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

Beta 0.01 版

By 阳哥

Android 自定义控件-1

优酷菜单&广告条&下拉选择框&滑动开关

1. 自定义控件简介（★★）

Android 本身提供了很多控件 比如我们常用的有：

- 文本控件 TextView 和 EditText

TextView 控件继承自 View 类。TextView 控件的功能是向用户显示文本内容 同时可选择性让用户编辑文本。其中 TextView 不允许编辑。

EditText 控件 EditText 控件继承自 TextView。EditText 与 TextView 最大的不同是 EditText 是可以编辑的

- 按钮控件 Button 和 ImageButton

Button 控件继承自 TextView 类 Button 的用法比较简单 主要是为 Button 控件设置 View.OnClickListener. 监听器 并在监听器的实现代码中编写按钮按下事件的处理代码。

ImageButton 控件 ImageButton 继承自 ImageView。ImageButton 与 Button 最大的区别是 ImageButton 没有 text 属性 既按钮中将显示图片而不是文本。 ImageButton 控件中设置显示图片可以通过 android:src 属性 也可以通过 setImageResource(int) 方法来实现

- 状态开关按钮 ToggleButton

ToggleButton 控件是继承自 Button。ToggleButton 的状态只能是选中和未选中 并且需要为不同的状态设置不同的显示文本。除了继承自父类的一些属性和方法之外 ToggleButton 也具有一些自己的 ToggleButton 属性。

- 单选复选按钮 RadioButton 和 RadioGroup
- 单选按钮和复选按钮 CheckBox

CheckBox 和 RadioButton 都只有选中和未选中两种状态,可以通过 checked 属性来设置.

不同的是 RadioButton 是单选按钮,需要编制到一个 RadioGroup 中同一时刻一个 RadioGroup 中只能有一个按钮处于选中状态.

CheckBox 和 RadioButton 都是继承自 CompoundButton 中继承了一些成员。

- 图片控件 ImageView

ImageView 控件负责显示图片,其图片来源既可以是资源文件的 id,也可以是 Drawable 对象或 Bitmap 对象,还可以是 Content Provider 的 Uri.

- 时钟控件 AnalogClock 和 DigitalClock

AnalogClock 继承自 View,AnalogClock 控件显示模拟时钟 只显示时针和分针

DigitalClock 继承自 TextView。DigitalClock 显示数字时钟 可精确到秒。 时钟控件比较简单 只需要在布局文件中声明控件即可。

- 进度条 ProgressBar 和日期与时间选择控件 DatePicker 和 TimePicker 等。

DatePicker 继承自 FrameLayout 类 日期选择控件的主要功能是向用户提供包含年、月、日的日期数据并允许用户对其进行选择。如果要捕获用户修改日期选择控件中数据的事件 需要为 DatePicker 添加 onDateChangedListener 监听器。

TimePicker 同样继承自 FrameLayout 类。时间选择控件向用户显示一天中的时间 可以为 24 小时制 可以为 AM/PM 制并允许用户进行选择。如果要捕获用户修改时间数据的事件 便需要为 TimePicker 添加 OnTimeChangedListener 监听器

但是这些控件并不能满足我们所有的要求。有的时候我们必须要自己定义控件来满足我们的要求。

2. 优酷菜单（★★）

优酷菜单是一个典型的组合控件，这里我们要进一步学习组合控件的使用。优酷菜单效果图如下：

图中由中间往外，分别是一级菜单、二级菜单、三级菜单。其基本用法是，点击一级菜单后加载二级菜单，再点击二级菜单加载三级菜单，再点击一级菜单分别隐藏三级、二级菜单。

2.1 优酷菜单布局

布局整体采用 RelativeLayout，最外层是一个 RelativeLayout，每一级菜单都是一个 RelativeLayout。

布局文件 activity_main.xml，清单如下：

```
<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"
    tools:context=".MainActivity" >

    <RelativeLayout
        android:id="@+id/rl_level1"
        android:layout_width="100dp"
        android:layout_height="50dp"
        android:layout_alignParentBottom="true"
        android:layout_centerHorizontal="true"
        android:background="@drawable/level1" >

        <ImageButton
            android:id="@+id/ib_home"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_alignParentBottom="true"
            android:layout_centerHorizontal="true"
            android:background="@android:color/transparent"
            android:src="@drawable/icon_home" />
    </RelativeLayout>
```

```
    <RelativeLayout
        android:id="@+id/rl_level2"
        android:layout_width="180dp"
        android:layout_height="90dp"
        android:layout_alignParentBottom="true"
        android:layout_centerHorizontal="true"
        android:background="@drawable/level2" >

        <ImageButton
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_alignParentBottom="true"
            android:layout_marginBottom="5dp"
            android:layout_marginLeft="10dp"
            android:background="@android:color/transparent"
            android:src="@drawable/icon_search" />

        <ImageButton
            android:id="@+id/ib_menu"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_centerHorizontal="true"
            android:layout_marginTop="5dp"
            android:background="@android:color/transparent"
            android:src="@drawable/icon_menu" />

        <ImageButton
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_alignParentBottom="true"
            android:layout_alignParentRight="true"
            android:layout_marginRight="10dp"
            android:background="@android:color/transparent"
            android:src="@drawable/icon_myyouku" />
    </RelativeLayout>

    <RelativeLayout
        android:id="@+id/rl_level3"
        android:layout_width="280dp"
        android:layout_height="142dp"
        android:layout_alignParentBottom="true"
        android:layout_centerHorizontal="true"
        android:background="@drawable/level3" >
```

```
        <ImageButton
            android:id="@+id/bt_channel1"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_alignParentBottom="true"
            android:layout_alignParentLeft="true"
            android:layout_marginBottom="15dp"
            android:layout_marginLeft="10dp"
            android:background="@android:color/transparent"
            android:src="@drawable/channel1" />

        <ImageButton
            android:id="@+id/ib_channel2"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_above="@id/bt_channel1"
            android:layout_marginBottom="20dp"
            android:layout_marginLeft="40dp"
            android:background="@android:color/transparent"
            android:src="@drawable/channel2" />

        <ImageButton
            android:id="@+id/bt_channel3"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_marginTop="15dp"
            android:layout_toRightOf="@id/ib_channel2"
            android:background="@android:color/transparent"
            android:src="@drawable/channel3" />

        <ImageButton
            android:id="@+id/bt_channel4"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_alignParentTop="true"
            android:layout_centerHorizontal="true"
            android:background="@android:color/transparent"
            android:src="@drawable/channel4" />

        <ImageButton
            android:id="@+id/bt_channel5"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
```

```
            android:layout_alignParentBottom="true"
            android:layout_alignParentRight="true"
            android:layout_marginBottom="15dp"
            android:layout_marginRight="10dp"
            android:background="@android:color/transparent"
            android:src="@drawable/channel5" />

        <ImageButton
            android:id="@+id/ib_channel6"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_above="@id/bt_channel5"
            android:layout_alignParentRight="true"
            android:layout_marginBottom="20dp"
            android:layout_marginRight="40dp"
            android:background="@android:color/transparent"
            android:src="@drawable/channel6" />

        <ImageButton
            android:id="@+id/ib_channel7"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_marginTop="15dp"
            android:layout_toLeftOf="@id/ib_channel6"
            android:background="@android:color/transparent"
            android:src="@drawable/channel7" />
    </RelativeLayout>

</RelativeLayout>
```

2.2 优酷菜单业务逻辑实现

```
public class MainActivity extends Activity implements OnClickListener {
    //用于标记相应菜单显示状态
    private boolean isShowLevel3 = true;
    private boolean isShowLevel2 = true;
    private boolean isShowMenue = true;
    //一、二、三三个菜单是三个 RelativeLayout
    private RelativeLayout rl_level1;
    private RelativeLayout rl_level2;
```

```
private RelativeLayout rl_level3;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        //初始化菜单
        initView();
    }

    private void initView(){
        //初始化三个菜单控件
        rl_level1 = (RelativeLayout) findViewById(R.id.rl_level1);
        rl_level2 = (RelativeLayout) findViewById(R.id.rl_level2);
        rl_level3 = (RelativeLayout) findViewById(R.id.rl_level3);
        //初始化一级菜单的 home 按钮
        ImageButton ib_home = (ImageButton) findViewById(R.id.ib_home);
        //初始化二级菜单的 menu 按钮
        ImageButton ib_menu = (ImageButton) findViewById(R.id.ib_menu);
        //给 home、menu 按钮添加 click 事件，为了方便我们将当前 Activity 实现 OnClickListener
接口
        ib_home.setOnClickListener(this);
        ib_menu.setOnClickListener(this);

    }
    //覆写 OnClickListener 方法
    @Override
    public void onClick(View v) {
        //判断当前是否还有动画没有执行完闭
        if (AnimUtil.animationCount>0) {
            return;
        }
        //判断点击的控件时哪个
        switch (v.getId()) {
        //如果是 home 按钮
        case R.id.ib_home:
            //判断二级菜单是否显示
            if (isShowLevel2) {
                //如果二级菜单显示再判断三级菜单是否显示
                if (isShowLevel3) {
                    //如果三级菜单也显示则关闭三级菜单
                    AnimUtil.startOutAnimation(rl_level3);
                    //并将三级菜单显示状态设置为 false
```

```
                isShowLevel3 = false;
            }
            //如果仅仅二级菜单显示，三级菜单不显示则关闭二级菜单，动画演示 500 毫秒
            AnimUtil.startOutAnimation(rl_level2,500L);
        }else {
            //如果二级菜单都没显示则显示二级菜单
            AnimUtil.startInAnimation(rl_level2);
        }
        //不管二级菜单是什么状态，点击了 home 按钮，显示状态肯定要设置为相反的
        isShowLevel2 = !isShowLevel2;
        break;
        //如果点击的是 menu 按钮
    case R.id.ib_menu:
        //如果三级菜单处于显示状态，则设置为隐藏
        if(isShowLevel3){
            //隐藏
            AnimUtil.startOutAnimation(rl_level3);
        }else {
            //如果三级菜单处于隐藏状态，则显示
            AnimUtil.startInAnimation(rl_level3);
        }
        //将三级菜单显示状态反转
        isShowLevel3 = !isShowLevel3;
        break;
    default:
        break;
    }
}
/*
 * 覆写按钮点击事件，这里主要覆写 Android 手机 menu 键的点击事件，当点击手机的 menu 的时
 * 候如果菜单显示则全部隐藏，
 * 如果隐藏则显示全部，不管是隐藏还是显示都执行动画效果
 */

@Override
public boolean onKeyDown(int keyCode, KeyEvent event) {
    long delayTime = 0;
    //如果点击的是 menu 按钮
    if (keyCode==KeyEvent.KEYCODE_MENU) {
        //如果还有未执行完的动画则直接返回，不做任何操作
        if (AnimUtil.animationCount>0) {
            return true;
        }
```

```
            //如果优酷菜单的 menu 处于显示状态，则按照三级、二级、一级的顺序隐藏菜单
            if (isShowMenue) {
                if (isShowLevel3) {
                    //如果三级菜单显示，则隐藏，并执行延时动画
                    AnimUtil.startOutAnimation(rl_level3,delayTime);
                    isShowLevel3 = false;
                    //延时时间加 200 毫秒
                    delayTime=+200;
                }
                if (isShowLevel2) {
                    //如果二级菜单处于显示状态，则隐藏，并执行延时动画
                    AnimUtil.startOutAnimation(rl_level2,delayTime);
                    isShowLevel2 = false;
                    //延时时间加 200 毫秒
                    delayTime+=200;
                }
                //将一级菜单隐藏
                AnimUtil.startOutAnimation(rl_level1,delayTime);
                isShowMenue = !isShowMenue;
            }else {
                //如果菜单未显示则
                //按照一级、二级、三级的顺序显示菜单
                AnimUtil.startInAnimation(rl_level1);
                AnimUtil.startInAnimation(rl_level2, 200L);
                AnimUtil.startInAnimation(rl_level3, 400L);
                isShowMenue=isShowLevel2=isShowLevel3=true;
            }
            return true;
        }
        return super.onKeyDown(keyCode, event);
    }
}
```

2.3 AnimUtil 的业务逻辑实现

在 2.2 节中我们使用到了 AnimUtil 工具类，该类主要是完成了菜单控件显示和隐藏的动画效果，其业务逻辑代码如下：

```
public class AnimUtil {
    // 记录当前动画个数，每开始执行一个动画则加 1，结束一个动画则减 1
    public static int animationCount = 0;

    // 开始执行隐藏动画
    public static void startOutAnimation(RelativeLayout rl) {
        // 遍历 RelativeLayout 的所有子控件
        for (int i = 0; i < rl.getChildCount(); i++) {
            // 将子控件设置为不可用
            rl.getChildAt(i).setEnabled(false);
        }
        /*
         * 初始化一个旋转动画
         * 第一个参数 fromDegrees Rotation offset to apply at the start of the
         * animation.//其实角度，这里我们设置为 0
         * 第二个参数 toDegrees Rotation offset to apply at the end of the
         * animation.//目标角度，这是设置为-180 度
         * 第三个参数 pivotXType Specifies how pivotXValue should be
         * interpreted. One of Animation.ABSOLUTE, Animation.RELATIVE_TO_SELF,
         * or Animation.RELATIVE_TO_PARENT.//相对于 X 坐标类型，这里是相对于自己
         * 第四个参数 pivotXValue The X coordinate of the
         * point about which the object is being rotated, specified as an
         * absolute number where 0 is the left edge. This value can either be an
         * absolute number if pivotXType is ABSOLUTE, or a percentage (where 1.0
         * is 100%) otherwise.//相对于 X 坐标类型的值，这里是 0.5f,也就是 X 轴的一半
         * 第五个参数 pivotYType Specifies how pivotYValue should be
         * interpreted. One of Animation.ABSOLUTE, Animation.RELATIVE_TO_SELF,
         * or Animation.RELATIVE_TO_PARENT.//相对于 Y 坐标类型，这里是相对于自己
         * 第六个参数 pivotYValue The Y coordinate of the
         * point about which the object is being rotated, specified as an
         * absolute number where 0 is the top edge. This value can either be an
         * absolute number if pivotYType is ABSOLUTE, or a percentage (where 1.0
         * is 100%) otherwise.//相对于 Y 坐标类型的值，这里是 1.f,也就是一坐标最大处
         *
         * 上面六个参数的整体效果就是按照控件 x 轴中心，y 轴最下方（Y 轴是越往下 Y 值越大）逆
时针旋转 180 度
         */
        RotateAnimation rotateAnimation = new RotateAnimation(0, -180,
Animation.RELATIVE_TO_SELF, 0.5f, Animation.RELATIVE_TO_SELF, 1.f);
        //旋转演示 500 毫秒
        rotateAnimation.setDuration(500);
        //动画执行完后控件是否还停留在远处，这里设置为 true
        rotateAnimation.setFillAfter(true);
```

```
        //给动画添加一个监听器，用于监听动画的执行完毕，因为执行前需要将 animationCount+1，执行完毕后需要 animationCount-1
        rotateAnimation.setAnimationListener(new MyAnimationListener());
        rl.startAnimation(rotateAnimation);
    }
    //方法重载，延时 millis 毫秒后再执行
    public static void startOutAnimation(RelativeLayout rl, Long millis) {
        for (int i = 0; i < rl.getChildCount(); i++) {
            rl.getChildAt(i).setEnabled(false);
        }
        RotateAnimation rotateAnimation = new RotateAnimation(0, -180,
Animation.RELATIVE_TO_SELF, 0.5f, Animation.RELATIVE_TO_SELF, 1.f);
        rotateAnimation.setDuration(500);
        rotateAnimation.setFillAfter(true);
        rotateAnimation.setStartOffset(millis);
        rotateAnimation.setAnimationListener(new MyAnimationListener());
        rl.startAnimation(rotateAnimation);
    }
    //显示控件，原理跟隐藏动画几乎一模一样，唯一不同的就是旋转参数不停
    public static void startInAnimation(RelativeLayout rl) {
        for (int i = 0; i < rl.getChildCount(); i++) {
            rl.getChildAt(i).setEnabled(true);
        }
        RotateAnimation rotateAnimation = new RotateAnimation(-180, 0,
Animation.RELATIVE_TO_SELF, 0.5f, Animation.RELATIVE_TO_SELF, 1.f);
        rotateAnimation.setDuration(500);
        rotateAnimation.setFillAfter(true);
        rotateAnimation.setAnimationListener(new MyAnimationListener());
        rl.startAnimation(rotateAnimation);
    }

    public static void startInAnimation(RelativeLayout rl, Long delayTime) {
        for (int i = 0; i < rl.getChildCount(); i++) {
            rl.getChildAt(i).setEnabled(true);
        }
        RotateAnimation rotateAnimation = new RotateAnimation(-180, 0,
Animation.RELATIVE_TO_SELF, 0.5f, Animation.RELATIVE_TO_SELF, 1.f);
        rotateAnimation.setDuration(500);
        rotateAnimation.setFillAfter(true);
        rotateAnimation.setStartOffset(delayTime);
        rotateAnimation.setAnimationListener(new MyAnimationListener());
        rl.startAnimation(rotateAnimation);
    }
```

```
    //动画监听器
    static class MyAnimationListener implements AnimationListener {

        @Override
        public void onAnimationStart(Animation animation) {
            //开始一个动画的时候
            animationCount++;
        }

        @Override
        public void onAnimationEnd(Animation animation) {
            //动画结束的时候
            animationCount--;
        }

        @Override
        public void onAnimationRepeat(Animation animation) {
        }
    }
}
```

3. 广告条（★★）

3.1 广告条布局

广告条做好以后如上图所示，其实就是我们经常见到的滚动广告，默认情况下每隔 N 秒会自动滚动，用手指滑动时也会切换到上一张或者下一张。

核心控件时 ViewPager 是高版本加入的新控件，为了向下兼容，我们使用 android.support.v4.view.ViewPager 类。

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
   >
    <android.support.v4.view.ViewPager
        android:id="@+id/viewPager"
        android:layout_width="fill_parent"
        android:layout_height="200dp" >
    </android.support.v4.view.ViewPager>
    <LinearLayout
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:layout_alignBottom="@id/viewPager"
        android:background="#55000000"
        android:gravity="center_horizontal"
        android:orientation="vertical"
        android:padding="5dip"
        >
        <TextView
            android:id="@+id/tv_img_desc"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="图片描述信息"
            android:textColor="#ffffff"
            android:textSize="16sp" />
        <LinearLayout
            android:id="@+id/ll_poin_group"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_marginTop="5dip"
            android:orientation="horizontal" >
        </LinearLayout>
    </LinearLayout>
</RelativeLayout>
```

上面布局整体是一个 RelativeLayout，在 RelativeLayout 中添加了 ViewPaper 和 LinearLayout 布局。

3.2 广告条业务逻辑实现

ViewPaper 的用法相对简单，跟 ListView 的用法有的一拼！

```
public class MainActivity extends Activity implements OnPageChangeListener {
    private ViewPager mViewPager;
    private LinearLayout mLinearLayout;
    private TextView mTextView;
    private List<ImageView> imageViews;
    private String[] imgDescs;
    private boolean isStop = true;
    private int priviousPosition = 0;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        // 初始化视图
        initView();
        // 在子线程中开启每 3 秒更新一次 ViewPager 对象
        new Thread(new Runnable() {

            @Override
            public void run() {
                while (isStop) {
                    SystemClock.sleep(3000);
                    /*
                     * 运行在主线程中的任务 Runs the specified action on the UI thread. If
                     * the current thread is the UI thread, then the action is
                     * executed immediately. If the current thread is not the UI
                     * thread, the action is posted to the event queue of the UI
                     * thread.
                     */
                    runOnUiThread(new Runnable() {

                        @Override
                        public void run() {
                            mViewPager.setCurrentItem(mViewPager.getCurrentItem() + 1);
                        }
```

```
                });

            }
        }
    }).start();
}
private void initView() {
    // 初始化控件
    mViewPager = (ViewPager) findViewById(R.id.viewPager);
    mLinearLayout = (LinearLayout) findViewById(R.id.ll_poin_group);
    mTextView = (TextView) findViewById(R.id.tv_img_desc);
    // 初始化数据
    initData();
}

private void initData() {
    int[] imgs = { R.drawable.a, R.drawable.b, R.drawable.c, R.drawable.d,
R.drawable.e };
    imgDescs = new String[] { "图片描述 1", "图片描述 2", "图片描述 3", "图片描述 4",
"图片描述 5", "图片描述 6" };
    // 设置第一个文本描述为当前文本图片描述信息
    mTextView.setText(imgDescs[0]);
    imageViews = new ArrayList<ImageView>();
    ImageView iv;
    LayoutParams params;
    // 有多少图片资源就创建多少个 ImageView 对象
    for (int i = 0; i < imgs.length; i++) {
        // 创建 ImageView 对象
        iv = new ImageView(this);
        // 给 ImageView 设置图片资源为背景
        iv.setBackgroundResource(imgs[i]);
        // 添加到集合中
        imageViews.add(iv);
        // 创建一个 View 对象，其实就是白色的点点，用于显示当前滚动到第几张图片
        View v = new View(this);
        v.setBackgroundResource(R.drawable.point_bg);
        // 布局参数
        params = new LayoutParams(5, 5);
        if (i != 0) {
            // 设置布局参数左边距为 5pix
            params.leftMargin = 5;
            v.setEnabled(false);
        } else {
```

```
            v.setEnabled(true);
        }
        // 给 View 对象设置布局参数
        v.setLayoutParams(params);
        // 在 LinearLayout 布局中添加 View 对象
        mLinearLayout.addView(v);
    }
    // 给 ViewPaper 设置更新监听器
    mViewPager.setOnPageChangeListener(this);
    // 给 ViewPaper 设置适配器（跟 ListView 的适配器有点儿类似）
    mViewPager.setAdapter(new MyAdapter());
    /*
     * 为了给用户以可以无限制循环滚动的效果，我们是默认位置在 Integer.MAX_VALUE 的中间
默认情况下，滚动到头就无法再滚动
     */
    int m = (Integer.MAX_VALUE / 2) % imageViews.size();
    int currentPosition = Integer.MAX_VALUE / 2 - m;
    mViewPager.setCurrentItem(currentPosition);
}

// ViewPaper 适配器
class MyAdapter extends PagerAdapter {
    // 返回总共有多少个 ViewPaper 对象，这里给出 Integer 的最大值
    @Override
    public int getCount() {
        return Integer.MAX_VALUE;
    }

    /*
     * Determines whether a page View is associated with a specific key
     * object as returned by instantiateItem(ViewGroup, int). This method is
     * required for a PagerAdapter to function properly.
     * 判断当前的 pageView 对象是否是通过 instantiateItem 创建的 Object 对象
     */
    @Override
    public boolean isViewFromObject(View view, Object object) {
        System.out.println("isViewFromObject");
        return view == object;
    }

    // 销毁 Item
    @Override
    public void destroyItem(View container, int position, Object object) {
```

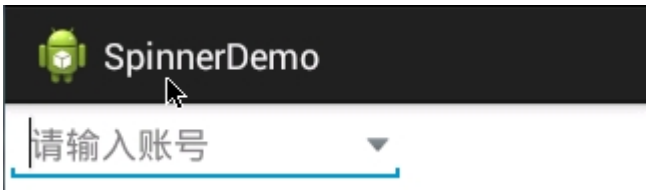
```
            // 从 ViewPaper 中销毁一个对象 imageViews 集合中第当前位置取模该集合的长度
            mViewPager.removeView(imageViews.get(position % imageViews.size()));
            System.out.println("destroyItem");
        }

        /*
         * Create the page for the given position. The adapter is responsible
         * for adding the view to the container given here, although it only
         * must ensure this is done by the time it returns from
         * finishUpdate(ViewGroup).
         * 创建一个 ViewPaper 对象
         */
        @Override
        public Object instantiateItem(ViewGroup container, int position) {
            mViewPager.addView(imageViews.get(position % imageViews.size()));
            System.out.println("instantiateItem");
            return imageViews.get(position % imageViews.size());
        }
    }

    @Override
    public void onPageScrolled(int position, float positionOffset, int
positionOffsetPixels) {

    }
    /*
     * 当选中某个 pager 的时候调用该方法
     */
    @Override
    public void onPageSelected(int position) {
        //获取当前 position 取模于 imageViews 的长度
        int newPosition = position % imageViews.size();
        //设置显示文本信息
        mTextView.setText(imgDescs[newPosition]);
        //将前一个对象设置为不可用
        mLinearLayout.getChildAt(priviousPosition).setEnabled(false);
        //将当前对象设置可用
        mLinearLayout.getChildAt(newPosition).setEnabled(true);
        //将当前对象赋值为前一个对象
        priviousPosition = newPosition;
    }

    @Override
```

```
    public void onPageScrollStateChanged(int state) {
    }
    @Override
    protected void onDestroy() {
        super.onDestroy();
        isStop = false;
    }
}
```

4. 下拉选择框（★★）

4.1 下拉选择框布局

下拉选择框主要是通过在 EditText 下用 PopupWindow 动态显示 ListView 控件来实现的。下拉选择框可以方便用户的输入效率，以此提升用户体验。实现后的效果如下图所示。



点击 EditText 控件右侧的倒三角形弹出如下可选项，点击可选项，那么内容将自动填充到 EditText 控件中。

布局文件为 activity_main.xml ,布局清单如下所示 :这里的布局仅仅是在 EditText 的右侧添加了一个倒三角图标。

```
<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"
    tools:context=".MainActivity" >
    <EditText
        android:id="@+id/et"
        android:layout_width="200dp"
        android:layout_height="wrap_content"
        android:hint="请输入账号" />
    <ImageButton
        android:id="@+id/ib"
        android:onClick="click"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:src="@drawable/down_arrow"
        android:layout_alignRight="@id/et"
        android:background="@android:color/transparent"
        android:layout_alignBottom="@id/et"
        android:layout_alignTop="@id/et"
        />

</RelativeLayout>
```

点击倒三角弹出的 ListView 条目布局文件为 listview_item.xml，其清单如下：

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:padding="5dp"
    android:layout_gravity="center_vertical"
    android:descendantFocusability="blocksDescendants"
    android:orientation="horizontal" >
    <ImageView
        android:background="@android:color/transparent"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:src="@drawable/user" />
    <TextView
        android:id="@+id/tv_number"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
```

```
        android:layout_weight="1"
        android:layout_marginLeft="10dp"
        android:text="1000" />

    <ImageView
        android:id="@+id/iv_delete"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:background="@android:color/transparent"
        android:src="@drawable/delete" />

</LinearLayout>
```

4.2 下拉选择框业务逻辑实现

```
public class MainActivity extends Activity implements OnItemClickListener {

    private List<String> listData;
    private ListView listView;
    private EditText et_number;
    private PopupWindow popupWindow;
    private MyAdapter adapter;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        et_number = (EditText) findViewById(R.id.et);
        // 初始化 ListView
        initListView();
    }

    private void initListView() {
        listData = new ArrayList<String>();
        // 模拟 30 个数据
        for (int i = 0; i < 30; i++) {
            listData.add((10000 + i) + "");
        }
        listView = new ListView(this);
        // 将 ListView 的分割线高度设置为 0，其实保留着也行，可能比较难看就是
        listView.setDividerHeight(0);
```

```
        // 给 ListView 设置背景图片
        listView.setBackgroundResource(R.drawable.listview_background);
        // 将 ListView 的滚动条隐藏
        listView.setVerticalScrollBarEnabled(false);
        // 给 ListView 设置 ListView 点击事件，点击以后以获取其中的数据
        listView.setOnItemClickListener(this);
        adapter = new MyAdapter();
        // 设置适配器
        listView.setAdapter(adapter);
    }

    // 点击倒三角图标触发该事件
    public void click(View view) {
        // 如果 popupWindow 处于显示状态则隐藏
        if (popupWindow != null && popupWindow.isShowing()) {
            popupWindow.dismiss();
        } else {
            /*
             * 如果 popupWindow 不显示则创建 第一个参数 contentView the popup's content
             * popupWindow 的内容，这里是 ListView 对象 第二个参数 width the
             * 宽度，这里是 EditText 的宽度-4 个 pix（自己多次调整的结果） popup's width 第
三个参数 height the
             * popup's height
             */
            popupWindow = new PopupWindow(listView, et_number.getWidth() - 4, 300);
            // 设置可以获取焦点
            popupWindow.setFocusable(true);
            /*
             * Controls whether the pop-up will be informed of touch events
             * outside of its window. This only makes sense for pop-ups that are
             * touchable but not focusable, which means touches outside of the
             * window will be delivered to the window behind. The default is
             * false.
             *
             * If the popup is showing, calling this method will take effect
             * only the next time the popup is shown or through a manual call to
             * one of the update() methods.
             *  就是在 popupWindow 控件外面是否让触摸事件生效，这里我们让其生效，效果就是触
摸 popupWindow 外面屏幕时，会自动隐藏 popupWindow
             */
            popupWindow.setOutsideTouchable(true);
            /*
             * Change the background drawable for this popup window. The
```

```
            * background can be set to null. 设置背景，要求不能为 null
            */
           popupWindow.setBackgroundDrawable(new BitmapDrawable());
           /*
            * Display the content view in a popup window anchored to the
            * bottom-left corner of the anchor view offset by the specified x
            * and y coordinates. If there is not enough room on screen to show
            * the popup in its entirety, this method tries to find a parent
            * scroll view to scroll. If no parent scroll view can be scrolled,
            * the bottom-left corner of the popup is pinned at the top left
            * corner of the anchor view.
            *
            * If the view later scrolls to move anchor to a different location,
            * the popup will be moved correspondingly.
            *
            * 简单的说就是给 popupWindow 设置一个父 View，然后设置相对该父 View 的 x 偏移量
和 y 偏移量
            */
           popupWindow.showAsDropDown(et_number, 2, -2);
       }
   }

   class MyAdapter extends BaseAdapter {

       @Override
       public int getCount() {
           return listData.size();
       }

       @Override
       public Object getItem(int position) {
           return null;
       }

       @Override
       public long getItemId(int position) {
           return 0;
       }

       @Override
       public View getView(final int position, View convertView, ViewGroup parent) {
           ViewHolder holder;
           if (convertView == null) {
```

```
            convertView = View.inflate(MainActivity.this, R.layout.listview_item,
null);
            holder = new ViewHolder();
            holder.imageView = (ImageView)
convertView.findViewById(R.id.iv_delete);
            holder.textView = (TextView) convertView.findViewById(R.id.tv_number);
            convertView.setTag(holder);
        } else {
            holder = (ViewHolder) convertView.getTag();
        }
        holder.textView.setText(listData.get(position));
        holder.imageView.setOnClickListener(new OnClickListener() {

            @Override
            public void onClick(View v) {
                System.out.println("删除了" + position);
                listData.remove(position);
                adapter.notifyDataSetChanged();
                if (listData.size() == 0) {
                    popupWindow.dismiss();
                }

            }
        });
        return convertView;
    }
}

class ViewHolder {
    ImageView imageView;
    TextView textView;
}
//点击后将点击的号码设置个 EditText
@Override
public void onItemClick(AdapterView<?> parent, View view, int position, long id)
{
    String number = listData.get(position);
    et_number.setText(number);
}
}
```

5. 滑动开关（★★）

5.1 滑动开关布局

滑动开关效果如下图所示，这是再常见不过的控件了。

布局文件为 activity_main.xml，其清单如下：

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:itheima="http://schemas.android.com/apk/res/com.example.togglebutton"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity" >
<com.example.togglebutton.ToggleButton
   android:id="@+id/tb"
   itheima:switchBackgroundID="@drawable/slide_button_background"
   itheima:slideButtonBackgroundID="@drawable/switch_background"
   itheima:toggleState="true"
   android:layout_width="wrap_content"
   android:layout_height="wrap_content"
   android:layout_centerInParent="true"
    />

</RelativeLayout>
```

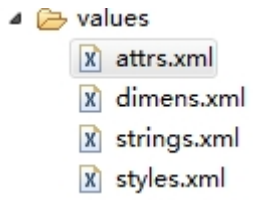
Tips：布局其实并没啥，主要是 com.example.togglebutton.ToggleButton 类的实现和自定义属性的使用。

5.2 滑动开关业务逻辑实现

1 添加自定义属性。在 values 目录下创建 attrs.xml 文件，文件清单如下：

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <declare-styleable name="ToggleButton">
    <attr name="switchBackgroundID" format="reference" ></attr>
    <attr name="slideButtonBackgroundID" format="reference" ></attr>
    <attr name="toggleState" format="boolean" ></attr>
    </declare-styleable>
</resources>
```

目录结构如下图所示：



2 在布局文件中用到自定义属性时需要引入自定义的命名空间。

在 activity_main.xml 布局中引入如下命名空间：

xmlns:itheima="http://schemas.android.com/apk/res/com.example.togglebutton"

Com.example.togglebutton 是包名，其实名字可以自定义，并不是非得用类名，但是我们习惯用类型作为自定义属性的命名空间。

3 自定义类 ToggleButton 继承 View 类。

ToggleButton 类代码清单如下：

```
public class ToggleButton extends View {
    // 当前是开或者关，开关的状态
    private boolean currentState = false;
    // 该按钮其实是在一个背景图片上叠加一个图片
    private Bitmap mSwitchBackground;
    private Bitmap mSlideButtonBackground;
    private int currentX = 0;

    // 自定义控件必须覆写父类的三个构造函数
    public ToggleButton(Context context, AttributeSet attrs, int defStyle) {
        super(context, attrs, defStyle);
    }

    public ToggleButton(Context context, AttributeSet attrs) {
```

```
        super(context, attrs);
        // 通过下面函数可以获取自定义属性的值，不过值为字符串类型，这里不用这种方法，如果是简单的字符串值推荐使用这种方法
        // String switchBackgroundID =
        //
attrs.getAttributeValue("http://schemas.android.com/apk/res/com.example.togglebutton",
        // "switchBackgroundID");
        // 从 attrs 中获取自定义属性的类型
        TypedArray typedArray = context.obtainStyledAttributes(attrs,
R.styleable.ToggleButton);
        // 获取类型总数
        int indexCount = typedArray.getIndexCount();
        // 遍历类型
        for (int i = 0; i < indexCount; i++) {
            // 属性是按照顺序排列的
            int index = typedArray.getIndex(i);
            switch (index) {
            case R.styleable.ToggleButton_slideButtonBackgroundID:
                // 因为是引用类型，因此获取其 id
                int resourceId = typedArray.getResourceId(index, -1);
                setSlideButtonBackground(resourceId);
                break;
            case R.styleable.ToggleButton_switchBackgroundID:
                int resourceId2 = typedArray.getResourceId(index, -1);
                setSwitchBackground(resourceId2);
                break;
            case R.styleable.ToggleButton_toggleState:
                // 获取 Boolean 类型属性值
                boolean b = typedArray.getBoolean(index, false);
                setCurrentState(b);
                break;

            default:
                break;
            }
        }
        // 让其循环，一遍下次再使用，我们可以将其遍历后指针跑到了最后面，为了方便下次使用，因此这里将指针移动到最前面，如果确定不用了，可以不执行该句代码，但是建议执行
        typedArray.recycle();

    }
```

```
    public ToggleButton(Context context) {
        super(context);
    }

    /**
     * 设置当前开关图片资源 id
     */
    public void setSwitchBackground(int slideButtonBackground) {
        mSwitchBackground = BitmapFactory.decodeResource(getResources(),
slideButtonBackground);
    }

    /**
     * 设置滑块的图片资源 id
     */
    public void setSlideButtonBackground(int switchBackground) {
        mSlideButtonBackground = BitmapFactory.decodeResource(getResources(),
switchBackground);
    }

    public void setCurrentState(boolean currentState) {
        this.currentState = currentState;
    }

    // 测量
    @Override
    protected void onMeasure(int widthMeasureSpec, int heightMeasureSpec) {
        super.onMeasure(widthMeasureSpec, heightMeasureSpec);
        // 测量背景图片的大小
        setMeasuredDimension(mSlideButtonBackground.getWidth(),
mSlideButtonBackground.getHeight());
    }

    // 绘制
    @Override
    protected void onDraw(Canvas canvas) {
        int left = 0;
        /*
         * 绘制背景图
         * 第一个参数 bitmap The bitmap to be drawn
         * 第二个参数 left The position of the left side of the bitmap being drawn top
The position of the top side of
         * the bitmap being drawn
```

```
         * 第三个参数 paint The paint used to draw the bitmap (may be null)
         */
        canvas.drawBitmap(mSlideButtonBackground, 0, 0, null);
        //计算第二张（上面那个按钮）图片左侧的位置
        left = currentX - mSwitchBackground.getWidth() / 2;
        //如果 left<0 则等于 0，不然第二张图片就跑出第一张图片的范围了
        if (left < 0) {
            left = 0;
            //第二张图片右侧也不能跑出第一张图片右侧边界
        } else if (left + mSwitchBackground.getWidth() >
mSlideButtonBackground.getWidth()) {
            left = mSlideButtonBackground.getWidth() - mSwitchBackground.getWidth();
        }
        //绘制第二张图片，偏移量为 left
        canvas.drawBitmap(mSwitchBackground, left, 0, null);
    }
    //触摸事件
    @Override
    public boolean onTouchEvent(MotionEvent event) {
        switch (event.getAction()) {
        case MotionEvent.ACTION_DOWN:
            //按下时记录当前 x 坐标
            currentX = (int) event.getX();
            break;
        case MotionEvent.ACTION_UP:
            currentX = (int) event.getX();
            //如果滑动到的 x 坐标小于 1/2 第一张图片坐标，则将当前坐标设置为 0，这样放手时不至
于第二张图片处于一个尴尬的位置（要么在左侧，要么在右侧）
            if (currentX < mSlideButtonBackground.getWidth() / 2) {
                currentX = 0;
                currentState = false;
                Toast.makeText(getContext(), "开关已经关闭", 1).show();
            } else {
                //相反的，如果滑动的第二张图的大于 1/2 处，则将当前 x 设置为最右侧
                currentX = mSlideButtonBackground.getWidth() -
mSwitchBackground.getWidth() / 2;
                currentState = true;
                Toast.makeText(getContext(), "开关已经开启", 0).show();
            }
            break;
        case MotionEvent.ACTION_MOVE:
            currentX = (int) event.getX();
            break;
```

```
        default:
            break;
        }
        //让控件重新绘制
        invalidate();
        return true;
    }
}
```

至此，本文档完！

2015 年 2 月 25 日 星期三 18:28:31

河南省济源市梨林镇