河北师范大学软件学院
Software College of Hebei Normal University

Android中动画的使用

智能设备教研室

- Android中的动画简介

  - 帧动画（Frame Animation）
  - 补间动画（Tween Animation）
- 属性动画（Property animation）

Animation简介

- Android 平台提供了一套完整的动画框架，使得开发者可以用它来开发各种动画效果，总的来说Android动画可以分为两类，最初的传统动画和Android3.0 之后出现的属性动画。
- 传统动画又包括帧动画（Frame Animation）和补间动画（Tweened Animation）。

Animation简介

- 传统动画：
  - 帧动画（Frame Animation）：顺序播放一系列的图像。
    - 存在多幅图像，在不同图像之间切换构成动画。
  - 补间动画（Tween Animation）：对场景里的对象不断做图像变换（平移、缩放、旋转、透明度）从而实现动画效果。
    - 指定某个View的开始和结束状态；
    - 指定从开始到结束的变换方式。
- 对于Android中的动画来说，既可以使用XML文件方式声明，也可以在代码中实现。

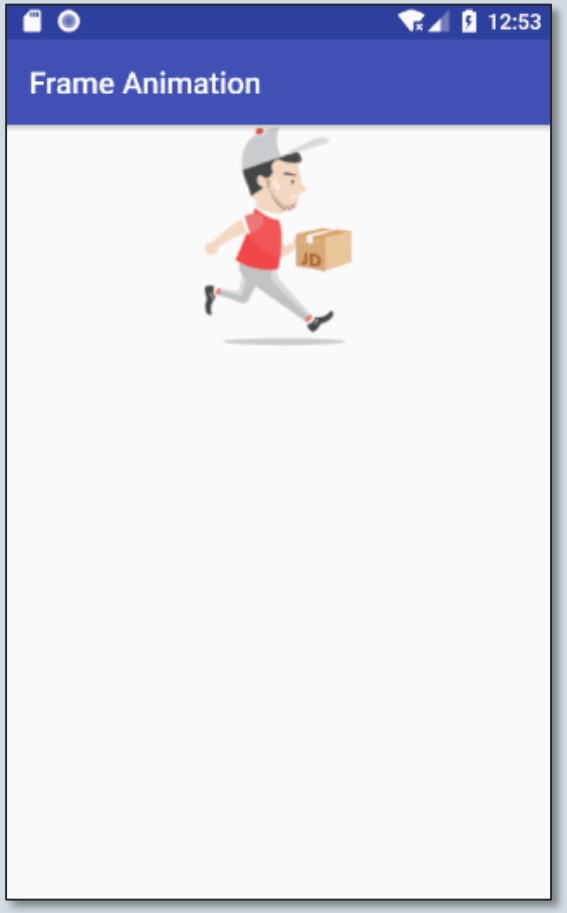
- Android中的动画简介
  - 帧动画（Frame Animation）
  - 补间动画（Tween Animation）
- 属性动画（Property animation）

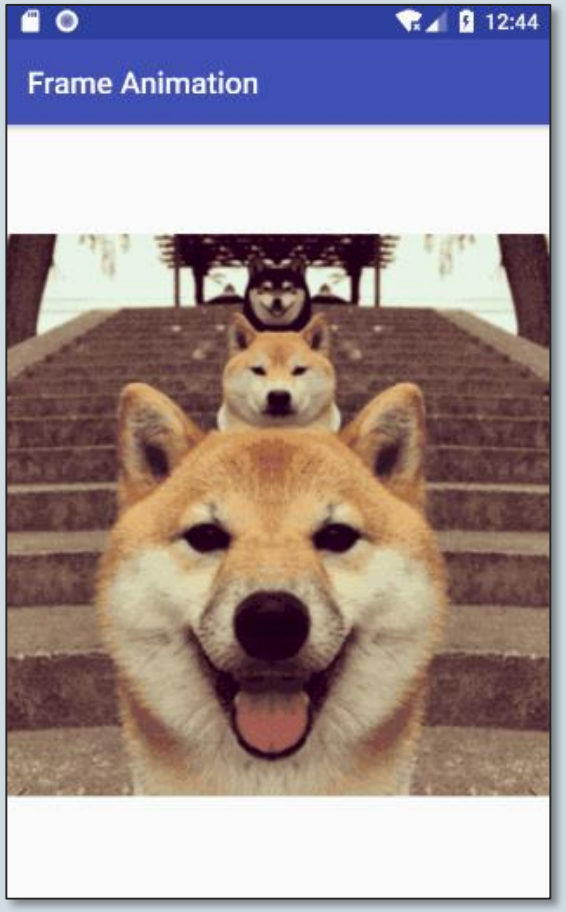
帧动画 Frame Animation

- Frame动画：顺序播放事先准备好的图像从而产生动画效果。
- 创建Frame动画的基本流程：
  - 在动画中加载待播放的图片（XML文件或Java代码）；
  - 为视图控件或视图页面启动Frame动画。

帧动画 Frame Animation

- 实例：借助Frame Animation模拟GIF动画。





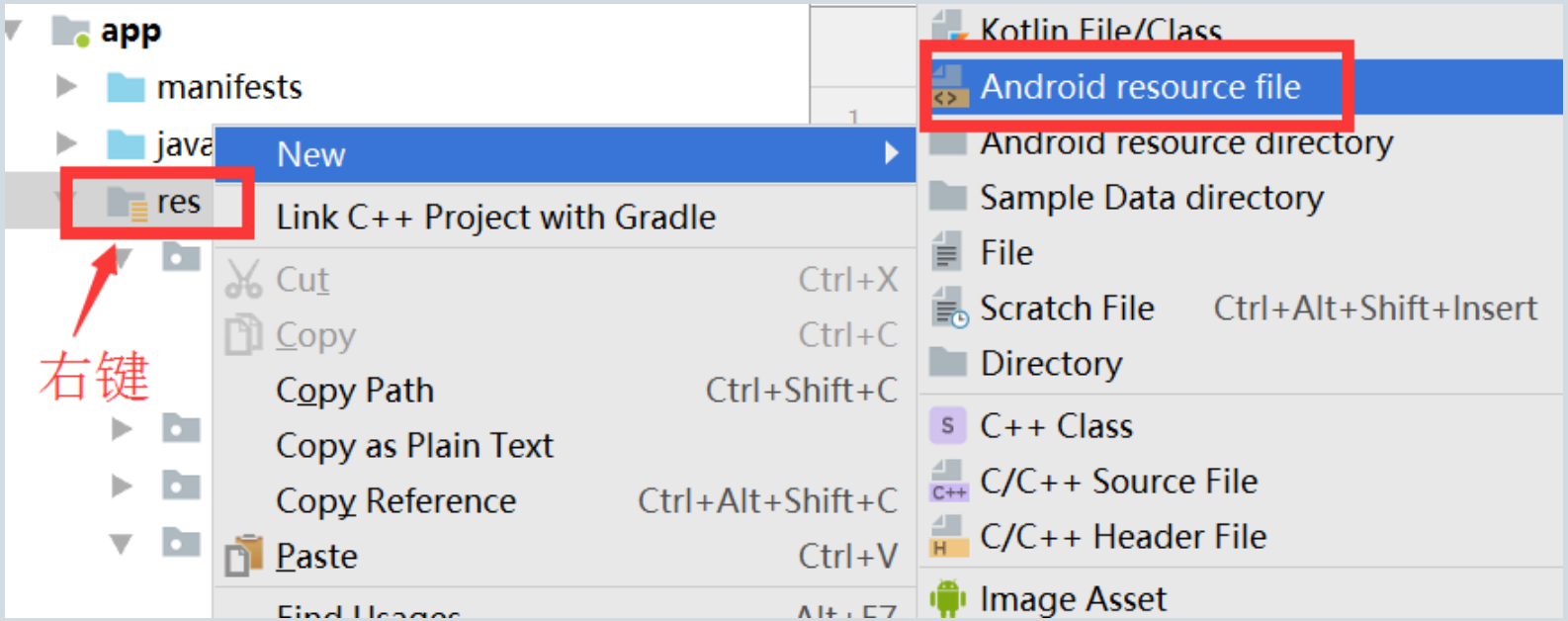
Step1：加载待播放图片及XML文件

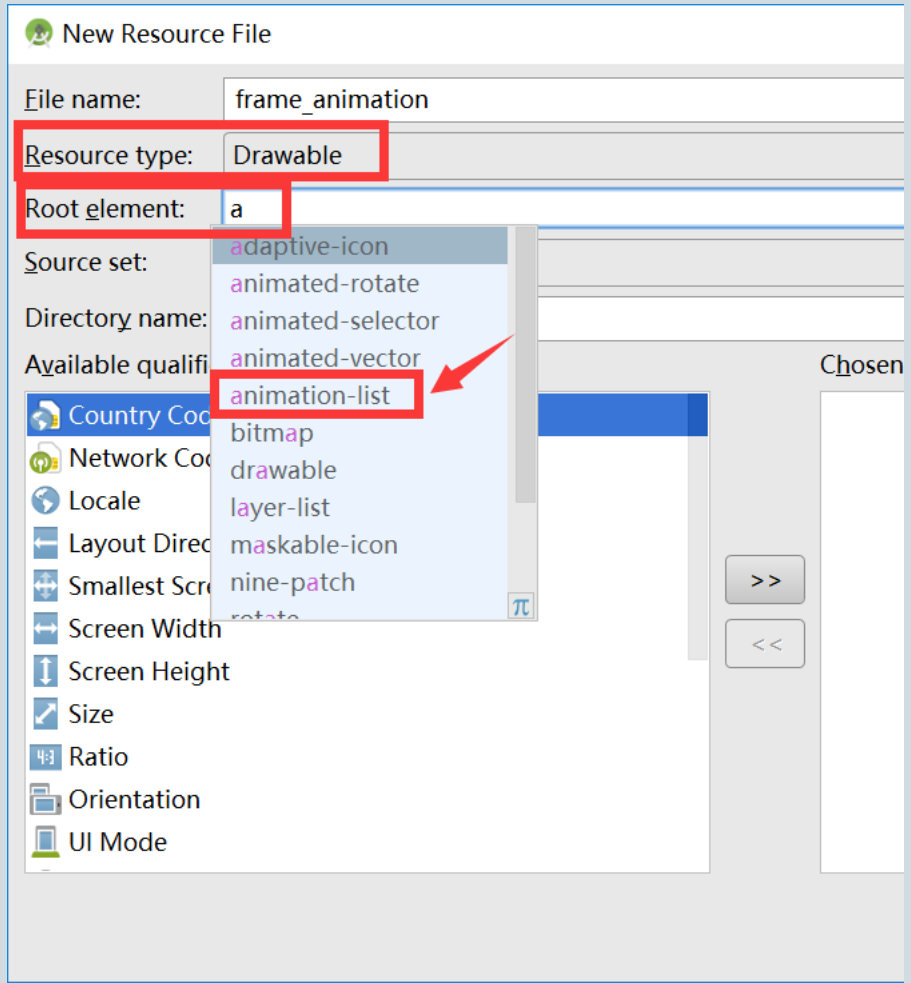
- 在res/drawable目录中添加图片。
- 添加res/drawable/***.xml文件。

```
<?xml version="1.0" encoding="utf-8"?>
<animation-list android:oneshot="false"
    xmlns:android="http://schemas.android.com/apk/res/android">
    <item android:drawable="@mipmap/jd_1" android:duration="90" />
    <item android:drawable="@mipmap/jd_2" android:duration="90" />
    <item android:drawable="@mipmap/jd_3" android:duration="90" />
</animation-list>
```

Step1：加载待播放图片及XML文件

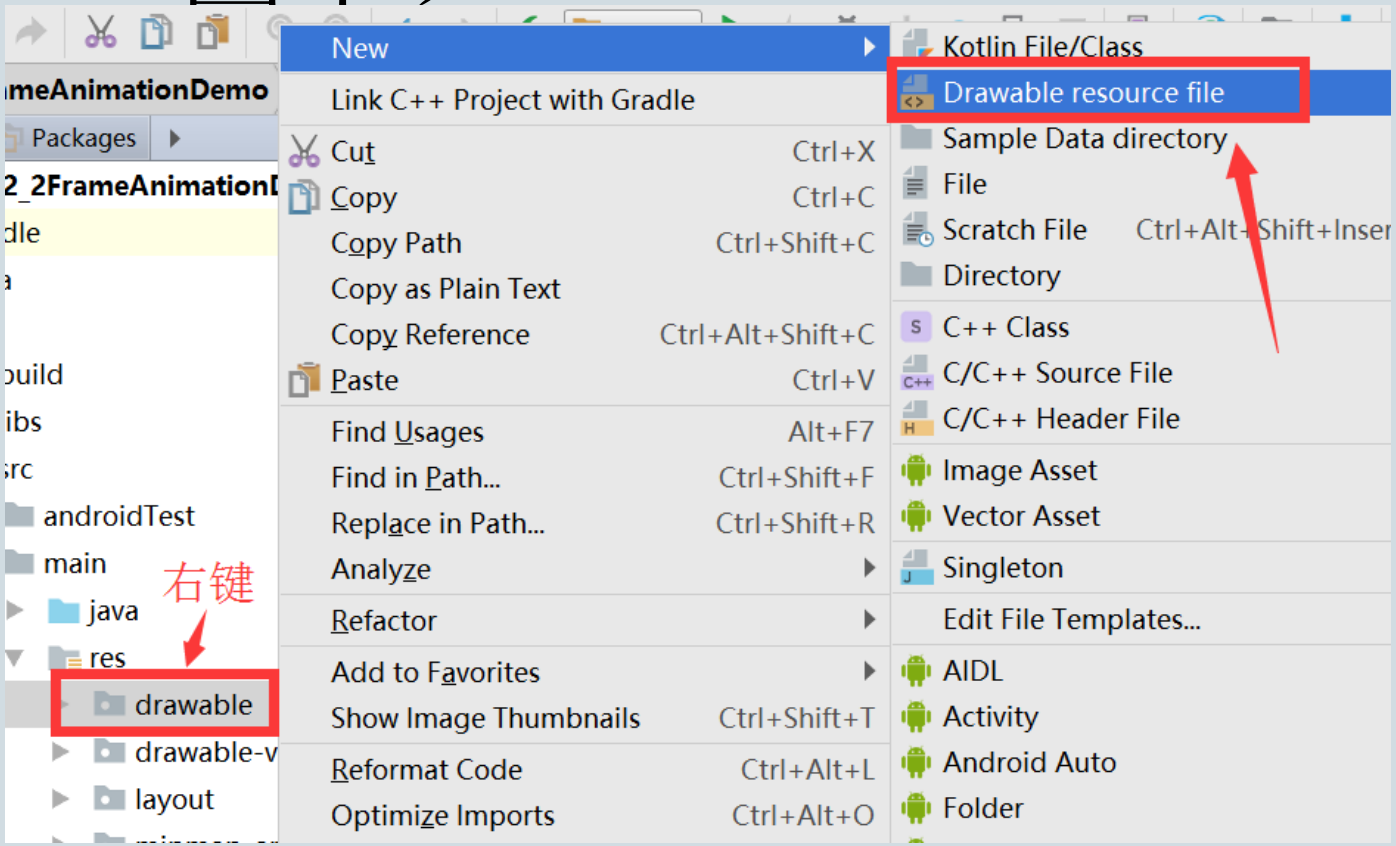
- 添加动画列表的xml文件（方法1： Android视图下）

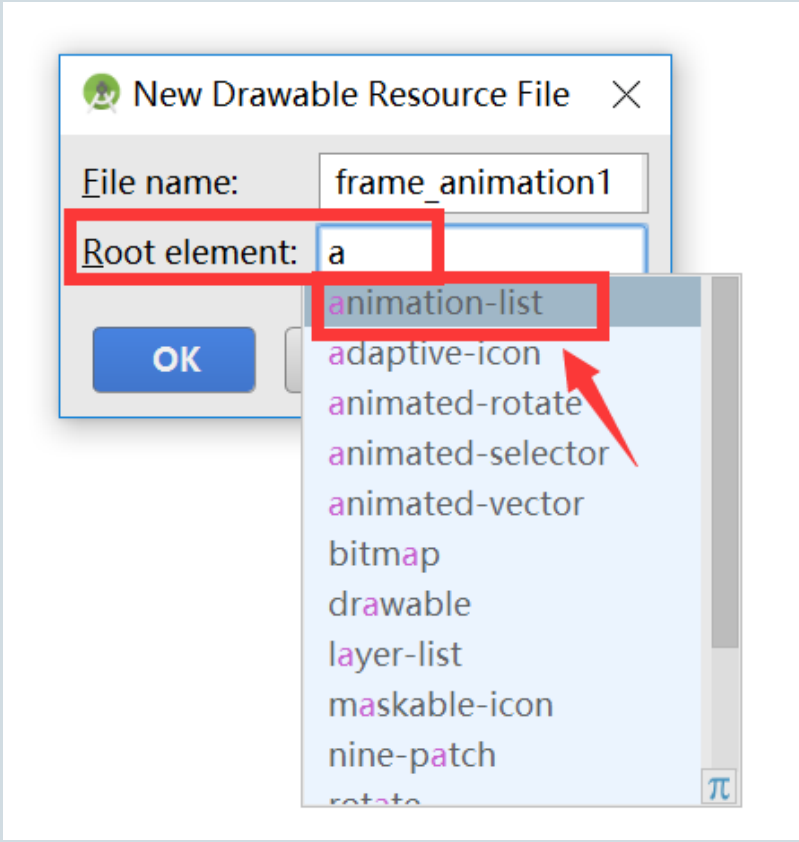




Step1：加载待播放图片及XML文件

- 添加动画列表的xml文件（方法2：Project视图下）





XML文件节点属性

- 在res/drawable/***.xml文件中
  - <animation-list>标签：
    - oneshot属性：
      - true：动画执行一次。
      - false：动画循环执行。
  - <item>子标签：
    - drawable属性：表示图片的资源ID。
    - duration属性：表示该图片的显示时间，毫秒。

Step2：加载待播放图片

- 在Activity中加载背景图片资源。
  - 帧动画是作为背景图片资源插入到视图中的。
  - 方法一：在XML布局资源文件中配置。

```
<ImageView android:id="@+id/iv_loading"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:src="@drawable/loading_jd"/>
```

  - 方法二：在Java源文件文件中配置。

```
ImageView iv_loading = findViewById(R.id.iv_loading);
iv_loading.setImageResource(R.drawable.loading);
```

Step3：启动动画

- 在Activity中启动动画。
- 在Activity中停止动画执行。

```
ImageView iv_loading = findViewById(R.id.iv_loading);
AnimationDrawable loadingDrawable
        =(AnimationDrawable) iv_loading.getDrawable();
loadingDrawable.start();    // 开始动画
// loadingDrawable.stop(); // 停止动画
```

补充：Java代码实现Frame动画

```
// 实例化AnimationDrawable对象
AnimationDrawable frameAnimation = new AnimationDrawable();
// 添加帧资源
for (int i=1; i<4; ++i) {
    int id = getResources().getIdentifier("img_"+i,
                    "drawable", getPackageName());
    frameAnimation.addFrame(getResources().getDrawable(id), 90);
}
// 设置循环并播放
frameAnimation.setOneShot(false);
frameAnimation.start();
// 使用动画
ImageView iv_loading = findViewById(R.id.iv_loading);
iv_loading.setImageDrawable(frameAnimation);
```

- Android中的动画简介
  - 帧动画（Frame Animation）
  - 补间动画（Tween Animation）
- 属性动画（Property animation）

补间动画 Tween Animation

- Tween动画：对于一个对象（视图控件或页面）实行图像变换。
- 创建Tween动画的基本流程：
  1. 设置对象开始和结束时的状态；
  2. 设置动画效果（XML方式或Java代码方式）；
  3. 为视图控件或视图页面设定（或启动）Tween动画。

设置Tween动画效果

- 常用的Tween动画效果。
  - Alpha：改变对象透明度从而产生动画。
  - Translate：改变对象的位置从而产生动画。
  - Scale：改变对象的尺寸从而产生动画。
  - Rotate：令对象旋转从而产生动画。

设置Tween动画效果

- 实例：在XML文件中设置动画效果。

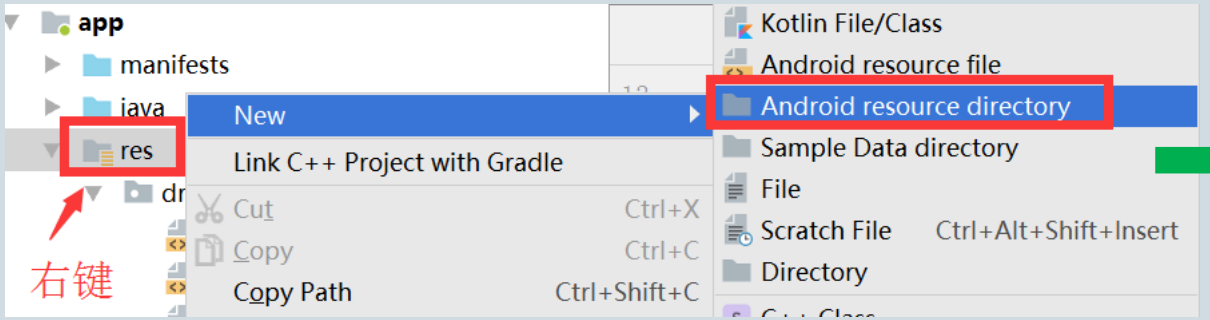
Step1：在XML文件中设置动画效果

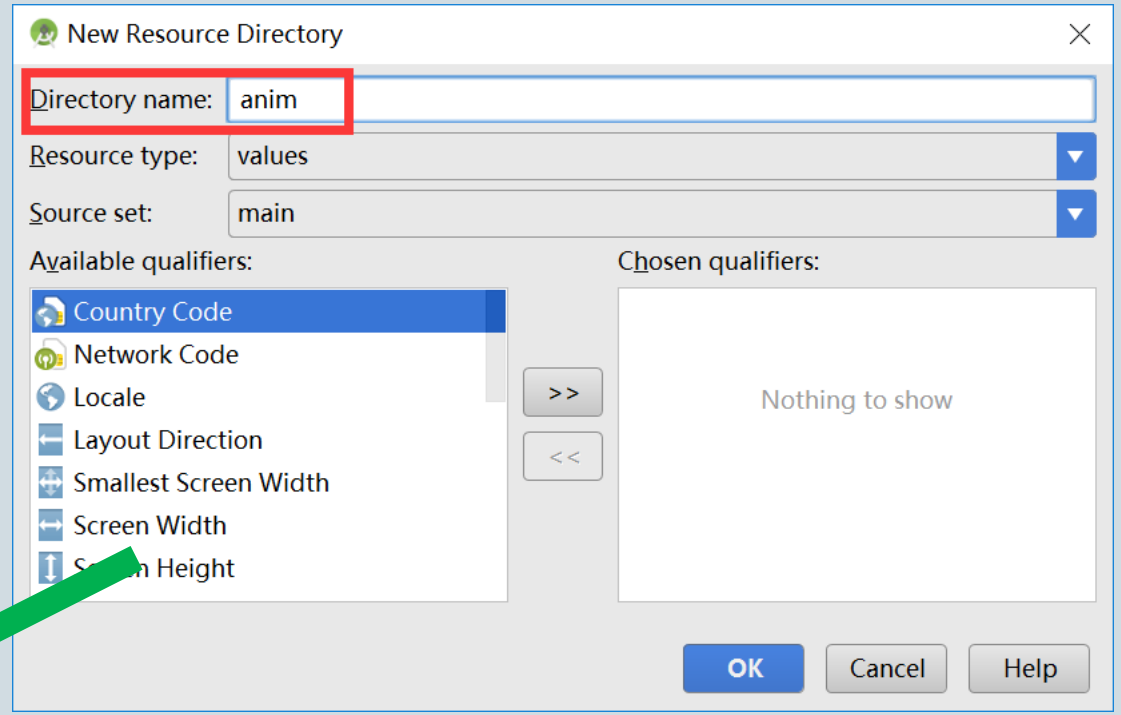
- XML文件路径：res/anim/***.xml
- XML文件基本结构：

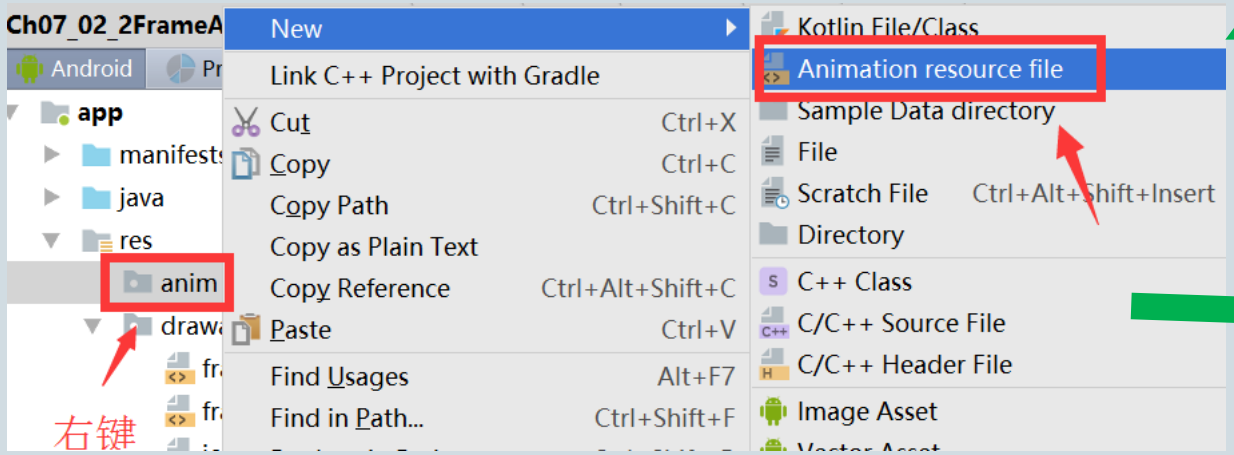
```
<?xml version="1.0" encoding="utf-8"?>
<set
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:shareInterpolator="false">
    <alpha
        android:fromAlpha="1.0"
        android:toAlpha="0.0"
        android:duration="5000" />
</set>
```

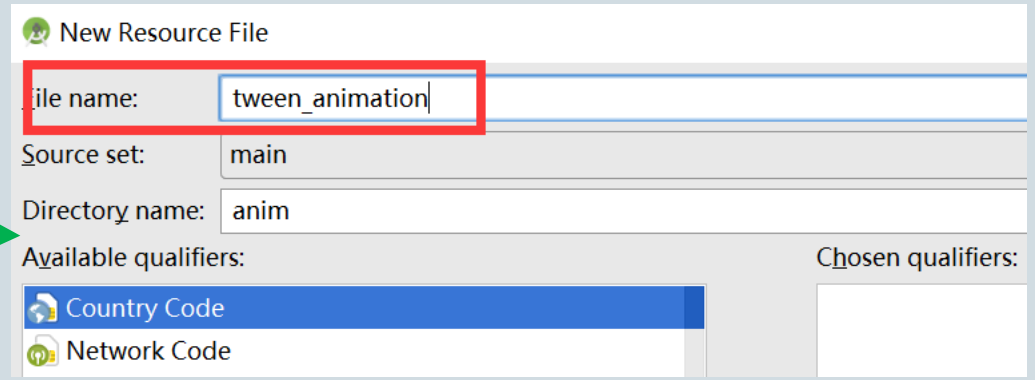
Step1：在XML文件中设置动画效果

- 创建XML文件方法









Step1：在XML文件中设置动画效果

- <set>标签：
  - shareInterpolator属性：是否本标签下的所有动画效果共享加速器属性。
  - interpolator属性：加速器属性。
    - AccelerateDecelerateInterpolator：在动画开始与结束的地方速率改变比较慢，在中间的时候速率快。
    - AccelerateInterpolator：在动画开始的地方速率改变比较慢，然后开始加速。
    - CycleInterpolator：动画循环播放特定的次数，速率改变沿着正弦曲线。
    - DecelerateInterpolator：在动画开始的地方速率改变比较慢，然后开始减速。
    - LinearInterpolator：动画以均匀的速率改变。

Step1：在XML文件中设置动画效果

- 动画效果的通用属性。

属性名	属性值	说明
duration	long，毫秒	对象的持续时间
fillAfter	true/false	动画结束后，对象位置是否为新位置
repeatCount	int	动画的执行次数
repeatMode	int	动画重复执行时采用的模式
startOffset	long，毫秒	动画绑定到对象上后开始播放的延迟时间
interpolator	interpolator对象	设置动画的加速器效果

Step1：在XML文件中设置动画效果

- 不同动画效果的属性

类型	属性名	属性值	说明
Alpha	fromAlpha / toAlpha	float	动画开始前/结束后的透明度
Translate	fromXDelta / toXDelta fromYDelta / toYDelta	float	动画开始前/结束后的X，Y坐标值
Scale	fromXScale / toXScale fromYScale / toYScale	float	动画开始前/结束后X，Y方向的大小
Scale/Rotate	pivotX / pivotY	float	动画变换的中心点位置
Rotate	fromDegrees / toDegrees	float	对象旋转前/旋转后的角度

Step2：为视图控件设定动画

- 在Activity文件中为View绑定动画：

```
ImageView imageView = findViewById(R.id.iv_loading);
Animation animation = AnimationUtils.loadAnimation(
                    MainActivity.this, R.anim.anim_alpha);
imageView.startAnimation(animation);
```

- AnimationUtils.loadAnimation( )：从XML文件中加载动画对象。
- View.startAnimation( )：启动动画。

实例：多个Tween Animation效果

- 实例：实现多个Tween Animation效果的依次执行。

实例：多个Tween Animation效果

```
<set xmlns:android="http://schemas.android.com/apk/res/android"
    android:shareInterpolator="true"
    android:interpolator="@android:anim/linear_interpolator" >
    <scale android:fromXScale="1.0"
        android:fromYScale="1.0"
        android:toXScale="0.0"
        android:toYScale="0.0"
        android:duration="4000"
        android:pivotY="50%"
        android:pivotX="50%"/>
    <rotate android:pivotX="50%"
        android:pivotY="50%"
        android:fromDegrees="0.0"
        android:toDegrees="360.0"
        android:duration="1000"
        android:repeatCount="4"/>
</set>
```

实例：实现Activity切换动画

- 实例：实现页面之间的平滑切换。

实例：实现Activity切换动画

– XML文件： anim_in_right.xml

```
<set xmlns:android="http://schemas.android.com/apk/res/android">
    <translate android:duration="1000"
        android:fromXDelta="100%p"
        android:toXDelta="0"/>
</set>
```

– XML文件：anim_out_left.xml

```
<set xmlns:android="http://schemas.android.com/apk/res/android">
    <translate android:duration="1000"
        android:fromXDelta="0"
        android:toXDelta="-100%p"/>
</set>
```

实例：实现Activity切换动画

– Activity文件： MainActivity.java

```
startActivity(new Intent(MainActivity.this,
                         MyActivity.class));
overridePendingTransition(R.anim.anim_in_right,
                          R.anim.anim_out_left);
```

补充：使用Java代码创建动画效果

- 在Activity文件中：

```
// 创建动画集合
AnimationSet animationSet = new AnimationSet(true);
// 创建一个平移动画
TranslateAnimation translateAnimation
        = new TranslateAnimation(0, 200.0f, 0, 0);
// 为平移动画设置属性
translateAnimation.setDuration(1000);
translateAnimation.setStartOffset(0);
translateAnimation.setRepeatCount(3);
// 将动画添加到动画集合中
animationSet.addAnimation(translateAnimation);
// 为控件绑定动画
imageView.startAnimation(animationSet);
```

补充：为动画绑定事件监听器

- 在Activity文件中：

```
translateAnimation.setAnimationListener(new
Animation.AnimationListener() {
    @Override
    public void onAnimationStart(Animation animation) { // TODO }
    @Override
    public void onAnimationEnd(Animation animation) { // TODO }
    @Override
    public void onAnimationRepeat(Animation animation) { // TODO }
});
```

补充：为动画添加加速器

- Interpolat作用：
  - 定义了一个动画的变化率，这使得基本的动画效果(alpha, scale, translate, rotate）得以加速，减速等。
  - 定义了动画的变化速度，可以实现匀速、正加速、负加速、无规则变加速等。
  - Interpolator 是基类，封装了所有 Interpolator 的共同方法，它只有一个方法，即 getInterpolation (float input) 。

补充：为动画添加加速器

- Android 中常用的Interpolator 子类。

加速器	作用
AccelerateDecelerateInterpolator	在动画开始与结束的地方速率改变比较慢，在中间的时候加速
AccelerateInterpolator	在动画开始的地方速率改变比较慢，然后开始加速
CycleInterpolator	动画循环播放特定的次数，速率改变沿着正弦曲线
DecelerateInterpolator	在动画开始的地方速率改变比较快，然后开始减速
LinearInterpolator	在动画的以均匀的速率改变

- 只需要在XML文件添加相应属性即可。

- Android中的动画简介
  - 帧动画（Frame Animation）
  - 补间动画（Tween Animation）
- 属性动画（Property animation）

属性动画 Property animation

- 属性动画（Property Animation）是在 Android 3.0（API 11）后为了弥补补间动画的缺陷才提供的一种全新动画模式。
- 补间动画的缺陷：
  1. 作用对象局限，只能作用View上，不能作用非View对象；
  2. 效果单一，只包含移动、缩放、旋转、淡入淡出效果；
  3. 没有改变View的属性，只是改变视觉效果。

属性动画 Property animation

- 属性动画特点：
  - 作用对象：任意 Java 对象。
  - 实现的动画效果：可自定义各种动画效果。
- 工作原理：在一定时间间隔内，通过不断对值进行改变，并不断将该值赋给对象的属性，从而实现该对象在该属性上的动画效果。

ValueAnimator类

- ValueAnimator 是整个属性动画机制中最核心的一个类，它的内部使用一种时间循环的机制来计算值与值之间的动画过渡，只需要将初始值和结束值提供给 ValueAnimator，并且设置动画运行时长，那么 ValueAnimator 就会自动完成从初始值平滑过渡到结束值这样的效果。

ValueAnimator类

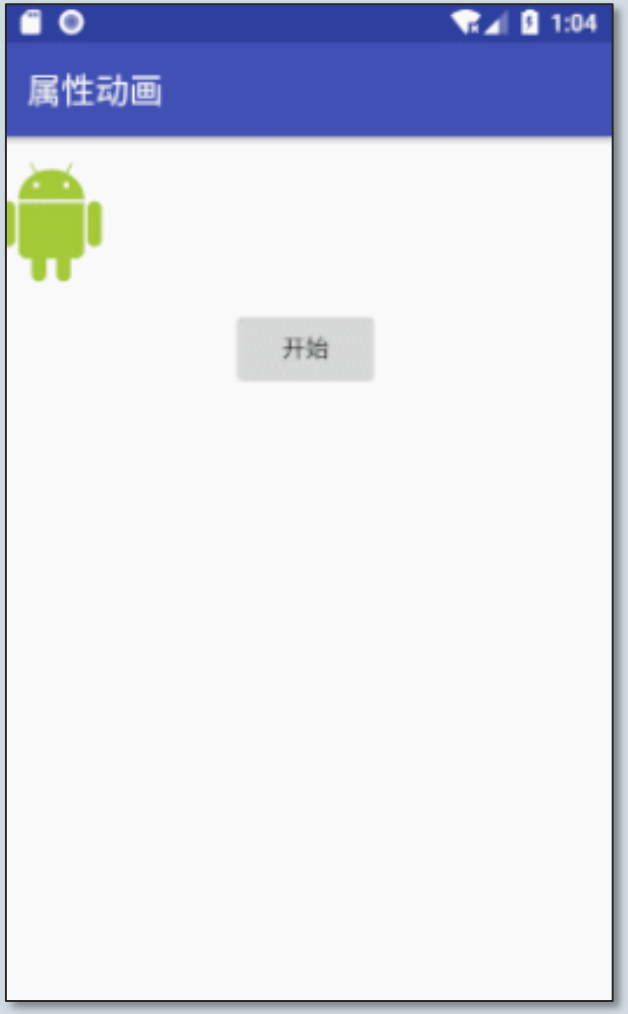
- ValueAnimator用法：

```
ValueAnimator anim = ValueAnimator.ofFloat(0f, 1f);
anim.setDuration(300);
anim.start();
```

  - ofFloat()方法当中允许传入多个float类型的参数。

ValueAnimator类

- 实例：实现图片的平滑移动。



Step1：创建动画并进行设置

```
ValueAnimator anim = ValueAnimator.ofInt(0, 400, 0);
// 设置动画运行的时长
anim.setDuration(2000);
// 设置动画延迟播放时间
anim.setStartDelay(500);
// 设置动画重复播放次数 = 重放次数 + 1
// 动画播放次数 = ValueAnimator.INFINITE时，动画无限重复
anim.setRepeatCount(0);
// 设置重复播放动画模式
// ValueAnimator.RESTART(默认):正序重放
// ValueAnimator.REVERSE:倒序回放
anim.setRepeatMode(ValueAnimator.RESTART);
```

Step2：设置监听器

```
anim.addUpdateListener(
        new ValueAnimator.AnimatorUpdateListener() {
    @Override
    public void onAnimationUpdate(
            ValueAnimator animation) {
        // 获得改变后的值，并对相应控件进行设置
        int currentValue = (Integer) animation
                .getAnimatedValue();
        imageView.setTranslationX(currentValue);
    }
});
anim.start(); // 启动动画
```

ObjectAnimator类

- ObjectAnimator类继承自ValueAnimator类，是直接对对象的属性值进行改变操作，从而实现动画效果。

```
ObjectAnimator.ofFloat(Object object,
                       String property,
                       float ....values);
```

- Object object　：需要操作的对象。
- String property：需要操作的对象的属性。
- float ....values　：动画初始值 & 结束值（不固定长度）。

ObjectAnimator类

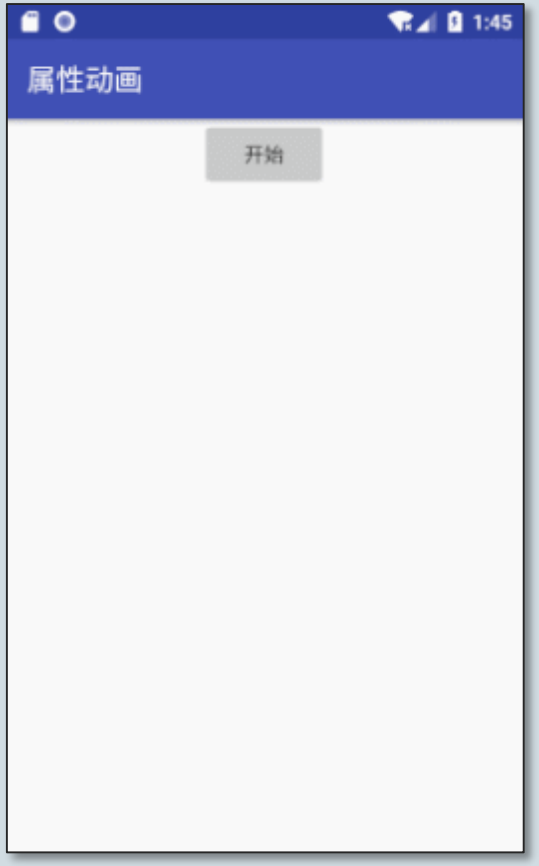
- 常见的对象属性值：

属性名	属性值	数值类型
alpha	控制View的透明度	float
translationX	控制X方向的位移	float
translationY	控制Y方向的位移	float
scaleX	控制X方向的缩放倍数	float
scaleY	控制Y方向的缩放倍数	float
rotation	控制以屏幕方向为轴的旋转度数	float
rotationX	控制以X轴为轴的旋转度数	float
rotationY	控制以Y轴为轴的旋转度数	float

ObjectAnimator类

- 实例：实现按钮的横向缩放。

```
ObjectAnimator animator = ObjectAnimator
        .ofFloat(button, "scaleX",
                1f, 3f, 1f);
animator.setDuration(5000);
animator.start();
```



顺序性控制

- AnimatorSet：可以灵活控制多个属性动画
  - play(ObjectAnimator动画对象)：播放参数动画对象
  - with(ObjectAnimator动画对象)：同时播放参数动画对象
  - before(ObjectAnimator动画对象)：参数动画对象播放前播放
  - after(ObjectAnimator动画对象)：参数动画对象播放后播放
  - playTogether(…可变长ObjectAnimator动画对象)：参数动画对象同时播放
  - playSequentially(… 可变长ObjectAnimator动画对象)：参数动画对象顺序播放

顺序性控制

- animatorSet.play(objectAnimator1).with(objectAnimator2).before(***).after(***);
- animatorSet.playTogether(objectAnimator, objectAnimator1, objectAnimator2);
- animatorSet.playSequentially(objectAnimator, objectAnimator1, objectAnimator2);

- Android中的动画简介
  - 帧动画（Frame Animation）
  - 补间动画（Tween Animation）
- 属性动画（Property animation）

Thank You!