Android 快速开发系列 打造万能的ListView GridView 适配器

标签： Android BaseAdapterHelper 通过的BaseAdapter
2014-08-28 21:32 45806人阅读 评论(163) 收藏 举报

分类：
【android 进阶之路】（61） 【Android 快速开发】（10）

目录(?) [+]

转载请标明出处：http://blog.csdn.net/lmj623565791/article/details/38902805 ，本文出自【张鸿洋的博客】

1、概述

相信做Android开发的写得最多的就是ListView，GridView的适配器吧，记得以前开发一同事开发项目，一个项目下来基本就一直在写ListView的Adapter都快吐了~~~对于Adapter一般都继承BaseAdapter复写几个方法，getView里面使用ViewHolder模式，其实大部分的代码基本都是类似的。

本篇博客为快速开发系列的第一篇，将一步一步带您封装出一个通用的Adapter。

2、常见的例子

首先看一个最常见的案例，大家一目十行的扫一眼

1、布局文件

主布局文件：

```
[html]
01. <RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
02.     xmlns:tools="http://schemas.android.com/tools"
03.     android:layout_width="match_parent"
04.     android:layout_height="match_parent" >
05. 
06.     <ListView
07.         android:id="@+id/id_lv_main"
08.         android:layout_width="fill_parent"
09.         android:layout_height="fill_parent" />
10. 
11. </RelativeLayout>
```

Item的布局文件：

```
[html]
01. <?xml version="1.0" encoding="utf-8"?>
02. <TextView xmlns:android="http://schemas.android.com/apk/res/android"
03.     android:id="@+id/id_tv_title"
04.     android:layout_width="match_parent"
05.     android:layout_height="50dp"
06.     android:background="#aa111111"
07.     android:gravity="center_vertical"
08.     android:paddingLeft="15dp"
09.     android:textColor="#ffffff"
10.     android:text="hello"
11.     android:textSize="20sp"
12.     android:textStyle="bold" >
13. 
14. </TextView>
```

2、Adapter

```
[java]
01. package com.example.zhy_baseadapterhelper;
02. 
03. import java.util.List;
```

```
04.
05. import android.content.Context;
06. import android.view.LayoutInflater;
07. import android.view.View;
08. import android.view.ViewGroup;
09. import android.widget.BaseAdapter;
10. import android.widget.TextView;
11.
12. public class MyAdapter extends BaseAdapter
13. {
14.     private LayoutInflater mInflater;
15.     private Context mContext;
16.     private List<String> mDatas;
17.
18.     public MyAdapter(Context context, List<String> mDatas)
19.     {
20.         mInflater = LayoutInflater.from(context);
21.         this.mContext = context;
22.         this.mDatas = mDatas;
23.     }
24.
25.     @Override
26.     public int getCount()
27.     {
28.         return mDatas.size();
29.     }
30.
31.     @Override
32.     public Object getItem(int position)
33.     {
34.         return mDatas.get(position);
35.     }
36.
37.     @Override
38.     public long getItemId(int position)
39.     {
40.         return position;
41.     }
42.
43.     @Override
44.     public View getView(int position, View convertView, ViewGroup parent)
45.     {
46.         ViewHolder viewHolder = null;
47.         if (convertView == null)
48.         {
49.             convertView = mInflater.inflate(R.layout.item_single_str, parent,
50.                     false);
51.             viewHolder = new ViewHolder();
52.             viewHolder.mTextView = (TextView) convertView
53.                     .findViewById(R.id.id_tv_title);
54.             convertView.setTag(viewHolder);
55.         } else
56.         {
57.             viewHolder = (ViewHolder) convertView.getTag();
58.         }
59.         viewHolder.mTextView.setText(mDatas.get(position));
60.         return convertView;
61.     }
62.
63.     private final class ViewHolder
64.     {
65.         TextView mTextView;
66.     }
67.
68. }
```

3、Activity

```
[java]
01. package com.example.zhy_baseadapterhelper;
02.
03. import java.util.ArrayList;
04. import java.util.Arrays;
```

```
import java.util.List;

import android.app.Activity;
import android.os.Bundle;
import android.widget.ListView;

public class MainActivity extends Activity
{

    private ListView mListView;
    private List<String> mDatas = new ArrayList<String>(Arrays.asList("Hello",
            "World", "Welcome"));
    private MyAdapter mAdapter;

    @Override
    protected void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        mListView = (ListView) findViewById(R.id.id_lv_main);
        mListView.setAdapter(mAdapter = new MyAdapter(this, mDatas));

    }

}
```

上面这个例子大家应该都写了无数遍了，MyAdapter集成BaseAdapter，然后getView里面使用ViewHolder模式；一般情况下，我们的写法是这样的：对于不同布局的ListView，我们会有一个对应的Adapter，在Adapter中又会有一个ViewHolder类来提高效率。

这样出现ListView就会出现与之对于的Adapter类、ViewHolder类；那么有没有办法减少我们的编码呢？

下面首先拿ViewHolder开刀~

3、通用的ViewHolder

首先分析下ViewHolder的作用，通过convertView.setTag与convertView进行绑定，然后当convertView复用时，直接从与之对于的ViewHolder(getTag)中拿到convertView布局中的控件，省去了findViewById的时间~

也就是说，实际上们每个convertView会绑定一个ViewHolder对象，这个viewHolder主要用于帮convertView存储布局中的控件。

那么我们只要写出一个通用的ViewHolder，然后对于任意的convertView，提供一个对象让其setTag即可；

既然是通用，那么我们这个ViewHolder就不可能含有各种控件的成员变量了，因为每个Item的布局是不同的，最好的方式是什么呢？

提供一个容器，专门存每个Item布局中的所有控件，而且还要能够查找出来；既然需要查找，那么ListView肯定是不行了，需要一个键值对进行保存，键为控件的Id，值为控件的引用，相信大家立刻就能想到Map；但是我们不用Map，因为有更好的替代类，就是我们android提供的SparseArray这个类，和Map类似，但是比Map效率，不过键只能为Integer.

下面看我们的ViewHolder类：

```
[java]
package com.example.zhy_baseadapterhelper;

import android.content.Context;
import android.util.Log;
import android.util.SparseArray;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;

public class ViewHolder
{
    private final SparseArray<View> mViews;
    private View mConvertView;

    private ViewHolder(Context context, ViewGroup parent, int layoutId,
            int position)
    {
        this.mViews = new SparseArray<View>();
```

```
        mConvertView = LayoutInflater.from(context).inflate(layoutId, parent,
                false);
        //setTag
        mConvertView.setTag(this);


    }

    /**
     * 拿到一个ViewHolder对象
     * @param context
     * @param convertView
     * @param parent
     * @param layoutId
     * @param position
     * @return
     */
    public static ViewHolder get(Context context, View convertView,
            ViewGroup parent, int layoutId, int position)
    {

        if (convertView == null)
        {
            return new ViewHolder(context, parent, layoutId, position);
        }
        return (ViewHolder) convertView.getTag();
    }


    /**
     * 通过控件的Id获取对于的控件，如果没有则加入views
     * @param viewId
     * @return
     */
    public <T extends View> T getView(int viewId)
    {

        View view = mViews.get(viewId);
        if (view == null)
        {
            view = mConvertView.findViewById(viewId);
            mViews.put(viewId, view);
        }
        return (T) view;
    }

    public View getConvertView()
    {
        return mConvertView;
    }


}
```

与传统的ViewHolder不同，我们使用了一个SparseArray<View>用于存储与之对于的convertView的所有的控件，当需要拿这些控件时，通过getView(id)进行获取；

下面看使用该ViewHolder的MyAdapter；

```
[java]
@Override
    public View getView(int position, View convertView, ViewGroup parent)
    {
        //实例化一个viewHolder
        ViewHolder viewHolder = ViewHolder.get(mContext, convertView, parent,
                R.layout.item_single_str, position);
        //通过getView获取控件
        TextView tv = viewHolder.getView(R.id.id_tv_title);
        //使用
```

```
        tv.setText(mDatas.get(position));
        return viewHolder.getConvertView();
    }
```

只看getView，其他方法都一样；首先调用ViewHolder的get方法，如果convertView为null，new一个ViewHolder实例，通过使用mInflater.inflate加载布局，然后new一个SparseArray用于存储View，最后setTag(this)；

如果存在那么直接getTag

最后通过getView(id)获取控件，如果存在则直接返回，否则调用findViewById，返回存储，返回。

好了，一个通用的ViewHolder写好了，以后一个项目几十个Adapter一个ViewHolder直接hold住全场~~大家可以省点时间斗个小地主了~~

4、打造通用的Adapter

有了通用的ViewHolder大家肯定不能满足，怎么也得省出dota的时间，人在塔在~~

下面看如何打造一个通过的Adapter，我们叫做CommonAdapter

继续分析，Adapter一般需要保持一个List对象，存储一个Bean的集合，不同的ListView，Bean肯定是不同的，这个CommonAdapter肯定需要支持泛型，内部维持一个List<T>，就解决我们的问题了；

于是我们初步打造我们的CommonAdapter

```
[java]
package com.example.zhy_baseadapterhelper;

import java.util.List;

import android.content.Context;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.BaseAdapter;
import android.widget.TextView;

public abstract class CommonAdapter<T> extends BaseAdapter
{
    protected LayoutInflater mInflater;
    protected Context mContext;
    protected List<T> mDatas;

    public CommonAdapter(Context context, List<T> mDatas)
    {
        mInflater = LayoutInflater.from(context);
        this.mContext = context;
        this.mDatas = mDatas;
    }

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

    @Override
    public Object getItem(int position)
    {
        return mDatas.get(position);
    }

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

}
```

我们的CommonAdapter依然是一个抽象类，除了getView以外我们把其他的代码都实现了，这样的话，在使用我们的Adapter只要实现一个

getView，然后getView里面再使用我们打造的通过的ViewHolder是不是感觉还不错~

现在我们的MyAdapter是这样的：

```
[java]
01. package com.example.zhy_baseadapterhelper;
02.
03. import java.util.List;
04.
05. import android.content.Context;
06. import android.view.View;
07. import android.view.ViewGroup;
08. import android.widget.TextView;
09.
10. public class MyAdapter<T> extends CommonAdapter<T>
11. {
12.     public MyAdapter(Context context, List<T> mDatas)
13.     {
14.         super(context, mDatas);
15.     }
16.
17.     @Override
18.     public View getView(int position, View convertView, ViewGroup parent)
19.     {
20.         ViewHolder viewHolder = ViewHolder.get(mContext, convertView, parent,
21.                 R.layout.item_single_str, position);
22.         TextView mTitle = viewHolder.getView(R.id.id_tv_title);
23.         mTitle.setText((String) mDatas.get(position));
24.         return viewHolder.getConvertView();
25.     }
26.
27. }
```

所有的代码加起来也就10行左右，是不是神清气爽~~稍等，我先去dota一把~

但是我们是否就这样满足了呢？显然还可以简化。

5、进一步铸造

注意我们的getView里面的代码，虽然只有4行，但是我觉得所有的Adapter的

第一行（ViewHolder viewHolder = getViewHolder(position, convertView,parent);）和

最后一行：return viewHolder.getConvertView();一定是一样的。

那么我们可以这样做：我们把第一行和最后一行写死，把中间变化的部分抽取出来，这不就是OO的设计原则嘛。现在CommonAdapter是这样的：

```
[java]
01. package com.example.zhy_baseadapterhelper;
02.
03. import java.util.List;
04.
05. import android.content.Context;
06. import android.view.LayoutInflater;
07. import android.view.View;
08. import android.view.ViewGroup;
09. import android.widget.BaseAdapter;
10.
11. public abstract class CommonAdapter<T> extends BaseAdapter
12. {
13.     protected LayoutInflater mInflater;
14.     protected Context mContext;
15.     protected List<T> mDatas;
16.     protected final int mItemLayoutId;
17.
18.     public CommonAdapter(Context context, List<T> mDatas, int itemLayoutId)
19.     {
20.         this.mContext = context;
21.         this.mInflater = LayoutInflater.from(mContext);
```

```
        this.mDatas = mDatas;
        this.mItemLayoutId = itemLayoutId;
    }

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

    @Override
    public T getItem(int position)
    {
        return mDatas.get(position);
    }

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

    @Override
    public View getView(int position, View convertView, ViewGroup parent)
    {
        final ViewHolder viewHolder = getViewHolder(position, convertView,
                parent);
        convert(viewHolder, getItem(position));
        return viewHolder.getConvertView();

    }

    public abstract void convert(ViewHolder helper, T item);

    private ViewHolder getViewHolder(int position, View convertView,
            ViewGroup parent)
    {
        return ViewHolder.get(mContext, convertView, parent, mItemLayoutId,
                position);
    }

}
```

对外公布了一个convert方法，并且还把viewHolder和本Item对于的Bean对象给传出去，现在convert方法里面需要干嘛呢？

通过ViewHolder把View找到，通过Item设置值；

现在我觉得代码简化到这样，我已经不需要单独写一个Adapter了，直接MainActivity匿名内部类走起~

[java]

```
@Override
    protected void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        mListView = (ListView) findViewById(R.id.id_lv_main);

        //设置适配器
        mListView.setAdapter(mAdapter = new CommonAdapter<String>(
                getApplicationContext(), mDatas, R.layout.item_single_str)
        {
            @Override
            public void convert(ViewHolder c, String item)
            {
                TextView view = viewHolder.getView(R.id.id_tv_title);
                view.setText(item);
            }

        });

    }
```

可以看到效果咋样，不错吧。你觉得还能简化么？我觉得还能改善。

6、Adapter最后的封魔

我们现在在convertView里面需要这样:

```
@Override
public void convert(ViewHolder viewHolder, String item)
{
    TextView view = viewHolder.getView(R.id.id_tv_title);
    view.setText(item);
}
```

我们细想一下，其实布局里面的View常用也就那么几种：ImageView,TextView,Button,CheckBox等等；

那么我觉得ViewHolder还可以封装一些常用的方法，比如setText(id,String)；setImageResource(viewId, resId)；

setImageBitmap(viewId, bitmap)；

那么现在ViewHolder是：

[java]

```
package com.example.zhy_baseadapterhelper;

import android.content.Context;
import android.graphics.Bitmap;
import android.util.SparseArray;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.ImageView;
import android.widget.TextView;

import com.example.zhy_baseadapterhelper.ImageLoader.Type;

public class ViewHolder
{
    private final SparseArray<View> mViews;
    private int mPosition;
    private View mConvertView;

    private ViewHolder(Context context, ViewGroup parent, int layoutId,
            int position)
    {
        this.mPosition = position;
        this.mViews = new SparseArray<View>();
        mConvertView = LayoutInflater.from(context).inflate(layoutId, parent,
                false);
        // setTag
        mConvertView.setTag(this);
    }

    /**
     * 拿到一个ViewHolder对象
     *
     * @param context
     * @param convertView
     * @param parent
     * @param layoutId
     * @param position
     * @return
     */
    public static ViewHolder get(Context context, View convertView,
            ViewGroup parent, int layoutId, int position)
    {
        if (convertView == null)
        {
            return new ViewHolder(context, parent, layoutId, position);
```

```
        }
        return (ViewHolder) convertView.getTag();
    }

    public View getConvertView()
    {
        return mConvertView;
    }

    /**
     * 通过控件的Id获取对于的控件，如果没有则加入views
     *
     * @param viewId
     * @return
     */
    public <T extends View> T getView(int viewId)
    {
        View view = mViews.get(viewId);
        if (view == null)
        {
            view = mConvertView.findViewById(viewId);
            mViews.put(viewId, view);
        }
        return (T) view;
    }

    /**
     * 为TextView设置字符串
     *
     * @param viewId
     * @param text
     * @return
     */
    public ViewHolder setText(int viewId, String text)
    {
        TextView view = getView(viewId);
        view.setText(text);
        return this;
    }

    /**
     * 为ImageView设置图片
     *
     * @param viewId
     * @param drawableId
     * @return
     */
    public ViewHolder setImageResource(int viewId, int drawableId)
    {
        ImageView view = getView(viewId);
        view.setImageResource(drawableId);

        return this;
    }

    /**
     * 为ImageView设置图片
     *
     * @param viewId
     * @param drawableId
     * @return
     */
    public ViewHolder setImageBitmap(int viewId, Bitmap bm)
    {
        ImageView view = getView(viewId);
        view.setImageBitmap(bm);
        return this;
    }

    /**
     * 为ImageView设置图片
     *
     * @param viewId
     * @param drawableId
     * @return
     */
```

```
123.     public ViewHolder setImageByUrl(int viewId, String url)
124.     {
125.         ImageLoader.getInstance(3, Type.LIFO).loadImage(url,
126.                 (ImageView) getView(viewId));
127.         return this;
128.     }
129. 
130.     public int getPosition()
131.     {
132.         return mPosition;
133.     }
134. 
135. }
```

现在的MainActivity只需要这么写：

```
[java]
01. mAdapter = new CommonAdapter<String>(getApplicationContext(),
02.                 R.layout.item_single_str, mDatas)
03.         {
04.             @Override
05.             protected void convert(ViewHolder viewHolder, String item)
06.             {
07.                 viewHolder.setText(R.id.id_tv_title, item);
08.             }
09.         };
```

convertView里面只要一行代码了~~~

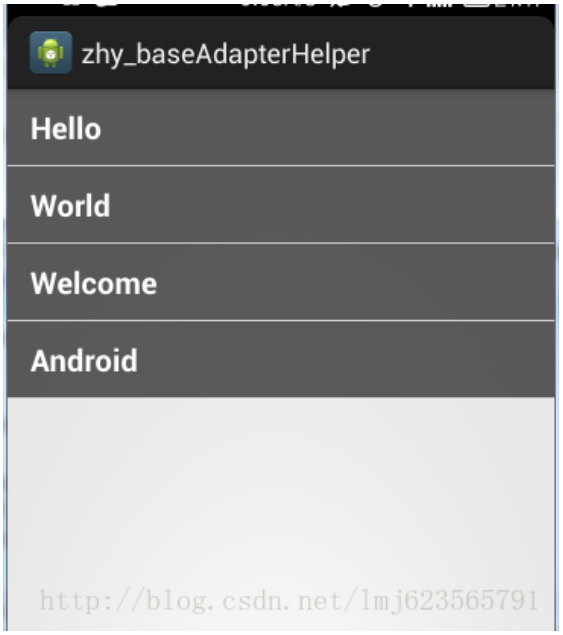
好了，到此我们的通用的Adapter已经一步一步铸造完毕~咋样，以后写项目省下来的时间是不是可以陪我切磋dota了（ps:11昵称：血魔哥404）~~

注：关于ViewHolder里面的setText，setImageResource这类的方法，大家可以在使用的过程中不断的完善，今天发现这个控件可以这么设置值，好，放进去；时间长了，基本就完善了。还有那个ImageLoader是我另一篇博客里的，大家可以使用UIL，Volley或者自己写个图片加载器；

7、实践

说了这么多，还是得拿出来让我们的实践检验检验，顺便来几张套图，俗话说，没图没正相。

1、我们的实例代码的图是这样的：



关于Adapter和ViewHolder的代码是这样的：

```java
// 设置适配器
        mListView.setAdapter(mAdapter = new CommonAdapter<String>(
                getApplicationContext(), mDatas, R.layout.item_single_str)
        {
            @Override
            public void convert(ViewHolder helper, String item)
            {
                helper.setText(R.id.id_tv_title,item);
            }

        });
```


哎哟，我是不是只要贴一行；

2、来个复杂点的布局

```html
<?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="wrap_content"
    android:background="#ffffff"
    android:orientation="vertical"
    android:padding="10dp" >

    <TextView
        android:id="@+id/tv_title"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:singleLine="true"
        android:text="红色钱包"
        android:textSize="16sp"
        android:textColor="#444444" >
    </TextView>

    <TextView
        android:id="@+id/tv_describe"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_below="@id/tv_title"
        android:layout_marginTop="10dp"
        android:maxLines="2"
        android:minLines="1"
        android:text="周三早上丢失了红色钱包，在食堂二楼"
        android:textColor="#898989"
        android:textSize="16sp" >
    </TextView>

    <TextView
        android:id="@+id/tv_time"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_below="@id/tv_describe"
        android:layout_marginTop="10dp"
        android:text="20130240122"
        android:textColor="#898989"
        android:textSize="12sp" >
    </TextView>

    <TextView
        android:id="@+id/tv_phone"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentRight="true"
        android:layout_below="@id/tv_describe"
        android:layout_marginTop="10dp"
        android:background="#5cbe6c"
        android:drawableLeft="@drawable/icon_photo"
        android:drawablePadding="5dp"
        android:paddingBottom="3dp"
        android:paddingLeft="5dp"
```

```
        android:paddingRight="5dp"
        android:paddingTop="3dp"
        android:text="138024249542"
        android:textColor="#ffffff"
        android:textSize="12sp" >
    </TextView>

</RelativeLayout>
```

效果图是这样的：

布局是不是挺复杂的了~~

但是代码是这样的：

[java]

```
// 设置适配器
        mListView.setAdapter(mAdapter = new CommonAdapter<Bean>(
                getApplicationContext(), mDatas, R.layout.item_list)
        {
            @Override
            public void convert(ViewHolder helper, Bean item)
            {
                helper.setText(R.id.tv_title, item.getTitle());
                helper.setText(R.id.tv_describe, item.getDesc());
                helper.setText(R.id.tv_phone, item.getPhone());
                helper.setText(R.id.tv_time, item.getTime());

//              helper.getView(R.id.tv_title).setOnClickListener(l)
            }

        });
```

从一个字符串的布局到这样的布局，Adapter加ViewHolder的改变就这么多，加起来3行左右代码~~~

到此，Android 快速开发系列 打造万能的ListView GridView 适配器结束；

最后给大家推荐一个gitHub项目：https://github.com/JoanZapata/base-adapter-helper ,这个项目所做的，和我上面写的基本一致。

还有上面的布局文件来自网络，感谢Bmob的提供~

好了，我要去快乐的玩耍了~~

以下为最新更新==>

添加了多种Item类型的支持，源码地址：https://github.com/hongyangAndroid/base-adapter .