Animator属性动画

Android中动画类型：

1. 帧动画：逐帧图片轮放
2. 补间动画：对View进行一系列操作（包括淡入淡出、缩放、平移、旋转四种）
3. 属性动画（Android 3.0开始）：可以对任何对象进行动画操作，而不仅仅是View对象，如操作保存了View属性的Points对象，进而实现View的动画操作，也可以进行动态变换View的背景，它实际上是一种不断地对值进行操作的机制，并将值赋值到指定对象的指定属性上，可以是任意对象的任意属性

补间动画还有一个致命的缺陷，就是它只是改变了View的显示效果而已，而不会真正去改变View的属性

什么意思呢？比如说，现在屏幕的左上角有一个按钮，然后我们通过补间动画将它移动到了屏幕的右下角，现在你可以去尝试点击一下这个按钮，点击事件是绝对不会触发的，因为实际上这个按钮还是停留在屏幕的左上角，只不过补间动画将这个按钮绘制到了屏幕的右下角而已。

1. ValueAnimator ：对值进行操作的动画

将一个值在5秒内从0过渡到5，再过渡到3，再过渡到10

1. ValueAnimator anim = ValueAnimator.ofFloat(0f, 5f, 3f, 10f);
2. anim.setDuration(5000);
3. anim.start();

ValueAnimator animator = ValueAnimator.*ofFloat*(0.0f,1.0f);  
// 设置动画延迟播放的时间  
 animator.setStartDelay(2000);  
 //设置循环播放模式为重新播放  
// animator.setRepeatMode(ValueAnimator.RESTART);  
 //设置循环播放模式为倒序播放  
 animator.setRepeatMode(ValueAnimator.*REVERSE*);  
// 设置动画播放时长  
 animator.setDuration(1000);

ValueAnimator

---ObjectAnimator(继承自ValueAnimator)

1. ObjectAnimator ：对任意对象的任意属性进行动画操作

1）将一个TextView在5秒中内从常规变换成全透明，再从全透明变换成常规

1. ObjectAnimator animator = ObjectAnimator.ofFloat(textview, "alpha", 1f, 0f, 1f);
2. animator.setDuration(5000);
3. animator.start();

2）将TextView进行一次360度的旋转

1. ObjectAnimator animator = ObjectAnimator.ofFloat(textview, "rotation", 0f, 360f);
2. animator.setDuration(5000);
3. animator.start();

3）将TextView先向左移出屏幕，然后再移动回来

1. **float** curTranslationX = textview.getTranslationX();
2. ObjectAnimator animator = ObjectAnimator.ofFloat(textview, "translationX", curTranslationX, -500f, curTranslationX);
3. animator.setDuration(5000);
4. animator.start();
5. 将TextView在垂直方向上放大3倍再还原
6. ObjectAnimator animator = ObjectAnimator.ofFloat(textview, "scaleY", 1f, 3f, 1f);
7. animator.setDuration(5000);
8. animator.start();

组合动画：

AnimatorSet

* after(Animator anim)   将现有动画插入到传入的动画之后执行
* after(long delay)   将现有动画延迟指定毫秒后执行
* before(Animator anim)   将现有动画插入到传入的动画之前执行
* with(Animator anim)   将现有动画和传入的动画同时执行

让TextView先从屏幕外移动进屏幕，然后开始旋转360度，旋转的同时进行淡入淡出操作:

1. ObjectAnimator moveIn = ObjectAnimator.ofFloat(textview, "translationX", -500f, 0f);
2. ObjectAnimator rotate = ObjectAnimator.ofFloat(textview, "rotation", 0f, 360f);
3. ObjectAnimator fadeInOut = ObjectAnimator.ofFloat(textview, "alpha", 1f, 0f, 1f);
4. AnimatorSet animSet = **new** AnimatorSet();
5. animSet.play(rotate).with(fadeInOut).after(moveIn);
6. animSet.setDuration(5000);
7. animSet.start();

这里我们先是把三个动画的对象全部创建出来，然后new出一个AnimatorSet对象之后将这三个动画对象进行播放排序，让旋转和淡入淡出动画同时进行，并把它们插入到了平移动画的后面，最后是设置动画时长以及启动动画

设置动画的监听器：

1.使用addListener()

1. anim.addListener(**new** AnimatorListener() {
2. @Override
3. **public** **void** onAnimationStart(Animator animation) {
4. }
6. @Override
7. **public** **void** onAnimationRepeat(Animator animation) {
8. }
10. @Override
11. **public** **void** onAnimationEnd(Animator animation) {
12. }
14. @Override
15. **public** **void** onAnimationCancel(Animator animation) {
16. }
17. });
18. 使用AnimatorListenerAdapter，不用每次都要去实现上述四个方法，避免冗余
19. anim.addListener(**new** AnimatorListenerAdapter() {
20. @Override
21. **public** **void** onAnimationEnd(Animator animation) {
22. }
23. });

## 使用XML编写动画

1. 在res目录下面新建一个animator文件夹

2. XML文件中我们一共可以使用如下三种标签：

* <animator>  对应代码中的ValueAnimator
* <objectAnimator>  对应代码中的ObjectAnimator
* <set>  对应代码中的AnimatorSet

1. 实现一个从0到100平滑过渡的动画，在XML当中就可以这样写：
2. **<animator** xmlns:android="http://schemas.android.com/apk/res/android"
3. android:valueFrom="0"
4. android:valueTo="100"
5. android:valueType="intType"**/>**
6. 将一个视图的alpha属性从1变成0
7. **<objectAnimator** xmlns:android="http://schemas.android.com/apk/res/android"
8. android:valueFrom="1"
9. android:valueTo="0"
10. android:valueType="floatType"
11. android:propertyName="alpha"**/>**
12. 将一个视图先从屏幕外移动进屏幕，然后开始旋转360度，旋转的同时进行淡入淡出操作
13. **<set** xmlns:android="http://schemas.android.com/apk/res/android"
14. android:ordering="sequentially" **>**
16. **<objectAnimator**
17. android:duration="2000"
18. android:propertyName="translationX"
19. android:valueFrom="-500"
20. android:valueTo="0"
21. android:valueType="floatType" **>**
22. **</objectAnimator>**
24. **<set** android:ordering="together" **>**
25. **<objectAnimator**
26. android:duration="3000"
27. android:propertyName="rotation"
28. android:valueFrom="0"
29. android:valueTo="360"
30. android:valueType="floatType" **>**
31. **</objectAnimator>**
33. **<set** android:ordering="sequentially" **>**
34. **<objectAnimator**
35. android:duration="1500"
36. android:propertyName="alpha"
37. android:valueFrom="1"
38. android:valueTo="0"
39. android:valueType="floatType" **>**
40. **</objectAnimator>**
41. **<objectAnimator**
42. android:duration="1500"
43. android:propertyName="alpha"
44. android:valueFrom="0"
45. android:valueTo="1"
46. android:valueType="floatType" **>**
47. **</objectAnimator>**
48. **</set>**
49. **</set>**
51. **</set>**

在代码中使用XML属性动画

1. Animator animator = AnimatorInflater.loadAnimator(context, R.animator.anim\_file);
2. animator.setTarget(view);
3. animator.start();

调用AnimatorInflater的loadAnimator来将XML动画文件加载进来，然后再调用setTarget()方法将这个动画设置到某一个对象上面，最后再调用start()方法启动动画就可以了

1. ViewAnimator和ObjectAnimator的高级用法：

博客：http://blog.csdn.net/guolin\_blog/article/details/43816093

即创建自己的TypeEvaluator（实现TypeEvalutor接口中的evaluate方法），去实现控制动画的变化，如控制动画过程中点的移动

1. **public** **class** PointEvaluator **implements** TypeEvaluator{
3. @Override
4. **public** Object evaluate(**float** fraction, Object startValue, Object endValue) {
5. Point startPoint = (Point) startValue;
6. Point endPoint = (Point) endValue;
7. **float** x = startPoint.getX() + fraction \* (endPoint.getX() - startPoint.getX());
8. **float** y = startPoint.getY() + fraction \* (endPoint.getY() - startPoint.getY());
9. Point point = **new** Point(x, y);
10. **return** point;
11. }
13. }

调用：

1. ValueAnimator anim = ValueAnimator.ofObject(**new** PointEvaluator(), startPoint, endPoint);

2. anim.setDuration(5000);

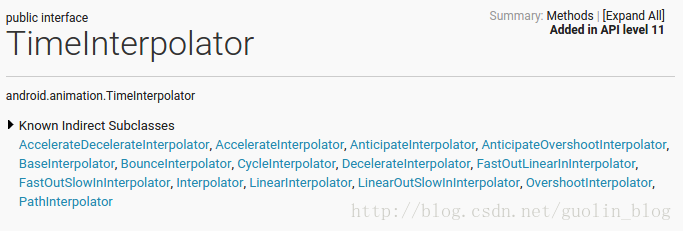
3.  anim.start();

这种比较难写。。。。。

## 三、Interpolator的用法

控制动画变化速率，如加速变化、减速变化、匀速变化

系统提供了很多实现类供我们调用



1. ValueAnimator anim = ValueAnimator.ofObject(**new** PointEvaluator(), startPoint, endPoint);
2. // 设置回弹效果的动画插值器
3. anim.setInterpolator(**new** BounceInterpolator());

所有的插值器实现类都实现了TimeInterpolator接口

## 四、ViewPropertyAnimator的用法

通过面向对象的方式最简单的去实现动画

之前通过借助ValueAnimator和ObjectAnimator这两个类去实现View的动画，

通过ViewPropertyAnimator类可以直接一行代码实现简单及组合动画

将TextView移动到坐标点（500,500）

1. textview.animate().x(500).y(500);

设置动画执行时间

1. textview.animate().x(500).y(500).setDuration(5000);

设置插值器

1. textview.animate().x(500).y(500).setDuration(5000)
2. .setInterpolator(**new** BounceInterpolator());

原理： view.animate() : 该方法获取到ViewPropertyAnimator对象，通过该对象对View完成动画的实现

注意点：

* 整个ViewPropertyAnimator的功能都是建立在View类新增的animate()方法之上的，这个方法会创建并返回一个ViewPropertyAnimator的实例，之后的调用的所有方法，设置的所有属性都是通过这个实例完成的。
* 大家注意到，在使用ViewPropertyAnimator时，我们自始至终没有调用过start()方法，这是因为新的接口中使用了隐式启动动画的功能，只要我们将动画定义完成之后，动画就会自动启动。并且这个机制对于组合动画也同样有效，只要我们不断地连缀新的方法，那么动画就不会立刻执行，等到所有在ViewPropertyAnimator上设置的方法都执行完毕后，动画就会自动启动。当然如果不想使用这一默认机制的话，我们也可以显式地调用start()方法来启动动画。
* ViewPropertyAnimator的所有接口都是使用连缀的语法来设计的，每个方法的返回值都是它自身的实例，因此调用完一个方法之后可以直接连缀调用它的另一个方法，这样把所有的功能都串接起来，我们甚至可以仅通过一行代码就完成任意复杂度的动画功能。