.playTogether(oa, oa2, oa3, oa4);
51.     as.start();
52. }

```

13 通知栏介绍

Notification 通知

Notification.Builder 设置通知的内容使用builder

NotificationManager 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

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

④ 获取NotificationManager 对象 使用NotificationManager 发送通知 notify

```

1. public void send(View v){
2.     //通过Notification.Builder 创建一个notification
3.     Notification.Builder builder = new Builder(this);
4.     //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
5.     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");
6.     //设置当前的通知 用户点击之后就消失掉
7.     builder.setAutoCancel(true);

```

```

8.      //设置在通知栏中显示的大标题
9.      builder.setTitle("账户异动通知");
10.     //设置在通知栏中显示的小的文字内容
11.     builder.setText("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");
12.     //设置在状态栏显示的小图标
13.     builder.setSmallIcon(R.drawable.ic_launcher);
14.     //builder.setOngoing(true);
15.     Intent intent2 = new Intent(this,MainActivity.class);
16.     //pendingIntent 延迟执行的意图 当通知被点击的时候 就会执行intent
17.     //如果想操作activity getActivity
18.     //如果想操作service getService
19.     //第二个参数 请求码 用来区分不同的pendingIntent
20.     //第三个参数 意图
21.     //第四个参数 传FLAG_UPDATE_CURRENT 更新pendingintent
22.     PendingIntent intent = PendingIntent.getActivity(getApplicationContext(),
1, intent2, PendingIntent.FLAG_UPDATE_CURRENT);
23.     builder.setContentIntent(intent);
24.     Notification notification = builder.build();
25.     //获取notificationManager
26.     NotificationManager mananger = (NotificationManager) getSystemService(NOTI
FICATION_SERVICE);
27.     //调用这个方法 通知会发送到通知栏中
28.     //第一个参数 通知的id
29.     mananger.notify(1, notification);
30.     mananger.cancel(1);
31. }

```

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