宝贵建议请发送至：wangzhenyang@itcast.cn

黑马程序员 itheima.com -编程，始于黑马

Android 课程同步笔记

Alpha 0.01 版

Android-样式主题&国际化&动画

1.Android 中的样式和主题（★★）

1.1 样式

样式是作用在控件上的,是它是一个包含一个或者多个 view 控件属性的集合，例如定义属性 fontColor、fontSize、layout_width、layout_height 等，以独立的资源文件存放在 XML 文件中，并设置样式的名称。

Android Style 类似网页设计中的级联样式 CSS 设计思路，可以让设计与内容分离，并且可以方便的继承、覆盖、重用。

下面通过一个简单的案例演示自定义样式的用法，在该案例中，我们自定义一个样式用于渲染 Button 控件的显示效果。

我们新创建一个 Android 工程，工程名称《样式和主题》。直接使用默认布局文件和默认 Activity 类。

1 打开工程中 res->values->styles.xml 文件，添加如下样式。

```
<style name="btn_style">
        <item name="android:layout_height">wrap_content</item>
        <item name="android:layout_width">wrap_content</item>
        <item name="android:textSize">20sp</item>
        <item name="android:textColor">#ff0000</item>
        <item name="android:text">自定义样式</item>
    </style>
   <!--  继承上一个样式，相同的属性则覆盖父类 -->
     <style name="btn_style_child" parent="btn_style">
        <item name="android:textColor">#0000ff</item>
    </style>
```

2 在默认布局文件中使用上面的自定义样式。我们只需在如下布局文件中给 Button 一个 style="@style/btn_style"属性，那么所有属性都会作用在该 Button 上。

```
<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=".MainActivity" >

    <TextView
        android:layout_gravity="center_horizontal"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Android 中的样式和主题" />
    <Button
        style="@style/btn_style"
        />

</LinearLayout>
```

执行上面代码的效果如图：

Tips:同 CSS 一样，样式的引入遵循就近原则，在控件上定义的属性会覆盖被引入的样式中

的同一个属性。

1.2 主题

主题的定义与样式的定义相同，都是定义在 styles.xml 文件下，且均可以通过设置 parent 属性来继承一个父样式，不同之处在于主题是作用在 Activity 上的。

主题通过定义 AndroidManifest.xml 文件中的<application>和<activity>节点下的”android:theme”属性作用在整个应用或者某个 Activity，主题对整个应用或某个 Activity 进行全局性影响。如果一个应用使用了主题，同时应用下的 view 也使用了样式，那么当主题和样式属性发生冲突时，样式的优先级高于主题。

android 系统也定义了一些主题，例如：<activity android:theme=“@android:style/Theme.Dialog”>，该主题可以让 Activity 看起来像一个对话框，还有透明主题：@android:style/Theme.Translucent。如果需要查阅这些主题，可以在文档的 reference-->android-->R.style 中查看。

1 继续使用本文档 1.1 中创建的工程。在 res->values->styles.xml 中添加如下样式：

```
<!--
    主题也是通过在 styles.xml 文件中定义<style>节点来定义。
    同样式一样,指定一个全局唯一的名字给主题,通过 parent 属性继承父样式。
 -->
<style name="theme_noTitle">
    <!-- 该属性使 window 界面无标题 -->
    <item name="android:windowNoTitle">true</item>
</style>
   <!-- 该属性使 window 界面全屏 -->
<style name="them_noTitle_full" parent="theme_noTitle">
    <item name="android:windowFullscreen">true</item>
</style>
```

2 在 AndroidManifest.xml 中给添加样式，如下图清单黄色高亮部分。

```
<!-- 在 application 中引入主题 则所有 activity 都适用-->
 <application
     android:allowBackup="true"
     android:icon="@drawable/ic_launcher"
     android:label="@string/app_name"
     android:theme="@style/theme_noTitle" >
     <!-- 在 activity 中引入主题 则当前 Activity 适用-->
     <activity
         android:theme="@style/them_noTitle_full"
         android:name="com.itheima.styleAndTheme.MainActivity"
         android:label="@string/app_name" >
         <intent-filter>
             <action android:name="android.intent.action.MAIN" />
```

也可以在 Activity 类中通过 Java 代码动态设置样式。

```
public class MainActivity extends Activity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        //给当前 Activity 设置主题
        setTheme(R.style.them_noTitle_full);
    }

}
```

程序运行效果图比较简单，在这里就不再展示。

2.Android 实现国际化（★★）

国际化的英文单词是 Internationalization ,因为这个单词太长了 ,有时也简称为 I18N，其中的 I 是这个单词的第一个字符，18 表示中间省略的字母个数，而 N 代表这个单词的最后一个字母。所以，I18N 也就是国际化的意思。

Android 程序国际化，也就是程序可以根据系统所使用的语言，将界面中的文字翻译成与之对应的语言。这样，可以让程序更加通用。Android 可以通过资源文件非常方便的实现程序的国际化。

2.1 Android 中如何实现国际化

在编写 Android 项目时，通常都是将程序中要使用的字符串资源放置在 res/values 目录下的 strings.xml 文件中，为了给这些字符串资源实现国际化，可以在 Android 项目的 res 目录下，创建对应于各个语言的资源文件夹（例如，为了让程序兼容简体中文、繁体中文和美式英文，可以分别创建名称为 values-zh-rCN、values-zh-rTW 和 values-en-rUS 的文件夹），然后在每个文件夹中创建一个对应的 strings.xml 文件，并在该文件中定义对应语言的字符串即可。这样，当程序运行时，就会自动根据操作系统所使用的语言来显示对应的字符串信息了。

图片也可以进行国际化，同字符串的国际化类似，只需根据程序要兼容的语言版本，分别创建名称类似 drawable-zh-rCN, drawable-zh-rTW 这样的文件夹，将图片资源存放在文件夹下即可。

2.2 国际化示例

下面通过一个案例来演示国际化的使用方法。

1 在 res 文件下分别创建 values-zh-rCN 和 values-zh-rTW 文件夹，分别在两个文件夹下创建 strings.xml 文件。

给 values-zh-rCN/settings.xml 添加内容如下：

```
<?xml version="1.0" encoding="utf-8"?>
<resources>

    <string name="app_name">国际化</string>
    <string name="action_settings">设置简体</string>
    <string name="hello_world">简体字，心怀大志就莫虚度光阴！ </string>

</resources>
```

给 values-zh-rTW/settings.xml 添加内容如下：

```
<?xml version="1.0" encoding="utf-8"?>
<resources>

    <string name="app_name">國際化</string>
    <string name="action_settings">設置</string>
    <string name="hello_world">繁體字，心懷大志就莫虛度光陰！ </string>

</resources>
```

2 在 res 文件下分别创建 drawable-zh-rCN 和 drawable-zh-rTW 文件夹 ,向里面各自添加一个 flag.jpg 的图片。第一个为为五星红旗、第二个为青天白日旗。图片的名字必须一致。

3 修改该工程的默认布局文件

```
<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=".MainActivity" >

    <TextView
```

```
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/hello_world" />
    <ImageView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:src="@drawable/flag"
        />
</LinearLayout>
```

4 设置模拟器的语言为简体中文。Menu->System settings->Language&input->Language->中文(简体)。然后运行改程序,效果图如下：

5 设置模拟器的语言为简体中文。Menu->System

settings->Language&input->Language->中文(繁体)。然后运行改程序，效果图如下：

3.Android 中的动画（★★★）

Android 3.0 以前 ,Android 支持两种动画模式 ,tween animation,frame animation，在 android3.0 中又引入了一个新的动画系统：property animation，这三种动画模式在 SDK 中被称为 property animation,view animation,drawable animation。 在本文档中只介绍 tween animation 和 frame animation。

Frame Animation(帧动画)：创建一个 Drawable 序列，这些 Drawable 可以按照指定的时间间隔一个一个的显示，也就是顺序播放事先做好的图像。

Tween Animation(渐变动画)：通过对特定的对象做图像变换如平移、缩放、旋转、淡出/淡入等产生动画效果。

3.1 帧动画 FrameAnimation

创建一个 Drawable 序列，这些 Drawable 可以按照指定的时间间隔一个一个的显示，也就是顺序播放事先做好的图像，跟放胶片电影类似。

下面通过一个案例来演示帧动画的使用方法。新创建一个工程《Android 中的动画》。

1 将准备好的图片文件放到 res/drawable-hdpi 目录中。

如果大家缺乏图片资源可以随便放进去几张图片就行，只要起到练习代码的作用就行。

2 在项目的 res 目录下创建文件夹 drawable，然后在文件夹下面定义动画 XML 文件，文件名称可以自定义（也可以使用 AnimationDrawable 类，采用代码方式定义动画效果），这里给改 xml 文件起名为 frame_anim.xml。

3 打开创建好的 xml 文件 ,在里面添加根节点<animation-list>,可以在此根节点中设置属性”android:oneshot”来控制动画只播放一次，否则系统将默认持续播放。

在根节点<animation-list>下为帧动画的每幅图片添加一个<item>节点，节点的”android:drawable”属性是图片的资源 id，”android:duration”属性指定图片展示的时间（一般每秒展示 5-8 张图片就可以感受到动画的效果）。

```
<?xml version="1.0" encoding="utf-8"?>
<animation-list
xmlns:android="http://schemas.android.com/apk/res/android"
    android:oneshot="true">
  <item android:drawable="@drawable/a1" android:duration="200"></item>
 <item android:drawable="@drawable/a2" android:duration="200"></item>
    <!-----------------更多 item 不再展示--------------------->
</animation-list>
```

4 编写默认的 Activity 类。

```
public class MainActivity extends Activity {
    private ImageView iv;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        iv = (ImageView) findViewById(R.id.iv);
    }
    public void startAnimation(View view){
        iv.setImageResource(R.drawable.frame_anim);
        AnimationDrawable animationDrawable =  (AnimationDrawable)
iv.getDrawable();
        animationDrawable.start();
    }
}
```

运行该程序效果如下图：

3.2 渐变动画 TweenAnimation

渐变动画也叫补间动画。补间动画通过对 View 的内容进行一系列的图形变换（包括平移、缩放、旋转、改变透明度)来实现动画效果。动画效果的定义可以采用 XML 来做也可以采用 java 代码来做。补间动画有 4 种类型：

动画的类型	Xml定义动画使用的配置节点	编码定义动画使用的类
渐变透明度动画效果	<alpha/>	AlphaAnimation
渐变尺寸缩放动画效果	<scale/>	ScaleAnimation
画面位置移动动画效果	<translate/>	TranslateAnimation
画面旋转动画效果	<rotate/>	RotateAnimation

补间动画的常用方法：

- Animation：
  - ★ setDuration 设置动画的执行时间
  - ★ setRepeatCount 设置动画的重复次数
  - ★ setRepeatMode 指定重复的模式（如：反转）

★ setFillAfter 指示动画指定完毕之后控件的状态是否停留在动画停止的时候

★ setAnimationListener设置动画的事件监听器

ImageView：

★ startAnimation(Animation a) 让 ImageView 执行某动画

3.2.1 Alpha 渐变动画

渐变动画在代码中使用的是AlphaAnimation类来定义，在XML文件中使用<alpha>节点来定义。

下面分别演示使用 XML 文件和 Java 代码的方式实现 Alpha 渐变动画。在这里依然使用本文档 3.1 章节中的工程。

使用 XML 文件实现 Alpha 动画

1. 在 res 目录下创建 anim 文件夹
2. 在 anim 文件夹中创建 alpha_anim.xml 文件，文件名自定义
3. 编辑 alpha_anim.xml 文件，文件清单如下。

```
<?xml version="1.0" encoding="utf-8"?>
<alpha xmlns:android="http://schemas.android.com/apk/res/android"
    android:duration="1000"
    android:fromAlpha="1"
    android:repeatCount="2"
    android:toAlpha="0"
    android:repeatMode="restart"
    >
</alpha>
```

4. 添加 Java 逻辑代码:使用 AnimationUtils 工具类加载 xml 文件 获取 Animation

对象；调用 startAnimation 让 ImageView 执行此动画。

Tips：这里需要在 3.1 章节的默认布局文件（activity_main.xml)中添加一个 Button，并为该 Button 指定 android:onClick="startAlphaAnimation"属性，在 MainActivity 中实现该方法。方法清单如下：

```
public void startAlphaAnimation(View view){
        //通过 AnimationUtils 加载定义的动画文件
        Animation animation = AnimationUtils.loadAnimation(this,
R.anim.alpha_anima);
        //给 iv 设置一个图片
        iv.setImageResource(R.drawable.me);
        //iv 开始播放动画
        iv.startAnimation(animation);
    }
```

运行效果截图比较简单，且截图是静态的，因此不再给出。

使用编码方式实现 Alpha 动画

```
public void startAlphaAnimation(View view){
        /*
         * 创建一个动画类 AlphaAnimation
         * 第一个参数是开始透明度
         * 第二个参数是结束透明度
         */
        Animation animation = new AlphaAnimation(0, 1);
        //设置持续时间
        animation.setDuration(2000);
        //设置重复次数
        animation.setRepeatCount(2);
        //设置重复方式 Animation.RESTART 代表重新开始播放
        animation.setRepeatMode(Animation.RESTART);
        //设置是否保持动画结束后的状态
        animation.setFillAfter(true);
        //绑定动画事件
        animation.setAnimationListener(new AnimationListener() {

            @Override
            public void onAnimationStart(Animation animation) {
                Toast.makeText(MainActivity.this, "动画开始播放",
0).show();
            }
            @Override
            public void onAnimationRepeat(Animation animation) {
                Toast.makeText(MainActivity.this, "动画重复中...",
0).show();
            }
            @Override
            public void onAnimationEnd(Animation animation) {
                Toast.makeText(MainActivity.this, "动画结束", 0).show();
            }
        });
        //给 iv 设置一个图片
        iv.setImageResource(R.drawable.me);
        //iv 开始播放动画
        iv.startAnimation(animation);
    }
```

3.2.2 Scale 伸缩动画

在本文档3.1章节中创建的工程中的布局文件中添加ScaleAnimation按钮，在Activity类中设置改按钮的绑定事件。布局文件十分简单，因此这里就不再给出详细布局清单。效果如下图所示。

Activity 类中添加 scaleAnimation(View view)方法，方法清单如下：

```
public void scaleAnimation(View view){
        /*
         * 第一个参数 fromX x 轴起始大小(这个大小指倍数，它内部会用这个倍数去乘实际像素)
         * 第二参数 toX 轴截止大小(若起始大小=截止大小就是指 x 轴不伸缩)
```

```
* 第三个参数 fromY Y 轴的起始大小
        * 第四个参数 toY 轴的截止大小
        * 第五个参数 pivotXType X 轴的原点的类型（相对于自己而言还是相对于父容器而言）
        * 第六个参数 pivotXValue 开始伸缩时的 X 轴的原点(例:0.5 就是指以图片宽度的二分之一的位置作为 X 轴的原点)
        * 第七个参数 pivotYType Y 轴的原点的类型
        * 第八个参数 pivotYValue 开始伸缩时的 Y 轴的原点
        */
       Animation sa = new ScaleAnimation(1,1.5f, 1, 2, Animation.RELATIVE_TO_SELF, 0.5f, Animation.RELATIVE_TO_SELF,0.5f);
       sa.setDuration(2000);
       sa.setRepeatCount(5);
       sa.setRepeatMode(Animation.REVERSE);
       sa.setFillBefore(true);
       iv.setImageResource(R.drawable.me);
       iv.setAnimation(sa);
       sa.start();
    }
```

3.2.3 Translate 位移动画

在本文档 3.1 章节中创建的工程中的布局文件中添加 TranslateAnimation 按钮，在 Activity 类中设置改按钮的绑定事件。布局文件十分简单，因此这里就不再给出详细布局清单。Activity 类中添加 translateAnimation(View view)方法，方法清单如下：

```
public void translateAnimation(View view){
       /*
        * 第一个参数 fromXType 位移的 x 轴起始坐标的类型(相对于自己还是相对父容器)
        * 第二参数 fromXValue x 轴起点(0:自身最左边的 x 坐标  1：最右边的 x 坐标)
        * 第三个参数 toXType X 轴终点坐标的类型
        * 第四个参数 toXValue X 轴的终点
        * 第五个参数 fromYType Y 轴起始坐标的类型
        * 第六个参数 fromYValue Y 轴的起始坐标
```

```
         * 第七个参数 toYType Y 轴的终点坐标的类型
         * 第八个参数 toYValue Y 轴的终点坐标
         *
         * 下面代码的效果是：以自身的左上角为位移的原点，向右平移，平移距离=
自身大小
         */
        Animation sa = new
TranslateAnimation(Animation.RELATIVE_TO_SELF,0f,
Animation.RELATIVE_TO_SELF, 1, Animation.RELATIVE_TO_SELF, 0f,
Animation.RELATIVE_TO_SELF,0.f);
        sa.setDuration(2000);
        sa.setRepeatCount(5);
        sa.setRepeatMode(Animation.REVERSE);
        sa.setFillBefore(true);
        iv.setImageResource(R.drawable.me);
        iv.setAnimation(sa);
        sa.start();
    }
```

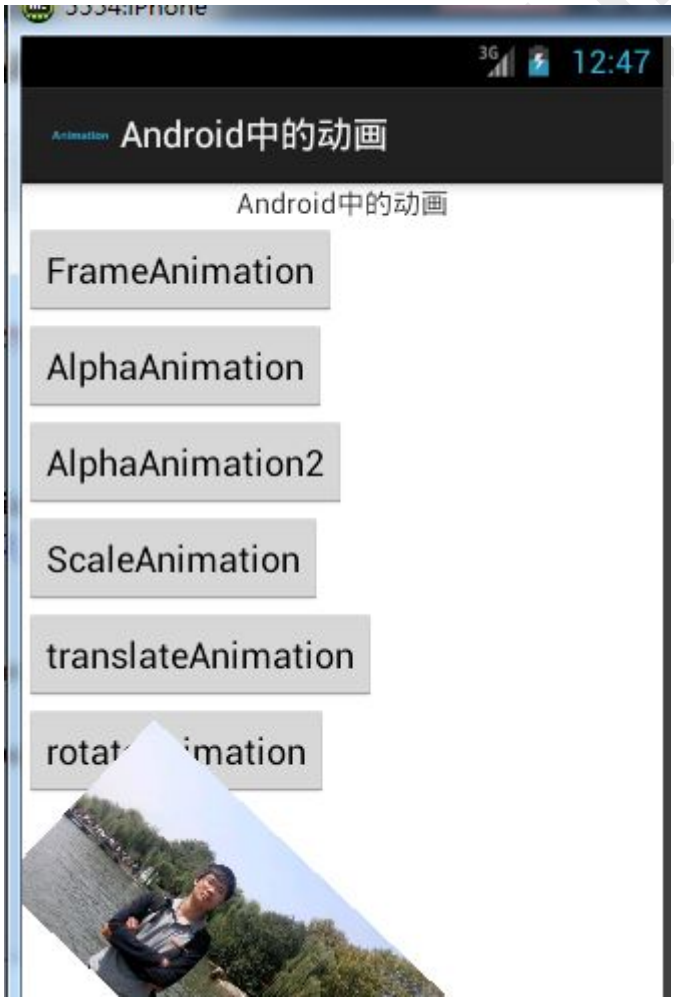
3.2.4 Rotate 旋转

在本文档 3.1 章节中创建的工程中的布局文件中添加 RotateAnimation 按钮，在 Activity 类中设置改按钮的绑定事件。布局文件十分简单，因此这里就不再给出详细布局清单。Activity 类中添加 RotateAnimation(View view)方法，方法清单如下：

```
public void rotateAnimation(View view){
        /*
         *第一个参数 fromDegrees：旋转的起始角度
         *第二个参数 toDegrees:旋转的结束角度
         *第三个参数 pivotXType:X 轴原点的类型(相对于自身还是相对于父容器)
         *第四个参数 pivotXValue:原点的 X 轴坐标
         *第五个参数 pivotYType： Y 轴原点的类型
         *第六个参数 pivotYValue:原点的 Y 轴坐标
         *
         * 下面代码的效果是：以自身中心点为旋转的原点，从左 45 度角摆动到右 45
```

```
度角
        */
       Animation ar = new
RotateAnimation(-45,45,RotateAnimation.RELATIVE_TO_SELF,
0.5f,RotateAnimation.RELATIVE_TO_SELF, 0.5f);
       ar.setDuration(2000);
       ar.setRepeatCount(5);
       ar.setRepeatMode(Animation.REVERSE);
       ar.setFillBefore(true);
       iv.setImageResource(R.drawable.me);
       iv.setAnimation(ar);
       ar.start();
    }
```

运行效果如图：



3.2.5 AnimationSet 动画的集合

动画集合在代码中使用的是 AnimationSet 类来定义，在 XML 文件中使用<set>节点来定义。

下面分别演示使用 XML 文件和 Java 代码的方式实现动画集合。在这里依然使用本文档 3.1 章节中的工程。

使用 XML 文件实现动画

在 res/anim 目录下创建 xml 文件，anim_set.xml。文件清单如下：

```
<?xml version="1.0" encoding="utf-8"?>
<set xmlns:android="http://schemas.android.com/apk/res/android"
    android:shareInterpolator="false"
    >
   <alpha
        android:duration="2000"
        android:fillAfter="true"
        android:fromAlpha="1"
        android:repeatCount="2"
        android:repeatMode="reverse"
        android:toAlpha="0" >
    </alpha>

    <scale
        android:duration="2000"
        android:fillAfter="true"
        android:fromXScale="0"
        android:fromYScale="0"
        android:repeatCount="2"
        android:repeatMode="reverse"
        android:toXScale="3"
        android:toYScale="3" >
    </scale>

    <translate
        android:duration="2000"
```

```
android:fillAfter="true"
        android:fromXDelta="0"
        android:fromYDelta="0%"
        android:repeatCount="2"
        android:repeatMode="reverse"
        android:toXDelta="500%"
        android:toYDelta="300%" >
    </translate>

    <rotate
        android:duration="2000"
        android:fillAfter="true"
        android:fromDegrees="0"
        android:pivotX="50%"
        android:pivotY="50%"
        android:repeatCount="2"
        android:repeatMode="reverse"
        android:toDegrees="360" >
    </rotate>

</set>
```

在 Activity 中添加如下方法，实现业务功能：

```
public void animationSet(View view) {
        //加载动画集合文件
        Animation animation = AnimationUtils.loadAnimation(this,
R.anim.anim_set);
        iv.setAnimation(animation);
        animation.start();
    }
```

使用 Java 代码实现动画

在该工程的默认布局文件中添加 Button，并给该 Button 指定一个 onClick 事件，触发的方法名为 animationSet2 在 Activity 类中添加如下方法：

```
public void animationSet2(View view) {
	//创建 AnimationSet 实例
	AnimationSet set = new AnimationSet(false);
	//创建一个普通动画实例
	AlphaAnimation aa = new AlphaAnimation(0.f, 1.f);
	aa.setDuration(2000);
	iv.setAnimation(aa);
	//创建一个普通动画实例
	Animation sa = new ScaleAnimation(1, 1.5f, 1, 2,
Animation.RELATIVE_TO_SELF, 0.5f, Animation.RELATIVE_TO_SELF, 0.5f);
	sa.setDuration(2000);
	sa.setRepeatMode(Animation.REVERSE);
	iv.setAnimation(sa);
	//将普通动画实例添加到 AnimationSet 中
	set.addAnimation(aa);
	set.addAnimation(sa);
	//启动集合动画
	set.start();

   }
```

至此，章节所有内容完！