# Github博客原始文档

## 亮点

### 亮点：windowSoftInputMode

|  |
| --- |
| <activity android:name=*".app.CustomTitle"*  android:label=*"@string/activity\_custom\_title"*  android:windowSoftInputMode=*"stateVisible|adjustPan"*> |

### 2．亮点：listView 过滤文字

AbsListView.setTextFilterEnabled(true);

这个方法的作用是用来过滤选项的.

例如在软键盘上打出一个a,则会过滤掉除了a开头的所有选项.

## 功能点

### 功能：自定义 titleBar

|  |
| --- |
| requestWindowFeature(Window.***FEATURE\_CUSTOM\_TITLE***);  setContentView(R.layout.***custom\_title***);  getWindow().setFeatureInt(Window.***FEATURE\_CUSTOM\_TITLE***,  R.layout.***custom\_title\_1***);  **final** TextView leftText = (TextView) findViewById(R.id.***left\_text***); |

1. 功能：监听apk安装卸载，获取安装卸载apk包名和uid

|  |
| --- |
| 注册广播 <receiverandroid:name=*"com.cs.cspermission.InstallApkReceiver"* >  <intent-filter>  <action android:name=*"android.intent.action.PACKAGE\_ADDED"* />  <action android:name=*"android.intent.action.PACKAGE\_REMOVED"* />  <data android:scheme=*"package"* />  </intent-filter>  </receiver> |

|  |
| --- |
| 获取apk uid 和 packageName  **if**(Intent.***ACTION\_PACKAGE\_ADDED***.equalsIgnoreCase(intent.getAction())){  Uri uri = **null**;  uri = intent.getData();  **int** uid = intent.getIntExtra(Intent.***EXTRA\_UID***, -4);  String packageName = uri.getEncodedSchemeSpecificPart();  ……  …… |

### 功能：在values string中定义 数组，代码中使用

|  |
| --- |
| <string-array name=*"dialog\_title\_time"*>  <item>今日</item>  <item>本月</item>  </string-array>  **New** ArrayList<String>(Arrays.*asList*(  getResources().getStringArray(R.array.***dialog\_title\_time***))) |

### 功能：一段文字中包含电话等url高亮显示并可点击触发事件(打电话，调起浏览器)

|  |
| --- |
| **if** (text **instanceof** Spannable) {  **int** end = text.length();  Spannable sp = (Spannable) mTextInfo.getText();  URLSpan[] urls = sp.getSpans(0, end, URLSpan.**class**);  SpannableStringBuilder style = **new**SpannableStringBuilder(text);  style.clearSpans();// should clear old spans  **for** (URLSpan url : urls) {  MyURLSpan myURLSpan = **new** MyURLSpan(url.getURL());  style.setSpan(myURLSpan, sp.getSpanStart(url), sp.getSpanEnd(url), Spannable.***SPAN\_EXCLUSIVE\_INCLUSIVE***);  }  mTextInfo.setText(style);  } |

### 5.功能：文件 IO

|  |
| --- |
| FileOutputStream outStream = this.openFileOutput("itcast.txt", Context.MODE\_PRIVATE);          outStream.write("要写入的数据".getBytes());          outStream.close(); |

第一个参数：指定文件名称，不能包含路径分隔符“/” ，如果文件不存在，Android 自动创建它。保存在/data/data/<package name>/files目录中，如： /data/data/cn.itcast.action/files/ceshi.txt

## 注意点

1. setEnable 会使点击事件失效
2. singleTask 会将在其上面栈的Activity 挤掉，普通模式 直接新建一个singleTop和standard模式，都会将intent发送新的实例（后两种模式不发送到新的实例，如果已经有了的话）。不过，singleTop要求如果创建intent的时候栈顶已经有要创建的Activity的实例，则将intent发送给该实例，而不发送给新的实例。
3. 在 startActivityForResult 如果设置 为 singleInstance 将 无法 返回
4. 这两个LaunchMode标识只能用在startActivity()的方法中，而不能使用在startActivityForResult方法中。因为从Task的角度看，Android认为不同Task之间的Activity是不能传递数据的。所以也不会有有用的result返回。
5. 一个 edittext 在没有初始化之前 传递给一个函数，如果再初始化 edittext ，那么函数中的edittext 将不会被初始化。若是 先初始化 再传给 函数，那么函数中的修改同样作用于之前的edittext，即他们此时 是同一个edittext

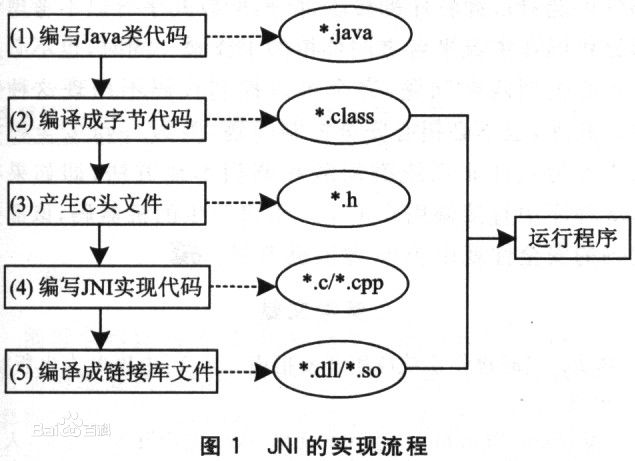
android.intent.action.MAIN： Start as a main entry point, does not expect to receive data.

Input: nothing

Output: nothing

作为一个主入口，没有输入值也没有输出值

android.intent.category.LAUNCHER 表示为一个桌面入口，作为桌面入口必须有android.intent.action.MAIN 的Action

1. android 资源文件位置例如D:\android-sdk-windows\platforms\android-4\data\res\layout
2. Android 事件传递
3. jni实现流程  
   

## 大点

### 1.fragment：

#### 使用方式，xml中定义 FrameLayout，调用FragmentTransaction replace方法

|  |
| --- |
| <FrameLayout  android:id=*"@+id/frameDetail"*  android:layout\_width=*"match\_parent"*  android:layout\_height=*"match\_parent"* >  </FrameLayout>  **private** **void** setFragment() {  FragmentManager fmg = getFragmentManager();  FragmentTransaction ftc = fmg.beginTransaction();  mFragDetail = **new** FragmentDetail(mCurrentTime, mCurrentFrom);  ftc.replace(R.id.***frameDetail***, mFragDetail);  ftc.commit();  } |

#### fragment 生命周期与 activity 生命周期关系



### 2.ScrollView

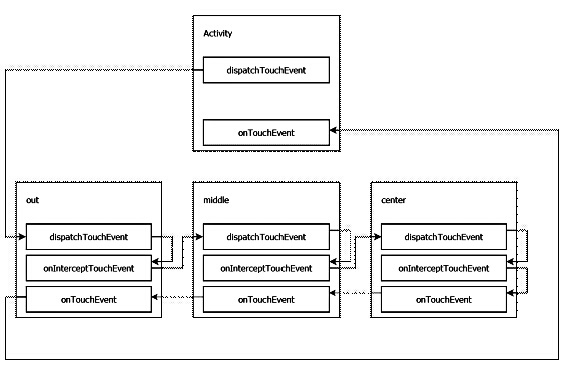
setFillViewport(boolean) scrollView 设置当前滚动视图是否将内容高度拉伸以填充视图可视范围

### 3.View

#### 事件

在一个view中顺序依次1、dispatchTouchEvent，2、 setOnTouchListener的onTouch，3、onTouchEvent。 如果mOnTouchListener.onTouch(this, event)返回true则onTouchEvent(event）不会被执行了。

Viewgroup事件拦截



默认是不拦截的，即返回false；如果你需要拦截，只要return true就行了，这要该事件就不会往子View传递了，并且如果你在DOWN retrun true ，则DOWN,MOVE,UP子View都不会捕获事件。

requestDisallowInterceptTouchEvent(boolean) 用于设置是否允许拦截

1. getParent().requestDisallowInterceptTouchEvent(**true**);

总结：

1、如果ViewGroup找到了能够处理该事件的View，则直接交给子View处理，自己的onTouchEvent不会被触发；

2、可以通过复写onInterceptTouchEvent(ev)方法，拦截子View的事件（即return true），把事件交给自己处理，则会执行自己对应的onTouchEvent方法

3、子View可以通过调用getParent().requestDisallowInterceptTouchEvent(true);  阻止ViewGroup对其MOVE或者UP事件进行拦截；

#### 重要方法说明

**protected** **boolean** overScrollBy(**int** deltaX, **int** deltaY,

**int** scrollX, **int** scrollY,

**int** scrollRangeX, **int** scrollRangeY,

**int** maxOverScrollX, **int** maxOverScrollY,

**boolean** isTouchEvent)

该方法就相当于在scrollTo/By的基础上添加了对overScroll情况的处理, 但父类view中只处理数据,没有实际的scroll操作,父类view处理完数据后将其记录在onOverScrolled方法中,  
子类继承onOverScrolled方法再根据得到的数据scrollTo/By处理即可~

scrollTo() 和scrollBy()方法是将控件内容移动而不是控件本身

offsetLeftAndRight(int offset)用于移动控件本身

#### 重要类说明

Scroller：

Scroller只是用于跟踪控件滑动的轨迹,此类不会移动控件,只相当于一个滚动轨迹记录工具所以需要你在View的一个回调函数computerScroll()中使用Scroller对象获取滑动的数据,  
最后还是用scrollTo/By方法来实际控制某个View的滚动行为