Android 自定义控件

讲师：杨光福

微博：http://weibo.com/321chinavideo

Day1

1_系统控件回顾-12

01_什么是自定义控件

Android 系统中，继承 Android **系统自带的 View** 或者 **ViewGroup 控件**或者**系统自带的控件**，并在这基础上增加或者重新组合成我们想要的效果。

02_为什么用自定义控件

系统控件无法满足需求时，需要自定义控件。

1、系统的控件在不同手机长得不一样，我们希望在不同手机实现相同的效果；

2、有些手机上的控件长得不好看，希望好看一些。

3、系统控件的功能有限，需要在基础上增加功能。

03_怎么用自定义控件-三种方式

1、使用系统控件，重新组合，实现自定义的效果，案例有：

优酷环形菜单、广告条(ViewPager)、下拉菜单(spinner)

2、自己定义一个类继承 View ，实现特定的效果，案例有：

自定义开关按钮、水波纹效果、自定义属性

3、自己定义一个类继承 ViewGroup，实现特定的效果，案例有：

仿 ViewPager 的效果实现 、 仿网易侧滑菜单

4、自定义属性：给自己的控件，添加自己的属性，通过 demo 了解系统解析属性的过程，

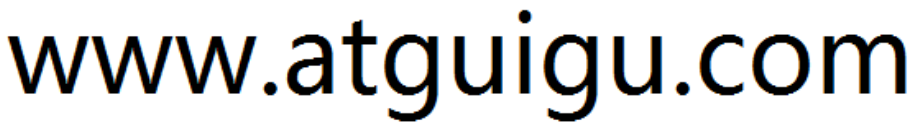
并给上一个例子开关按钮，添加新属性。

04_Android 常用控件回顾

Android 本身提供了很多控件，如：
文本控件　　TextView 和 EditText；
图片控件　　ImageView
按钮控件　　Button 和 ImageButton
进度条　　ProgressBar
单选按钮　　RadioButton 和 RadioGroup
复选按钮　　CheckBox
状态开关按钮 ToggleButton
时钟控件　　AnalogClock 和 DigitalClock
日期与时间选择控件 DatePicker 和 TimePicker 等。
　　。。。

使用原则：尽量使用系统的控件，在系统控件没法达到我们的需求的时候才需要自定义控件。再定义控件会带来工作量，例如修改 bug.

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



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 内容提供者（Content Provider）的 Uri.

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 设置一个点击事件监听器，并在编写按钮点击事件的处理代码。

ImageButton 控件 继承自 ImageView。

ImageButton 与 Button 相同之处：都用于响应按钮的点击事件

不同之处：ImageButton 只能显示图片；Button 用于显示文字

进度条 ProgressBar

ProgressBar 继承自 View，用于显示正在运行的状态。有两种显示形式：一种是环形显示只用于显示状态，没有具体的进度。第二种是水平显示，可以显示具体的进度。

通过设置不同的 Style 显示不同的样式：

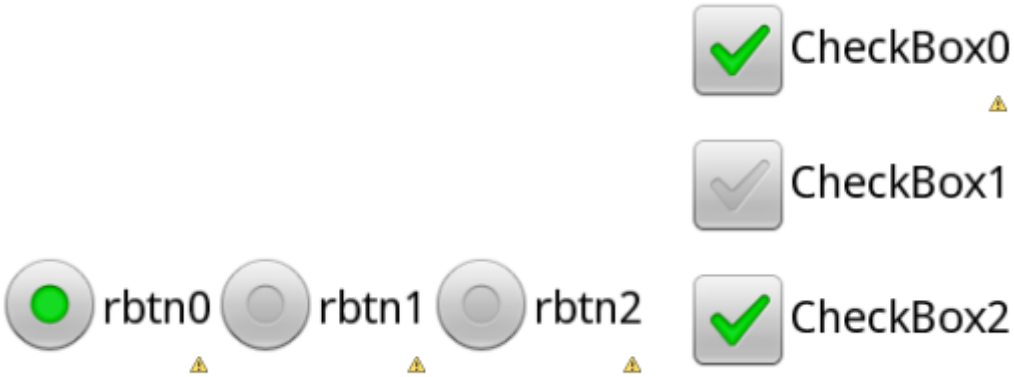
style="?android:attr/progressBarStyleLarge"　　环形样式

style="?android:attr/progressBarStyleHorizontal"　水平样式

单选按钮 RadioButton 和复选按钮 CheckBox

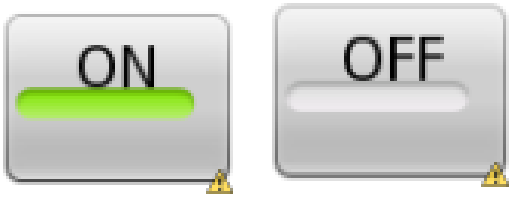
CheckBox 和 RadioButton 都继承自 CompoundButton，都只有选中和未选中两种状态,可以通过 checked 属性来设置。

不同的是 RadioButton 是单选按钮,在一个 RadioGroup 中只能有一个 RadioButton 按钮处于选中状态；CheckBox 则可以有多个按钮被选中。



状态开关按钮 ToggleButton

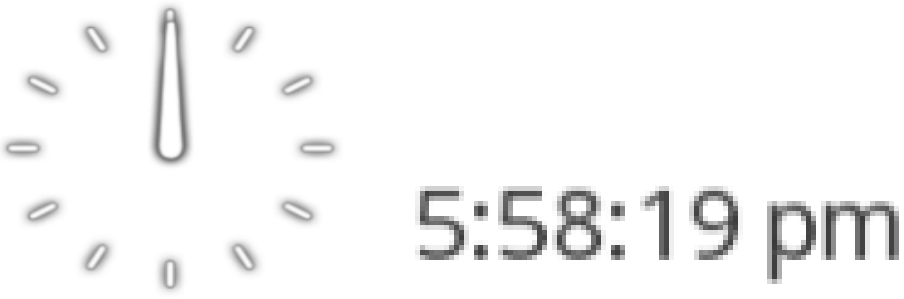
ToggleButton 控件是继承自 CompoundButton。ToggleButton 的状态只能是选中和未选中,并且需要为不同的状态设置不同的显示文本。除了继承自父类的一些属性和方法之外,ToggleButton 也具有一些自己的属性。



时钟控件 AnalogClock 和 DigitalClock

AnalogClock 继承自 View，用于显示模拟时钟只显示时针和分针。

DigeitalClock 继承自 TextView。用于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。



日期选择器 DatePicker 和时间选择器 TimePicker

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，并允许用户对其修改。如果要捕获这个修改,可以为 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，并允许用户进行修改。如果要捕获用户的修改事件,需要为 TimePicker 添加 OnTimeChangedListener 监听器

知识链接：

http://www.myexception.cn/android/1236819.html

系统提供的控件虽然很丰富，但是，还远远不够。有的时候我们必须要自己定义控件来满足我们的要求。

2_优酷效果

运行演示做好的优酷菜单效果，并且讲解实现思路；

01_优酷布局-25

1_创建工程 YukuMenuDemo，图片全部拷贝到 drawable-hdpi 目录下

2_实现三个圆环-最里面的圆环

```
<RelativeLayoutxmlns:android="http://schemas.android.com/apk/res/androi
```

```
d"
android:layout_width="match_parent"
android:layout_height="match_parent">

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

</RelativeLayout>
```

3_实现三个圆环-中间园环

```
<RelativeLayoutxmlns:android="http://schemas.android.com/apk/res/androi
d"
android:layout_width="match_parent"
android:layout_height="match_parent">

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

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

</RelativeLayout>
```

4_实现三个圆环-最外环

```
<RelativeLayoutxmlns:android="http://schemas.android.com/apk/res/androi
d"
android:layout_width="match_parent"
android:layout_height="match_parent">

<RelativeLayout
 android:id="@+id/level3"
 android:layout_width="280dip"
        android:layout_height="140dip"
        android:layout_alignParentBottom="true"
        android:layout_centerHorizontal="true"
android:background="@drawable/level3">
</RelativeLayout>

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

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

</RelativeLayout>
```

5_最里环的的图标

```
<RelativeLayoutxmlns:android="http://schemas.android.com/apk/res/androi
```

```
d"
android:layout_width="match_parent"
android:layout_height="match_parent">

<RelativeLayout
android:id="@+id/level3"
android:layout_width="280dip"
android:layout_height="140dip"
android:layout_alignParentBottom="true"
android:layout_centerHorizontal="true"
android:background="@drawable/level3">
</RelativeLayout>

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

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

<ImageView
  android:layout_width="wrap_content"
 android:layout_height="wrap_content"
 android:layout_centerInParent="true"
 android:src="@drawable/icon_home" />
</RelativeLayout>

</RelativeLayout>
```

6_中间环的图标

```
<RelativeLayoutxmlns:android="http://schemas.android.com/apk/res/a
ndroid"
android:layout_width="match_parent"
android:layout_height="match_parent">

<RelativeLayout
android:id="@+id/level3"
android:layout_width="280dip"
android:layout_height="140dip"
android:layout_alignParentBottom="true"
android:layout_centerHorizontal="true"
android:background="@drawable/level3">
</RelativeLayout>

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

<ImageView
          android:layout_width="wrap_content"
          android:layout_height="wrap_content"
          android:layout_alignParentBottom="true"
          android:layout_margin="10dip"
          android:src="@drawable/icon_search" />

<ImageView
          android:layout_width="wrap_content"
          android:layout_height="wrap_content"
          android:layout_centerHorizontal="true"
          android:layout_marginTop="5dip"
          android:src="@drawable/icon_menu" />

<ImageView
          android:layout_width="wrap_content"
          android:layout_height="wrap_content"
          android:layout_alignParentBottom="true"
          android:layout_alignParentRight="true"
```

```
            android:layout_margin="10dip"
            android:src="@drawable/icon_myyouku" />
</RelativeLayout>

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

<ImageView
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:layout_centerInParent="true"
android:src="@drawable/icon_home"/>
</RelativeLayout>

</RelativeLayout>
```

7_最外环的图标的左边部分

```
<RelativeLayoutxmlns:android="http://schemas.android.com/apk/res/androi
d"
android:layout_width="match_parent"
android:layout_height="match_parent">

<RelativeLayout
android:id="@+id/level3"
android:layout_width="280dip"
android:layout_height="140dip"
android:layout_alignParentBottom="true"
android:layout_centerHorizontal="true"
android:background="@drawable/level3">

<ImageView
            android:id="@+id/channel1"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_alignParentBottom="true"
            android:layout_marginBottom="10dip"
```

```
            android:layout_marginLeft="10dip"
            android:src="@drawable/channel1" />

<ImageView
            android:id="@+id/channel2"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
 android:layout_above="@id/channel1"
 android:layout_alignLeft="@id/channel1"
            android:layout_marginLeft="20dip"
            android:layout_marginBottom="10dip"
            android:src="@drawable/channel2" />

<ImageView
            android:id="@+id/channel3"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
 android:layout_above="@id/channel2"
            android:layout_alignLeft="@id/channel2"
            android:layout_marginBottom="8dp"
            android:layout_marginLeft="35dp"
            android:src="@drawable/channel3" />

<ImageView
            android:layout_marginTop="10dip"
            android:id="@+id/channel4"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
 android:layout_centerHorizontal="true"
            android:src="@drawable/channel4" />
</RelativeLayout>

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

..............
</RelativeLayout>
```

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

..............

</RelativeLayout>

</RelativeLayout>
```

8_最外环的图标的右边部分

```
<RelativeLayoutxmlns:android="http://schemas.android.com/apk/res/androi
d"
android:layout_width="match_parent"
android:layout_height="match_parent">

<RelativeLayout
android:id="@+id/level3"
android:layout_width="280dip"
android:layout_height="140dip"
android:layout_alignParentBottom="true"
android:layout_centerHorizontal="true"
android:background="@drawable/level3">

<ImageView
android:id="@+id/channel1"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:layout_alignParentBottom="true"
android:layout_marginBottom="10dip"
android:layout_marginLeft="10dip"
android:src="@drawable/channel1"/>

<ImageView
```

```
android:id="@+id/channel2"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:layout_above="@id/channel1"
android:layout_alignLeft="@id/channel1"
android:layout_marginBottom="10dip"
android:layout_marginLeft="20dip"
android:src="@drawable/channel2"/>

<ImageView
android:id="@+id/channel3"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:layout_above="@id/channel2"
android:layout_alignLeft="@id/channel2"
android:layout_marginBottom="8dp"
android:layout_marginLeft="35dp"
android:src="@drawable/channel3"/>

<ImageView
android:id="@+id/channel4"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:layout_centerHorizontal="true"
android:layout_marginTop="10dip"
android:src="@drawable/channel4"/>

<ImageView
            android:id="@+id/channel7"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
 android:layout_alignParentBottom="true"
            android:layout_alignParentRight="true"
            android:layout_marginBottom="10dip"
            android:layout_marginRight="10dip"
            android:src="@drawable/channel7" />

<ImageView
            android:id="@+id/channel6"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
android:layout_above="@id/channel7"
```

```
            android:layout_alignRight="@id/channel7"
 android:layout_marginBottom="10dip"
            android:layout_marginRight="20dip"
            android:src="@drawable/channel6" />

<ImageView
            android:id="@+id/channel5"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_above="@id/channel6"
            android:layout_alignRight="@id/channel6"
 android:layout_marginBottom="10dip"
            android:layout_marginRight="35dip"
            android:src="@drawable/channel7" />
</RelativeLayout>



.................

.................

</RelativeLayout>
```

02_优酷代码实现-48

1_初始化三环的控件，并设置 icon_menu 和 icon_menu 的点击事件

```
publicclass MainActivity extends Activity implements OnClickListener {

    private RelativeLayout level1;
    private RelativeLayout level2;
    private RelativeLayout level3;

    private ImageView icon_home;
    private ImageView icon_menu;

    @Override
    protectedvoid onCreate(Bundle savedInstanceState) {
```

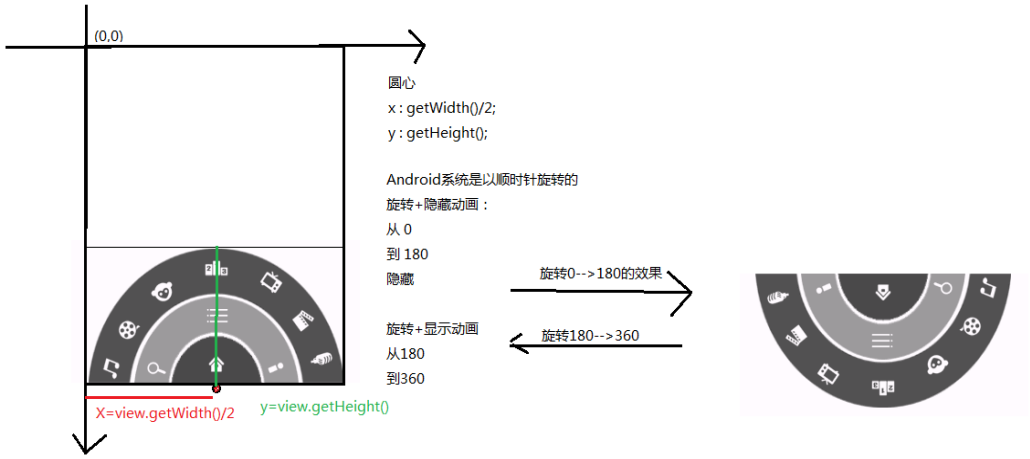
```
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        level1 = (RelativeLayout) findViewById(R.id.level1);
        level2 = (RelativeLayout) findViewById(R.id.level2);
        level3 = (RelativeLayout) findViewById(R.id.level3);
        icon_home = (ImageView) findViewById(R.id.icon_home);
        icon_menu = (ImageView) findViewById(R.id.icon_menu);

        icon_home.setOnClickListener(this);
        icon_menu.setOnClickListener(this);
    }
    @Override
    publicvoid onClick(View v) {
        switch (v.getId()) {
        case R.id.icon_home://响应home的点击事件

            break;

        case R.id.icon_menu://响应menu的点击事件
            break;
        }

    }
```

2_三级菜单的显示和隐藏

```
private boolean isLevel3Show = true;

@Override
    publicvoid onClick(View v) {
        switch (v.getId()) {
        case R.id.icon_home:// 相应home的点击事件

            break;

        case R.id.icon_menu:// 相应menu的点击事件
            if (isLevel3Show) {
                Tools.hideView(level3);
                isLevel3Show = false;
            } else {
                Tools.showView(level3);
```

```
                isLevel3Show = true;
            }
            break;
        }
    }
```

旋转原理画图分析



旋转工具类代码：

```
/**
 * @authorafu
 *
 */
publicclass Tools {

    publicstaticvoid hideView(View view) {
        /**
         * fromDegrees 从多少度开始
         * toDegrees 旋转到度
         * pivotX 中心点x坐标
         * pivotY 中心点y坐标
         */
        RotateAnimation ra = new RotateAnimation(0, 180, view.getWidth()/2,
view.getHeight());
        //播放时常
        ra.setDuration(500);
```

```
        //停留在播放完成状态
        ra.setFillAfter(true);
        view.startAnimation(ra);
    }

    publicstaticvoid showView(View view) {
        RotateAnimation ra = new RotateAnimation(180, 360, view.getWidth()
/ 2,
                view.getHeight());
        ra.setDuration(500);
        ra.setFillAfter(true);
        view.startAnimation(ra);
    }

}
```

3_二级菜单的显示和隐藏

```
@Override
    publicvoid onClick(View v) {
        switch (v.getId()) {
        case R.id.icon_home:// 相应home的点击事件
            if (isLevel2Show) {
                //如果二级菜单式显示的，隐藏二级菜单
                Tools.hideView(level2);
                //判断三级菜单的状态，如果是显示，同时也隐藏三级菜单
                if(isLevel3Show){
                    Tools.hideView(level3);
                }
                isLevel2Show = false;
            } else {
                //如果二级才能使隐藏的，那么显示二级菜单
                Tools.showView(level2);
                isLevel2Show = true;
            }

            break;

        case R.id.icon_menu:// 相应menu的点击事件
            ..................
            break;
```

```
        }
    }
```

4_设置延迟动画 setStartOffset()方法和代码重构

```
/**
 * @authorafu
 *
 */
publicclass Tools {

    publicstaticvoid hideView(View view) {
        hideView(view, 0);
    }

    publicstaticvoid showView(View view) {
        showView(view, 0);
    }

    /**
     * 延迟显示
     *
     * @param view
     * @param i
     */
    publicstaticvoid showView(View view, int i) {
        RotateAnimation ra = newRotateAnimation(180, 360, view.getWidth()
/ 2,
                view.getHeight());
        ra.setDuration(500);
        ra.setFillAfter(true);
        ra.setStartOffset(i);
        view.startAnimation(ra);
    }

    /**
     * 延迟隐藏
     *
     * @param view
     * @param i
     *            延迟隐藏的时间
```

```
     */
    publicstaticvoid hideView(View view, int i) {
        /**
         * fromDegrees 从多少度开 toDegrees 旋转到度 pivotX x坐标 pivotY y
坐标
         */
        RotateAnimation ra = newRotateAnimation(0, 180, view.getWidth() /
2,
                view.getHeight());
        // 播放时常
        ra.setDuration(500);
        // 停留在播放完成状态
        ra.setFillAfter(true);
        ra.setStartOffset(i);
        view.startAnimation(ra);

    }

}
```

5._监听手机 menu 按键实现菜单隐藏和显示

```
@Override
    publicboolean onKeyDown(int keyCode, KeyEvent event) {
        if (keyCode == KeyEvent.KEYCODE_MENU) {

            if (isLevel1Show) {
                // 如果一级菜单式显示的，那么隐藏 一级菜单
                Tools.hideView(level1);
                isLevel1Show = false;
                // 同时判断 隐藏二级、三级菜单
                if (isLevel2Show) {
                    Tools.hideView(level2, 200);
                    isLevel2Show = false;
                    if (isLevel3Show) {
                        Tools.hideView(level3, 300);
                        isLevel3Show = false;
                    }
                }
            } else {
                // 如果一级菜单式隐藏的，那么就要显示一级菜单
                Tools.showView(level1);
```

```
                isLevel1Show = true;
                // 同时要显示二级菜单
                Tools.showView(level2,200);
                isLevel2Show = true;

            }

            return true;
        }
        return super.onKeyDown(keyCode, event);
    }
```

03_优酷效果的完成和 bug 修复-9

1_用 View 和 ViewGroup 的区别解决 bug

```
/**
 * @authorafu
 *
 */
publicclass Tools {

    publicstaticvoid hideView(ViewGroup view) {
        hideView(view, 0);
    }

    publicstaticvoid showView(ViewGroup view) {
        showView(view, 0);
    }

    /**
     * 延迟显示
     *
     * @param view
     * @param i
     */
    publicstaticvoid showView(ViewGroup view, intstartOffset) {
        RotateAnimation ra = new RotateAnimation(180, 360, view.getWidth()
/ 2,
                view.getHeight());
        ra.setDuration(500);
```

```
        ra.setFillAfter(true);
        ra.setStartOffset(startOffset);
        view.startAnimation(ra);

        // view.setVisibility(View.VISIBLE);
        // view.setEnabled(true);
        for (int i = 0; i < view.getChildCount(); i++) {

            view.getChildAt(i).setEnabled(true);
        }
    }

    /**
     * 延迟隐藏
     *
     * @paramview
     * @param i
     *            延迟隐藏的时间
     */
    public static void hideView(ViewGroup view, intstartOffset) {
        /**
         * fromDegrees 从多少度开 toDegrees 旋转到度 pivotX x坐标 pivotY y
坐标
         */
    RotateAnimation ra = new RotateAnimation(0, 180, view.getWidth() / 2,
            view.getHeight());
        // 播放时常
        ra.setDuration(500);
        // 停留在播放完成状态
        ra.setFillAfter(true);
        ra.setStartOffset(startOffset);
        view.startAnimation(ra);
        // view.setVisibility(View.GONE);
        // view.setEnabled(false);
        for (int i = 0; i <view.getChildCount(); i++) {

            view.getChildAt(i).setEnabled(false);
        }

    }

}
```

2_用属性动画解决 bug

```
 public static void hideView(ViewGroup view, int startOffset) {
//      view.setRotation();
        ObjectAnimator animator =
ObjectAnimator.ofFloat(view,"rotation",0,180);
        animator.setDuration(500);
        animator.setStartDelay(startOffset);//设置延迟播放
        animator.start();//开始播放动画

        //单独设置中心点击
        view.setPivotX(view.getWidth()/2);
        view.setPivotY(view.getHeight());

    }

    public static void showView(ViewGroup view, int startOffset) {

        ObjectAnimator animator =
ObjectAnimator.ofFloat(view,"rotation",180,360);
        animator.setDuration(500);
        animator.setStartDelay(startOffset);//设置延迟播放
        animator.start();

        //单独设置中心点击
        view.setPivotX(view.getWidth() / 2);
        view.setPivotY(view.getHeight());
    }
```

3_广告条和首页推荐

1_广告条 ViewPage 的介绍-40

1_创建工程名：首页影片推广效果，包名为：com.atguigu.viewpager,并且拷贝图片到 drawable-hdpi 目录

2_写布局文件

```
<RelativeLayout
xmlns:android="http://schemas.android.com/apk/res/android"
android:layout_width="match_parent"
android:layout_height="match_parent">

<android.support.v4.view.ViewPager
android:id="@+id/viewpager"
 android:layout_width="match_parent"
android:layout_height="200dip"/>

<LinearLayout
```

```
android:layout_width="match_parent"
android:layout_height="wrap_content"
 android:layout_alignBottom="@id/viewpager"
android:background="#33000000"
android:gravity="center_horizontal"
android:orientation="vertical"
android:padding="5dip">

<TextView
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:text="三个火枪手"
android:textColor="#ffffff"
android:textSize="18sp"/>

<LinearLayout
android:id="@+id/ll_point_group"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
 android:layout_margin="5dip"
android:orientation="horizontal">
</LinearLayout>
</LinearLayout>

</RelativeLayout>
```

3_实例化 ViewPager 和关联其源代码

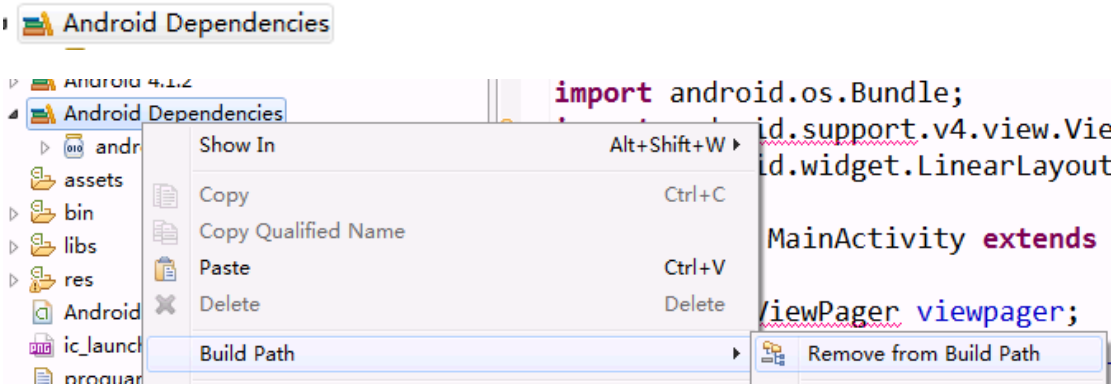
代码实例化：

```
publicclass MainActivity extends Activity {

    private ViewPager viewpager;
    private LinearLayout ll_point_group;

    @Override
    protectedvoid onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        viewpager = (ViewPager) findViewById(R.id.viewpager);
        ll_point_group = (LinearLayout) findViewById(R.id.ll_point_group);
    }
```

}

关联源代码

1.删除工程里面的 Android Depandencies，删除后会报错，不要理会。看下面



2.添加 libs 目录下的 Android-support-v4.jar 包

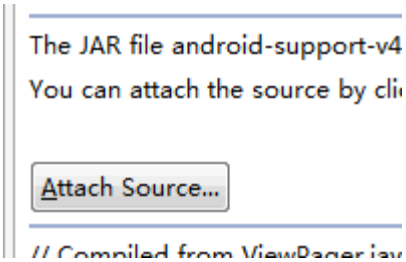
选中-->右键-->build path-->add to build path

3.关联源代码

目 录 ：

C:\android\adt-bundle-windows-x86_64-20130219\sdk\extras\android\support\v4\src\java

点击 ViewPager 类，出现图标；



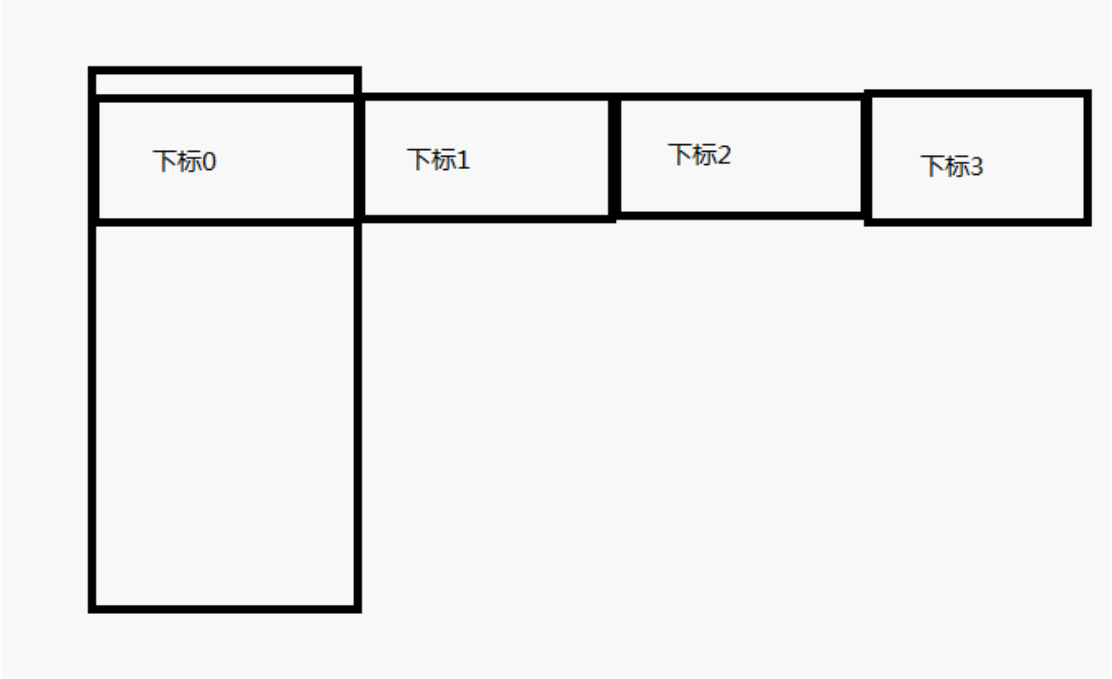
大家对于 v4 包都已经很熟悉了，现在在新建 android 项目时，v4 包是默认导入的。v7 包出来没多长时间，用的人也不多，主要对 3.0 以下版本提供 ActionBar 支持，以及 SearchView，PopupMenu 等控件的支持。因为一些开源框架已经实现对 3.0 以下版本 ActionBar 的支 持，所以 v7 包的使用意义也不是很大。

知识拓展：

如果 jar 包导入错误，怎么修改呢？

右键工程---->properties---->Java Build Path --->Libraries--> 选择 android-support-v4.jar 展开---->Editor--->External Folder

4.ViewPager 的原理



能显示很多页面，者些页面可以是图片也可以是布局文件。

4_设置图片资源 ID 和图片标题集合和准备 ImageView 列表数据

```
// 图片资源ID
    privatefinalint[] imageIds = {
            R.drawable.a,
            R.drawable.b,
            R.drawable.c,
            R.drawable.d,
            R.drawable.e };

    // 图片标题集合
    privatefinal String[] imageDescriptions = {
            "巩俐不低俗，我就不能低俗",
            "扑树又回来啦！再唱经典老歌引万人大合唱",
            "揭秘北京电影如何升级",
            "乐视网TV版大派送",
            "热血屌丝的反杀" };
```

```
//准备数据
        imageList = new ArrayList<ImageView>();
        for(int i=0;i<imageIds.length;i++){
            ImageView imageView = new ImageView(this);
            imageView.setBackgroundResource(imageIds[i]);
    imageList.add(imageView);
        }
```

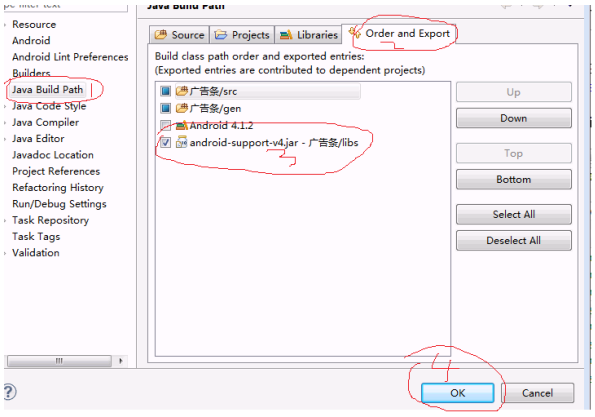
5_为 ViewPager 设置适配器

```
privateclass MyPagerAdapter extends PagerAdapter {

        @Override
        publicint getCount() {
            // 页面或者图片的总数
            returnimageList.size();
        }

        /**
         * 功能：给ViewPager添加指定的view
         * container 就是ViewPager，其实就是容器。
         * position 具体页面或者图片的位置
         */
        @Override
        public Object instantiateItem(ViewGroup container, int position) {
            System.out.println("instantiateItem=="+position);
            View view = imageList.get(position);
            container.addView(view);
            //返回的值，不一定是View ,也可以是和View有关系的任意的Object
//          return super.instantiateItem(container, position);
            return view;
        }

        /**
         * 判断某个page和object的关系
         * object 是 instantiateItem的返回值
         */
        @Override
        publicboolean isViewFromObject(View view, Object object) {
//          if(view ==object){
```

```
//              return true;
//          }else{
//              return false;
//          }
            return view ==object;

        }

        /**
         * 销毁指定位置上的View或者object
         */
        @Override
        publicvoid destroyItem(ViewGroup container, int position, Object
object) {
            System.out.println("destroyItem=="+position);
            container.removeView((View) object);
//          super.destroyItem(container, position, object);
        }

    }
```

6_解决运行报错

选中项目--->右键--->Java Build Path --->
order export--->勾选 android-support-v4.jar--->千万不要忘了 clean



2_广告条基本功能-34

1_根据不同图片显示不同描述信息

```
viewpager.setOnPageChangeListener(new OnPageChangeListener() {

            /**
             * 当页面被选择了回调
             * position 当前被显示的页面的位置：从0开始
             */
            @Override
            publicvoid onPageSelected(int position) {

    tv_image_desc.setText(imageDescriptions[position]);
        }
            /**
             * 当页面滑动了调用该方法
             */
            @Override
            publicvoid onPageScrolled(int position, float positionOffset,
                    int positionOffsetPixels) {
            }
            /**
             * 当页面状态发送变化的调用方法
             * 静止--滑动
             * 滑动-静止
             *
             */
            @Override
            publicvoid onPageScrollStateChanged(int state) {
            }
        });
```

2.用 shape 资源定义点和背景

创建 drawable 目录里面创建文件

point_normal.xml

```
<?xmlversion="1.0"encoding="utf-8"?>
<shape xmlns:android="http://schemas.android.com/apk/res/android"
    android:shape="oval">
<size android:height="5dip" android:width="5dip"/>
<solid android:color="#55000000"/>
</shape>
```

point_focused.xml

```
<?xmlversion="1.0"encoding="utf-8"?>
<shapexmlns:android="http://schemas.android.com/apk/res/android"
android:shape="oval">
<size android:height="5dip"android:width="5dip"/>
<solid android:color="#aaffffff"/>
</shape>
```

point_selsetor.xml

```
<?xmlversion="1.0"encoding="utf-8"?>
<selector xmlns:android="http://schemas.android.com/apk/res/android">
<item
  android:state_enabled="true"
android:drawable="@drawable/point_focused"/>
    <item
android:state_enabled="false"android:drawable="@drawable/point_normal"/
>
</selector>
```

3.代码里面添加指示点

```
for(int i=0;i<imageIds.length;i++){
            ImageView imageView = new ImageView(this);
            imageView.setBackgroundResource(imageIds[i]);
            imageViews.add(imageView);

            //添加指示点
            ImageView point = new ImageView(this);
            point.setBackgroundResource(R.drawable.point_selsetor);
            ll_point_group.addView(point);

            //默认情况下，第一个小点enable为true
            if(i ==0){
                point.setEnabled(true);
            }else{
                point.setEnabled(false);
            }

}
```

4_设置改变指示点的状态

如红色部分

```
/**
     * 上次的位置
     */
privateint lastPointIndex;

viewpager.setOnPageChangeListener(new OnPageChangeListener() {

            /**
             * 当页面被选择了回调
             * position 当前被显示的页面的位置：从0开始
             */
            @Override
            publicvoid onPageSelected(int position) {
                System.out.println("onPageSelected="+position);
                tv_image_desc.setText(imageDescriptions[position]);

                //设置指示点的状态 enable 的状态为true或者为false；
ll_point_group.getChildAt(lastPointIndex).setEnabled(false);

                ll_point_group.getChildAt(position).setEnabled(true);

                                lastPointIndex = position;
            }
            /**
             * 当页面滑动了调用该方法
             */
            @Override
            publicvoid onPageScrolled(int position, float positionOffset,
                    int positionOffsetPixels) {

            }
            /**
             * 但页面状态发送变化的调用防方法
             * 静止--滑动
             * 滑动-静止
             *
             */
            @Override
```

```
            publicvoid onPageScrollStateChanged(int state) {

    System.out.println("onPageScrollStateChanged===state=="+state);
            }
        });
```

5.设置指示点的间距

如红色部分

```
for(int i=0;i<imageIds.length;i++){
            ImageView imageView = new ImageView(this);
            imageView.setBackgroundResource(imageIds[i]);
            imageViews.add(imageView);

            //添加指示点
            ImageView point = new ImageView(this);

            LayoutParams              params              =              new
LayoutParams(LayoutParams.WRAP_CONTENT, -2);
            params.leftMargin = 15;
            point.setLayoutParams(params);

            point.setBackgroundResource(R.drawable.point_selsetor);
            ll_point_group.addView(point);

            //默认情况下，第一个小点enable为true
            if(i ==0){
                point.setEnabled(true);
            }else{
                point.setEnabled(false);
            }
        }
```

注意导入包的时候，当前控件放入什么布局就导入谁的 LayoutParams 的。

6_设置可以循环滑动

```
viewpager.setOnPageChangeListener(new OnPageChangeListener() {

            /**
             * 当页面被选择了回调
             * position 当前被显示的页面的位置：从0开始
```

```
             */
            @Override
            publicvoid onPageSelected(int position) {
                int  myIndex = position % imageViews.size();
                System.out.println("onPageSelected="+position);
                tv_image_desc.setText(imageDescriptions[myIndex]);

                //设置指示点的状态 enable 的状态为true或者为false；
                ll_point_group.getChildAt(myIndex).setEnabled(true);


    ll_point_group.getChildAt(lastPointIndex).setEnabled(false);
                lastPointIndex = myIndex;
            }
            ..............................
        });
    }

    privateclass MyPagerAdapter extends PagerAdapter {

        @Override
        publicint getCount() {
            //得到数据的总数
//          return imageViews.size();
            return Integer.MAX_VALUE;
        }

        /**
         * 给ViewPager添加指定的View
         * container 是ViewPage,他是一个容器
         * position 要实例化的view的位置
         */
        @Override
        public Object instantiateItem(ViewGroup container, int position) {
//          System.out.println("instantiateItem=="+position);
            //实例化View
            View view = imageViews.get(position%imageViews.size());
            container.addView(view);
            //返回值，不一定要是View对象，也可以是和View有关系的任意object
//          return super.instantiateItem(container, position);
            return view;
        }
```

```
        ......................

    }
```

7_解决左滑没有效果问题

```
//要求刚好是 imageViews.size()的整数倍
int item = Integer.MAX_VALUE/2-Integer.MAX_VALUE/2%imageViews.size();
//让ViewPager跳转到指定的位置，应该保证是imageView.size()的整数倍
viewpager.setCurrentItem(item );
//11 和 101
```

3_广告条自动翻页-10

实现方式有多种方案：

1.定时器 timer + Handler

2.2,while true 循环 sleep + Handler;

3,ClockManger + Handler ;

4,Handler

我们采用常用的方式 Handler

```
/**
     * 是否自定滑动运行中
     */
    privatebooleanisRunning = false;

    private Handler handler = new Handler(){
        publicvoid handleMessage(android.os.Message msg) {
            viewpager.setCurrentItem(viewpager.getCurrentItem()+1);
            if(isRunning){
                handler.sendEmptyMessageDelayed(0, 4000);
            }

        };
    };
```

在 onCreate 中写上

```
isRunning = true;
handler.sendEmptyMessageDelayed(0, 2000);
```

4_下来框-44

下拉框效果：
在 editText 的右边放置一个小箭头的图片，点击图片，在 editText 的下方弹出一个 popupWindow，并对 popupWindow 进行一些设置即得到想要的效果。

1_新建一个工程：下拉框，把需要的图片拷贝到工程中，包名：com.atguigu.popupwindow

2_写布局文件代码如下

```
<RelativeLayoutxmlns:android="http://schemas.android.com/apk/res/android"
android:layout_width="match_parent"
android:layout_height="match_parent">

<EditText
android:id="@+id/et_input"
android:paddingRight="40dip"
android:layout_marginTop="20dip"
android:layout_centerHorizontal="true"
android:layout_width="wrap_content"
```

```
android:layout_height="wrap_content"
android:text="@string/hello_world"/>
<ImageView
 android:id="@+id/dowan_arrow"
 android:layout_alignRight="@id/et_input"
 android:layout_alignTop="@id/et_input"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:layout_marginTop="5dip"
android:layout_marginRight="5dip"
android:background="@drawable/down_arrow"/>

</RelativeLayout>
```

3_实例化控件并准备数据

```
publicclass MainActivity extends Activity {
    private EditText et_input;
    private ImageView downArrow;

    /**
     * 装数据的集合
     */
    private ArrayList<String>msgList;
    @Override
    protectedvoid onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        et_input = (EditText) findViewById(R.id.et_input);
        downArrow = (ImageView) findViewById(R.id.dowan_arrow);
        //准备数据
        msgList = new ArrayList<String>();
        for(int i=0;i<30;i++){
            msgList.add("aaaaaaaaaa"+i);
        }
    }
}
```

4_设置向下箭头点击事件并实例化 popupwindow&TODO

简介

```
downArrow.setOnClickListener(this);

//浮悬的窗体
    private PopupWindow  popupWindow;
    @Override
    publicvoid onClick(View v) {
        switch (v.getId()) {
        case R.id.dowan_arrow:
            if(popupWindow == null){
                popupWindow = new PopupWindow(this);
                //设置高和宽
                popupWindow.setWidth(et_input.getWidth());
                popupWindow.setHeight(200);
                //设置窗体的内容
                //TODO ListView 还没有初始化
                popupWindow.setContentView(listView);

            }
            popupWindow.showAsDropDown(et_input, 0, 0);

            break;

        default:
            break;
        }
```

5_实例化 ListView 并且设置适配器

在 onCreate 方法中实例化 ListView

```
//实例化ListView
listView = new ListView(this);
listView.setAdapter(new MyAdapter());
```

自定义适配器

```
class MyAdapter extends BaseAdapter{
```

```
        @Override
        publicint getCount() {
            returnmsgList.size();
        }
        @Override
        public View getView(finalint position, View convertView, ViewGroup
parent) {
            View view;
            ViewHolder holder;
            if(convertView != null){
                view = convertView;
                holder = (ViewHolder) view.getTag();
            }else{
                view       =              View.inflate(MainActivity.this,
R.layout.list_popupwindow_item, null);
                holder = new ViewHolder();
                holder.iv_user                =                (ImageView)
view.findViewById(R.id.iv_user);
                holder.tv_tilte                =                (TextView)
view.findViewById(R.id.tv_tilte);
                holder.iv_delete                =                (ImageView)
view.findViewById(R.id.iv_delete);
                view.setTag(holder);
            }
            holder.tv_tilte.setText(msgList.get(position));
            holder.iv_delete.setOnClickListener(new OnClickListener() {

                @Override
                publicvoid onClick(View v) {
                    //1.把点击的条在列表中移除
                    msgList.remove(position);
                    //2.更新数据
                    notifyDataSetChanged();
                }
            });
            return view;
        }
        @Override
        public Object getItem(int position) {
            returnnull;
        }
```

```
        @Override
        publiclong getItemId(int position) {
            // TODO Auto-generated method stub
            return 0;
        }

    }

    class ViewHolder{
        ImageView iv_user;
        TextView tv_tilte;
        ImageView iv_delete;
    }
```

每条布局文件代码 **list_popupwindow_item.xml**

```
<?xmlversion="1.0"encoding="utf-8"?>
<RelativeLayout
xmlns:android="http://schemas.android.com/apk/res/android"
android:layout_width="fill_parent"
android:layout_height="55dip"
android:gravity="center_vertical"
android:padding="15dip">

<ImageView
android:id="@+id/iv_user"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:background="@drawable/user"
 android:padding="5dp"/>

<TextView
 android:id="@+id/tv_tilte"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
 android:layout_centerHorizontal="true"
android:text="aaaaaaaaa1"/>

<ImageView
android:id="@+id/iv_delete"
android:layout_width="wrap_content"
```

```
android:layout_height="wrap_content"
 android:layout_alignParentRight="true"
android:background="@drawable/delete"
android:padding="5dp"/>

</RelativeLayout>
```

演示运行看看效果

6_ListView 在低版本 2.3 的适配并且解决各个问题

设置输入框的宽为 200dip

```
<EditText
android:id="@+id/et_input"
android:paddingRight="40dip"
android:layout_marginTop="20dip"
android:layout_centerHorizontal="true"
android:layout_width="200dip"
android:layout_height="wrap_content"
android:text="@string/hello_world"/>
```

解决按下变白的问题：

```
listView = new ListView(this);
listView.setBackgroundResource(R.drawable.listview_background);
listView.setAdapter(new MyAdapter());
```

解决点击 popupwindow 外部，无法消掉问题

```
popupWindow.setOutsideTouchable(true);
```

设置选择某一条，并且显示在输入框中

```
listView.setOnItemClickListener(new OnItemClickListener() {

            @Override
            publicvoid onItemClick(AdapterView<?> parent, View view,
                    int position, long id) {
                et_input.setText(msgList.get(position));
```

```
            }
        });
```

注意需要设置 popupwindow 的焦点才起作用

```
popupWindow.setFocusable(true);
```

在 setOnItemClickListener 方法中消掉对话框

```
popupWindow.dismiss();
```

```
@Override
    publicvoid onClick(View v) {
        switch (v.getId()) {
        case R.id.iv_down_arrow://点击向下箭头
            if(window == null){
                window = new PopupWindow(this);

//              window.setBackgroundDrawable(new
ColorDrawable(color.transparent));
                window.setWidth(et_input.getWidth());
                window.setHeight(200);

                //TODO 设置 popupWindow 的内容
                window.setContentView(contentView);


//              window.setOutsideTouchable(true);
                //不一定要背景，主要是 setFocusable 要先执行，showAsDropDown
后执行
                window.setFocusable(true);

            }

            window.showAsDropDown(et_input, 0, 0);



            break;

        default:
            break;
        }
```

```
}
```

5_自定义开关按钮

1_自定义点击开关按钮-37

继承已有 View 实现自定义 View

通过对 android 原生控件的研究，可以发现 android 中的控件都是继承 view 类，如 textView、ImageView 等,通过重写相关的方法来实现新的效果，通过这个我们得到两点:
我们可以在已有控件的基础上，通过重写相关方法来实现我们的需求。
继承 view 类或 viewgroup 类,来创建我们所需要的控件。一般来讲,通过继承已有的控件,来自定义控件要简单一点。

1_创建工程：开关按钮，包名：com.atguigu.togglebutton，并把图片拷贝到工程中

2_自定义类 MyToggleButton 继承自 View

实现三个构造方法

```
/**
 * 自定按钮
 * @authorafu
 */
publicclass MyToggleButton extends View {

    // 增加一个默认显示样式时候使用
   public MyToggleButton(Context context, AttributeSet attrs, int defStyle)
{
        super(context, attrs, defStyle);
    }

    // 在布局文件中声明view的时候，该方法有系统调用
    public MyToggleButton(Context context, AttributeSet attrs) {
        super(context, attrs);
    }

    // 在代码中new实例化时调用
    public MyToggleButton(Context context) {
        super(context);
    }

}
```

在布局文件中使用

```
<RelativeLayoutxmlns:android="http://schemas.android.com/apk/res/androi
d"
android:layout_width="match_parent"
android:layout_height="match_parent">

    <com.atguigu.togglebutton.MyToggleButton
     android:layout_centerHorizontal="true"
     android:layout_width="wrap_content"
     android:layout_height="wrap_content" />
```

```
</RelativeLayout>
```

3_一个 View 从创建到显示屏幕的步骤

从执行view构造方法，创建对象

1.测量view大小measure()

onMeasure(int,int);来完成测量动作

2.指定view的位置:onLayout(),子View只有建议权，父View才有决定权；

onLayout(boolean,int,int,int ,int);

这个方法一般用不着，如果自定义继承ViewGoup才用到

3.绘制view的内容draw()

onDraw(canvas);

自定义 View 的时候一般重新 onMeasure(int,int)和 onDraw(canvas);

4_画个矩形背景和圆形

```
packagecom.atguigu.togglebutton;

import android.content.Context;
import android.graphics.Canvas;
import android.graphics.Color;
import android.graphics.Paint;
import android.util.AttributeSet;
import android.view.View;

/**
 * 自定按钮
 * @authorafu
 */
publicclass MyToggleButton extends View {

    /**
     * 一个View从创建到显示屏幕上的主要步骤:
     * 1.执行view构造方法，创建对象
     * 2.测量view大小
     *  onMeasure(int,int);来完成测量动作
     * 3.指定view的位置,子View只有建议权，父View才有决定权;
     * onLayout(boolean,int,int,int ,int);
```

```
     * 这个方法一般用不着，如果自定义ViewGoup才用到
     * 4.绘制view的内容
     * onDraw(canvas);
     *
     */

    private Paint paint;

    /**
     * 测量
     */
    @Override
protected void onMeasure(int widthMeasureSpec, int heightMeasureSpec) {
//      super.onMeasure(widthMeasureSpec, heightMeasureSpec);
        //设置当前view的测量大小
        setMeasuredDimension(100, 100);
    }
    /**
     * 绘制
     */
    @Override
    protectedvoid onDraw(Canvas canvas) {
//      super.onDraw(canvas);
        //绘制颜色，可以理解成背景颜色
        canvas.drawColor(Color.RED);
        //绘制圆形
        canvas.drawCircle(50, 50, 20, paint);
    }

    privatevoid init(Context context) {
        paint = new Paint();
        paint.setColor(Color.GREEN);
        //设置抗锯齿，让边缘圆滑，一般都会设置
        paint.setAntiAlias(true);

    }
    // 增加一个默认显示样式时候使用
   public MyToggleButton(Context context, AttributeSet attrs, int defStyle)
{
        super(context, attrs, defStyle);
        init(context);
    }
```

```
    // 在布局文件中声明view的时候，该方法有系统调用
    public MyToggleButton(Context context, AttributeSet attrs) {
        super(context, attrs);
        init(context);
    }

    // 在代码中new实例化时调用
    public MyToggleButton(Context context) {
        super(context);
        init(context);
    }


}
```

5_画按钮背景

```
packagecom.atguigu.togglebutton;


/**
 * 自定按钮
 * @authorafu
 */
publicclass MyToggleButton extends View {

    /**
     * 一个View从创建到显示屏幕上的主要步骤：
     * 1.执行view构造方法，创建对象
     * 2.测量view大小
     *  onMeasure(int,int);来完成测量动作
     * 3.指定view的位置,子View只有建议权，父View才有决定权；
     * onLayout(boolean,int,int,int ,int);
     * 这个方法一般用不着，如果自定义ViewGoup才用到
     * 4.绘制view的内容
     * onDraw(canvas);
     *
     */

    private Paint paint;
    private Bitmap backGroundBitmap;
```

```
    private Bitmap slideBitmap;
    private Context context;

    /**
     * 测量
     */
    @Override
    protectedvoid onMeasure(int widthMeasureSpec, int heightMeasureSpec) {
//      super.onMeasure(widthMeasureSpec, heightMeasureSpec);
        //设置当前view的测量大小
        setMeasuredDimension(backGroundBitmap.getWidth(),
backGroundBitmap.getHeight());
    }
    /**
     * 绘制
     */
    @Override
    protectedvoid onDraw(Canvas canvas) {
//      super.onDraw(canvas);
        //绘制颜色，可以理解成背景颜色
//      canvas.drawColor(Color.RED);
        //绘制圆形
//      canvas.drawCircle(50, 50, 20, paint);
        canvas.drawBitmap(backGroundBitmap, 0, 0, paint);
    }

    privatevoid init(Context context) {
        this.context = context;
        paint = new Paint();
        paint.setColor(Color.GREEN);
        //设置抗锯齿，让边缘圆滑，一般都会设置
        paint.setAntiAlias(true);

        //初始化图片-从资源文件中解析成Bitmap对象
        slideBitmap     =     BitmapFactory.decodeResource(getResources(),
R.drawable.slide_button);
        backGroundBitmap  =  BitmapFactory.decodeResource(getResources(),
R.drawable.switch_background);

    }
    // 增加一个默认显示样式时候使用
   public MyToggleButton(Context context, AttributeSet attrs, int defStyle)
```

```
{
        super(context, attrs, defStyle);
        init(context);
    }

    // 在布局文件中声明view的时候，该方法有系统调用
    public MyToggleButton(Context context, AttributeSet attrs) {
        super(context, attrs);
        init(context);
    }

    // 在代码中new实例化时调用
    public MyToggleButton(Context context) {
        super(context);
        init(context);
    }


}
```

6_画滑动按钮

```
canvas.drawBitmap(slideBitmap, 45, 0, paint);
```

分别设置 0 和 30 运行看看效果

7_点击时改变按钮状态

```
packagecom.atguigu.togglebutton;


/**
 * 自定按钮
 * @authorafu
 */
publicclass MyToggleButton extends View implements View.OnClickListener {

    /**
     * 一个View从创建到显示屏幕上的主要步骤:
     * 1.执行view构造方法，创建对象
     * 2.测量view大小
```

```
     *  onMeasure(int,int);来完成测量动作
     * 3.指定view的位置,子View只有建议权，父View才有决定权；
     * onLayout(boolean,int,int,int ,int);
     * 这个方法一般用不着，如果自定义ViewGoup才用到
     * 4.绘制view的内容
     * onDraw(canvas);
     *
     */

    private Paint paint;
    private Bitmap backGroundBitmap;
    private Bitmap slideBitmap;
    private Context context;
    /**
     * 距离左边的距离
     */
    private float slideLeft;
    /**
     * 判断当前开关状态
     * true为开
     * false为关
     */
    private boolean curStata = false;

    /**
     * 测量
     */
    @Override
    protectedvoid onMeasure(int widthMeasureSpec, int heightMeasureSpec) {
//      super.onMeasure(widthMeasureSpec, heightMeasureSpec);
        //设置当前view的测量大小
        setMeasuredDimension(backGroundBitmap.getWidth(),
backGroundBitmap.getHeight());
    }
    /**
     * 绘制
     */
    @Override
    protectedvoid onDraw(Canvas canvas) {
//      super.onDraw(canvas);
        //绘制颜色，可以理解成背景颜色
//      canvas.drawColor(Color.RED);
```

```
        //绘制圆形
//      canvas.drawCircle(50, 50, 20, paint);
        canvas.drawBitmap(backGroundBitmap, 0, 0, paint);
        //绘制滑动按钮
        canvas.drawBitmap(slideBitmap, slideLeft, 0, paint);
    }

    privatevoid init(Context context) {
        this.context = context;
        paint = new Paint();
        paint.setColor(Color.GREEN);
        //设置抗锯齿，让边缘圆滑，一般都会设置
        paint.setAntiAlias(true);

        //初始化图片-从资源文件中解析成Bitmap对象
        slideBitmap     =     BitmapFactory.decodeResource(getResources(),
R.drawable.slide_button);-
        backGroundBitmap  =  BitmapFactory.decodeResource(getResources(),
R.drawable.switch_background);

        setOnClickListener( MyToggleButton.this);

    }
    // 增加一个默认显示样式时候使用
   public MyToggleButton(Context context, AttributeSet attrs, int defStyle)
{
        super(context, attrs, defStyle);
        init(context);
    }

    // 在布局文件中声明view的时候，该方法有系统调用
    public MyToggleButton(Context context, AttributeSet attrs) {
        super(context, attrs);
        init(context);
    }

    // 在代码中new实例化时调用
    public MyToggleButton(Context context) {
        super(context);
        init(context);
    }
```

```
    @Override
    publicvoid onClick(View v) {
        curStata = !curStata;
        flushState();

    }
    /**
     * 刷新状态
     */
    private void flushState() {
        //设置距离左边的距离
        if(curStata){
            slideLeft =
backGroundBitmap.getWidth()-slideBitmap.getWidth();
        }else{
            slideLeft = 0;
        }

        /**
         * 刷新View,会导致当前View的onDraw方法执行
         */
        invalidate();
    }


}
```

2_自定义滑动开关按钮-38

1_实现滑动效果

实现思想，参照手机卫士中的拖动的原理

```
/**
     * 第一次按下的x坐标
     */
    int startX = 0;


    @Override
```

```
    publicboolean onTouchEvent(MotionEvent event) {
        super.onTouchEvent(event);
        switch (event.getAction()) {
        case MotionEvent.ACTION_DOWN://按下
            //1.记录第一次按下坐标
            startX = (int) event.getRawX();

            break;
        case MotionEvent.ACTION_MOVE://滑动
            //2.来到新的坐标
            int newX = (int) event.getRawX();
            //3.计算偏移量
            int dX = newX - startX;
            slideLeft += dX;
            //4.更新UI-onDraw方法即可--invalidate();
            invalidate();
            //5.重新记录坐标
            startX = (int) event.getRawX();
            break;
        case MotionEvent.ACTION_UP://离开

            break;

        default:
            break;
        }

        returntrue;
    }
```

2_取消点击事件，屏蔽非法滑动

```
publicclass MyToggleButton extends View implements OnClickListener {

    private Paint paint;

    /**
     * 一个View从创建到显示到屏幕过程中的步骤： 1.执行View的构造方法，实例化；通常在构造方法里面加载资源 2.测量view对象
```

```
     * onMeasure(int,int) 3.指定View的位置 - 一般的View用不到，自定义包括其他View进来这样的控才用到
     * onLayout(boolean,int,int,int,int) 4.绘制View对象 onDraw(canvas)
     *
     */

    private Bitmap backgroundBitmap;
    private Bitmap slideBitmap;

    /**
     * 滑动图片，距离左边的距离
     */
    private float slideLeft;
    /**
     * 按钮的状态 false为关闭 true为开
     */
    private boolean curState = false;

    // 测量
    @Override
    protectedvoid onMeasure(int widthMeasureSpec, int heightMeasureSpec) {
        // super.onMeasure(widthMeasureSpec, heightMeasureSpec);
        // 设置测量值
        setMeasuredDimension(backgroundBitmap.getWidth(),
                backgroundBitmap.getHeight());
    }

    // 绘制
    @Override
    protected void onDraw(Canvas canvas) {
        // super.onDraw(canvas);
        // canvas.drawColor(Color.GREEN);
        // canvas.drawCircle(50, 50, 20, paint);
        canvas.drawBitmap(backgroundBitmap, 0, 0, paint);
        canvas.drawBitmap(slideBitmap, slideLeft, 0, paint);
    }
    /**
     * 第一次按下的x坐标
     */
    intstartX = 0;
    int maxLeft;
```

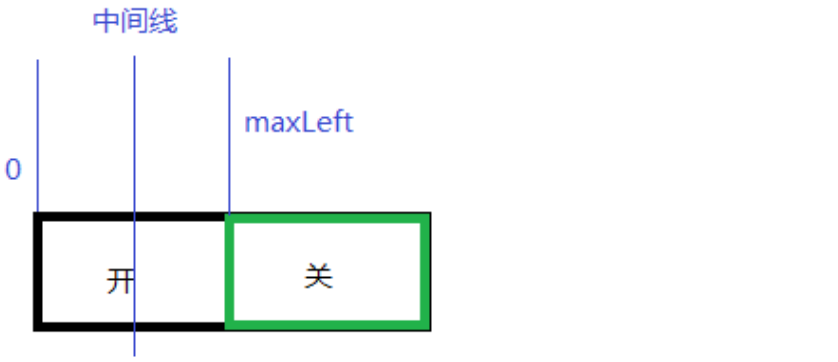
```
    @Override
    public boolean onTouchEvent(MotionEvent event) {
        super.onTouchEvent(event);
        switch (event.getAction()) {
        case MotionEvent.ACTION_DOWN://按下
            //1.记录第一次按下坐标
            startX = (int) event.getRawX();

            break;
        case MotionEvent.ACTION_MOVE://滑动
            //2.来到新的坐标
            int newX = (int) event.getRawX();
            //3.计算偏移量
            int dX = newX - startX;
            slideLeft += dX;
            //4.更新UI-onDraw方法即可--invalidate();
            flushView();
            //5.重新记录坐标
            startX = (int) event.getRawX();
            break;
        case MotionEvent.ACTION_UP://离开

            break;

        default:
            break;
        }

        returntrue;
    }

// 刷新View的状态，并且纠正非法滑动
    privatevoid flushView() {
        if(slideLeft < 0){
            slideLeft = 0;
        }

        if(slideLeft > maxLeft){
            slideLeft = maxLeft;
        }
        //屏蔽非法滑动
```

```
        invalidate();
    }

    privatevoid init(Context context) {
        paint = new Paint();
        paint.setColor(Color.RED);
        // 设置抗锯齿-使其变得光滑
        paint.setAntiAlias(true);
        // 加载资源图片
        backgroundBitmap = BitmapFactory.decodeResource(getResources(),
                R.drawable.switch_background);
        slideBitmap = BitmapFactory.decodeResource(getResources(),
                R.drawable.slide_button);

        //滑动图片距离左边的距离
        maxLeft = backgroundBitmap.getWidth()-slideBitmap.getWidth();

        // 设置点击事件
//      setOnClickListener(this);
    }

    // 一般会在代码中实例化
    public MyToggleButton(Context context) {
        super(context);
        init(context);
    }

    // 带有两个参数的构造方法，在布局文件中使用的时候，就会回调
    public MyToggleButton(Context context, AttributeSet attrs) {
        super(context, attrs);
        init(context);
    }

    // 我们需要设置默认的样式风格的时候
   public MyToggleButton(Context context, AttributeSet attrs, int defStyle)
{
        super(context, attrs, defStyle);
        init(context);
    }

    @Override
    publicvoid onClick(View v) {
```

```
        curState = !curState;

        flushState();
    }

    // 刷新View的状态
    privatevoid flushState() {
        if (curState) {

            slideLeft = maxLeft;
        } else {
            slideLeft = 0;
        }
        flushView();
    }
}
```

3_处理滑动到一小半时时不好看的问题

先画图分析



当UP事件发生的时候，由按钮的左边距离(btn_left)确定View的状态；
当btn_left >= maxLeft/2 设置为开状态
当btn_left < maxLeft/2 设置为 关闭状态

代码如下：

```
@Override
    publicboolean onTouchEvent(MotionEvent event) {
```

```
        super.onTouchEvent(event);
        switch (event.getAction()) {
        case MotionEvent.ACTION_DOWN://按下
            //1.记录第一次按下坐标
            startX = (int) event.getRawX();

            break;
        case MotionEvent.ACTION_MOVE://滑动
            //2.来到新的坐标
            int newX = (int) event.getRawX();
            //3.计算偏移量
            int dX = newX - startX;
            slideLeft += dX;
            //4.更新UI-onDraw方法即可--invalidate();
            flushView();
            //5.重新记录坐标
            startX = (int) event.getRawX();
            break;
        case MotionEvent.ACTION_UP://离开
            /**
             *  当UP事件发生的时候，由按钮的左边距离(btn_left)确定View的状态；
                当btn_left >= maxLeft/2 设置为开状态
                当btn_left < maxLeft/2  设置为 关闭状态
             */

            if(slideLeft >= maxLeft/2){
                curState = true;
            }else{
                curState = false;
            }
            flushState();
            break;

        default:
            break;
        }

        returntrue;
    }
```

恢复点击事件

演示会有 bug

3_ 解决点击事件和滑动事件导致的 bug

```
publicclass MyToggleButton extends View implements OnClickListener {

    private Paint paint;

    /**
     * 一个View从创建到显示到屏幕过程中的步骤： 1.执行View的构造方法，实例化；通常在构造方法里面加载资源 2.测量view对象
     * onMeasure(int,int) 3.指定View的位置 - 一般的View用不到，自定义包括其他View进来这样的控才用到
     * onLayout(boolean,int,int,int,int) 4.绘制View对象 onDraw(canvas)
     *
     */

    private Bitmap backgroundBitmap;
    private Bitmap slideBitmap;

    /**
     * 滑动图片，距离左边的距离
     */
    private float slideLeft;
    /**
     * 按钮的状态 false为关闭 true为开
     */
    private boolean curState = false;

    // 测量
    @Override
    protectedvoid onMeasure(int widthMeasureSpec, int heightMeasureSpec) {
        // super.onMeasure(widthMeasureSpec, heightMeasureSpec);
        // 设置测量值
        setMeasuredDimension(backgroundBitmap.getWidth(),
                backgroundBitmap.getHeight());
    }

    // 绘制
    @Override
    protectedvoid onDraw(Canvas canvas) {
        // super.onDraw(canvas);
```

```
        // canvas.drawColor(Color.GREEN);
        // canvas.drawCircle(50, 50, 20, paint);
        canvas.drawBitmap(backgroundBitmap, 0, 0, paint);
        canvas.drawBitmap(slideBitmap, slideLeft, 0, paint);
    }
    /**
     * 第一次按下的x坐标
     */
    intstartX = 0;
    /**
     * 最初的历史位置
     */
    int lastX = 0;
    /**
     * 滑动按钮距离左边的最大距离
     */
    int maxLeft;
    /**
     * 点击事件是否可用
     * true 可用
     * false 不可用
     */
    boolean isClickEnable = true;


    @Override
    publicboolean onTouchEvent(MotionEvent event) {
        super.onTouchEvent(event);
        switch (event.getAction()) {
        case MotionEvent.ACTION_DOWN://按下
            //1.记录第一次按下坐标
            lastX = startX = (int) event.getRawX();
            isClickEnable = true;
            break;
        case MotionEvent.ACTION_MOVE://滑动
            //2.来到新的坐标
            int newX = (int) event.getRawX();
            //3.计算偏移量
            int dX = newX - startX;
            slideLeft += dX;
            //4.更新UI-onDraw方法即可--invalidate();
            if(Math.abs(event.getRawX()-lastX)>5){
```

```
                isClickEnable = false;
            }
            flushView();
            //5.重新记录坐标
            startX = (int) event.getRawX();
            break;
        case MotionEvent.ACTION_UP://离开

            if(!isClickEnable){
                /**
                 *  当UP事件发生的时候，由按钮的左边距离(btn_left)确定View的状
态；
                    当btn_left >= maxLeft/2 设置为开状态
                    当btn_left < maxLeft/2  设置为 关闭状态
                 */

                if(slideLeft>= maxLeft/2){
                    curState = true;
                }else{
                    curState = false;
                }
                flushState();
            }

            break;

        default:
            break;
        }

        return true;
    }

// 刷新View的状态，并且纠正非法滑动
    privatevoid flushView() {
        if(slideLeft< 0){
            slideLeft = 0;
        }

        if(slideLeft>maxLeft){
            slideLeft = maxLeft;
        }
```

```
        //屏蔽非法滑动
        invalidate();
    }

    privatevoid init(Context context) {
        paint = new Paint();
        paint.setColor(Color.RED);
        // 设置抗锯齿-使其变得光滑
        paint.setAntiAlias(true);
        // 加载资源图片
        backgroundBitmap = BitmapFactory.decodeResource(getResources(),
                R.drawable.switch_background);
        slideBitmap = BitmapFactory.decodeResource(getResources(),
                R.drawable.slide_button);

        //滑动图片距离左边的距离
        maxLeft = backgroundBitmap.getWidth()-slideBitmap.getWidth();

        // 设置点击事件
        setOnClickListener(this);
    }

    // 一般会在代码中实例化
    public MyToggleButton(Context context) {
        super(context);
        init(context);
    }

    // 带有两个参数的构造方法，在布局文件中使用的时候，就会回调
    public MyToggleButton(Context context, AttributeSet attrs) {
        super(context, attrs);
        init(context);
    }

    // 我们需要设置默认的样式风格的时候
   public MyToggleButton(Context context, AttributeSet attrs, int defStyle)
{
        super(context, attrs, defStyle);
        init(context);
    }

    @Override
```

```
    publicvoid onClick(View v) {
        if(isClickEnable){
            curState = !curState;

            flushState();
        }

    }

    // 刷新View的状态
    privatevoid flushState() {
        if (curState) {

            slideLeft = maxLeft;
        } else {
            slideLeft = 0;
        }
        flushView();
    }
}
```