## 一、实现功能

本界面可对手机中的程序建立列表，并通过滑动打开按钮界面，通过按钮实现运行、注入、解注入等功能。并且主列表上会有相应的状态显示。(已关闭。并可通过右上角的按钮进入设置界面，对主界面的操控进行设置，如设置是否可以左滑、右滑，滑动打开的程度等。)



## 二、UML框图





## 三、代码解读

## 1、package com.fortysevendeg.android.swipelistview.sample.activities

SettingsActivity用来运行设置界面。

首先调用SettingsManager类（在**package** com.fortysevendeg.android.swipelistview.sample.utils中）。

**private** SettingsManager settings;

该类中有关于设置界面的输入和读出函数，并存储设置数据。

如下是设置以及返回 settingsManage的函数r:

**public** **static** SettingsManager getSettingsManager() {

**return** *settingsManager*;

}

**public** **static** **void** setSettingsManager(SettingsManager settingsManager) {

SettingsManager.*settingsManager* = settingsManager;

}

之后对每个选项，按钮编写函数。

如下是设置是否可以左滑,右滑或两侧都可以的单选按钮的函数：

RadioButton rbModeBoth = (RadioButton) findViewById(R.id.*mode\_both*);

rbModeBoth.setOnCheckedChangeListener(radiosListener);

RadioButton rbModeLeft = (RadioButton) findViewById(R.id.*mode\_left*);

rbModeLeft.setOnCheckedChangeListener(radiosListener);

RadioButton rbModeRight = (RadioButton) findViewById(R.id.*mode\_right*);

rbModeRight.setOnCheckedChangeListener(radiosListener);

**if** (settings.getSwipeMode() == SwipeListView.*SWIPE\_MODE\_BOTH*) {

rbModeBoth.setChecked(**true**);

} **else** **if** (settings.getSwipeMode() == SwipeListView.*SWIPE\_MODE\_LEFT*) {

rbModeLeft.setChecked(**true**);

} **else** **if** (settings.getSwipeMode() == SwipeListView.*SWIPE\_MODE\_RIGHT*) {

rbModeRight.setChecked(**true**);

}

SwipeListViewExampleActivity：生成主界面.

生成PackageAdapter类和PackageItem类的变量

两个类在**package** com.fortysevendeg.android.swipelistview.sample.adapters中。

PackageAdapter类编写了主界面，包括图片，text和各个按钮的具体函数。

如下是”打开”按钮的函数：

holder.bAction1.setOnClickListener(**new** View.OnClickListener() {//打开按钮

@Override

**public** **void** onClick(View v) {

Intent intent = context.getPackageManager().getLaunchIntentForPackage(item.getPackageName());

**if** (intent != **null**) {

context.startActivity(intent);

} **else** {

Toast.*makeText*(context, R.string.*cantOpen*, Toast.*LENGTH\_SHORT*).show();

}

}

});

PackageItem主要是存放主界面数据的类。

如下可设置和返回packageName :

**public** String getPackageName() {

**return** packageName;

}

**public** **void** setPackageName(String packageName) {

**this**.packageName = packageName;

}

## 2、package com.fortysevendeg.android.swipelistview.sample.utils;

PreferencesManager 父类,保存主界面数据

SettingsManager 设置界面的数据读写,父类

## 四、、关键机制的解读

## 1、package com.fortysevendeg.android.swipelistview.sample.adapters

**PackageAdapter**：

在ListView的使用中，有时候还需要在里面加入按钮等控件，实现单独的操作。也就是说，这个ListView不再只是展示数据，也不仅仅是这一行要来处理用户的操作，而是里面的控件要获得用户的焦点。读者可以试试用SimpleAdapter添加一个按钮到ListView的条目中，会发现可以添加，但是却无法获得焦点，点击操作被ListView的Item所覆盖。这时候最方便的方法就是使用灵活的适配器BaseAdapter了。

使用BaseAdapter必须写一个类继承它，同时BaseAdapter是一个抽象类，继承它必须实现它的方法。BaseAdapter的灵活性就在于它要重写很多方法，其中最重要的即为getView()方法。这些方法都有什么作用呢？我们通过分析ListView的原理来为读者解答。

当系统开始绘制ListView的时候，首先调用getCount()方法。得到它的返回值，即ListView的长度。

**public** **int** getCount() {//返回数组的长度

**return** data.size();

}

然后通过package com.fortysevendeg.android.swipelistview里的SwipeListViewTouchListener来设置ListView的长度。

**public** **void** resetItems() {

**if** (swipeListView.getAdapter() != **null**) {

**int** count = swipeListView.getAdapter().getCount();

**for** (**int** i = opened.size(); i <= count; i++) {

opened.add(**false**);

openedRight.add(**false**);

checked.add(**false**);

}

}

}

然后系统调用getView()方法，根据这个长度逐一绘制ListView的每一行。

**public** View getView(**final** **int** position, View convertView, ViewGroup parent){

**……**

**……**

**……**

**}**

getView()方法里边有对每一个ListView里的对象的界面的具体设置,包括图片,包名以及各个按钮和按钮对应的函数。

也就是说，如果让getCount()返回1，那么只显示一行。而getItem()和getItemId()则在需要处理和取得Adapter中的数据时调用。

**public** PackageItem getItem(**int** position) {

**return** data.get(position);

}

@Override

**public** **long** getItemId(**int** position) {//呵呵

**return** position;

}

当启动Activity呈现第一屏ListView的时候，convertView为零。当用户向下滚动ListView时，上面的条目变为不可见，下面出现新的条目。这时候convertView不再为空，而是创建了一系列的convertView的值。当又往下滚一屏的时候，发现第11行的容器用来容纳第22行，第12行的容器用来容纳第23行。也就是说convertView相当于一个缓存，开始为0，当有条目变为不可见，它缓存了它的数据，后面再出来的条目只需要更新数据就可以了，这样大大节省了系统资料的开销。

如下,为空时对其进行输入,当不为空时就取出缓存对象使用：

**if** (convertView == **null**) {

LayoutInflater li = (LayoutInflater)context.getSystemService(Context.*LAYOUT\_INFLATER\_SERVICE*);

convertView = li.inflate(R.layout.*package\_row*, parent, **false**);//那还输入干嘛？为了判断非空

holder = **new** ViewHolder();

/\*得到各个控件的对象\*/

holder.ivImage = (ImageView) convertView.findViewById(R.id.*example\_row\_iv\_image*);

holder.tvTitle = (TextView) convertView.findViewById(R.id.*example\_row\_tv\_title*);

holder.tvDescription = (TextView) convertView.findViewById(R.id.*example\_row\_tv\_description*);

holder.bAction1 = (Button) convertView.findViewById(R.id.*example\_row\_b\_action\_1*);

holder.bAction2 = (Button) convertView.findViewById(R.id.*example\_row\_b\_action\_2*);

holder.bAction3 = (Button) convertView.findViewById(R.id.*example\_row\_b\_action\_3*);

holder. bAction4 = (Button) convertView.findViewById(R.id.*example\_row\_b\_action\_4*);

convertView.setTag(holder);//绑定ViewHolder对象

} **else** {

holder = (ViewHolder) convertView.getTag();//取出ViewHolder对象 //使用缓存的

}

虽然重复利用了已经绘制的view，但是要得到其中的控件，需要在控件的容器中通过findViewById的方法来获得。如果这个容器非常复杂，这显然会增加系统资源的开销。在上面的例子中，引入了Tag的概念。或许不是最好的办法，但是它确实能使ListView变得更流畅。代码中，当convertView为空时，用setTag()方法为每个View绑定一个存放控件的ViewHolder对象。当convertView不为空，重复利用已经创建的view的时候，使用getTag()方法获取绑定的ViewHolder对象，这样就避免了findViewById对控件的层层查询，而是快速定位到控件。

## 2、package com.fortysevendeg.android.swipelistview.sample.activities;

**AsyncTask（在SwipeListViewExampleActivity.java中）**

AsyncTask,是android提供的轻量级的异步类,可以直接继承AsyncTask,在类中实现异步操作,**并**提供接口反馈当前**异步执行的程度**(可以通过接口实现UI进度更新),最后反馈执行的结果给UI主线程.

如下：

adapter.notifyDataSetChanged();//可以在修改适配器绑定的数组后，不用重新刷新Activity，通知Activity更新ListView

AsyncTask定义了三种泛型类型 Params，Progress和Result。

* Params 启动任务执行的输入参数，比如HTTP请求的URL。
* Progress 后台任务执行的百分比。
* Result 后台执行任务最终返回的结果，比如String。

本界面只需要使用result，即建立并返回List<PackageItem>。

使用AsyncTask 异步加载数据最少要重写以下这两个方法：

* doInBackground(Params…) 后台执行，比较耗时的操作都可以放在这里。注意这里不能直接操作UI。此方法在后台线程执行，完成任务的主要工作，通常需要较长的时间。在执行过程中可以 调用publicProgress(Progress…)来更新任务的进度。

本界面在此从手机中读取要输入item的数据，并进行检测，判断软件状态。

* onPostExecute(Result)  相当于Handler 处理UI的方式，在这里面可以使用在doInBackground 得到的结果处理操作UI。 此方法在主线程执行，任务执行的结果作为此方法的参数返回

使用AsyncTask类，以下是几条必须遵守的准则：

* Task的实例必须在UI thread中创建；
* execute方法必须在UI thread中调用；
* 不要手动的调用onPreExecute(), onPostExecute(Result)，doInBackground(Params...), onProgressUpdate(Progress...)这几个方法；
* 该task只能被执行一次，否则多次调用时将会出现异常；

## 3、package com.fortysevendeg.android.swipelistview.sample.utils;

**PreferencesManager**

其内部主要是有一个SharedPreferences类型的变量

很多软件会有配置文件，里面存放这程序运行当中的各个属性值，由于其配置信息并不多，如果采用数据库来存放并不划算，因为数据库连接跟操作等耗时大大影响 了程序的效率，因此我们使用键值这种一一对应的关系来存放这些配置信息。SharedPreferences正是Android中用于实现这中存储方式的 技术。

SharedPreferences的使用非常简单，能够轻松的存放数据和读取数据。SharedPreferences只能保存简单类型的数据，例如，String、int等。一般会将复杂类型的数据转换成Base64编码，然后将转换后的数据以字符串的形式保存在 XML文件中，再用SharedPreferences保存。

本界面中主要用来存储一个boolean类型的变量，用来进行判断，默认为true。

当为true时就加载AboutDialog定义的界面。当在AboutDialog中勾选“Don\'t show this message again”之后变量被存为false，以后就不再加载该界面。

## 五、功能设计

本界面通过SwipeListViewExampleActivity来生成主界面。其中通过读取手机控件的id来切换到设置界面。通过调取SettingsManager来对主界面是否可以左/右滑动等进行设置。通过AsyncTask来控制线程读取手机中的数据建立列表。

各个按钮的功能在PackageAdapter中进行设置，被SwipeListView调用，之后生成主界面时通过调用其函数来加载各个按键的功能，该函数每次点击操作之后都会运行。

设置界面可实时读取设置界面，更改设置数据。

## 六、后期改进（已实现）

通过在item中加入一个新的变量，并在生成数组（列表那个）时对其加载，以实现打开之后自动检测是否是否被注入，且不会出现卡顿。