## 一、概述

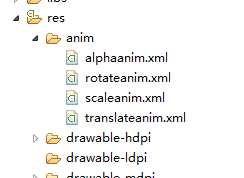
**[Android](http://lib.csdn.net/base/android" \o "Android知识库" \t "http://blog.csdn.net/harvic880925/article/details/_blank)**的animation由四种类型组成：alpha、scale、translate、rotate，对应**[android](http://lib.csdn.net/base/android" \o "Android知识库" \t "http://blog.csdn.net/harvic880925/article/details/_blank)**官方文档地址：[《Animation Resources》](http://developer.android.com/guide/topics/resources/animation-resource.html" \t "http://blog.csdn.net/harvic880925/article/details/_blank)

#### ****1、XML配置文件中****

|  |  |
| --- | --- |
| alpha | 渐变透明度动画效果 |
| scale | 渐变尺寸伸缩动画效果 |
| translate | 画面转换位置移动动画效果 |
| rotate | 画面转移旋转动画效果 |

下面我们逐个讲讲每个标签的属性及用法。

#### ****2、动作文件存放位置****

动作定义文件应该存放在res/anim文件夹下，访问时采用R.anim.XXX.xml的方式，位置如图：  


## 二、scale标签——调节尺寸

### 1、自有属性

scale标签是缩放动画，可以实现动态调控件尺寸的效果，有下面几个属性：

* android:fromXScale    起始的X方向上相对自身的缩放比例，浮点值，比如1.0代表自身无变化，0.5代表起始时缩小一倍，2.0代表放大一倍；
* android:toXScale        结尾的X方向上相对自身的缩放比例，浮点值；
* android:fromYScale    起始的Y方向上相对自身的缩放比例，浮点值，
* android:toYScale        结尾的Y方向上相对自身的缩放比例，浮点值；
* android:pivotX            缩放起点X轴坐标，可以是数值、百分数、百分数p 三种样式，比如 50、50%、50%p，当为数值时，表示在当前View的左上角，即原点处加上50px，做为起始缩放点；如果是50%，表示在当前控件的左上角加上自己宽度的50%做为起始点；如果是50%p，那么就是表示在当前的左上角加上父控件宽度的50%做为起始点x轴坐标。（具体意义，后面会举例演示）
* android:pivotY           缩放起点Y轴坐标，取值及意义跟android:pivotX一样。

下面看一个实例，当scale里的属性这样设置时，效果会怎样呢：

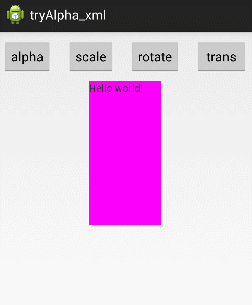
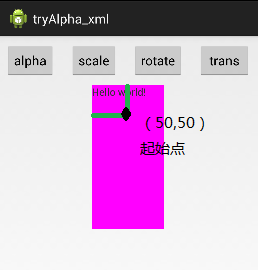
**[html]** [view plain](http://blog.csdn.net/harvic880925/article/details/39996643" \o "view plain" \t "http://blog.csdn.net/harvic880925/article/details/_blank) [copy](http://blog.csdn.net/harvic880925/article/details/39996643" \o "copy" \t "http://blog.csdn.net/harvic880925/article/details/_blank)

1. **<?xml** version="1.0" encoding="utf-8"**?>**
2. **<scale** xmlns:android="http://schemas.android.com/apk/res/android"
3. android:fromXScale="0.0"
4. android:toXScale="1.4"
5. android:fromYScale="0.0"
6. android:toYScale="1.4"
7. android:pivotX="50"
8. android:pivotY="50"
9. android:duration="700" **/>**

**（1）、pivotX取值数值时（50）**

这个控件，宽度和高度都是从0放大到1.4倍，起始点坐标在控件左上角（坐标原点），向x轴正方向和y轴正方向都加上50像素；

根据pivotX,pivotY的意义，控件的左上角即为控件的坐标原点，这里的起始点是在控件的原点的基础上向X轴和Y轴各加上50px，做为起始点，如下图中图二所示

                               图一                                                             图二  
    

**（2）、pivotX取值百分数时（50%）**  
下面再看看当pivotX、pivotY取百分数的时候，起始点又在哪里？

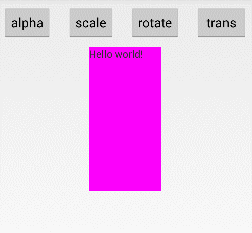
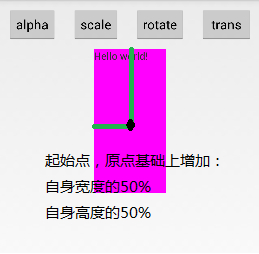
上面我们讲了，pivotX的值，当取50%时，表示在原点坐标的基础上加上的自己宽度的50%，看看效果：

**[html]** [view plain](http://blog.csdn.net/harvic880925/article/details/39996643" \o "view plain" \t "http://blog.csdn.net/harvic880925/article/details/_blank) [copy](http://blog.csdn.net/harvic880925/article/details/39996643" \o "copy" \t "http://blog.csdn.net/harvic880925/article/details/_blank)

1. **<?xml** version="1.0" encoding="utf-8"**?>**
2. **<scale** xmlns:android="http://schemas.android.com/apk/res/android"
3. android:fromXScale="0.0"
4. android:toXScale="1.4"
5. android:fromYScale="0.0"
6. android:toYScale="1.4"
7. android:pivotX="50%"
8. android:pivotY="50%"
9. android:duration="700" **/>**

缩放位置大小仍然从0-1.4，只改变pivotX和pivotY;起始点位置如下图中图二所示：

                               图一                                                                 图二

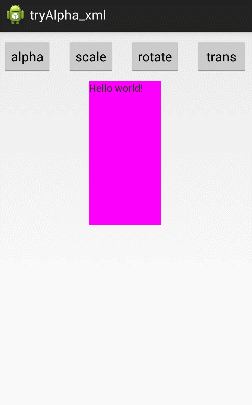
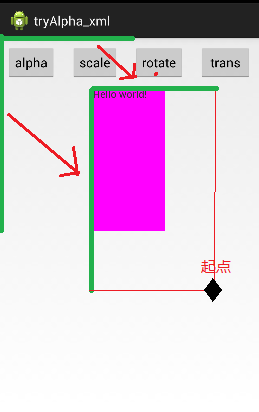
     
**（3）、pivotX取值50%p时**

前面说过，当取值在百分数后面加上一个字母p，就表示，取值的基数是父控件，即在原点的基础上增加的值是父标签的百分值。

**[html]** [view plain](http://blog.csdn.net/harvic880925/article/details/39996643" \o "view plain" \t "http://blog.csdn.net/harvic880925/article/details/_blank) [copy](http://blog.csdn.net/harvic880925/article/details/39996643" \o "copy" \t "http://blog.csdn.net/harvic880925/article/details/_blank)

1. **<?xml** version="1.0" encoding="utf-8"**?>**
2. **<scale** xmlns:android="http://schemas.android.com/apk/res/android"
3. android:fromXScale="0.0"
4. android:toXScale="1.4"
5. android:fromYScale="0.0"
6. android:toYScale="1.4"
7. android:pivotX="50%p"
8. android:pivotY="50%p"
9. android:duration="700" **/>**

效果图，及起始点坐标图如下所示：

### 2、从Animation类继承的属性

Animation类是所有动画（scale、alpha、translate、rotate）的基类，这里以scale标签为例，讲解一下，Animation类所具有的属性及意义。关于Animation类的官方文档位置为：[《Animation》](http://developer.android.com/reference/android/view/animation/Animation.html" \t "http://blog.csdn.net/harvic880925/article/details/_blank)

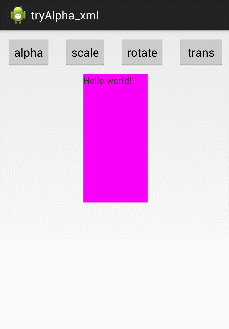
* android:duration        动画持续时间，以毫秒为单位
* android:fillAfter          如果设置为true，控件动画结束时，将保持动画最后时的状态
* android:fillBefore       如果设置为true,控件动画结束时，还原到开始动画前的状态
* android:fillEnabled    与android:fillBefore 效果相同，都是在动画结束时，将控件还原到初始化状态
* android:repeatCount 重复次数
* android:repeatMode 重复类型，有reverse和restart两个值，reverse表示倒序回放，restart表示重新放一遍，必须与repeatCount一起使用才能看到效果。因为这里的意义是重复的类型，即回放时的动作。
* android:interpolator  设定插值器，其实就是指定的动作效果，比如弹跳效果等，不在这小节中讲解，后面会单独列出一单讲解。

对于android:duration，就不再讲解了，就是动画的持续时长，以毫秒为单位，下面看看android:fillAfter和android:fillBefore

#### (1)android:fillAfter:保持动画结束的状态

**[html]** [view plain](http://blog.csdn.net/harvic880925/article/details/39996643" \o "view plain" \t "http://blog.csdn.net/harvic880925/article/details/_blank) [copy](http://blog.csdn.net/harvic880925/article/details/39996643" \o "copy" \t "http://blog.csdn.net/harvic880925/article/details/_blank)

1. **<?xml** version="1.0" encoding="utf-8"**?>**
2. **<scale** xmlns:android="http://schemas.android.com/apk/res/android"
3. android:fromXScale="0.0"
4. android:toXScale="1.4"
5. android:fromYScale="0.0"
6. android:toYScale="1.4"
7. android:pivotX="50%"
8. android:pivotY="50%"
9. android:duration="700"
10. android:fillAfter="true"
11. **/>**

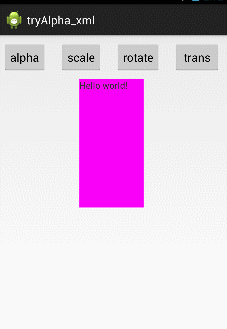
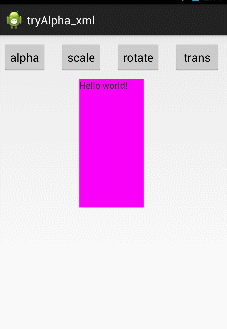


#### （2）android:fillBefore  还原初始化状态

**[html]** [view plain](http://blog.csdn.net/harvic880925/article/details/39996643" \o "view plain" \t "http://blog.csdn.net/harvic880925/article/details/_blank) [copy](http://blog.csdn.net/harvic880925/article/details/39996643" \o "copy" \t "http://blog.csdn.net/harvic880925/article/details/_blank)

1. **<?xml** version="1.0" encoding="utf-8"**?>**
2. **<scale** xmlns:android="http://schemas.android.com/apk/res/android"
3. android:fromXScale="0.0"
4. android:toXScale="1.4"
5. android:fromYScale="0.0"
6. android:toYScale="1.4"
7. android:pivotX="50%"
8. android:pivotY="50%"
9. android:duration="700"
10. android:fillBefore="true"
11. **/>**

                android:fillBefore="true"                                 android:fillEnable="true"

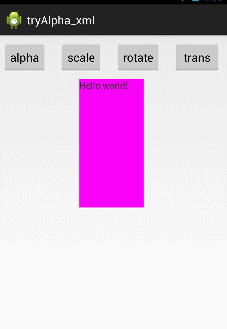
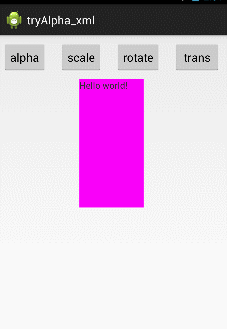
   

上面顺便列出了，当仅设定fillEanble为true时的效果，这两个的标签的效果完全相同。

#### （3）、android:repeatMode="restart /reverse"  设定回放类型

**[html]** [view plain](http://blog.csdn.net/harvic880925/article/details/39996643" \o "view plain" \t "http://blog.csdn.net/harvic880925/article/details/_blank) [copy](http://blog.csdn.net/harvic880925/article/details/39996643" \o "copy" \t "http://blog.csdn.net/harvic880925/article/details/_blank)

1. **<?xml** version="1.0" encoding="utf-8"**?>**
2. **<scale** xmlns:android="http://schemas.android.com/apk/res/android"
3. android:fromXScale="0.0"
4. android:toXScale="1.4"
5. android:fromYScale="0.0"
6. android:toYScale="1.4"
7. android:pivotX="50%"
8. android:pivotY="50%"
9. android:duration="700"
10. android:fillBefore="true"
11. android:repeatCount="1"
12. android:repeatMode="restart"
13. **/>**

        androidRepeatMode设为restart                       androidRepeatMode设为reverse  
       

## 三、alpha标签——调节透明度

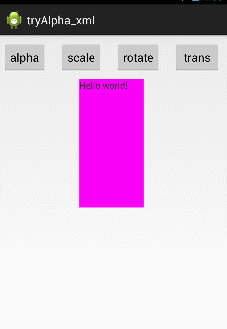
### 1、自身属性

* android:fromAlpha   动画开始的透明度，从0.0 --1.0 ，0.0表示全透明，1.0表示完全不透明
* android:toAlpha       动画结束时的透明度，也是从0.0 --1.0 ，0.0表示全透明，1.0表示完全不透明

使用示例：

**[html]** [view plain](http://blog.csdn.net/harvic880925/article/details/39996643" \o "view plain" \t "http://blog.csdn.net/harvic880925/article/details/_blank) [copy](http://blog.csdn.net/harvic880925/article/details/39996643" \o "copy" \t "http://blog.csdn.net/harvic880925/article/details/_blank)

1. **<?xml** version="1.0" encoding="utf-8"**?>**
2. **<alpha** xmlns:android="http://schemas.android.com/apk/res/android"
3. android:fromAlpha="1.0"
4. android:toAlpha="0.1"
5. android:duration="3000"
6. android:fillBefore="true"**>**
7. **</alpha>**



### 2、从Animation类继承的属性

* android:duration        动画持续时间，以毫秒为单位
* android:fillAfter          如果设置为true，控件动画结束时，将保持动画最后时的状态
* android:fillBefore       如果设置为true,控件动画结束时，还原到开始动画前的状态
* android:fillEnabled    与android:fillBefore 效果相同，都是在动画结束时，将控件还原到初始化状态
* android:repeatCount 重复次数
* android:repeatMode 重复类型，有reverse和restart两个值，reverse表示倒序回放，restart表示重新放一遍，必须与repeatCount一起使用才能看到效果。因为这里的意义是重复的类型，即回放时的动作。
* android:interpolator  设定插值器，其实就是指定的动作效果，比如弹跳效果等，不在这小节中讲解，后面会单独列出一单讲解。

与scale标签意义一样，就不再缀述。

## 四、rotate标签——旋转

### 1、自身属性

* android:fromDegrees     开始旋转的角度位置，正值代表顺时针方向度数，负值代码逆时针方向度数
* android:toDegrees         结束时旋转到的角度位置，正值代表顺时针方向度数，负值代码逆时针方向度数
* android:pivotX               缩放起点X轴坐标，可以是数值、百分数、百分数p 三种样式，比如 50、50%、50%p，具体意义已在scale标签中讲述，这里就不再重讲
* android:pivotY               缩放起点Y轴坐标，可以是数值、百分数、百分数p 三种样式，比如 50、50%、50%p

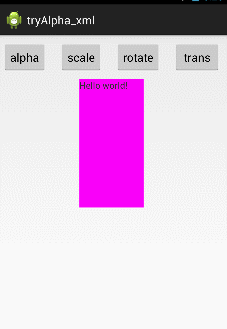
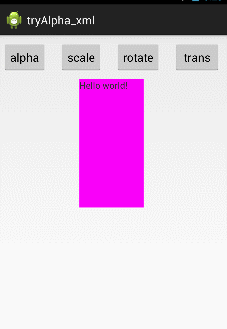
**[html]** [view plain](http://blog.csdn.net/harvic880925/article/details/39996643" \o "view plain" \t "http://blog.csdn.net/harvic880925/article/details/_blank) [copy](http://blog.csdn.net/harvic880925/article/details/39996643" \o "copy" \t "http://blog.csdn.net/harvic880925/article/details/_blank)

1. **<?xml** version="1.0" encoding="utf-8"**?>**
2. **<rotate** xmlns:android="http://schemas.android.com/apk/res/android"
3. android:fromDegrees="0"
4. android:toDegrees="-650"
5. android:pivotX="50%"
6. android:pivotY="50%"
7. android:duration="3000"
8. android:fillAfter="true"**>**
10. **</rotate>**

围绕自身从0度逆时针旋转650度                            围绕自身从0度顺时针旋转650度

android:fromDegrees="0"                                       android:fromDegrees="0"

android:toDegrees="-650"                                      android:toDegrees="650"

### 2、从Animation类继承的属性

* android:duration        动画持续时间，以毫秒为单位
* android:fillAfter          如果设置为true，控件动画结束时，将保持动画最后时的状态
* android:fillBefore       如果设置为true,控件动画结束时，还原到开始动画前的状态
* android:fillEnabled    与android:fillBefore 效果相同，都是在动画结束时，将控件还原到初始化状态
* android:repeatCount 重复次数
* android:repeatMode 重复类型，有reverse和restart两个值，reverse表示倒序回放，restart表示重新放一遍，必须与repeatCount一起使用才能看到效果。因为这里的意义是重复的类型，即回放时的动作。
* android:interpolator  设定插值器，其实就是指定的动作效果，比如弹跳效果等，不在这小节中讲解，后面会单独列出一单讲解。

与scale标签意义一样，就不再缀述。

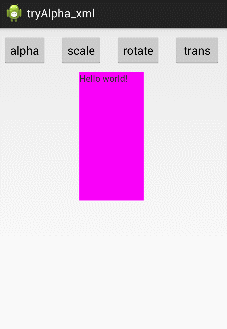
## 五、translate标签 —— 平移

### 1、自身属性

* android:fromXDelta     起始点X轴坐标，可以是数值、百分数、百分数p 三种样式，比如 50、50%、50%p，具体意义已在scale标签中讲述，这里就不再重讲
* android:fromYDelta    起始点Y轴从标，可以是数值、百分数、百分数p 三种样式；
* android:toXDelta         结束点X轴坐标
* android:toYDelta        结束点Y轴坐标

**[html]** [view plain](http://blog.csdn.net/harvic880925/article/details/39996643" \o "view plain" \t "http://blog.csdn.net/harvic880925/article/details/_blank) [copy](http://blog.csdn.net/harvic880925/article/details/39996643" \o "copy" \t "http://blog.csdn.net/harvic880925/article/details/_blank)

1. **<?xml** version="1.0" encoding="utf-8"**?>**
2. **<translate** xmlns:android="http://schemas.android.com/apk/res/android"
3. android:fromXDelta="0"
4. android:toXDelta="-80"
5. android:fromYDelta="0"
6. android:toYDelta="-80"
7. android:duration="2000"
8. android:fillBefore="true"**>**
9. **</translate>**



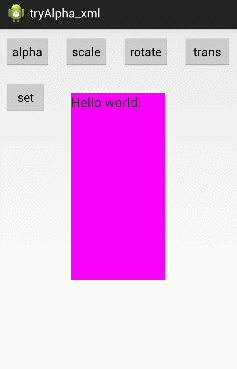
### 2、从Animation类继承的属性

* android:duration        动画持续时间，以毫秒为单位
* android:fillAfter          如果设置为true，控件动画结束时，将保持动画最后时的状态
* android:fillBefore       如果设置为true,控件动画结束时，还原到开始动画前的状态
* android:fillEnabled    与android:fillBefore 效果相同，都是在动画结束时，将控件还原到初始化状态
* android:repeatCount 重复次数
* android:repeatMode 重复类型，有reverse和restart两个值，reverse表示倒序回放，restart表示重新放一遍，必须与repeatCount一起使用才能看到效果。因为这里的意义是重复的类型，即回放时的动作。
* android:interpolator  设定插值器，其实就是指定的动作效果，比如弹跳效果等，不在这小节中讲解，后面会单独列出一单讲解。

与scale标签意义一样，就不再缀述。

## 六、set标签——定义动作合集

前面我们讲解了各个标签动画的意义及用法，但他们都是独立对控件起作用，假设我现在想上面的textView控件做一个动画——从小到大，旋转出场，而且透明度也要从0变成1，即下面的这个效果，该怎么办？



这就需要对指定的控件定义动作合集，Set标签就可以将几个不同的动作定义成一个组；

### 属性：

set标签自已是没有属性的，他的属性都是从Animation继承而来，但当它们用于Set标签时，就会对Set标签下的所有子控件都产生作用。

属性有：（从Animation类继承的属性）

* android:duration        动画持续时间，以毫秒为单位
* android:fillAfter          如果设置为true，控件动画结束时，将保持动画最后时的状态
* android:fillBefore       如果设置为true,控件动画结束时，还原到开始动画前的状态
* android:fillEnabled    与android:fillBefore 效果相同，都是在动画结束时，将控件还原到初始化状态
* android:repeatCount 重复次数
* android:repeatMode 重复类型，有reverse和restart两个值，reverse表示倒序回放，restart表示重新放一遍，必须与repeatCount一起使用才能看到效果。因为这里的意义是重复的类型，即回放时的动作。
* android:interpolator  设定插值器，其实就是指定的动作效果，比如弹跳效果等，不在这小节中讲解，后面会单独列出一单讲解。

与scale标签意义一样，就不再缀述。

上面这个效果，所对应的XML代码为：

**[html]** [view plain](http://blog.csdn.net/harvic880925/article/details/39996643" \o "view plain" \t "http://blog.csdn.net/harvic880925/article/details/_blank) [copy](http://blog.csdn.net/harvic880925/article/details/39996643" \o "copy" \t "http://blog.csdn.net/harvic880925/article/details/_blank)

1. **<?xml** version="1.0" encoding="utf-8"**?>**
2. **<set** xmlns:android="http://schemas.android.com/apk/res/android"
3. android:duration="3000"
4. android:fillAfter="true"**>**
6. **<alpha**
7. android:fromAlpha="0.0"
8. android:toAlpha="1.0"**/>**
10. **<scale**
11. android:fromXScale="0.0"
12. android:toXScale="1.4"
13. android:fromYScale="0.0"
14. android:toYScale="1.4"
15. android:pivotX="50%"
16. android:pivotY="50%"**/>**
18. **<rotate**
19. android:fromDegrees="0"
20. android:toDegrees="720"
21. android:pivotX="50%"
22. android:pivotY="50%"**/>**
24. **</set>**

## 七、实例——如何将动画XML文件应用于控件中

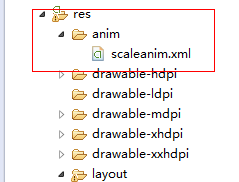
上面我仅仅是列出了每个标签及其属性的意义及应用之后的效果演示，但上面是如何将定义动画的xml应用到textView控件中的却迟迟没说，这一小节，就以scale动画为例，讲述如何将定义好的scle动作添加到指定控件中。

**先看最终效果图：**



### 1、新建工程、新建scale动画文件（scaleanim.xml）

新建一个工程，并且在res文件夹下，新建一个anim文件夹，然后再新建一个scaleanim.xml文件，结构如图所示：



scaleanim.xml的代码为：（从TextView中心点，从0放大到1.4倍，反复一次，最后还原到初始化状态）

**[html]** [view plain](http://blog.csdn.net/harvic880925/article/details/39996643" \o "view plain" \t "http://blog.csdn.net/harvic880925/article/details/_blank) [copy](http://blog.csdn.net/harvic880925/article/details/39996643" \o "copy" \t "http://blog.csdn.net/harvic880925/article/details/_blank)

1. **<?xml** version="1.0" encoding="utf-8"**?>**
2. **<scale** xmlns:android="http://schemas.android.com/apk/res/android"
3. android:fromXScale="0.0"
4. android:toXScale="1.4"
5. android:fromYScale="0.0"
6. android:toYScale="1.4"
7. android:pivotX="50%"
8. android:pivotY="50%"
9. android:duration="700"
10. android:fillBefore="true"
11. android:repeatCount="1"
12. android:repeatMode="restart"
13. **/>**

### 2、XML布局文件

**[html]** [view plain](http://blog.csdn.net/harvic880925/article/details/39996643" \o "view plain" \t "http://blog.csdn.net/harvic880925/article/details/_blank) [copy](http://blog.csdn.net/harvic880925/article/details/39996643" \o "copy" \t "http://blog.csdn.net/harvic880925/article/details/_blank)

1. **<LinearLayout** 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:orientation="vertical"
6. tools:context="com.harvic.animation\_demo.MainActivity" **>**
8. **<Button** android:id="@+id/btn\_animation"
9. android:layout\_width="match\_parent"
10. android:layout\_height="wrap\_content"
11. android:layout\_margin="10dip"
12. android:text="scale animation"**/>**
13. **<TextView**
14. android:id="@+id/tv"
15. android:layout\_width="100dip"
16. android:layout\_height="200dip"
17. android:background="#ff00ff"
18. android:text="@string/hello\_world"
19. android:layout\_gravity="center\_horizontal"**/>**
21. **</LinearLayout>**

### 3、JAVA代码

**[java]** [view plain](http://blog.csdn.net/harvic880925/article/details/39996643" \o "view plain" \t "http://blog.csdn.net/harvic880925/article/details/_blank) [copy](http://blog.csdn.net/harvic880925/article/details/39996643" \o "copy" \t "http://blog.csdn.net/harvic880925/article/details/_blank)

1. **public** **class** MainActivity **extends** Activity {
3. Button scaleBtn ;
4. Animation scaleAnimation;
6. TextView tv;
7. @Override
8. **protected** **void** onCreate(Bundle savedInstanceState) {
9. **super**.onCreate(savedInstanceState);
10. setContentView(R.layout.activity\_main);
12. scaleAnimation = AnimationUtils.loadAnimation(**this**, R.anim.scaleanim);
13. scaleBtn = (Button)findViewById(R.id.btn\_animation);
14. tv =(TextView)findViewById(R.id.tv);
16. scaleBtn.setOnClickListener(**new** View.OnClickListener() {
18. @Override
19. **public** **void** onClick(View v) {
20. // TODO Auto-generated method stub
21. tv.startAnimation(scaleAnimation);
22. }
23. });
25. }
27. }

（1）通过scaleAnimation = AnimationUtils.loadAnimation(this, R.anim.scaleanim);从XML文件中获取动画

（2）利用startAnimation将动画传递给指定控件显示。