上一篇Android 属性动画（Property Animation） 完全解析 （上）已经基本展示了属性动画的核心用法：

ObjectAnimator实现动画，ValueAnimator实现动画，AnimatorSet的使用等~

当然了属性动画还有一部分的知识点，也能做出很不错的效果，将在本篇博客为您展示~

1、如何使用xml文件来创建属性动画

大家肯定都清楚，View Animator 、Drawable Animator都可以在anim文件夹下创建动画，然后在程序中使用，甚至在Theme中设置为属性值。当然了，属性动画其实也可以在文件中声明：

首先在res下建立animator文件夹，然后建立res/animator/scalex.xml

[html] view plaincopy

```
<?xml version="1.0" encoding="utf-8"?>
<objectAnimator xmlns:android="http://schemas.android.com/apk/res/android"
    android:duration="1000"
    android:propertyName="scaleX"
    android:valueFrom="1.0"
    android:valueTo="2.0"
    android:valueType="floatType" >
</objectAnimator>
```

代码：

[java] view plaincopy

```java
public void scaleX(View view)
    {
        // 加载动画
        Animator anim = AnimatorInflater.loadAnimator(this, R.animator.scalex);
        anim.setTarget(mMv);
        anim.start();
    }
```

使用AnimatorInflater加载动画的资源文件，然后设置目标，就ok~~是不是很简单，这只是单纯横向的放大一倍~

如果我希望纵向与横向同时缩放呢？则可以怎么定义属性文件：

[html] view plaincopy

```xml
<?xml version="1.0" encoding="utf-8"?>
<set xmlns:android="http://schemas.android.com/apk/res/android"
    android:ordering="together" >

    <objectAnimator
        android:duration="1000"
        android:propertyName="scaleX"
        android:valueFrom="1"
        android:valueTo="0.5" >
    </objectAnimator>
    <objectAnimator
        android:duration="1000"
        android:propertyName="scaleY"
        android:valueFrom="1"
        android:valueTo="0.5" >
    </objectAnimator>

</set>
```

使用set标签，有一个orderring属性设置为together,【还有另一个值：sequentially（表示一个接一个执行）】。

上篇博客中忽略了一个效果，就是缩放、反转等都有中心点或者轴，默认中心缩放，和中间对称线为反转线，所以我决定这个横向，纵向缩小以左上角为中心点：

代码：

[java] view plaincopy

```java
// 加载动画
        Animator anim = AnimatorInflater.loadAnimator(this, R.animator.scale);
        mMv.setPivotX(0);
        mMv.setPivotY(0);
        //显示的调用invalidate
        mMv.invalidate();
        anim.setTarget(mMv);
        anim.start();
```

很简单，直接给View设置pivotX和pivotY，然后调用一下invalidate，就ok了。

下面看效果图：

好了，通过写xml声明动画，使用set嵌套set，结合orderring属性，也基本可以实现任何动画~~上面也演示了pivot的设置。

2、布局动画（Layout Animations）

主要使用LayoutTransition为布局的容器设置动画，当容器中的视图层次发生变化时存在过渡的动画效果。

基本代码为：

[java] view plaincopy

```java
LayoutTransition transition = new LayoutTransition();
    transition.setAnimator(LayoutTransition.CHANGE_APPEARING,
            transition.getAnimator(LayoutTransition.CHANGE_APPEARING));
    transition.setAnimator(LayoutTransition.APPEARING,
            null);
    transition.setAnimator(LayoutTransition.DISAPPEARING,
                null);
    transition.setAnimator(LayoutTransition.CHANGE_DISAPPEARING,
                null);
    mGridLayout.setLayoutTransition(transition);
```

过渡的类型一共有四种：

LayoutTransition.APPEARING 当一个View在ViewGroup中出现时，对**此View**设置的动画

LayoutTransition.CHANGE_APPEARING 当一个View在ViewGroup中出现时，对此View对其他View位置造成影响，对**其他View**设置的动画

LayoutTransition.DISAPPEARING 当一个View在ViewGroup中消失时，对**此View**设置的动画

LayoutTransition.CHANGE_DISAPPEARING 当一个View在ViewGroup中消失时，对此View对其他View位置造成影响，对**其他View**设置的动画

LayoutTransition.CHANGE 不是由于View出现或消失造成对其他View位置造成影响，然后对其他View设置的动画。

注意动画到底设置在谁身上，此View还是其他View。

好了下面看一个综合的例子：

布局文件：

[html] view plaincopy

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/id_container"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical" >

    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:onClick="addBtn"
        android:text="addBtns" />

    <CheckBox
        android:id="@+id/id_appear"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:checked="true"
        android:text="APPEARING" />

    <CheckBox
        android:id="@+id/id_change_appear"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:checked="true"
        android:text="CHANGE_APPEARING" />

    <CheckBox
        android:id="@+id/id_disappear"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:checked="true"
        android:text="DISAPPEARING" />

    <CheckBox
          android:id="@+id/id_change_disappear"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:checked="true"
        android:text="CHANGE_DISAPPEARING " />

</LinearLayout>
```

代码:

[java] view plaincopy

```java
package com.example.zhy_property_animation;

import android.animation.LayoutTransition;
import android.app.Activity;
import android.os.Bundle;
import android.view.View;
import android.view.View.OnClickListener;
import android.view.ViewGroup;
import android.widget.Button;
import android.widget.CheckBox;
import android.widget.CompoundButton;
import android.widget.CompoundButton.OnCheckedChangeListener;
import android.widget.GridLayout;

```

```java
public class LayoutAnimaActivity extends Activity implements
        OnCheckedChangeListener
{
    private ViewGroup viewGroup;
    private GridLayout mGridLayout;
    private int mVal;
    private LayoutTransition mTransition;

    private CheckBox mAppear, mChangeAppear, mDisAppear, mChangeDisAppear;

    @Override
    public void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.layout_animator);
        viewGroup = (ViewGroup) findViewById(R.id.id_container);

        mAppear = (CheckBox) findViewById(R.id.id_appear);
        mChangeAppear = (CheckBox) findViewById(R.id.id_change_appear);
        mDisAppear = (CheckBox) findViewById(R.id.id_disappear);
        mChangeDisAppear = (CheckBox) findViewById(R.id.id_change_disappear);

        mAppear.setOnCheckedChangeListener(this);
        mChangeAppear.setOnCheckedChangeListener(this);
        mDisAppear.setOnCheckedChangeListener(this);
        mChangeDisAppear.setOnCheckedChangeListener(this);

        // 创建一个GridLayout
        mGridLayout = new GridLayout(this);
        // 设置每列5个按钮
        mGridLayout.setColumnCount(5);
        // 添加到布局中
        viewGroup.addView(mGridLayout);
        //默认动画全部开启
        mTransition = new LayoutTransition();
        mGridLayout.setLayoutTransition(mTransition);

    }

    /**
     * 添加按钮
     *
     * @param view
     */
    public void addBtn(View view)
    {
        final Button button = new Button(this);
        button.setText((++mVal) + "");
        mGridLayout.addView(button, Math.min(1, mGridLayout.getChildCount()));
        button.setOnClickListener(new OnClickListener()
        {

            @Override
            public void onClick(View v)
            {
                mGridLayout.removeView(button);
            }
        });
    }

    @Override
    public void onCheckedChanged(CompoundButton buttonView, boolean isChecked)
    {
        mTransition = new LayoutTransition();
        mTransition.setAnimator(
                LayoutTransition.APPEARING,
                (mAppear.isChecked() ? mTransition
                        .getAnimator(LayoutTransition.APPEARING) : null));
        mTransition
                .setAnimator(
                        LayoutTransition.CHANGE_APPEARING,
                        (mChangeAppear.isChecked() ? mTransition
                                .getAnimator(LayoutTransition.CHANGE_APPEARING)
                                : null));
        mTransition.setAnimator(
                LayoutTransition.DISAPPEARING,
                (mDisAppear.isChecked() ? mTransition
                        .getAnimator(LayoutTransition.DISAPPEARING) : null));
        mTransition.setAnimator(
                LayoutTransition.CHANGE_DISAPPEARING,
                (mChangeDisAppear.isChecked() ? mTransition
                        .getAnimator(LayoutTransition.CHANGE_DISAPPEARING)
                        : null));
        mGridLayout.setLayoutTransition(mTransition);
    }
    }
```

效果图：

动画有点长，耐心点看，一定要注意，是对当前View还是其他Views设置的动画。

当然了动画支持自定义，还支持设置时间，比如我们修改下，添加的动画为：

[java] view plaincopy

```java
mTransition.setAnimator(LayoutTransition.APPEARING, (mAppear
                .isChecked() ? ObjectAnimator.ofFloat(this, "scaleX", 0, 1)
                : null));
```

则效果为：

原本的淡入，变成了宽度从中间放大的效果~~是不是还不错~~

3、View的anim方法

在SDK11的时候，给View添加了animate方法，更加方便的实现动画效果。

布局文件：

[html] view plaincopy

```html
<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"
    >

    <ImageView
        android:id="@+id/id_ball"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:src="@drawable/bol_blue" />

    <LinearLayout
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:layout_alignParentBottom="true"
        android:orientation="horizontal" >

        <Button
            android:layout_width="wrap_content"
```

```
                android:layout_height="wrap_content"
                android:onClick="viewAnim"
                android:text="View Anim" />

            <Button
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:onClick="propertyValuesHolder"
                android:text="PropertyValuesHolder " />


    </LinearLayout>

</RelativeLayout>
```

代码:

[java] view plaincopy

```java
package com.example.zhy_property_animation;

import android.animation.ObjectAnimator;
import android.animation.PropertyValuesHolder;
import android.app.Activity;
import android.os.Bundle;
import android.util.DisplayMetrics;
import android.util.Log;
import android.view.View;
import android.widget.ImageView;

public class ViewAnimateActivity extends Activity
{
    protected static final String TAG = "ViewAnimateActivity";

    private ImageView mBlueBall;
    private float mScreenHeight;

    @Override
    protected void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.view_animator);

        DisplayMetrics outMetrics = new DisplayMetrics();
        getWindowManager().getDefaultDisplay().getMetrics(outMetrics);
        mScreenHeight = outMetrics.heightPixels;
        mBlueBall = (ImageView) findViewById(R.id.id_ball);

    }

    public void viewAnim(View view)
    {
        // need API12
        mBlueBall.animate()//
                .alpha(0)//
                .y(mScreenHeight / 2).setDuration(1000)
                // need API 12
                .withStartAction(new Runnable()
                {
                    @Override
                    public void run()
                    {
                        Log.e(TAG, "START");
                    }
                    // need API 16
                }).withEndAction(new Runnable()
```

```
            {

                @Override
                public void run()
                {
                    Log.e(TAG, "END");
                    runOnUiThread(new Runnable()
                    {
                        @Override
                        public void run()
                        {
                            mBlueBall.setY(0);
                            mBlueBall.setAlpha(1.0f);
                        }
                    });
                }
            }).start();

}

                                  }
```

简单的使用mBlueBall.animate().alpha(0).y(mScreenHeight / 2).setDuration(1000).start()就能实现动画~~不过需要SDK11，此后在SDK12，SDK16又分别添加了withStartAction和withEndAction用于在动画前，和动画后执行一些操作。当然也可以.setListener(listener)等操作。

使用ObjectAnimator实现上面的变化，我们可以使用：PropertyValueHolder

[java] view plaincopy

```java
PropertyValuesHolder pvhX = PropertyValuesHolder.ofFloat("alpha", 1f,
            0f, 1f);
    PropertyValuesHolder pvhY = PropertyValuesHolder.ofFloat("y", 0,
            mScreenHeight / 2, 0);
    ObjectAnimator.ofPropertyValuesHolder(mBlueBall, pvhX, pvhY).setDuration(1000).start();
```

效果与上面一样。