Android 自定义控件

讲师：杨光福

微博：http://weibo.com/321chinavideo

Day3

1、联系人快速索引

1_界面布局

创建一个 module 名字叫 09 联系人快速索引

并且分析实现布局实现

主页面的布局

```
 <?xml version="1.0" encoding="utf-8"?>
<RelativeLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"

    >
    <ListView
        android:id="@+id/lv_main_contact"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        />
    <TextView
        android:background="#44000000"
        android:id="@+id/tv_main_word"
        android:layout_width="80dp"
        android:layout_height="80dp"
        android:layout_centerInParent="true"
        android:gravity="center"
        android:text="A"
```

```
        android:textSize="30sp" />
    <com.atguigu.quichindex.IndexView
        android:id="@+id/iv_main_words"
        android:background="#ff0000"
        android:layout_alignParentRight="true"
        android:layout_height="match_parent"
        android:layout_width="30dp"
        />
</RelativeLayout>
```

IndexView 类的代码如下：

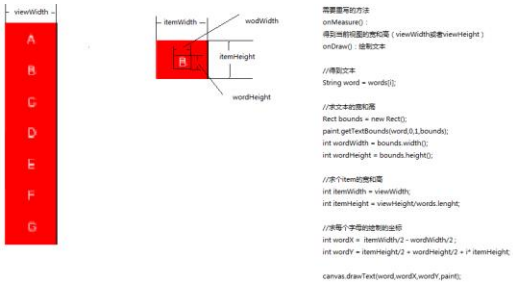
```
public class IndexView extends View {

    public IndexView(Context context, AttributeSet attrs) {
        super(context, attrs);
    }
}
```

2_初始化显示字母列表

实现步骤
1.重新 onMeasure():得到视图的宽和高，计算出 item 的高和宽
2.重写 onDraw():绘制所有字母（计算出字母的坐标）

画图分析（难点分析）



代码具体实现

```
public class IndexView extends View {
    /**
     * 每个item 的宽和高
     */
    private float itemWidth;
    private float itemHeight;
    private String[] words = {"A", "B", "C", "D", "E", "F", "G", "H", "I",
            "J", "K", "L", "M", "N", "O", "P", "Q", "R", "S", "T", "U", "V",
            "W", "X", "Y", "Z"};
    private Paint paint;

    public IndexView(Context context, AttributeSet attrs) {
        super(context, attrs);
        paint = new Paint();
        paint.setColor(Color.WHITE);
        //设置粗体字
       paint.setTypeface(Typeface.DEFAULT_BOLD);
    }

    @Override
    protected void onMeasure(int widthMeasureSpec, int heightMeasureSpec)
{
        super.onMeasure(widthMeasureSpec, heightMeasureSpec);
        //得到item 的宽和高
       itemWidth = getMeasuredWidth();
        itemHeight = getMeasuredHeight()/words.length;
    }
```

```
    @Override
    protected void onDraw(Canvas canvas) {
        super.onDraw(canvas);
        for(int i=0;i<words.length;i++){
            String word = words[i];

            Rect bounds = new Rect();
            paint.getTextBounds(word,0,1,bounds);

            //计算每个文字的宽和高
           int wordWidth = bounds.width();
            int wordHeight = bounds.height();

            float wordX = itemWidth/2 - wordWidth/2;
            float wordY = itemHeight/2 + wordHeight/2 + i * itemHeight;
            canvas.drawText(word,wordX,wordY,paint);
        }
    }
}
```

3_在按下和移动的时候是操作字母变色

实现步骤分析

1).在按下和移动时候，使操作的字母变色

a.重写 onTouchEvent(),返回 true

b.在 down/move 时，计算出操作的下标，并且在 onDraw(),设置不同颜色画笔，强制绘制

c.在 up 时，重新操作下标，强制重绘制

代码如下：

```
/**
 字母的下标位置
 */
private int touchIndex = -1;


/**
 1).在按下和移动时候，使操作的字母变色
```

```
    a.重写onTouchEvent(),返回true
    b.在down/move时，计算出操作的下标，并且在onDraw(),设置不同颜色画笔，强制绘制
    c.在up时，重新操作下标，强制重绘制
 * @param event
 * @return
 */
@Override
public boolean onTouchEvent(MotionEvent event) {
    super.onTouchEvent(event);
    switch (event.getAction()){
        case MotionEvent.ACTION_DOWN:
        case MotionEvent.ACTION_MOVE:
           float Y = event.getY();//只能用这个

           int index = (int) (Y/itemHeight);
            if(index != touchIndex){//表示不同的字母位置
              //当前字母的下标位置
              touchIndex = index;
               //强制绘制
              invalidate();//会导致onDraw(),需要设置不同颜色的画笔
           }

             break;
        case MotionEvent.ACTION_UP://离开
           touchIndex = -1;//重置下标位置
           invalidate();
            break;
    }
        return true;
}
```

在 onDraw()方法中设置画笔颜色

```
@Override
protected void onDraw(Canvas canvas) {
    super.onDraw(canvas);
    for(int i=0;i<words.length;i++){

        //设置当前下标对应的字母为灰色，其他为白色
       if(i == touchIndex){
```

```
            paint.setColor(Color.GRAY);
        }else {
            paint.setColor(Color.WHITE);
        }

        String word = words[i];

        Rect bounds = new Rect();//矩形
        paint.getTextBounds(word,0,1,bounds);

        //计算每个文字的宽和高
        int wordWidth = bounds.width();
        int wordHeight = bounds.height();

        float wordX = itemWidth/2 - wordWidth/2;
        float wordY = itemHeight/2 + wordHeight/2 + i * itemHeight;
        canvas.drawText(word,wordX,wordY,paint);
    }
}
```

4_在按下和移动时显示更新提示字母

定义接口

```
/**
 * 监听字母下标的变化
 */
public interface OnIndexChangeListener{
    /**
     * 当字母下标位置变化的时候，回调该方法
     * @param word 字母
     */
    public void onIndexChange(String word);
}
```

```
private  OnIndexChangeListener onIndexChangeListener;

/**
 * 设置监听下标位置变化
```

```
 * @param onIndexChangeListener
 */
public void setOnIndexChangeListener(OnIndexChangeListener
onIndexChangeListener) {
    this.onIndexChangeListener = onIndexChangeListener;
}
```

调用接口

```
@Override
public boolean onTouchEvent(MotionEvent event) {
    super.onTouchEvent(event);
    switch (event.getAction()){
        case MotionEvent.ACTION_DOWN:
        case MotionEvent.ACTION_MOVE:
           float Y = event.getY();//只能用这个

            int index = (int) (Y/itemHeight);
            if(index != touchIndex){//表示不同的字母位置
                //当前字母的下标位置
                touchIndex = index;
                //强制绘制
                invalidate();//会导致 onDraw(),需要设置不同颜色的画笔

                //调用接口的方法
                if(onIndexChangeListener !=
null&&touchIndex<words.length){

onIndexChangeListener.onIndexChange(words[touchIndex]);
                }
            }

             break;
        case MotionEvent.ACTION_UP://离开
            touchIndex = -1;//重置下标位置
            invalidate();
            break;
    }
        return true;
}
```

在 Activity 中使用接口

```
public class MainActivity extends Activity {


    private ListView lv_main_contact;
    private TextView tv_main_word;
    private IndexView iv_main_words;

    private Handler handler = new Handler();

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        lv_main_contact = (ListView) findViewById(R.id.lv_main_contact);
        tv_main_word = (TextView) findViewById(R.id.tv_main_word);
        iv_main_words = (IndexView) findViewById(R.id.iv_main_words);

        //设置页面监听
        iv_main_words.setOnIndexChangeListener(new
IndexView.OnIndexChangeListener() {
            @Override
            public void onIndexChange(String word) {
                tv_main_word.setVisibility(View.VISIBLE);
                tv_main_word.setText(word);

                //把所有消息移除
                handler.removeCallbacksAndMessages(null);
                //发消息 2 秒钟后自动消失
                handler.postDelayed(new Runnable() {
                    @Override
                    public void run() {
                        tv_main_word.setVisibility(View.GONE);
                    }
                },2000);
            }
        });
    }
}
```

5_列表显示联系人

添加 pinyin4j-2.5.0.jar 和工具类

```
/**
 * 作者：杨光福 on 2016/4/14 13:57
 * 微信：yangguangfu520
 * QQ 号：541433511
 * 作用：把汉字转换成拼音
 */
public class PinYinUtils {
    /**
     * 得到指定汉字的拼音
     * 注意:不应该被频繁调用，它消耗一定内存
     * @param hanzi
     * @return
     */
    public static String getPinYin(String hanzi){
        String pinyin = "";

        HanyuPinyinOutputFormat format = new HanyuPinyinOutputFormat();//
控制转换是否大小写，是否带音标
        format.setCaseType(HanyuPinyinCaseType.UPPERCASE);//大写
        format.setToneType(HanyuPinyinToneType.WITHOUT_TONE);//不要音标

        //由于不能直接对多个汉字转换，只能对单个汉字转换
        char[] arr = hanzi.toCharArray();
        for (int i = 0; i < arr.length; i++) {
            if(Character.isWhitespace(arr[i]))continue;//如果是空格，则不处
理，进行下次遍历

            //汉字是 2 个字节存储，肯定大于 127，所以大于 127 就可以当为汉字转换
            if(arr[i]>127){
                try {
                    //由于多音字的存在，单 dan shan
                      String[] pinyinArr =
PinyinHelper.toHanyuPinyinStringArray(arr[i], format);

                    if(pinyinArr!=null){
                        pinyin += pinyinArr[0];
                    }else {
                        pinyin += arr[i];
                    }
```

```
                } catch (BadHanyuPinyinOutputFormatCombination e) {
                    e.printStackTrace();
                    //不是正确的汉字
                    pinyin += arr[i];
                }
            }else {
                //不是汉字,
                pinyin += arr[i];
            }
        }
        return pinyin;
    }
}
```

写 Person 对象

```
public class Person {

    /**
     * 姓
     */
    private String name;

    /**
     * 拼音
     */
    private String pinyin;

    public Person(String name){
        this.name = name;
        this.pinyin = PinYinUtils.getPinYin(name);

    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public String getPinyin() {
```

```
        return pinyin;
    }

    public void setPinyin(String pinyin) {
        this.pinyin = pinyin;
    }

    @Override
    public String toString() {
        return "Person{" +
                "name='" + name + '\'' +
                ", pinyin='" + pinyin + '\'' +
                '}';
    }
}
```

准备适配器的数据并且排序

```
/**
 * 初始化数据
 */
private void initData() {

    persons = new ArrayList<>();
    persons.add(new Person("张晓飞"));
    persons.add(new Person("杨光福"));
    persons.add(new Person("胡继群"));
    persons.add(new Person("刘畅"));

    persons.add(new Person("钟泽兴"));
    persons.add(new Person("尹革新"));
    persons.add(new Person("安传鑫"));
    persons.add(new Person("张骞壬"));

    persons.add(new Person("温松"));
    persons.add(new Person("李凤秋"));
    persons.add(new Person("刘甫"));
    persons.add(new Person("娄全超"));
    persons.add(new Person("张猛"));

    persons.add(new Person("王英杰"));
    persons.add(new Person("李振南"));
```

```
        persons.add(new Person("孙仁政"));
        persons.add(new Person("唐春雷"));
        persons.add(new Person("牛鹏伟"));
        persons.add(new Person("姜宇航"));

        persons.add(new Person("刘挺"));
        persons.add(new Person("张洪瑞"));
        persons.add(new Person("张建忠"));
        persons.add(new Person("侯亚帅"));
        persons.add(new Person("刘帅"));

        persons.add(new Person("乔竞飞"));
        persons.add(new Person("徐雨健"));
        persons.add(new Person("吴亮"));
        persons.add(new Person("王兆霖"));

        persons.add(new Person("阿三"));


        //排序
        Collections.sort(persons, new Comparator<Person>() {
            @Override
            public int compare(Person lhs, Person rhs) {
                return lhs.getPinyin().compareTo(rhs.getPinyin());
            }
        });

    }
```

item_main.xml 布局

```
 <?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="wrap_content"
    android:orientation="vertical">

    <TextView
        android:id="@+id/tv_word"
        android:text="A"
        android:textSize="20sp"
        android:background="#44000000"
```

```
        android:padding="5dp"
        android:layout_width="match_parent"
        android:layout_height="wrap_content" />

    <TextView
        android:id="@+id/tv_name"
        android:text="姓名"
        android:textSize="20sp"
        android:padding="5dp"
        android:layout_width="match_parent"
        android:layout_height="wrap_content" />

</LinearLayout>
```

设置在适配器中显示

```
class MyAdapter extends BaseAdapter{

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

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

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

    @Override
    public View getView(int position, View convertView, ViewGroup parent)
{
        ViewHolder viewHolder;
        if(convertView == null){
            convertView =
View.inflate(MainActivity.this,R.layout.item_main,null);
            viewHolder = new ViewHolder();
            viewHolder.tv_name = (TextView)
```

```
convertView.findViewById(R.id.tv_name);
            viewHolder.tv_word = (TextView)
convertView.findViewById(R.id.tv_word);
            convertView.setTag(viewHolder);
        }else {
            viewHolder = (ViewHolder) convertView.getTag();
        }

        //根据位置得到数据
        Person person = persons.get(position);
        viewHolder.tv_name.setText(person.getName());

        String word = person.getPinyin().substring(0,1);//A

         viewHolder.tv_word.setText(word);
        //隐藏不是第0个字母的item
         if(position == 0){
            viewHolder.tv_word.setVisibility(View.VISIBLE);
        }else{

            //得到前一个item的首个汉字的首字母
            String preWord =
persons.get(position-1).getPinyin().substring(0,1);
            if(preWord.equals(word)){
                viewHolder.tv_word.setVisibility(View.GONE);
            }else{
                viewHolder.tv_word.setVisibility(View.VISIBLE);
            }

        }

        return convertView;
    }
}
```

6_在按下和移动是列表更新

//把设置字母的抽取成额外方法

```
 //设置页面监听
iv_main_words.setOnIndexChangeListener(new
IndexView.OnIndexChangeListener() {
    @Override
    public void onIndexChange(String word) {
        updateWord(word);
        updateListView(word);
    }
});


 /**
  更新列表
 */
private void updateListView(String word) {

    for(int i = 0 ; i < persons.size() ; i++){
        //查找每个名字的汉字的首字母
        String preWord = persons.get(i).getPinyin().substring(0,1);
        //判断是否相同
        if(preWord.equals(word)){
            lv_main_contact.setSelection(i);
            return;
        }
    }


}
```

2、侧滑删除菜单

1_正常初始化显示 item 的布局

创建一个 module 名字叫 08 侧滑删除菜单

并且分析实现原理，重点事件冲突的解决

主页面的布局

```
 <?xml version="1.0" encoding="utf-8"?>
<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="com.atguigu.slidemenuitem.MainActivity">

    <ListView
        android:id="@+id/listview"
        android:layout_width="match_parent"
        android:layout_height="match_parent" />
</RelativeLayout>
```

自定义 item 类 SlideLayout

```
 public class SlideLayout extends FrameLayout {

    public SlideLayout(Context context, AttributeSet attrs) {
        super(context, attrs);
    }
}
```

item 的布局 item_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<com.atguigu.slidemenuitem.SlideLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="wrap_content">


    <include
        android:id="@+id/item_content"
        layout="@layout/item_content" />

    <include
        android:id="@+id/item_menu"
```

```
        layout="@layout/item_menu" />

</com.atguigu.slidemenuitem.SlideLayout>
```

item_content.xml 布局

```
 <?xml version="1.0" encoding="utf-8"?>
<TextView xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="60dp"
    android:textColor="#000000"
    android:text="content"
    android:background="#aaaaaa"
    android:gravity="center"
    android:textSize="20sp">

</TextView>
```

item_menu.xml 布局

```
<?xml version="1.0" encoding="utf-8"?>
<TextView xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="wrap_content"
    android:layout_height="60dp"
    android:background="#44000000"
    android:textColor="#ff0000"
    android:text="Delete"
    android:gravity="center"
    android:textSize="20sp">

</TextView>
```

把 item_main 加载到屏幕看看

2_正常初始化显示 item 的代码实现

写流程

```
1.正常显示item 代码实现
* 1.1).得到子View 对象（ContentView,MenuView）-->onFinishInflate()
* 1.2).得到子View 的宽和高-->onMeasure()
* 1.3).对item 视图进行重新布局-->onLayout
```

得到子 View 对象

```
/**
 *
 *当布局文件加载完成后回调这个方法
 *  1.1).得到子View 对象（ContentView,MenuView）-->onFinishInflate()
 */
@Override
protected void onFinishInflate() {
    super.onFinishInflate();
    contentView = getChildAt(0);
    menuView = getChildAt(1);
}
```

得到子 View 的宽和高

```
/**
     * 1.2).得到子View 的宽和高-->onMeasure()
     * @param widthMeasureSpec
     * @param heightMeasureSpec
     */
    @Override
   protected void onMeasure(int widthMeasureSpec, int heightMeasureSpec) {
        super.onMeasure(widthMeasureSpec, heightMeasureSpec);

        contentWidth = contentView.getMeasuredWidth();
//         contentWidth = getMeasuredWidth();//可以
        menuWidth = menuView.getMeasuredWidth();
//         menuWidth = getMeasuredWidth();//不可以

        ViewHeight = getMeasuredHeight();
```

```
    }
```

对子 View 进行重新布局

```
/**
 *  * 1.3).对子View进行重新布局-->onLayout
  */
@Override
protected void onLayout(boolean changed, int left, int top, int right, int
bottom) {
    super.onLayout(changed, left, top, right, bottom);
    menuView.layout(contentWidth,0,contentWidth+menuWidth,ViewHeight);
}
```

3_通过手势拖动打开或者关闭 menu

实现左右滑动

```
/**
     * 第一次按下的值
     */
    private int lastX;

    @Override
    public boolean onTouchEvent(MotionEvent event) {
        int eventX = (int) event.getRawX();
        switch (event.getAction()) {
            case MotionEvent.ACTION_DOWN:
                //1.记录起始坐标
                lastX = eventX;
                break;
            case MotionEvent.ACTION_MOVE:
                //2.计算偏移量
                int distencX = eventX - lastX;

                int toScrollX = getScrollX() - distencX;

                System.out.println(toScrollX);
                //屏蔽非法值
                if (toScrollX < 0) {
```

```
                toScrollX = 0;
            }else if(toScrollX > menuWidth){
                toScrollX = menuWidth;
            }

//              scrollTo(toScrollX, 0);//也可以
            scrollTo(toScrollX, getScrollY());

            //重新付值
            lastX = eventX;
            break;
        case MotionEvent.ACTION_UP:
            break;
    }

//      return super.onTouchEvent(event);
     return true;
}
```

运行演示看看

4_判断是平滑的打开还是关闭

当 up 时，判断是平滑的打开还是关闭

```
case MotionEvent.ACTION_UP:
  // 2.3).当up的时候，计算总的偏移量，判断是平滑的关闭或者打开
   int totallScrollX = getScrollX();

   if(totallScrollX < menuWidth/2){
       System.out.println("totallScrollX < menuWidth/2");
       closeMenu();
   }else{
       System.out.println("totallScrollX >= menuWidth/2");
       openMenu();
   }
   break;
```

```
private void openMenu() {//--->menuWidth

scroller.startScroll(getScrollX(),getScrollY(),menuWidth-getScrollX(),g
etScrollY());
        invalidate();//会导致 执行computeScroll
}

private void closeMenu() {//--->0

scroller.startScroll(getScrollX(),getScrollY(),0-getScrollX(),getScroll
Y());
    invalidate();//会导致 执行computeScroll
}

@Override
public void computeScroll() {
    super.computeScroll();
    if(scroller.computeScrollOffset()){
        scrollTo(scroller.getCurrX(),scroller.getCurrY());
        invalidate();//强制重绘制
    }
}
```

5_在 ListView 中显示侧滑 item

```
public class MainActivity extends Activity {

    private ListView listview;

    private List<MyBean> myBeans;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        listview = (ListView) findViewById(R.id.listview);
        myBeans = new ArrayList<>();
        for(int i=0 ; i < 100 ;i++){
```

```
            myBeans.add(new MyBean("content"+i));
        }

        //设置适配器
        listview.setAdapter(new MyAdapter());
    }

    class MyAdapter extends BaseAdapter{

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

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

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

        @Override
        public View getView(int position, View convertView, ViewGroup parent)
{
            ViewHolder viewHolder;
            if(convertView == null){
                convertView =
View.inflate(MainActivity.this,R.layout.item_main,null);
                viewHolder = new ViewHolder();
                viewHolder.item_content = (TextView)
convertView.findViewById(R.id.item_content);
                viewHolder.item_menu = (TextView)
convertView.findViewById(R.id.item_menu);
                convertView.setTag(viewHolder);
            }else{
                viewHolder = (ViewHolder) convertView.getTag();
            }

            MyBean myBean = myBeans.get(position);
            viewHolder.item_content.setText(myBean.getName());
```

```
            return convertView;
        }
    }


    static class ViewHolder{
        TextView item_content;
        TextView item_menu;
    }
}
```

运行演示，看滑动效果

6_解决 item 滑动后不能自动打开和关闭

1.原因分析

事件被 ListView 拦截,也就是说，当前 ListView 与子 item 的冲突

反拦截

2.代码实现

```
 /**
     * 第一次按下的值
     */
    private int lastX;
    private int downX;
    private int lastY;
    private int downY;

    @Override
    public boolean onTouchEvent(MotionEvent event) {
        int eventX = (int) event.getRawX();
        int eventY = (int) event.getRawY();
        switch (event.getAction()) {
            case MotionEvent.ACTION_DOWN:
                //1.记录起始坐标
                downX = lastX = eventX;
                downY = lastY = eventX;
                break;
            case MotionEvent.ACTION_MOVE:
                //2.计算偏移量
```

```
                int distenceX = eventX - lastX;
                int distenceY = eventY - lastY;

                int toScrollX = getScrollX() - distenceX;

                System.out.println(toScrollX);
                //屏蔽非法值
                if (toScrollX < 0) {
                    toScrollX = 0;
                }else if(toScrollX > menuWidth){
                    toScrollX = menuWidth;
                }

//                  scrollTo(toScrollX, 0);//也可以
                scrollTo(toScrollX, getScrollY());

                //重新付值
                lastX = eventX;

                int dX = Math.abs(eventX - downX);
                int dY = Math.abs(eventY - downY);

                if(dX > dY&& dX >8){
                    getParent().requestDisallowInterceptTouchEvent(true);
                }


                break;
            case MotionEvent.ACTION_UP:
               // 2.3).当up的时候，计算总的偏移量，判断是平滑的关闭或者打开
                int totallScrollX = getScrollX();
                if(totallScrollX < menuWidth/2){
                    System.out.println("totallScrollX < menuWidth/2");
                    //关闭菜单
                    closeMenu();
                }else{
                    System.out.println("totallScrollX >= menuWidth/2");
                    //打开菜单
                    openMenu();
                }
                break;
        }
```

```
//         return super.onTouchEvent(event);
        return true;
    }
```

7_内容视图设置点击事件时不能滑动 item

在 getView 方法中设置点击事件

```
viewHoler.item_content.setTag(position);
viewHoler.item_content.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        int position = (int) v.getTag();
        MyBean bean = myBeans.get(position);
        Toast.makeText(MainActivity.this, "bean==" + bean.getName(),
Toast.LENGTH_SHORT).show();
    }
});
viewHoler.item_menu.setTag(position);
viewHoler.item_menu.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {

        SlideLayout slideLayout = (SlideLayout) v.getParent();
        slideLayout.closeMenu();
        int position = (int) v.getTag();
        myBeans.remove(position);
        myAdapte.notifyDataSetChanged();

    }
});
```

分析原因

事件被点击 TextView 事件消费

解决方法，在 item 中拦截

```
@Override
public boolean onInterceptTouchEvent(MotionEvent event) {
    boolean intercept = false;
    int eventX = (int) event.getRawX();
    int eventY = (int) event.getRawY();
```

```
    switch (event.getAction()) {
        case MotionEvent.ACTION_DOWN:
            //1.记录起始坐标
            downX = lastX = eventX;
            downY = lastY = eventX;
            break;
        case MotionEvent.ACTION_MOVE:
            //2.计算偏移量
            int dX = Math.abs(eventX - downX);
            if( dX >8){
                intercept = true;
            }

            break;
    }
    return intercept;
}
```

解决删除后还显示打开的删除 TextView

```
viewHolder.item_menu.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {

        SlideLayout slideLayout = (SlideLayout) v.getParent();
        slideLayout.closeMenu();
        int position = (int) v.getTag();
        myBeans.remove(position);
        myAdapter.notifyDataSetChanged();
    }
});
```

8_限制只能打开一个 item

在 SlideLayout 中定义接口

```
 public interface OnStateChangeListener{
    /**
     * 当item 被打开的时候回调
     * @param layout
     */
    public void onOpen(SlideLayout layout);
```

```
    /**
     * 当item关闭的时候回调
     * @param layout
     */
    public void onClose(SlideLayout layout);

    /**
     * 当item按下的时候被回调
     * @param layout
     */
    public void onDown(SlideLayout layout);
}


private OnStateChangeListener onStateChangeListener;


public void setStateChangeListener(OnStateChangeListener
stateChangeListener) {
    this.onStateChangeListener = stateChangeListener;
}
```

在 SlideLayout 中调用接口

```
public void openMenu() {//--->menuWidth
    scroller.startScroll(getScrollX(), getScrollY(), menuWidth -
getScrollX(), getScrollY());
    invalidate();//会导致 执行computeScroll
    if(onStateChangeListener != null){
        onStateChangeListener.onOpen(this);
    }
}

public void closeMenu() {//--->0

scroller.startScroll(getScrollX(),getScrollY(),0-getScrollX(),getScroll
Y());
    invalidate();//会导致 执行computeScroll
    if(onStateChangeListener != null){
        onStateChangeListener.onClose(this);
```

```
    }
}
```

在 SlideLayout 中的 onInterceptTouchEvent 方法使用

```
@Override
public boolean onInterceptTouchEvent(MotionEvent event) {
    boolean intercept = false;
    int eventX = (int) event.getRawX();
    int eventY = (int) event.getRawY();
    switch (event.getAction()) {
        case MotionEvent.ACTION_DOWN:
            //1.记录起始坐标
           downX = lastX = eventX;
            downY = lastY = eventX;
            if(onStateChangeListener != null){
                onStateChangeListener.onDown(this);
            }
            break;
        case MotionEvent.ACTION_MOVE:
            //2.计算偏移量
           int dX = Math.abs(eventX - downX);
            if( dX >8){
                intercept = true;
            }

            break;
    }
    return intercept;
}
```

在 MainActivity 的适配器 getView()方法中使用接口-设置点击事件

```
SlideLayout slideLayout = (SlideLayout) convertView;
slideLayout.setOnStateChangeListener(new MyOnStateChangeListener());
```

在 MainActivity 使用接口-回调的处理

```
private SlideLayout  slideLayout;
class MyOnStateChangeListener implements
SlideLayout.OnStateChangeListener {

    @Override
    public void onClose(SlideLayout layout) {
        if(slideLayout ==layout){//保持的置为空
            slideLayout = null;
        }

    }

    @Override
    public void onOpen(SlideLayout layout) {
        slideLayout = layout;//保持到内存中
    }

    @Override
    public void onDraw(SlideLayout layout) {
        if(slideLayout != null && slideLayout != layout){
            slideLayout.closeMenu();
        }

    }
}
```