**ObjectAnimator Состояния анимации**

[Методы анимации](http://developer.alexanderklimov.ru/android/animation/objectanimator4.php#methods)  
[Слушатели](http://developer.alexanderklimov.ru/android/animation/objectanimator4.php#listeners)

Класс **Animator** появился относительно недавно, а в API 19 в него были добавлены новые методы.

**Методы анимации**

Класс удобен тем, что у него есть готовые методы для старта, паузы, продолжения, окончания и отмены анимации. И пользоваться ими просто. И имена у них говорящие.

* Animator.start()
* Animator.end()
* Animator.cancel()
* Animator.pause() (API 19)
* Animator.resume() (API 19)

Стоит задержаться на небольшой разнице между методами **cancel()** и **end()**. Допустим у нас вращается картинка. Отмена анимации (**cancel()**) останавливает картинку в той позиции, в которой она находилась, например, повёрнутой на 56 градусов. Метод **end()** принудительно переводить картинку в конечную точку анимации.

Метод **pause()** вызывает тот же визуальный эффект, что и **cancel()**, но при следующем вызове метода **resume()** анимация начнётся не с начала, а продолжит работу. В этом заключается приятная дополнительная возможность, добавленная в API 19.

Подготовим разметку с картинкой и пятью кнопками для вызова каждого метода.

<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"

android:paddingLeft="@dimen/activity\_horizontal\_margin"

android:paddingRight="@dimen/activity\_horizontal\_margin"

android:paddingTop="@dimen/activity\_vertical\_margin"

android:paddingBottom="@dimen/activity\_vertical\_margin"

tools:context=".MainActivity">

<ImageView

android:id="@+id/some\_image"

android:layout\_width="150dp"

android:layout\_height="150dp"

android:layout\_alignParentTop="true"

android:layout\_centerHorizontal="true"

android:layout\_marginTop="20dp"

android:src="@drawable/android\_cat" />

<LinearLayout

android:id="@+id/status\_display"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:layout\_below="@id/some\_image"

android:layout\_centerHorizontal="true"

android:layout\_marginBottom="20dp"

android:orientation="vertical">

<TextView

android:id="@+id/status\_is\_started"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content" />

<TextView

android:id="@+id/status\_is\_running"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content" />

<TextView

android:id="@+id/status\_is\_paused"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content" />

</LinearLayout>

<LinearLayout

android:id="@+id/button\_row1"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:layout\_below="@id/status\_display"

android:layout\_centerHorizontal="true"

android:layout\_marginBottom="20dp">

<Button

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:onClick="startAnimation"

android:text="Start" />

<Button

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:onClick="endAnimation"

android:text="End" />

<Button

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:onClick="cancelAnimation"

android:text="Cancel" />

</LinearLayout>

<LinearLayout

android:id="@+id/button\_row2"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:layout\_below="@id/button\_row1"

android:layout\_centerHorizontal="true"

android:layout\_marginBottom="20dp">

<Button

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:onClick="pauseAnimation"

android:text="Pause" />

<Button

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:onClick="resumeAnimation"

android:text="Resume" />

</LinearLayout>

</RelativeLayout>

В классе объявим переменную для **ObjectAnimator** и создадим анимацию вращения картинки пять раз на 360 градусов. Для каждой кнопки свой метод.

package ru.alexanderklimov.animatordemo;

import android.animation.ObjectAnimator;

import android.support.v7.app.ActionBarActivity;

import android.os.Bundle;

import android.view.Menu;

import android.view.MenuItem;

import android.view.View;

import android.widget.ImageView;

public class MainActivity extends ActionBarActivity {

private ObjectAnimator mAnimator;

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_main);

ImageView imageView = (ImageView) findViewById(R.id.some\_image);

mAnimator = ObjectAnimator.ofFloat(imageView, "rotation", 0, 360);

mAnimator.setDuration(1000);

mAnimator.setRepeatCount(5);

mAnimator.setRepeatMode(ObjectAnimator.RESTART);

}

public void startAnimation(View view) {

mAnimator.start();

}

public void endAnimation(View view) {

mAnimator.end();

}

public void cancelAnimation(View view) {

mAnimator.cancel();

}

public void pauseAnimation(View view) {

mAnimator.pause();

}

public void resumeAnimation(View view) {

mAnimator.resume();

}

}

Запустив программу, вы можете включать анимацию и проверять, как работают методы. Всё достаточно прозрачно.

Также вы можете отслеживать статус анимации через методы, возвращающие **true** или **false**:

* **isStarted()** (API 14)
* **isRunning()**
* **isPaused()** (API 19)

Метод **isStarted()** возвращает **true**, когда анимация запущена, но не закончена или отменена.

Метод **isRunning()** возвращает **true**, когда анимация действительно совершается. В этом его отличие от **isStarted()**, которая вернёт **true**, даже если в анимации заложена задержка и объект ещё не начал двигаться.

Метод **isPaused()** возвращает **true**, когда анимация стоит на паузе, в остальное время возвращается **false**.

Чтобы увидеть их работу, добавим новые переменные для текстовых полей.

private TextView isStartedTextView;

private TextView isRunningTextView;

private TextView isPausedTextView;

// onCreate()

isStartedTextView = (TextView) findViewById(R.id.status\_is\_started);

isRunningTextView = (TextView) findViewById(R.id.status\_is\_running);

isPausedTextView = (TextView) findViewById(R.id.status\_is\_paused);

public void setStatusTexts() {

isStartedTextView.setText("isStarted = " + mAnimator.isStarted());

isRunningTextView.setText("isRunning = " + mAnimator.isRunning());

isPausedTextView.setText("isPaused = " + mAnimator.isPaused());

}

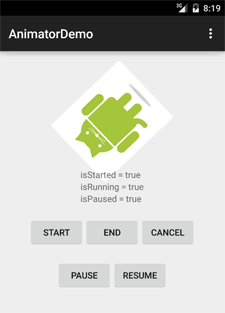
В каждый из пяти созданных нами ранее методов добавим вызов метода **setStatusTexts()**.

public void startAnimation(View view) {

mAnimator.start();

setStatusTexts();

}



**Слушатели**

Для мониторинга состояния анимации можно использовать интерфейс **Animator.AnimatorListener**, который реализует четыре метода

* **void onAnimationStart(Animator animation)**
* **void onAnimationRepeat(Animator animation)**
* **void onAnimationEnd(Animator animation)**
* **void onAnimationCancel(Animator animation)**

Регистрируется слушатель стандартным образом.

public void addListener(Animator.AnimatorListener listener)

По названию методы слушателя похожи на методы, которые мы изучили выше. Они связаны между собой.

Переработаем предыдущий пример и добавим слушателей. У всех методов, реагирующих на щелчки кнопок удалим вызов метода **setStatusTexts()**. Вместо этого в методе **onCreate()** зарегистрируем слушатель.

mAnimator.addListener(animatorListener);

В классе создадим слушатель со всеми его методами. Нам также понадобится дополнительный компонент **TextView**, в котором будем выводить информацию.

Animator.AnimatorListener animatorListener = new Animator.AnimatorListener() {

@Override

public void onAnimationStart(Animator animation) {

setStatusTexts();

listenerTextView.setText("started");

}

@Override

public void onAnimationEnd(Animator animation) {

setStatusTexts();

listenerTextView.setText(listenerTextView.getText() + " -- ended");

}

@Override

public void onAnimationCancel(Animator animation) {

setStatusTexts();

listenerTextView.setText("cancelled");

}

@Override

public void onAnimationRepeat(Animator animation) {

setStatusTexts();

listenerTextView.setText("repeating");

}

};

В API 19 был добавлен ещё один слушатель **AnimatorPauseListener** с двумя методами:

* **void onAnimationPause(Animator animation)**
* **void onAnimationResume(Animator animation)**

Регистрируется слушатель через метод **addPauseListener()**. Попробуйте самостоятельно добавить код для этого слушателя. Там всё тоже самое.

**Использованные материалы**

[Android Property Animations: Controlling Animation Flow | Cogito Learning](http://cogitolearning.co.uk/?p=1482)

[Android Property Animations: Animator Listeners | Cogito Learning](http://cogitolearning.co.uk/?p=1498)