

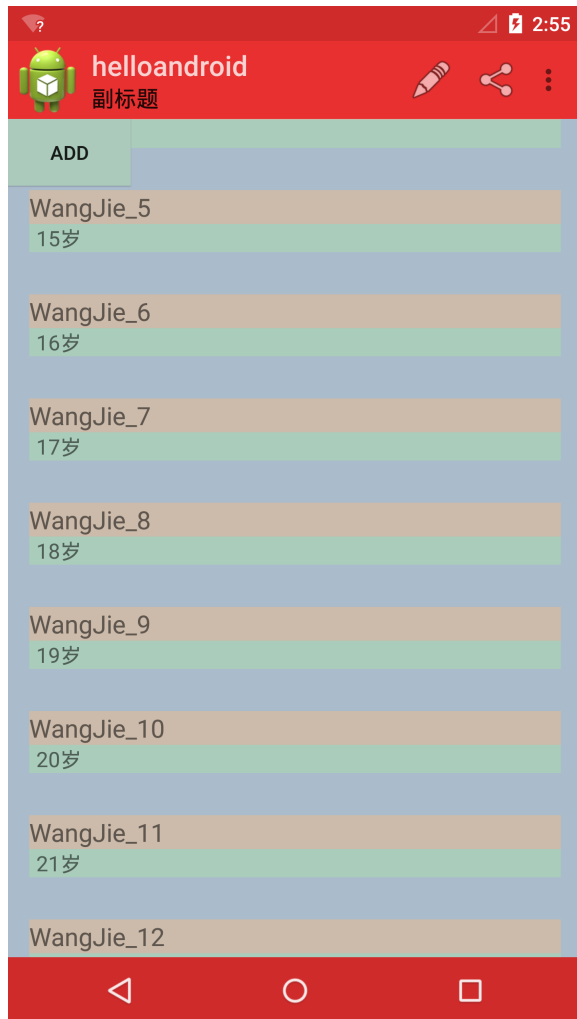
[Android]使用RecyclerView替代ListView（一）

以下内容原创，欢迎转载，转载请注明

来自天天博客：<http://www.cnblogs.com/tiantianbyconan/p/4232560.html>

RecyclerView是一个比ListView更灵活的一个控件，以后可以直接抛弃ListView了。具体好在哪些地方，往下看就知道了。

首先我们来使用RecyclerView来实现ListView的效果，一个滚动列表，先看下效果图（除了有动画之外，没什么特别——）：



每个item的布局如下：



```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/recycler_view_test_item_person_view"
    android:orientation="vertical"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:padding="15dp"
    android:background="#aabbcc"
    >
    <TextView
```

```
        android:id="@+id/recycler_view_test_item_person_name_tv"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:textSize="18sp"
        android:background="#ccbbaa"
    />
    <TextView
        android:id="@+id/recycler_view_test_item_person_age_tv"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:paddingLeft="5dp"
        android:background="#aaccbb"
        android:textSize="15sp"
    />
</LinearLayout>
```



item的布局很简单，只有两个TextView，一个用来显示名字，一个用来显示年龄。

Person的实体类就不贴代码了，两个属性：名字和年龄。

然后需要使用到RecyclerView，所以需要把support v7添加到class path，并在布局中添加该控件：

```
<android.support.v7.widget.RecyclerView
    android:id="@+id/recycler_view_test_rv"
    android:scrollbars="vertical"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="#bbccaa"
/>
```



然后在onCreate中：

```
1    recyclerView.setHasFixedSize(true);
2
3    RecyclerView.LayoutManager layoutManager = new LinearLayoutManager(context);
4    recyclerView.setLayoutManager(layoutManager);
5
6    initData();
7    adapter = new PersonAdapter(personList);
8    adapter.setOnRecyclerViewListener(this);
9    recyclerView.setAdapter(adapter);
```



如上述代码：

Line1: 使RecyclerView保持固定的大小,这样会提高RecyclerView的性能。

Line3: LinearLayoutManager，如果你需要显示的是横向滚动的列表或者竖直滚动的列表，则使用这个LayoutManager。显然，我们要实现的是ListView的效果，所以需要使用它。生成这个LinearLayoutManager之后可以设置他滚动的方向，默认竖直滚动，所以这里没有显式地设置。

Line6: 初始化数据源。

Line7~9: 跟ListView一样, 需要设置RecyclerView的Adapter, 但是这里的Adapter跟ListView使用的Adapter不一样, 这里的Adapter需要继承RecyclerView.Adapter, 需要实现3个方法:

- onCreateViewHolder()
- onBindViewHolder()
- getItemCount()

直接看代码:



```
1 package com.wangjie.helloandroid.sample.recycler.person;
2
3 import android.support.v7.widget.RecyclerView;
4 import android.view.LayoutInflater;
5 import android.view.View;
6 import android.view.ViewGroup;
7 import android.widget.LinearLayout;
8 import android.widget.TextView;
9 import com.wangjie.androidbucket.log.Logger;
10 import com.wangjie.helloandroid.R;
11
12 import java.util.List;
13
14 /**
15  * Author: wangjie
16  * Email: tiantian.china.2@gmail.com
17  * Date: 1/17/15.
18  */
19 public class PersonAdapter extends RecyclerView.Adapter {
20     public static interface OnRecyclerViewListener {
21         void onItemClick(int position);
22         boolean onItemLongClick(int position);
23     }
24
25     private OnRecyclerViewListener onRecyclerViewListener;
26
27     public void setOnRecyclerViewListener(OnRecyclerViewListener onRecyclerViewListener) {
28         this.onRecyclerViewListener = onRecyclerViewListener;
29     }
30
31     private static final String TAG = PersonAdapter.class.getSimpleName();
32     private List<Person> list;
33
34     public PersonAdapter(List<Person> list) {
35         this.list = list;
36     }
37
38     @Override
39     public RecyclerView.ViewHolder onCreateViewHolder(ViewGroup viewGroup, int i) {
40         Logger.d(TAG, "onCreateViewHolder, i: " + i);
41         View view =
42             LayoutInflater.from(viewGroup.getContext()).inflate(R.layout.recycler_view_test_item_person, null);
43         LinearLayout.LayoutParams lp = new
44             LinearLayout.LayoutParams(ViewGroup.LayoutParams.MATCH_PARENT, ViewGroup.LayoutParams.WRAP_CONTENT);
```

```
43     view.setLayoutParams(lp);
44     return new PersonViewHolder(view);
45 }
46
47 @Override
48 public void onBindViewHolder(RecyclerView.ViewHolder viewHolder, int i) {
49     Logger.d(TAG, "onBindViewHolder, i: " + i + ", viewHolder: " + viewHolder);
50     PersonViewHolder holder = (PersonViewHolder) viewHolder;
51     holder.position = i;
52     Person person = list.get(i);
53     holder.nameTv.setText(person.getName());
54     holder.ageTv.setText(person.getAge() + "岁");
55 }
56
57 @Override
58 public int getItemCount() {
59     return list.size();
60 }
61
62 class PersonViewHolder extends RecyclerView.ViewHolder implements View.OnClickListener,
View.OnLongClickListener
63 {
64     public View rootView;
65     public TextView nameTv;
66     public TextView ageTv;
67     public int position;
68
69     public PersonViewHolder(View itemView) {
70         super(itemView);
71         nameTv = (TextView) itemView.findViewById(R.id.recycler_view_test_item_person_name_tv);
72         ageTv = (TextView) itemView.findViewById(R.id.recycler_view_test_item_person_age_tv);
73         rootView = itemView.findViewById(R.id.recycler_view_test_item_person_view);
74         rootView.setOnClickListener(this);
75         rootView.setOnLongClickListener(this);
76     }
77
78     @Override
79     public void onClick(View v) {
80         if (null != onRecyclerViewListener) {
81             onRecyclerViewListener.onItemClick(position);
82         }
83     }
84
85     @Override
86     public boolean onLongClick(View v) {
87         if (null != onRecyclerViewListener) {
88             return onRecyclerViewListener.onItemLongClick(position);
89         }
90         return false;
91     }
92 }
93
94 }
```



如上代码所示:


```
public RecyclerView.ViewHolder onCreateViewHolder(ViewGroup viewGroup, int i)
```

这个方法主要生成为每个Item inflater出一个View，但是该方法返回的是一个ViewHolder。方法是把View直接封装在ViewHolder中，然后我们面向的是ViewHolder这个实例，当然这个ViewHolder需要我们去编写。直接省去了当初的convertView.setTag(holder)和convertView.getTag()这些繁琐的步骤。


```
public void onBindViewHolder(RecyclerView.ViewHolder viewHolder, int i)
```

这个方法主要用于适配渲染数据到View中。方法提供给你一个viewHolder，而不是原来的convertView。

对比下以前的写法就一目了然了：



```
1 @Override
2     public View getView(int position, View convertView, ViewGroup parent) {
3         ViewHolder holder;
4         if(null == convertView){
5             holder = new ViewHolder();
6             LayoutInflater mInflater = (LayoutInflater)
context.getSystemService(Context.LAYOUT_INFLATER_SERVICE);
7             convertView = mInflater.inflate(R.layout.item, null);
8             holder.btn = (Button) convertView.findViewById(R.id.btn);
9             holder.tv = (TextView) convertView.findViewById(R.id.tv);
10            holder.iv = (TextView) convertView.findViewById(R.id.iv);
11
12            convertView.setTag(holder);
13        }else{
14            holder = (ViewHolder) convertView.getTag();
15        }
16        final HashMap<String, Object> map = list.get(position);
17
18        holder.iv.setImageResource(Integer.valueOf(map.get("iv").toString()));
19        holder.tv.setText(map.get("tv").toString());
20
21        holder.btn.setOnClickListener(new View.OnClickListener() {
22            @Override
23            public void onClick(View v) {
24                Toast.makeText(context, map.get("btn").toString(), Toast.LENGTH_SHORT).show();
25            }
26        });
27
28        return convertView;
29    }
30
31    class ViewHolder{
32        Button btn;
33        ImageView iv;
34        TextView tv;
35
36    }
```



对比后可以发现：

旧的写法中Line5~Line12+Line28部分的代码其实起到的作用相当于新的写法的onCreateViewHolder()；

旧的写法中Line14~Line26部分的代码其实起到的作用相当于新的写法的onBindViewHolder()；

既然是这样，那我们就把原来相应的代码搬到对应的onCreateViewHolder()和onBindViewHolder()这两个方法中就可以了。

因为RecyclerView帮我们封装了Holder，所以我们自己写的ViewHolder就需要继承RecyclerView.ViewHolder，只有这样，RecyclerView才能帮你去管理这个ViewHolder类。

既然getView方法的渲染数据部分的代码相当于onBindViewHolder()，所以如果调用adapter.notifyDataSetChanged()方法，应该也会重新调用onBindViewHolder()方法才对吧？实验后，果然如此！

除了adapter.notifyDataSetChanged()这个方法之外，新的Adapter还提供了其他的方法，如下：



```
public final void notifyDataSetChanged()
public final void notifyItemChanged(int position)
public final void notifyItemRangeChanged(int positionStart, int itemCount)
public final void notifyItemInserted(int position)
public final void notifyItemMoved(int fromPosition, int toPosition)
public final void notifyItemRangeInserted(int positionStart, int itemCount)
public final void notifyItemRemoved(int position)
public final void notifyItemRangeRemoved(int positionStart, int itemCount)
```



基本上看到方法的名字就知道这个方法干嘛的了，

第一个方法没什么好讲的，跟以前一样。

notifyItemChanged(int position)，position数据发生了改变，那调用这个方法，就会回调对应position的onBindViewHolder()方法了，当然，因为ViewHolder是复用的，所以如果position在当前屏幕以外，也就不会回调了，因为没有意义，下次position滚动会当前屏幕以内的时候同样会调用onBindViewHolder()方法刷新数据了。其他的方法也是同样的道理。

public final void notifyItemRangeChanged(int positionStart, int itemCount)，顾名思义，可以刷新从positionStart开始itemCount数量的item了（这里的刷新指回调onBindViewHolder()方法）。

public final void notifyItemInserted(int position)，这个方法是在第position位置被插入了一条数据的时候可以使用这个方法刷新，注意这个方法调用后会有插入的动画，这个动画可以使用默认的，也可以自己定义。

public final void notifyItemMoved(int fromPosition, int toPosition)，这个方法是从fromPosition移动到toPosition为止的时候可以使用这个方法刷新

public final void notifyItemRangeInserted(int positionStart, int itemCount)，显然是批量添加。

public final void notifyItemRemoved(int position)，第position个被删除的时候刷新，同样会有动画。

public final void notifyItemRangeRemoved(int positionStart, int itemCount)，批量删除。

这些方法分析完之后，我们来实现一个点击一个按钮，新增一条数据，长按一个item，删除一条数据的场景。

以下是新增一条数据的代码：

```
1 Person person = new Person(i, "WangJie_" + i, 10 + i);
2 adapter.notifyItemInserted(2);
3 personList.add(2, person);
4 adapter.notifyItemRangeChanged(2, adapter.getItemCount());
```

如上代码:

Line2: 表示在position为2的位置, 插入一条数据, 这个时候动画开始执行。

Line3: 表示在数据源中position为2的位置新增一条数据 (其实这个才是真正的新增数据啦)。

Line4: 为什么要刷新position为2以后的数据呢? 因为, 在position为2的位置插入了一条数据后, 新数据的position变成了2, 那原来的position为2的应该变成了3, 3的应该变成了4, 所以2以后的所有数据的position都发生了改变, 所以需要把position2以后的数据都要刷新。理论上是这样, 但是实际上刷新的数量只有在屏幕上显示的position为2以后的数据而已。如果这里使用notifyDataSetChanged()来刷新屏幕上显示的所有item可以吗? 结果不会出错, 但是会有一个问题, 前面调用了notifyItemInserted()方法后会在执行动画, 如果你调用notifyDataSetChanged()刷新屏幕上显示的所有item的话, 必然也会刷新当前正在执行动画的那个item, 这样导致的结果是, 前面的动画还没执行完, 它马上又被刷新了, 动画就看不见了。所以只要刷新2以后的item就可以了。

看了RecyclerView的api, 发现没有setOnItemClickListener——, 所以还是自己把onItemClick从Adapter中回调出来吧。这个很简单, 就像上面PersonAdaper中写的OnRecyclerViewListener那样。

长按删除的代码如下:

```
1 adapter.notifyItemRemoved(position);  
2 personList.remove(position);  
3 adapter.notifyItemRangeChanged(position, adapter.getItemCount());
```

代码跟之前插入的代码基本一致。先通知执行动画, 然后删除数据源中的数据, 然后通知position之后的数据刷新就可以了。

这样ListView的效果就实现了。

示例代码:

<https://github.com/wangjiegulu/RecyclerViewSample>

[\[Android\]使用RecyclerView替代ListView \(二\):](#)

<http://www.cnblogs.com/tiantianbyconan/p/4242541.html>

[\[Android\]使用RecyclerView替代ListView \(三\):](#)

<http://www.cnblogs.com/tiantianbyconan/p/4268097.html>