

YUZHIBOYI的专栏


Make your life a story worth telling !

目录视图

摘要视图

RSS 订阅

个人资料



YAnG\_Linux

访问: 996387次

积分: 8383

等级: BLOG 5

排名: 第1628名

原创: 31篇 转载: 217篇

译文: 0篇 评论: 188条

文章搜索

文章分类

Linux (28)

Android (171)

IOS (7)

算法 (4)

嵌入式软件 (8)

Cocos2d-x (19)

Android进阶 (14)

文章存档

2015年06月 (1)

2014年08月 (16)

2014年07月 (10)

2014年06月 (8)

2013年09月 (4)

展开

阅读排行

Android中pendingIntent (70750)

Android的三种网络通信 (64920)

Android中的AnimationS (53080)

深度学习代码专栏 攒课--我的学习我做主 开启你的知识管理, 知识库个人图谱上线

Android中的AnimationSet使用

标签: android animation button xml float getter

2012-07-10 10:35 53102人阅读 评论(16) 收藏 举报

分类: Android (170)

目录(?) [+]

3.0以前, android支持两种动画模式, tween animation,frame animation, 在android3.0中又引入了一个新的动画系统: property animation, 这三种动画模式在SDK中被称为property animation,view animation,drawable animation。

### 1. View Animation ( Tween Animation )

View Animation ( Tween Animation ): 补间动画, 给出两个关键帧, 通过一些算法将给定属性值在给定的时间内在两个关键帧间渐变。

View animation只能应用于View对象, 而且只支持一部分属性, 如支持缩放旋转而不支持背景颜色的改变。而且对于View animation, 它只是改变了View对象绘制的位置, 而没有改变View对象本身, 比如, 你有一个Button, 坐标 (100,100), Width:200,Height:50, 而你有一个动画使其变为Width: 100, Height: 100, 你会发现动画过程中触 发按钮点击的区域仍是(100,100)-(300,150)。

View Animation就是一系列View形状的变换, 如大小的缩放, 透明度的改变, 位置的改变, 动画的定义既可以用代码定义也可以用XML定义, 当然, 建议用XML定义。

可以给一个View同时设置多个动画, 比如从透明至不透明的淡入效果, 与从小到大的放大效果, 这些动画可以同时进行, 也可以在一个完成之后开始另一个。

用XML定义的动画放在/res/anim/文件夹内, XML文件的根元素可以为<alpha>,<scale>,<translate>,<rotate>,<interpolator> 元素或<set>(表示以上几个动画的集合, set可以嵌套)。默认情况下, 所有动画是同时进行的, 可以通过startOffset属性设置 各个动画的开始偏移 (开始时间) 来达到动画顺序播放的效果。

可以通过设置interpolator属性改变动画渐变的方式, 如AccelerateInterpolator, 开始时慢, 然后逐渐加快。默认为AccelerateDecelerateInterpolator。

定义好动画的XML文件后, 可以通过类似下面的代码对指定View应用动画。

```
ImageView spaceshipImage = (ImageView)findViewById(R.id.spaceshipImage);
Animation hyperspaceJumpAnimation=AnimationUtils.loadAnimation(this, R.anim.hyperspace_jump);
spaceshipImage.startAnimation(hyperspaceJumpAnimation);
```

复制代码

### 2. Drawable Animation ( Frame Animation )

Drawable Animation ( Frame Animation ): 帧动画, 就像GIF图片, 通过一系列Drawable依次显示来模拟动画的效果。在XML中的定义方式如下:

```
?
1<animation-list xmlns:android="http://schemas.android.com/apk/res/android"
2    android:oneshot="true">
3    <item android:drawable="@drawable/rocket_thrust1" android:duration="200" />
4    <item android:drawable="@drawable/rocket_thrust2" android:duration="200" />
5    <item android:drawable="@drawable/rocket_thrust3" android:duration="200" />
```

http://blog.csdn.net/yuzhiboyi/article/details/7731826

1/9

Android中使用dimen定	(46306)
Android中的Environmen	(39657)
Android之CursorAdapte	(36480)
利用onSaveInstanceState	(33713)
Android里Service的bind	(33283)
Linux IP代理筛选系统 (	(24496)
Android之ActionBar详解	(21503)

评论排行	
Android的三种网络通信	(18)
Android中pendingIntent	(16)
Android中的AnimationS	(16)
Android中的Environmen	(10)
Android中ViewPager实	(8)
Android里Service的bind	(8)
Tabhost嵌套以及Tab中多	(8)
用ActivityGroup解决Tab	(7)
Android中使用dimen定	(7)
Android底部菜单栏 (用T	(6)

推荐文章	
* 2016 年最受欢迎的编程语言是什么？	
* Chromium扩展（Extension）的页面（Page）加载过程分析	
* Android Studio 2.2 来啦	
* 手把手教你做音乐播放器（二）技术原理与框架设计	
* JVM 性能调优实战之：使用阿里开源工具 TProfiler 在海量业务代码中精确定位性能代码	

最新评论	
Android中的Environment.getExt	真的很沉默_: 谢谢,总结的很好.
Android各种Layout特性和使用汇	忘了现实的真: 这排版
Android中pendingIntent的深入理	GoodBoy21: 文章误导人
pendingIntent与普通的Intent的	关键区别并不是立刻执行的问题、
Android的三种网络通信方式	Christophe: @xrb402874097:这是JSP的内置对象，不需要用户自己去实例化，直接拿来用就是了。
Android里Service的bindService	mybuffer1: 不错。。。。。。。。
Android中ViewPager实现动态L	吕晓冬: 像上面的例子的滑动会和左右滑动有冲突，这样的话怎么解决，求解答？？
Android网络通信的六种方式	lv_zhen123: 真心不错
Android的SharedPreferences和	Adan0520: 能够感觉得出来，写这篇博客花了很多心思，向你学习啦，真的很好
Android网络通信的六种方式	csdn_long: "webService是基于SAOP协议的远程调用标准"，应该是SOAP吧？俗称“肥皂协议”。
Android中的AnimationSet使用	jayoss152: AnimationSet 说错了

```
6</animation-list>

    必须以<animation-list>为根元素，以<item>表示要轮换显示的图片，duration属性表示各项显示的时间。
XML文件要放在/res/drawable/目录下。示例：

protected void onCreate(Bundle savedInstanceState) {
    // TODO Auto-generated method stub
    super.onCreate(savedInstanceState);
    setContentView(R.layout.main);
    imageView = (ImageView) findViewById(R.id.imageView1);
    imageView.setBackgroundResource(R.drawable.drawable_anim);
    anim = (AnimationDrawable) imageView.getBackground();
}

public boolean onTouchEvent(MotionEvent event) {
    if (event.getAction() == MotionEvent.ACTION_DOWN) {
        anim.stop();
        anim.start();
        return true;
    }
    return super.onTouchEvent(event);
}
```

复制代码

我在实验中遇到两点问题：

- 1. 要在代码中调用Imageview的setBackgroundResource方法，如果直接在XML布局文件中设置其src属性当触发动画时会FC。
- 2. 在动画start()之前要先stop()，不然在第一次动画之后会停在最后一帧，这样动画就只会触发一次。
- 3. 最后一点是SDK中提到的，不要在onCreate中调用start，因为AnimationDrawable还没有完全跟Window相关联，如果想要界面显示时就开始动画的话，可以在onWindowFoucsChanged()中调用start()。

3. Property Animation

属性动画，这个是在Android 3.0中才引进的，以前学WPF时里面的动画机制好像就是这个，它更改的是对象的实际属性，在View Animation（Tween Animation）中，其改变的是View的绘制效果，真正的View的属性保持不变，比如无论你在对话框中如何缩放Button的大小，Button的有效点击区域还是没有应用动画时的区域，其位置与大小都不变。而在Property Animation中，改变的是对象的实际属性，如Button的缩放，Button的位置与大小属性值都改变了。而且Property Animation不止可以应用于View，还可以应用于任何对象。Property Animation只是表示一个值在一段时间内的改变，当值改变时要做什么事情完全是你自己决定的。

在Property Animation中，可以对动画应用以下属性：

- Duration：动画的持续时间
- TimeInterpolation：属性值的计算方式，如先快后慢
- TypeEvaluator：根据属性的开始、结束值与TimeInterpolation计算出的因子计算出当前时间的属性值
- Repeat Country and behavoir：重复次数与方式，如播放3次、5次、无限循环，可以此动画一直重复，或播放完后再反向播放
- Animation sets：动画集合，即可以同时对一个对象应用几个动画，这些动画可以同时播放也可以对不同动画设置不同开始偏移
- Frame refreash delay：多少时间刷新一次，即每隔多少时间计算一次属性值，默认为10ms，最终刷新时间还受系统进程调度与硬件的影响

3.1 Property Animation的工作方式

对于下图的动画，这个对象的X坐标在40ms内从0移动到40 pixel.按默认的10ms刷新一次，这个对象会移动4次，每次移动40/4=10pixel。



也可以改变属性值的改变方法，即设置不同的interpolation，在下图中运动速度先逐渐增大再逐渐减小



下图显示了与上述动画相关的关键对象



ValueAnimator即表示一个动画，包含动画的开始值，结束值，持续时间等属性。

ValueAnimator封装了一个TimeInterpolator，TimeInterpolator定义了属性值在开始值与结束值之间的插值方法。

ValueAnimator还封装了一个TypeAnimator，根据开始、结束值与TimeInterpolator计算得到的值计算出属性值。

ValueAnimator根据动画已进行的时间跟动画总时间（duration）的比计算出一个时间因子（0~1），然后根据TimeInterpolator计算出另一个因子，最后TypeAnimator通过这个因子计算出属性值，如上例中10ms时：

首先计算出时间因子，即经过的时间百分比： $t=10ms/40ms=0.25$

经插值计算(inteplator)后的插值因子:大约为0.15，上述例子中用了AccelerateDecelerateInterpolator，计算公式为（input即为时间因子）：

```
(Math.cos((input + 1) * Math.PI / 2.0f) + 0.5f);
```

最后根据TypeEvaluator计算出在10ms时的属性值： $0.15 * (40-0) = 6\text{pixel}$ 。上例中TypeEvaluator为

FloatEvaluator，计算方法为：

```
public Float evaluate(float fraction, Number startValue, Number endValue) {
    float startFloat = startValue.floatValue();
    return startFloat + fraction * (endValue.floatValue() - startFloat);
}
```

[复制代码](#)

参数分别为上一步的插值因子，开始值与结束值。

### 3.2 ValueAnimator

ValueAnimator包含Property Animation动画的所有核心功能，如动画时间，开始、结束属性值，相应时间属性值计算方法等。应用Property Animation有两个步骤：

1. 计算属性值
2. 根据属性值执行相应的动作，如改变对象的某一属性。

ValueAnimator只完成了第一步工作，如果要完成第二步，需要实现ValueAnimator.OnUpdateListener接口，如：

```
ValueAnimator animation = ValueAnimator.ofFloat(0f, 1f);
animation.setDuration(1000);
animation.addUpdateListener(new AnimatorUpdateListener() {
    @Override
    public void onAnimationUpdate(ValueAnimator animation) {
        Log.i("update", ((Float) animation.getAnimatedValue()).toString());
    }
});
animation.setInterpolator(new CycleInterpolator(3));
animation.start();
```

[复制代码](#)

此示例中只是向Logcat输出了一些信息，可以改为想做的工作。

#### Animator.AnimatorListener

```
onAnimationStart()

onAnimationEnd()

onAnimationRepeat()

onAnimationCancel
```

[复制代码](#)

#### ValueAnimator.AnimatorUpdateListener

onAnimationUpdate() //通过监听这个事件在属性的值更新时执行相应的操作，对于ValueAnimator一般要监听此事件执行相应的动作，不然Animation没意义（可用于计时），在ObjectAnimator（继承自ValueAnimator）中会自动更新属性，如无必要不必监听。在函数中会传递一个ValueAnimator参数，通过此参数的getAnimatedValue()取得当前动画属性值。

#### 复制代码

可以继承AnimatorListenerAdapter而不是实现AnimatorListener接口来简化操作，这个类对AnimatorListener中的函数都定义了一个空函数体，这样我们就只用定义想监听的事件而不用实现每个函数却只定义一空函数体。

```
ObjectAnimator oa=ObjectAnimator.ofFloat(tv, "alpha", 0f, 1f);
oa.setDuration(3000);
oa.addListener(new AnimatorListenerAdapter() {
    public void onAnimationEnd(Animator animation){
        Log.i("Animation","end");
    }
});
oa.start();
```

#### 复制代码

### 3.3 ObjectAnimator

继承自ValueAnimator，要指定一个对象及该对象的一个属性，当属性值计算完成时自动设置为该对象的相应属性，即完成了Property Animation的全部两步操作。实际应用中一般都会用ObjectAnimator来改变某一对象的某一属性，但用ObjectAnimator有一定的限制，要想使用ObjectAnimator，应该满足以下条件：

- 对象应该有一个setter函数：set<PropertyName>（驼峰命名法）
- 如上面的例子中，像ofFloat之类的工厂方法，第一个参数为对象名，第二个为属性名，后面的参数为可变参数，如果values...参数只设置了一个值的话，那么会假定为目的值，属性值的变化范围围为的值，为了获得当前值，该对象要有相应属性的getter方法：get<PropertyName>
- 如果有getter方法，其应返回值类型应与相应的setter方法的参数类型一致。

如果上述条件不满足，则不能用ObjectAnimator，应用ValueAnimator代替。

```
tv=(TextView)findViewById(R.id.textview1);
btn=(Button)findViewById(R.id.button1);
btn.setOnClickListener(new OnClickListener() {
    @Override
    public void onClick(View v) {
        ObjectAnimator oa=ObjectAnimator.ofFloat(tv, "alpha", 0f, 1f);
        oa.setDuration(3000);
        oa.start();
    }
});
```

#### 复制代码

把一个TextView的透明度在3秒内从0变至1。

根据应用动画的对象或属性的不同，可能需要在onAnimationUpdate函数中调用invalidate()函数刷新视图。

### 3.4 通过AnimationSet应用多个动画

AnimationSet提供了一个把多个动画组合成一个组合的机制，并可设置组中动画的时序关系，如同时播放，顺序播放等。

以下例子同时应用5个动画：

1. 播放anim1;
2. 同时播放anim2,anim3,anim4;
3. 播放anim5。

?

```
1AnimatorSet bouncing = new AnimatorSet();
```

```
2bouncer.play(anim1).before(anim2);
3bouncer.play(anim2).with(anim3);
4bouncer.play(anim2).with(anim4);
5bouncer.play(anim5).after(anim2);
6animatorSet.start();
```

### 3.5 TypeEvaluators

根据属性的开始、结束值与TimeInterpolation计算出的因子计算出当前时间的属性值，android提供了以下几个evaluator：

- IntEvaluator：属性的值类型为int；
- FloatEvaluator：属性的值类型为float；
- ArgbEvaluator：属性的值类型为十六进制颜色值；
- TypeEvaluator：一个接口，可以通过实现该接口自定义Evaluator。

自定义TypeEvaluator很简单，只需要实现一个方法，如FloatEvaluator的定义：

?

```
1public class FloatEvaluator implements TypeEvaluator {
2    public Object evaluate(float fraction, Object startValue, Object endValue) {
3        float startFloat = ((Number) startValue).floatValue();
4        return startFloat + fraction * (((Number) endValue).floatValue() - startFloat);
5    }
6}
```

根据动画执行的时间跟应用的Interpolator，会计算出一个0~1之间的因子，即evaluate函数中的fraction参数，通过上述FloatEvaluator应该很好看出其意思。

### 3.6 TimeInterpolator

time interpolator定义了属性值变化的方式，如线性均匀改变，开始慢然后逐渐快等。在Property Animation中是TimeInterpolator，在View Animation中是Interpolator，这两个是一样的，在3.0之前只有Interpl实现代码转移至了 TimeInterpolator。Interpolator继承自TimeInterpolator，内部没有任何其他代码。

- |                                    |   |
|------------------------------------|---|
| • AccelerateInterpolator           | 加速，开始时慢中间加速   |
| • DecelerateInterpolator           | 减速，开始时快然后减速   |
| • AccelerateDecelerateInterpolator | 先加速后减速，开始结束时慢，中间加速  |
| • AnticipateInterpolator           | 反向，先向相反方向改变一段再加速播放  |
| • AnticipateOvershootInterpolator  | 反向加超越，先向相反方向改变，再加速播放，会超出目的值然后缓慢移动至目的值   |
| • BounceInterpolator               | 跳跃，快到目的值时会跳跃，如目的值100，后面的值可能依次为85，77，70，80，90，100  |
| • CycleInterpolator                | 循环，动画循环一定次数，值的改变为一正弦函数： $\text{Math.sin}(2 * \text{mCycles} * \text{Math.PI} * \text{input})$ |
| • LinearInterpolator               | 线性，线性均匀改变   |
| • OvershootInterpolator            | 超越，最后超出目的值然后缓慢改变到目的值  |
| • TimeInterpolator                 | 一个接口，允许你自定义interpolator，以上几个都是实现了这个接口   |

### 3.7 当Layout改变时应用动画

ViewGroup中的子元素可以通过setVisibility使其Visible、Invisible或Gone，当有子元素可见性改变时，可以向其应用动画，通过LayoutTransition类应用此类动画：

```
transition.setAnimator(LayoutTransition.DISAPPEARING, customDisappearingAnim);
```

通过setAnimator应用动画，第一个参数表示应用的情境，有以下4种类型：

- APPEARING 当一个元素变为Visible时对其应用的动画
- CHANGE\_APPEARING 当一个元素变为Visible时，因系统要重新布局有一些元素需要移动，这些要移动的元素应用的动画
- DISAPPEARING 当一个元素变为InVisible时对其应用的动画

- **CHANGE\_DISAPPEARING** 当一个元素变为Gone时，因系统要重新布局有一些元素需要移动，这些要移动的元素应用的动画 **disappearing from the container**.

第二个参数为一**Animator**。

```
mTransitioner.setStagger(LayoutTransition.CHANGE_APPEARING, 30);
```

此函数设置动画持续时间，参数分别为类型与时间。

### 3.8 Keyframes

**keyFrame**是一个 时间/值 对，通过它可以定义一个在特定时间的特定状态，而且在两个**keyFrame**之间可以定义不同的**Interpolator**，就相当多个动画的拼接，第一个动画的结束点是第二个动画的开始点。**KeyFrame**是抽象类，要通过**ofInt()**、**ofFloat()**、**ofObject()**获得适当的 **KeyFrame**，然后通过**PropertyValuesHolder.ofKeyframe**获得**PropertyValuesHolder**对象，如以下 例子：

```
Keyframe kf0 = Keyframe.ofInt(0, 400);
Keyframe kf1 = Keyframe.ofInt(0.25f, 200);
Keyframe kf2 = Keyframe.ofInt(0.5f, 400);
Keyframe kf4 = Keyframe.ofInt(0.75f, 100);
Keyframe kf3 = Keyframe.ofInt(1f, 500);
PropertyValuesHolder pvhRotation = PropertyValuesHolder.ofKeyframe("width", kf0, kf1, kf2, kf4, kf3);
ObjectAnimator rotationAnim = ObjectAnimator.ofPropertyValuesHolder(btn2, pvhRotation);
rotationAnim.setDuration(2000);
```

#### 复制代码

上述代码的意思为：设置btn对象的width属性值使其：

- 开始时 **Width=400**
- 动画开始1/4时 **Width=200**
- 动画开始1/2时 **Width=400**
- 动画开始3/4时 **Width=100**
- 动画结束时 **Width=500**

第一个参数为时间百分比，第二个参数是在第一个参数的时间时的属性值。

定义了一些**Keyframe**后，通过**PropertyValuesHolder**类的方法**ofKeyframe**封装，然后通过**ObjectAnimator.ofPropertyValuesHolder**获得**Animator**。

用下面的代码可以实现同样的效果：

```
ObjectAnimator oa=ObjectAnimator.ofInt(btn2, "width", 400,200,400,100,500);
oa.setDuration(2000);
oa.start();
```

#### 复制代码

### 3.9 Animating Views

在**View Animation**中，对**View**应用**Animation**并没有改变**View**的属性，动画的实现是通过其**Parent View**实现的，在**View**被drawn时**Parents View**改变它的绘制参数，**draw**后再改变参数**invalidate**，这样虽然**View**的大小或旋转角度等改变了，但**View**的实际属性没变，所以有效 区域还是应用动画之前的区域，比如你把一按钮放大两倍，但还是放大这前的区域可以触发点击事件。为了改变这一点，在**Android 3.0**中给**View**增加了一些参数并对这些参数增加了相应的**getter/setter**函数（**ObjectAnimator**要用这些函数改变这些属性）：

- **translationX,translationY**:转换坐标（control where the View is located as a delta from its left and top coordinates which are set by its layout container.）
- **rotation,rotationX,rotationY**:旋转，**rotation**用于2D旋转角度，3D中用到后两个
- **scaleX,scaleY**:缩放
- **x,y**:**View**的最终坐标（utility properties to describe the final location of the View in its container, as a sum of the left and top values and translationX and translationY values.）
- **alpha**:透明度

跟位置有关的参数有3个，以X坐标为例，可以通过`getLeft()`、`getX()`、`getTranslationX()`获得，若有一Button `btn2`，布局时其坐标为(40,0)：

```
//应用动画之前
btn2.getLeft();    //40
btn2.getX();       //40
btn2.getTranslationX();    //0
//应用translationX动画
ObjectAnimator oa=ObjectAnimator.ofFloat(btn2,"translationX", 200);
oa.setDuration(2000);
oa.start();
/*应用translationX动画后
btn2.getLeft();    //40
btn2.getX();       //240
btn2.getTranslationX();    //200
*/
//应用X动画，假设没有应用之前的translationX动画
ObjectAnimator oa=ObjectAnimator.ofFloat(btn2, "x", 200);
oa.setDuration(2000);
oa.start();
/*应用X动画后
btn2.getLeft();    //40
btn2.getX();       //200
btn2.getTranslationX();    //160
*/
```

#### 复制代码

无论怎样应用动画，原来的布局时的位置通过`getLeft()`获得，保持不变；

**X**是View最终的位置；

**translationX**为最终位置与布局时初始位置这差。

所以若就用**translationX**即为在原来基础上移动多少，**X**为最终多少

**getX()**的值为`getLeft()`与`getTranslationX()`的和

对于**X**动画，源代码是这样的：

```
case X:
    info.mTranslationX = value - mView.mLeft;
    break;
```

#### 复制代码

Property Animation也可以在XML中定义

- <set> - AnimatorSet
- <animator> - ValueAnimator
- <objectAnimator> - ObjectAnimator

XML文件应放大/res/animator/中，通过以下方式应用动画：

```
AnimatorSet set = (AnimatorSet) AnimatorInflater.loadAnimator(mContext,
R.anim.property_animator);
set.setTarget(myObject);
set.start();
```

#### 复制代码

### 3.10 ViewPropertyAnimator

如果需要对一个View的多个属性进行动画可以用ViewPropertyAnimator类，该类对多属性动画进行了优化，会合并一些invalidate()来减少刷新视图，该类在3.1中引入。

以下两段代码实现同样的效果：



```
PropertyValuesHolder pvhX = PropertyValuesHolder.ofFloat("x", 50f);
PropertyValuesHolder pvhY = PropertyValuesHolder.ofFloat("y", 100f);
ObjectAnimator.ofPropertyValuesHolder(myView, pvhX, pvhY).start();
```

复制代码

```
myView.animate().x(50f).y(100f);
```

顶

0

踩

1

我的同类文章

Android (170)

• java反射机制详解 及 Meth...

2015-06-08

阅读 1042

• Google I/O开发者大会之A...

2014-07-12

阅读 871

• android显示和隐藏键盘

2013-09-01

阅读 969

• Webview--如何让加载进...

2013-09-01

阅读 1194

• Android引入第三方jar包...

2013-03-30

阅读 764

• Android 开发之使用Eclips...

2013-03-23

阅读 804

• Android中SoundPool 类...

2014-08-06

阅读 978

• Intent 和 PendingIntent 区...

2013-09-17

阅读 964

• android 属性汇总

2013-09-01

阅读 1563

• Android的View类详解

2013-04-10

阅读 964

• Android游戏引擎汇总

2013-03-25

阅读 809

更多文章

猜你在找

- 【Android APP开发】Android高级商业布局快速实现

Android 30以前版本使用Fragment提示
- Android之动画全讲

Android v4包使用NotificationCompatBuilder 遇到
- Qt基础与Qt on Android入门

使用android studio 报错 undefined reference to
- 解析移动应用的身份认证，数据分析及信息推送

Android应用开发之所有动画使用详解
- Android移植基础

Android应用开发之所有动画使用详解

查看评论

13楼 [jayoss152](#) 2016-02-04 13:53发表

C

AnimationSet 说错了

12楼 [Bran886](#) 2015-10-13 17:54发表

C

6666666666

11楼 [阿俊\\_](#) 2015-08-27 19:14发表

一篇文章标题都错了，搞毛，直接pass

10楼 [taothreeyears](#) 2015-04-21 17:20发表

C

AnimationSet不是AnimatorSet啊，楼主这么久了也不改下

Re: [KougamiShinya](#) 2015-08-05 09:28发表





回复taothreeyears: 什么啊，就是AnimatorSet啊。。。你再去API查下看？AnimationSet里都是些什么方法啊，根本不能用。。。

9楼

ys997653450

2014-11-02 12:31发表



多谢

8楼

ys997653450

2014-10-31 11:47发表



谢谢

7楼

人未眠

2014-09-26 11:37发表



不知道在哪里copy的，写的啥哦，bouncer.play(anim1)，anim1是什么？

6楼

29boy

2014-06-19 10:04发表



1.要在代码中调用Imageview的setBackgroundResource方法，如果直接在XML布局文件中设置其src属性当触发动画时会FC。  
这个不是楼主说的。  
而是imageview，你设置的src就get他，如果你设置的background。那就getbackground。对应的。  
和在哪里设置没关系。  
很多人出错是因为，xml里设置个src。。结果代码里 getbackground。你都没设置background，你获取的肯定是空了。  
  
Re: xianmengyu 2014-09-11 14:20发表



回复29boy: 有道理！

5楼

simingshen

2014-05-13 17:35发表



感谢分享

4楼

subzero88

2014-01-10 09:13发表



帮了我大忙了。。。之前自己知道的太片面了

3楼

yashirx

2013-07-12 10:59发表



楼主标题写的是AnimationSet，结果自己说的，举得例子都是AnimatorSet，AnimatorSet和AnimationSe的！！错误这么明显。自己都没搞清楚啊！  
  
Re: KougamiShinya 2015-08-05 09:31发表



回复yashirx: 那只能说是标题错了。。。但内容是对的呀。。而且我感觉这篇文章写得比那什么《疯子Android讲义》要好太多。。。通俗易懂，细致入微。。。。那啥李刚的疯子安卓讲的什么鬼。。看了我3个小时一个程序都没看懂。。。看此文一遍就懂了。。。

2楼

longtianguo

2013-04-02 17:20发表



那四种基本动画 scale，translate，alpha,rotate 都属于哪种，scale，跟rotate不是 tween吗？怎么说的像property anim

1楼

ssy\_neo

2013-01-16 19:47发表



多谢楼主的总结！

您还没有登录,请[\[登录\]](#)或[\[注册\]](#)

\* 以上用户言论只代表其个人观点，不代表CSDN网站的观点或立场

核心技术类目

全部主题

Hadoop

AWS

移动游戏

Java

Android

iOS

Swift

智能硬件

Docker

OpenStack

VPN

Spark

ERP

IE10

Eclipse

CRM

JavaScript

数据库

Ubuntu

NFC

WAP

jQuery

BI

HTML5

Spring

Apache

.NET

API

HTML

SDK

IIS

Fedora

XML

LBS

Unity

Splashtop

UML

components

Windows Mobile

Rails

QEMU

KDE

Cassandra

CloudStack

FTC

coremail

OPhone

CouchBase

云计算

iOS6

Rackspace

Web App

SpringSide

Maemo

Compuware

大数据

aptech

Perl

Tornado

Ruby

Hibernate

ThinkPHP

HBase

Pure

Solr

Angular

Cloud Foundry

Redis

Scala

Django

Bootstrap