安卓入门

# 前言

基础平台环境说明：

WIN10 Verson 1809 X64环境 + Intel i5 8259 2.3GHz + 32G +512G

JAVA8 + idea (IDE 或者是Android Studio 3.X也可以)+gradle+一些插件

Android SDK 版本是28，

## 环境初始化略；

大致流程配置好系统JDK环境,IDE开发环境

# 项目开始：初始化

使用idea初始化一个安卓空项目

# 基本组件

# 布局

# [Drawable类及XMLDrawable的使用](https://www.cnblogs.com/rookiechen/p/5310558.html)

### 性质

可直接使用.png、.jpg、.gif、9.png等图片作为资源,也可使用多种XML文件作为资源。（就是这些资源都能生成Drawable对象）。并对XML文件作出相关处理

### XMLDrawable

## StateListDrawable

作用：StateListDrawable对象所显示的Drawable对象会随着目标组件状态的改变而改变

组成：

根元素<selector/>，子元素<item/>

子元素<item/>的属性：android:color或android:drawable

                                android:state\_xxx:状态

举例：

<selector xmlns:android="http://schemas.android.com/apk/res/android">

<item android:state\_pressed="false"

android:drawable="@mipmap/start"/>

<item android:state\_pressed="true"

android:drawable="@mipmap/start\_down"/>

</selector>

**相关目标组件的状态不列出来**

## LayerDrawable

作用：可包含一个Drawable数组，系统会按照Drawable对象的数组顺序绘制，索引越大越被绘制在上层

根元素：<layer-list>  子元素：<item/>

子元素的属性：android:drawable 作为LayerDrawable的Drawable对象

                    android:id 为Drawable对象指定标识符

                    android:buttom|top...等 指定Drawable的绘制位置

实例1：

<layer-list xmlns:android="http://schemas.android.com/apk/res/android">

<!--调用android自带的id修改,利用@android:id 修改父style的background-->

<item android:drawable="@mipmap/p\_1"

android:id="@android:id/background"/>

<item android:drawable="@mipmap/p\_2"

android:id="@android:id/progress"/>

</layer-list>

实例2：

<layer-list xmlns:android="http://schemas.android.com/apk/res/android">

<!--可以在item里面创建各种各样的XMLDrawable-->

<item>

<bitmap android:src="@mipmap/p\_3"

android:gravity="center"/>

</item>

<item>

<bitmap android:src="@mipmap/p\_4"

android:gravity="center"/>

</item>

</layer-list>

## ShapeDrawable

作用：设置一个基本的几何图形（矩形、圆形、线条灯）

根元素：<shape/>    根元素的属性：android:shape=["rectangle"|"oval"|"line"|"ring"]

子元素：<corners/>：设置整体或者四个边角的弧度

           <gradient/>:渐变（可选择渐变的角度但必须是45的倍数，默认为0，渐变的中心点，渐变的类型，渐变的半径和开始和终止的颜色）

           <padding/>:内边距 （可以控制四周的边距）

           <size/>:形状的大小 （设置形状的宽高）

           <solid/>:单种颜色填充

  <stroke/>:绘制边框 （可设置画笔的颜色和粗细 并 设置每画一条线的长度和间距且必须两者都设置才有效）

详情参照：<http://blog.csdn.net/lonelyroamer/article/details/8254592>

实例：

<shape xmlns:android="http://schemas.android.com/apk/res/android"

android:shape="rectangle">

<corners android:radius="3dp"/>

<padding android:left="7dp"

android:right="7dp"

android:bottom="7dp"

android:top="7dp"/>

<gradient android:angle="45"

android:startColor="#0000"

android:endColor="#ffff"/>

</shape>

## ClipDrawable

作用：从Drawable上截取一个“图片片段”

根元素：<clip>  不使用子元素。

根元素属性： android:drawable: 选定Drawable对象

                  android:clipOrientation:指定截取方向

                  android:gravity:从什么地方开始截取

总结：选定图片并选择方向与位置截取图片

使用：从java中获取ClipDrawable并用setLevel()改变截取大小 //setLevel()只能从0~10000

<!--res/drawable/test\_clip.xml-->

<clip xmlns:android="http://schemas.android.com/apk/res/android"

android:drawable="@mipmap/start"

android:clipOrientation="horizontal"

android:gravity="center">

</clip>

<!--res/layout/activity\_main.xml-->

<LinearLayout

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:orientation="vertical"

tools:context="com.chen.android.test.MainActivity">

<ImageView

android:id="@+id/imageView"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:src="@drawable/test\_clip"/>

</LinearLayout>

/\*实现图片渐渐展开的效果\*/

public class MainActivity extends AppCompatActivity {

int data = 0;

int what = 0X11;

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_main);

ImageView img = (ImageView)findViewById(R.id.imageView);

//ImageView.getDrawable()获取的是当前控件里的图片，返回的是Drawable类型,还有说明Drawable对象可随意变成子对象并调用子对象的方法

final ClipDrawable clipDrawable = (ClipDrawable)img.getDrawable();

//创建Handler等待计时器传送的信息，使图片扩展

final Handler mHandler = new Handler(){

@Override

public void handleMessage(Message msg) {

super.handleMessage(msg);

if (msg.what == what){

clipDrawable.setLevel(data);//扩大截取的图片面积

data += 200;

}

}

};

//创建计时器

final Timer timer = new Timer();

timer.schedule(new TimerTask() {

@Override

public void run() {

if (data >= 10000){

timer.cancel();

}

mHandler.obtainMessage(what).sendToTarget();

}

},0,300);

}

}

## AnimationDrawable

简介：放在res/anim下，支持逐帧动画和补间动画

根元素：<set>   根元素属性：android:interpolator="参数"

                       参数：@android:anim/ 为开头 选择 linear\_interpolator：匀速变换 | accelerate\_interpolar：加速变换 | decelerate\_interpolator：减速变换

                       android:shareInterpolator= "true|false" ：是否让资源的interpolator与根元素相同

                       android:duration="时间"：定义持续时间

子元素(同样可以设置duration)：<alpha>：设置开始和结束的透明度

           <scale>：设置缩放的中心、开始的X，Y的尺寸和结束时X,Y的尺寸

           <translate>：设置图片的开始位置和结束位置进行位移

           <rotate>：设置旋转的中心、开始的角度和结束时候的角度

注意：利用android:fillAfter="true|false"：设置保留后的状态（哪个状态想保留就用这个，如果都像就放在<set>中）

使用：利用AnimationUtils的静态方法loadAninmation(Context context,int resId)

<!--在res/anim/test\_animtaion中-->

<set xmlns:android="http://schemas.android.com/apk/res/android"

android:interpolator="@android:anim/linear\_interpolator"

android:shareInterpolator="true"

android:fillAfter="true">

<alpha android:fromAlpha="50.0"

android:toAlpha="100.0"

/>

<scale android:pivotX="50%"

android:pivotY="50%"

android:fromXScale="1.0"

android:fromYScale="1.0"

android:toXScale="1.5"

android:toYScale="1.5"

android:duration="3000"

/>

<translate android:fromXDelta="30"

android:toXDelta="300"

android:fromYDelta="40"

android:toYDelta="90"

android:duration="3000"/>

</set>

/\*实现动画\*/

public class MainActivity extends AppCompatActivity {

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_main);

ImageView img = (ImageView)findViewById(R.id.imageView);

/\*利用工具类获取对象\*/

Animation animation= AnimationUtils.loadAnimation(this,R.anim.test\_animation);

/\*将动画附加在图片上\*/

img.startAnimation(animation);

}

}

# 网络权限

## 新建一个项目，在AndroidManiifest中添加

<uses-permission android:name="android.permission.INTERNET"/>

如果不添加这一句，那么在运行时候会像下面这样



## 在layout界面中添加一个WebView控件

<WebView

android:layout\_width="match\_parent"

android:layout\_height="match\_parent"

android:id="@+id/wv"></WebView>

## 在MainActivity中进行相应操作

public class MainActivity extends AppCompatActivity {

private WebView webView;

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_main);

webView = (WebView) findViewById(R.id.wv);

//如果只写下面这一句，会提示无法访问

//加载网页需要连接互联网的权限，需要在AndroidManifest中进行声明

//这样的好处：可以让用户很清楚的看到app所需要的权限

//用户并不清楚和重视权限问题，造成了病毒问题的泛滥。其实是可以避免的！

webView.loadUrl("http://www.jikexueyuan.com");

}

}

## 再进行发布打开app时候，会像下面这样



我在努力，虽然依旧很菜。

# [使用Spring RestTemplate for Android进行经过身份验证的POST请求](https://answer-id.com/53832569)

https://answer-id.com/53832569