

## 1、概述

Android提供了几种动画类型：View Animation、Drawable Animation、Property Animation。View Animation相当简单，不过只能支持简单的缩放、平移、旋转、透明度基本的动画，且有一定的局限性。比如：你希望View有一个颜色的切换动画；你希望可以使用3D旋转动画；你希望当动画停止时，View的位置就是当前的位置；这些View Animation都无法做到。这就是Property Animation产生的原因，本篇博客详细介绍Property Animation的用法。至于Drawable Animation，嗯，略~

## 2、相关API

Property Animation顾名思义就是通过动画的方式改变对象的属性了，我们首先需要了解几个属性：

Duration动画的持续时间，默认300ms。

Time interpolation：时间差值，乍一看不知道是什么，但是我说LinearInterpolator、AccelerateDecelerateInterpolator，大家一定知道是干嘛的了，定义动画的变化率。

Repeat count and behavior：重复次数、以及重复模式；可以定义重复多少次；重复时从头开始，还是反向。

Animator sets: 动画集合，你可以定义一组动画，一起执行或者顺序执行。

Frame refresh delay: 帧刷新延迟, 对于你的动画, 多久刷新一次帧; 默认为10ms, 但最终依赖系统的当前状态; 基本不用管。

相关的类

ObjectAnimator 动画的执行类, 后面详细介绍

ValueAnimator 动画的执行类, 后面详细介绍

AnimatorSet 用于控制一组动画的执行: 线性, 一起, 每个动画的先后执行等。

AnimatorInflater 用户加载属性动画的xml文件

TypeEvaluator 类型估值, 主要用于设置动画操作属性的值。

TimeInterpolator 时间插值, 上面已经介绍。

总的来说, 属性动画就是, 动画的执行类来设置动画操作的对象的属性、持续时间, 开始和结束的属性值, 时间差值等, 然后系统会根据设置的参数动态的变化对象的属性。

### 3、ObjectAnimator实现动画

之所以选择ObjectAnimator为第一个~~是因为, 这个实现最简单~~一行代码, 秒秒钟实现动画, 下面看个例子:

布局文件：

[html] view plaincopy

```
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:id="@+id/id_container" >
6.
7.     <ImageView
8.         android:id="@+id/id_ball"
9.         android:layout_width="wrap_content"
10.        android:layout_height="wrap_content"
11.        android:layout_centerInParent="true"
12.        android:src="@drawable/mv"
13.        android:scaleType="centerCrop"
14.        android:onClick="rotateyAnimRun"
15.    />
16.
17. </RelativeLayout>
```

很简单，就一张妹子图片~

Activity代码：

[java] view plaincopy

```
1. package com.example.zhy_property_animation;
2.
```

```

3. import android.animation.ObjectAnimator;
4. import android.app.Activity;
5. import android.os.Bundle;
6. import android.view.View;
7.
8. public class ObjectAnimActivity extends Activity
9. {
10.     @Override
11.     protected void onCreate(Bundle savedInstanceState)
12.     {
13.         super.onCreate(savedInstanceState);
14.         setContentView(R.layout.xml_for_anim);
15.     }
16.
17.     public void rotateyAnimRun(View view)
18.     {
19.         ObjectAnimator//
20.             .ofFloat(view, "rotationX", 0.0F, 360.0F)//
21.             .setDuration(500)//
22.             .start();
23.     }
24.
25. }

```

效果：

是不是一行代码就能实现简单的动画~~

对于ObjectAnimator

1、提供了ofInt、ofFloat、ofObject，这几个方法都是设置动画作用的元素、作用的属性、动画开始、结束、以及中间的任意个属性值。

当对于属性值，只设置一个的时候，会认为当然对象该属性的值为开始（getPropName反射获取），然后设置的值为终点。如果设置两个，则一

个为开始、一个为结束~~~

动画更新的过程中，会不断调用setPropertyName更新元素的属性，所有使用ObjectAnimator更新某个属性，必须得有getter（设置一个属性值的时候）和setter方法~

2、如果你操作对象的该属性方法里面，比如上例的setRotationX如果内部没有调用view的重绘，则你需要自己按照下面方式手动调用。

[java] view plaincopy

```
1. anim.addUpdateListener(new AnimatorUpdateListener()
2.     {
3.         @Override
4.         public void onAnimationUpdate(ValueAnimator animation)
5.         {
6.             // view.postInvalidate();
7.             // view.invalidate();
8.         }
9.     });
```

3、看了上面的例子，因为设置的操作的属性只有一个，那么如果我希望一个动画能够让View既可以缩小、又能够淡出（3个属性scaleX,scaleY,alpha），只使用ObjectAnimator咋弄？

想法是不是很不错，可能会说使用AnimatorSet啊，这一看就是一堆动画塞一起执行，但是我偏偏要用一个ObjectAnimator实例实现呢~下面看代码：

[java] view plaincopy

```
1. public void rotateyAnimRun(final View view)
2. {
3.     ObjectAnimator anim = ObjectAnimator//
4.         .ofFloat(view, "zhy", 1.0F, 0.0F)//
5.         .setDuration(500);//
6.     anim.start();
7.     anim.addUpdateListener(new AnimatorUpdateListener()
8.     {
9.         @Override
10.         public void onAnimationUpdate(ValueAnimator animation)
11.         {
12.             float cVal = (Float) animation.getAnimatedValue();
13.             view.setAlpha(cVal);
14.             view.setScaleX(cVal);
15.             view.setScaleY(cVal);
16.         }
17.     });
18. }
```

把设置属性的那个字符串，随便写一个该对象没有的属性，就是不管~~咱们只需要它按照时间插值和持续时间计算的那个值，我们自己手动调用~

效果：

这个例子就是想说明一下，有时候换个思路不要被API所约束，利用部分

API提供的功能也能实现好玩的效果~~~

比如：你想实现抛物线的效果，水平方向100px/s，垂直方向加速度200px/s\*s，咋实现呢~~可以自己用ObjectAnimator试试~

4、其实还有更简单的方式，实现一个动画更改多个效果：使用propertyValuesHolder

[java] view plaincopy

```
1. public void propertyValuesHolder(View view)
2.     {
3.         PropertyValuesHolder pvhX = PropertyValuesHolder.
ofFloat("alpha", 1f,
4.             0f, 1f);
5.         PropertyValuesHolder pvhY = PropertyValuesHolder.
ofFloat("scaleX", 1f,
6.             0, 1f);
7.         PropertyValuesHolder pvhZ = PropertyValuesHolder.
ofFloat("scaleY", 1f,
8.             0, 1f);
9.         ObjectAnimator.ofPropertyValuesHolder(view, pvhX,
pvhY, pvhZ).setDuration(1000).start();
10.    }
```

## 4、ValueAnimator实现动画

和ObjectAnimator用法很类似，简单看一下用view垂直移动的动画代码：

```
[java] view plaincopy
```

```
1. public void verticalRun(View view)
2.     {
3.         ValueAnimator animator = ValueAnimator.ofFloat(0,
4.             mScreenHeight
5.             - mBlueBall.getHeight());
6.         animator.setTarget(mBlueBall);
7.         animator.setDuration(1000).start();
8.     }
```

给你的感觉是不是，坑爹啊，这和ValueAnimator有毛线区别~但是仔细看，你看会发现，没有设置操作的属性~~也就是说，上述代码是没有任何效果的，没有指定属性~

这就是和ValueAnimator的区别之处：ValueAnimator并没有在属性上做操作，你可能会问这样有啥好处？我岂不是还得手动设置？

好处：不需要操作的对象的属性一定要有getter和setter方法，你可以自己根据当前动画的计算值，来操作任何属性，记得上例的那个【我希望一个动画能够让View既可以缩小、又能够淡出（3个属性scaleX,scaleY,alpha）】吗？其实就是这么个用法~

实例：



布局文件：

[html] view plaincopy

```
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:id="@+id/id_container"
6.
7.     >
8.
9.     <ImageView
10.         android:id="@+id/id_ball"
11.         android:layout_width="wrap_content"
12.         android:layout_height="wrap_content"
13.         android:src="@drawable/bol_blue" />
14.
15.     <LinearLayout
16.         android:layout_width="fill_parent"
17.         android:layout_height="wrap_content"
18.         android:layout_alignParentBottom="true"
19.         android:orientation="horizontal" >
20.
21.         <Button
22.             android:layout_width="wrap_content"
23.             android:layout_height="wrap_content"
24.             android:onClick="verticalRun"
25.             android:text="垂直" />
26.
27.         <Button
28.             android:layout_width="wrap_content"
29.             android:layout_height="wrap_content"
30.             android:onClick="paowuxian"
31.             android:text="抛物线" />
```

```

32.
33.     </LinearLayout>
34.
35. </RelativeLayout>

```

左上角一个小球，底部两个按钮~我们先看一个自由落体的代码：

**[java]** view plaincopy

```

1.  /**
2.      * 自由落体
3.      * @param view
4.      */
5.     public void verticalRun( View view)
6.     {
7.
8.         ValueAnimator animator = ValueAnimator.ofFloat(0,
9.             mScreenHeight
10.            - mBlueBall.getHeight());
11.         animator.setTarget(mBlueBall);
12.         animator.setDuration(1000).start();
13.         // animator.setInterpolator(value)
14.
15.         animator.addUpdateListener(new AnimatorUpdateList
16.             ener()
17.             {
18.                 @Override
19.
20.                 public void onAnimationUpdate(ValueAnimator a
21.                     nimation)
22.                 {
23.
24.                     mBlueBall.setTranslationY((Float) animati
25.                         on.getAnimatedValue());
26.                 }
27.             }
28.         );
29.     }
30. }

```

```
19.         });  
20.     }
```

与ObjectAnimator不同的就是我们自己设置元素属性的更新~虽然多了几行代码，但是貌似提高灵活性~

下面再来一个例子，如果我希望小球抛物线运动【实现抛物线的效果，水平方向100px/s，垂直方向加速度200px/s\*s】，分析一下，貌似只和时间有关系，但是根据时间的变化，横向和纵向的移动速率是不同的，我们该咋实现呢？此时就要重写TypeValue的时候了，因为我们在时间变化的同时，需要返回给对象两个值，x当前位置，y当前位置：

代码：

[java] view plaincopy

```
1.  /**  
2.     * 抛物线  
3.     * @param view  
4.     */  
5.     public void paowuxian(View view)  
6.     {  
7.  
8.  
           ValueAnimator valueAnimator = new ValueAnimator()  
           ;  
9.           valueAnimator.setDuration(3000);  
10.           valueAnimator.setObjectValues(new PointF(0, 0));  
11.
```

```

        valueAnimator.setInterpolator(new LinearInterpolato
tor());
12.        valueAnimator.setEvaluator(new TypeEvaluator<Poin
tF>()
13.        {
14.            // fraction = t / duration
15.            @Override
16.            public PointF evaluate(float fraction, PointF
startValue,
17.                PointF endValue)
18.            {
19.                Log.e(TAG, fraction * 3 + "");
20.                // x方向200px/s , 则y方向0.5 * 10 * t
21.                PointF point = new PointF();
22.                point.x = 200 * fraction * 3;
23.                point.y = 0.5f * 200 * (fraction * 3) * (
fraction * 3);
24.                return point;
25.            }
26.        });
27.
28.        valueAnimator.start();
29.
        valueAnimator.addUpdateListener(new AnimatorUpdat
eListener()
30.        {
31.            @Override
32.            public void onAnimationUpdate(ValueAnimator a
nimation)
33.            {
34.                PointF point = (PointF) animation.getAnim
atedValue();
35.                mBlueBall.setX(point.x);
36.                mBlueBall.setY(point.y);

```

```
37.  
38.      }  
39.      });  
40.  }
```

可以看到，因为ofInt,ofFloat等无法使用，我们自定义了一个TypeValue，每次根据当前时间返回一个PointF对象，（PointF和Point的区别就是x,y的单位一个是float,一个是int;RectF,Rect也是）PointF中包含了x,y的当前位置～然后我们在监听器中获取，动态设置属性：

效果图：

有木有二个铁球同时落地的感觉～～对，我应该搞二个球～～ps:物理公式要是错了，就当没看见哈

自定义TypeEvaluator传入的泛型可以根据自己的需求，自己设计个Bean。

好了，我们已经分别讲解了ValueAnimator和ObjectAnimator实现动画；二者区别；如何利用部分API，自己更新属性实现效果；自定义TypeEvaluator实现我们的需求；但是我们并没有讲如何设计插值，其实我觉得把，这个插值默认的那一串实现类够用了～～很少，会自己去设计个超级变态的～嗯～所以：略。

## 5、监听动画的事件

对于动画，一般都是一些辅助效果，比如我要删除个元素，我可能希望是个淡出的效果，但是最终还是要删掉，并不是你透明度没有了，还占着位

置，所以我们需要知道动画如何结束。

所以我们可以添加一个动画的监听：

[java] view plaincopy

```
1. public void fadeOut(View view)
2.     {
3.
4.         ObjectAnimator anim = ObjectAnimator.ofFloat(mBlueBall, "alpha", 0.5f);
5.         anim.addListener(new AnimatorListener()
6.         {
7.
8.             @Override
9.
10.             public void onAnimationStart(Animator animation)
11.             {
12.                 Log.e(TAG, "onAnimationStart");
13.             }
14.
15.             @Override
16.
17.             public void onAnimationRepeat(Animator animation)
18.             {
19.                 // TODO Auto-generated method stub
20.                 Log.e(TAG, "onAnimationRepeat");
21.             }
22.
23.             @Override
24.
25.             public void onAnimationEnd(Animator animation)
```

```

23.         {
24.             Log.e(TAG, "onAnimationEnd");
25.
26.             ViewGroup parent = (ViewGroup) mBlueBall.
                getParent();
27.             if (parent != null)
28.                 parent.removeView(mBlueBall);
29.         }
30.         @Override
31.
32.         public void onAnimationCancel(Animator animat
                ion)
33.         {
34.             // TODO Auto-generated method stub
35.             Log.e(TAG, "onAnimationCancel");
36.         }
37.         anim.start();
38.     }

```

这样就可以监听动画的开始、结束、被取消、重复等事件~但是有时候会觉得，我只要知道结束就行了，这么长的代码我不能接收，那你可以使用 **AnimatorListenerAdapter**

[java] view plaincopy

```

1. anim.addListener(new AnimatorListenerAdapter()
2. {
3.     @Override
4.     public void onAnimationEnd(Animator animation)
5.     {
6.         Log.e(TAG, "onAnimationEnd");

```

```

7.
        ViewGroup parent = (ViewGroup) mBlueBall.getParent() ;
8.         if (parent != null)
9.             parent.removeView(mBlueBall);
10.    }
11. });

```

AnimatorListenerAdapter继承了AnimatorListener接口，然后空实现了所有的方法~

效果图：

animator还有cancel()和end()方法：cancel动画立即停止，停在当前的位置；end动画直接到最终状态。

## 6、AnimatorSet的使用

实例：

布局文件：

[html] view plaincopy

```

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:id="@+id/id_container"
6.
7.      >
8.
9.      <ImageView
10.         android:id="@+id/id_ball"
11.         android:layout_width="wrap_content"
12.         android:layout_height="wrap_content"
13.         android:layout_centerInParent="true"
14.         android:src="@drawable/bol_blue" />
15.
16.      <LinearLayout
17.         android:layout_width="fill_parent"
18.         android:layout_height="wrap_content"
19.         android:layout_alignParentBottom="true"
20.         android:orientation="horizontal" >
21.
22.         <Button
23.            android:layout_width="wrap_content"
24.            android:layout_height="wrap_content"
25.            android:onClick="togetherRun"
26.            android:text="简单的多动画Together" />
27.
28.         <Button
29.            android:layout_width="wrap_content"
30.            android:layout_height="wrap_content"
31.            android:onClick="playWithAfter"
32.            android:text="多动画按次序执行" />
33.
34.
35.      </LinearLayout>
36.
37. </RelativeLayout>

```

继续玩球~

代码:

[java] view plaincopy

```
1. package com.example.zhy_property_animation;
2.
3. import android.animation.AnimatorSet;
4. import android.animation.ObjectAnimator;
5. import android.app.Activity;
6. import android.os.Bundle;
7. import android.view.View;
8. import android.view.animation.LinearInterpolator;
9. import android.widget.ImageView;
10.
11. public class AnimatorSetActivity extends Activity
12. {
13.     private ImageView mBlueBall;
14.
15.     @Override
16.     protected void onCreate(Bundle savedInstanceState)
17.     {
18.         super.onCreate(savedInstanceState);
19.         setContentView(R.layout.anim_set);
20.
21.         mBlueBall = (ImageView) findViewById(R.id.id_ball
22.     );
23.     }
24.
25.     public void togetherRun(View view)
26.     {
27.
28.         ObjectAnimator anim1 = ObjectAnimator.ofFloat(mBl
```

```

        ueBall, "scaleX",
28.            1.0f, 2f);
29.
        ObjectAnimator anim2 = ObjectAnimator.ofFloat(mBl
ueBall, "scaleY",
30.            1.0f, 2f);
31.        AnimatorSet animSet = new AnimatorSet();
32.        animSet.setDuration(2000);
33.
        animSet.setInterpolator(new LinearInterpolator())
        ;
34.        //两个动画同时执行
35.        animSet.playTogether(anim1, anim2);
36.        animSet.start();
37.    }
38.
39.    public void playWithAfter(View view)
40.    {
41.        float cx = mBlueBall.getX();
42.
43.
        ObjectAnimator anim1 = ObjectAnimator.ofFloat(mBl
ueBall, "scaleX",
44.            1.0f, 2f);
45.
        ObjectAnimator anim2 = ObjectAnimator.ofFloat(mBl
ueBall, "scaleY",
46.            1.0f, 2f);
47.
        ObjectAnimator anim3 = ObjectAnimator.ofFloat(mBl
ueBall,
48.            "x", cx , 0f);
49.
        ObjectAnimator anim4 = ObjectAnimator.ofFloat(mBl
ueBall,
50.            "x", cx);
51.
52.        /**
53.        * anim1, anim2, anim3同时执行

```

```
54.      * anim4接着执行
55.      */
56.      AnimatorSet animSet = new AnimatorSet();
57.      animSet.play(anim1).with(anim2);
58.      animSet.play(anim2).with(anim3);
59.      animSet.play(anim4).after(anim3);
60.      animSet.setDuration(1000);
61.      animSet.start();
62.  }
63. }
```

写了两个效果：

第一：使用playTogether两个动画同时执行，当然还有playSequentially依次执行~~

第二：如果我们有一堆动画，如何使用代码控制顺序，比如1，2同时；3在2后面；4在1之前等~就是效果2了

有一点注意：animSet.play().with();也是支持链式编程的，但是不要想着狂点，比如

animSet.play(anim1).with(anim2).before(anim3).before(anim5); 这样是不行的，系统不会根据你写的这一长串来决定先后的顺序，所以麻烦你按照上面例子的写法，多写几行：

效果图：

好了，由于篇幅~~关于属性动画还有点知识：

1、xml文件创建属性动画

2、布局动画

3、View的animate方法等。

那就考虑写到下一篇了，不过核心的功能就这些了~~

对了，如果使用11以下的SDK，请导入nineoldandroids动画库，用法基本完全一致~