

2.5.0 构建一个可复用的自定义BaseAdapter

分类 [Android 基础入门教程](#)

Android 基础入门教程(Q群号 : 153836263)

本节引言：

如题，本节给大家带来的是构建一个可复用的自定义BaseAdapter，我们每每涉及到ListView GridView等其他的Adapter控件，都需要自己另外写一个BaseAdapter类，这样显得非常麻烦，又比如，我们想在一个界面显示两个ListView的话，我们也是需要些两个BaseAdapter... 这，程序员都是喜欢偷懒的哈，这节我们就来写一个可复用的自定义BaseAdapter类~

1.我们一点点开始改：

首先我们把上节写的自定义BaseAdapter贴下，等下我们就要对他进行升级改造

```
/**
 * Created by Jay on 2015/9/21 0021.
 */
public class MyAdapter extends BaseAdapter {

    private Context mContext;
    private LinkedList<Data> mData;

    public MyAdapter() {
    }

    public MyAdapter(LinkedList<Data> mData, Context mContext)
    {
        this.mData = mData;
        this.mContext = mContext;
    }

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

1.0 Android基础入门教程

1.0.1 2015年最新Android基...

1.1 背景相关与系统架构分析

1.2 开发环境搭建

1.2.1 使用Eclipse + ADT + S...

1.2.2 使用Android Studio开...

1.3 SDK更新不了问题解决

1.4 Genymotion模拟器安装

1.5.1 Git使用教程之本地仓...

1.5.2 Git之使用GitHub搭建...

1.6 .9(九妹)图片怎么玩

1.7 界面原型设计

1.8 工程相关解析(各种文件...

1.9 Android程序签名打包

1.11 反编译APK获取代码&...

2.1 View与ViewGroup的概念

2.2.1 LinearLayout(线性布局)

2.2.2 RelativeLayout(相对布...

2.2.3 TableLayout(表格布局)

2.2.4 FrameLayout(帧布局)

2.2.5 GridLayout(网格布局)

2.2.6 AbsoluteLayout(绝对...

2.3.1 TextView(文本框)详解

2.3.2 EditText(输入框)详解

2.3.3 Button(按钮)与ImageB...

2.3.4 ImageView(图像视图)

2.3.5.RadioButton(单选按钮...

2.3.6 开关按钮ToggleButton...

2.3.7 ProgressBar(进度条)

2.3.8 SeekBar(拖动条)

2.3.9 RatingBar(星级评分条)

2.4.1 ScrollView(滚动条)

```

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

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

@Override
public View getView(int position, View convertView, ViewGroup parent) {
    ViewHolder holder = null;
    if (convertView == null) {
        convertView = LayoutInflater.from(mContext).inflate(
            R.layout.item_list, parent, false);
        holder = new ViewHolder();
        holder.img_icon = (ImageView) convertView.findViewById(R.id.img_icon);
        holder.txt_content = (TextView) convertView.findViewById(R.id.txt_content);
        convertView.setTag(holder);
    } else {
        holder = (ViewHolder) convertView.getTag();
    }
    holder.img_icon.setImageResource(mData.get(position).getImgId());
    holder.txt_content.setText(mData.get(position).getContent());
    return convertView;
}

//添加一个元素
public void add(Data data) {
    if (mData == null) {
        mData = new LinkedList<>();
    }
    mData.add(data);
    notifyDataSetChanged();
}

//往特定位置，添加一个元素
public void add(int position, Data data) {
    if (mData == null) {
        mData = new LinkedList<>();
    }
    mData.add(position, data);
    notifyDataSetChanged();
}

public void remove(Data data) {
    if (mData != null) {
        mData.remove(data);
    }
}

```

2.4.2 Date & Time组件(上)

2.4.3 Date & Time组件(下)

2.4.4 Adapter基础讲解

2.4.5 ListView简单实用

2.4.6 BaseAdapter优化

2.4.7 ListView的焦点问题

2.4.8 ListView之checkbox错...

2.4.9 ListView的数据更新问题

2.5.0 构建一个可复用的自定义BaseAdapter

2.5.1 ListView Item多布局的...

2.5.2 GridView(网格视图)的...

2.5.3 Spinner(列表选项框)...

2.5.4 AutoCompleteTextView...

2.5.5 ExpandableListView(...

2.5.6 ViewPager(翻转视图)...

2.5.7 Toast(吐司)的基本使用

2.5.8 Notification(状态栏通...

2.5.9 AlertDialog(对话框)详解

2.6.0 其他几种常用对话框基...

2.6.1 PopupWindow(悬浮框)...

2.6.2 菜单(Menu)

2.6.3 ViewPager的简单使用

2.6.4 DrawerLayout(官方侧...

3.1.1 基于监听的事件处理机制

3.2 基于回调的事件处理机制

3.3 Handler消息传递机制浅析

3.4 TouchListener PK OnTouchListener

3.5 监听EditText的内容变化

3.6 响应系统设置的事件(CoordinatorLayout)

3.7 AsyncTask异步任务

3.8 Gestures(手势)

4.1.1 Activity初学乍练

4.1.2 Activity初窥门径

4.1.3 Activity登堂入室

4.2.1 Service初涉

4.2.2 Service进阶

4.2.3 Service精通

4.3.1 BroadcastReceiver牛...

4.3.2 BroadcastReceiver庖...

4.4.1 ContentProvider初探

```
    }
    notifyDataSetChanged();
}

public void remove(int position) {
    if(mData != null) {
        mData.remove(position);
    }
    notifyDataSetChanged();
}

public void clear() {
    if(mData != null) {
        mData.clear();
    }
    notifyDataSetChanged();
}

private class ViewHolder {
    ImageView img_icon;
    TextView txt_content;
}
}
```

升级1：将Entity设置成泛型

好的，毕竟我们传递过来的Entity实体类可能千奇百怪，比如有Person，Book，Wether等，所以我们将Entity设置成泛型，修改后的代码如下：

```
<pre>
public class MyAdapter<T> extends BaseAdapter {

    private Context mContext;
    private LinkedList<T> mData;

    public MyAdapter() {
    }

    public MyAdapter(LinkedList<T> mData, Context mContext) {
        this.mData = mData;
        this.mContext = mContext;
    }

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

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

[4.4.2 ContentProvider再探...](#)[4.5.1 Intent的基本使用](#)[4.5.2 Intent之复杂数据的传递](#)[5.1 Fragment基本概述](#)[5.2.1 Fragment实例精讲—...](#)[5.2.2 Fragment实例精讲—...](#)[5.2.3 Fragment实例精讲—...](#)[5.2.4 Fragment实例精讲—...](#)[5.2.5 Fragment实例精讲—...](#)[6.1 数据存储与访问之——文...](#)[6.2 数据存储与访问之——S...](#)[6.3.1 数据存储与访问之——...](#)[6.3.2 数据存储与访问之——...](#)[7.1.1 Android网络编程要学...](#)[7.1.2 Android Http请求头与...](#)[7.1.3 Android HTTP请求方...](#)[7.1.4 Android HTTP请求方...](#)[7.2.1 Android XML数据解析](#)[7.2.2 Android JSON数据解析](#)[7.3.1 Android 文件上传](#)[7.3.2 Android 文件下载（1）](#)[7.3.3 Android 文件下载（2）](#)[7.4 Android 调用WebService](#)[7.5.1 WebView\(网页视图\)基...](#)[7.5.2 WebView和JavaScip...](#)[7.5.3 Android 4.4后WebVie...](#)[7.5.4 WebView文件下载](#)[7.5.5 WebView缓存问题](#)[7.5.6 WebView处理网页返...](#)[7.6.1 Socket学习网络基础准备](#)[7.6.2 基于TCP协议的Socket...](#)[7.6.3 基于TCP协议的Socket...](#)[7.6.4 基于UDP协议的Socke...](#)[8.1.1 Android中的13种Draw...](#)[8.1.2 Android中的13种Draw...](#)[8.1.3 Android中的13种Draw...](#)[8.2.1 Bitmap\(位图\)全解析 P...](#)[8.2.2 Bitmap引起的OOM问题](#)[8.3.1 三个绘图工具类详解](#)[8.3.2 绘图类实战示例](#)

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

@Override
public View getView(int position, View convertView, ViewGroup parent) {
    ViewHolder holder = null;
    if (convertView == null) {
        convertView = LayoutInflater.from(mContext).inflate(
            R.layout.item_list, parent, false);
        holder = new ViewHolder();
        holder.img_icon = (ImageView) convertView.findViewById(R.id.img_icon);
        holder.txt_content = (TextView) convertView.findViewById(R.id.txt_content);
        convertView.setTag(holder);
    } else {
        holder = (ViewHolder) convertView.getTag();
    }
    holder.img_icon.setImageResource(mData.get(position).getImgId());
    holder.txt_content.setText(mData.get(position).getContent());
    return convertView;
}

//添加一个元素
public void add(T data) {
    if (mData == null) {
        mData = new LinkedList<>();
    }
    mData.add(data);
    notifyDataSetChanged();
}

//往特定位置，添加一个元素
public void add(int position, T data) {
    if (mData == null) {
        mData = new LinkedList<>();
    }
    mData.add(position, data);
    notifyDataSetChanged();
}

public void remove(T data) {
    if (mData != null) {
        mData.remove(data);
    }
    notifyDataSetChanged();
}

public void remove(int position) {
```

8.3.3 Paint API之—— Mask...

8.3.4 Paint API之—— Xferm...

8.3.5 Paint API之—— Xferm...

8.3.6 Paint API之—— Xferm...

8.3.7 Paint API之—— Xferm...

8.3.8 Paint API之—— Xferm...

8.3.9 Paint API之—— Color...

8.3.10 Paint API之—— Colo...

8.3.11 Paint API之—— Colo...

8.3.12 Paint API之—— Path...

8.3.13 Paint API之—— Sha...

8.3.14 Paint几个枚举/常量值...

8.3.15 Paint API之——Type...

8.3.16 Canvas API详解(Part 1)

8.3.17 Canvas API详解(Part...

8.3.18 Canvas API详解(Part...

8.4.1 Android动画合集之帧...

8.4.2 Android动画合集之补...

8.4.3 Android动画合集之属...

8.4.4 Android动画合集之属...

9.1 使用SoundPool播放音...

9.2 MediaPlayer播放音频与...

9.3 使用Camera拍照

9.4 使用MediaRecord录音

10.1 TelephonyManager(电...

10.2 SmsManager(短信管理...

10.3 AudioManager(音频管...

10.4 Vibrator(振动器)

10.5 AlarmManager(闹钟服务)

10.6 PowerManager(电源服...

10.7 WindowManager(窗口...

10.8 LayoutInflater(布局服务)

10.9 WallpaperManager(壁...

10.10 传感器专题(1)——相...

10.11 传感器专题(2)——方...

10.12 传感器专题(3)——加...

10.12 传感器专题(4)——其...

10.14 Android GPS初涉

11.0 《2015最新Android基...

```

        if(mData != null) {
            mData.remove(position);
        }
        notifyDataSetChanged();
    }

    public void clear() {
        if(mData != null) {
            mData.clear();
        }
        notifyDataSetChanged();
    }

    private class ViewHolder {
        ImageView img_icon;
        TextView txt_content;
    }
}

```

好的，上面我们做的事仅仅是将Data类型换成了泛型T！

升级2：ViewHolder类的升级改造：

我们先来看看前面我们的ViewHolder干了什么？答：

findViewById，设置控件状态；下面我们想在完成这个基础上，将getView()方法大部分的逻辑写到ViewHolder类里，这个ViewHolder要做的事：

定义一个查找控件的方法，我们的思路是通过暴露公共的方法，调用方法时传递过来 控件id，以及设置的内容，比如TextView设置文本：public ViewHolder setText(int id, CharSequence text){文本设置}

将convertView复用部分搬到这里，那就需要传递一个context对象了，我们把需要获取 的部分都写到构造方法中！

写一堆设置方法(public)，比如设置文字大小颜色，图片背景等！

好的，接下来我们就来一步步改造我们的ViewHolder类

1) 相关参数与构造方法：

```

public static class ViewHolder {

    private SparseArray<View> mViews;    //存储ListView 的 item中
    的View

    private View item;                  //存放convertView
    private int position;                //游标
    private Context context;            //Context上下文

    //构造方法，完成相关初始化
    private ViewHolder(Context context, ViewGroup parent, int layoutRes) {
        mViews = new SparseArray<>();
    }
}

```

```

        this.context = context;

        View convertView = LayoutInflater.from(context).inflate
(layoutRes, parent, false);
        convertView.setTag(this);
        item = convertView;
    }

    ImageView img_icon;
    TextView txt_content;
}

```

2) 绑定ViewHolder与Item

在上面的基础上我们再添加一个绑定的方法

```

//绑定ViewHolder与item
public static ViewHolder bind(Context context, View convertView
, ViewGroup parent,
                                int layoutRes, int position) {
    ViewHolder holder;
    if(convertView == null) {
        holder = new ViewHolder(context, parent, layoutRes);
    } else {
        holder = (ViewHolder) convertView.getTag();
        holder.item = convertView;
    }
    holder.position = position;
    return holder;
}

```

3) 根据id获取集合中保存的控件

```

public <T extends View> T getView(int id) {
    T t = (T) mViews.get(id);
    if(t == null) {
        t = (T) item.findViewById(id);
        mViews.put(id, t);
    }
    return t;
}

```

4) 接着我们再定义一堆暴露出来的方法

```

/**
 * 获取当前条目
 */
public View getItemView() {
    return item;
}

/**

```

```
    * 获取条目位置
    */

    public int getItemPosition() {
        return position;
    }

    /**
     * 设置文字
     */
    public ViewHolder setText(int id, CharSequence text) {
        View view = getView(id);
        if (view instanceof TextView) {
            ((TextView) view).setText(text);
        }
        return this;
    }

    /**
     * 设置图片
     */
    public ViewHolder setImageResource(int id, int drawableRes) {
        View view = getView(id);
        if (view instanceof ImageView) {
            ((ImageView) view).setImageResource(drawableRes);
        } else {
            view.setBackgroundResource(drawableRes);
        }
        return this;
    }

    /**
     * 设置点击监听
     */
    public ViewHolder setOnClickListener(int id, View.OnClickListener listener) {
        getView(id).setOnClickListener(listener);
        return this;
    }

    /**
     * 设置可见
     */
    public ViewHolder setVisibility(int id, int visible) {
        getView(id).setVisibility(visible);
        return this;
    }

    /**
     * 设置标签
     */
    public ViewHolder setTag(int id, Object obj) {
        getView(id).setTag(obj);
    }
}
```

```
        return this;
    }

    //其他方法可自行扩展
```

好的，ViewHolder的改造升级完成~

升级3：定义一个抽象方法，完成ViewHolder与Data数据集的绑定

```
public abstract void bindView(ViewHolder holder, T obj);
```

我们创建新的BaseAdapter的时候，实现这个方法就好，另外，别忘了把我们自定义的BaseAdapter改成abstract抽象的！

升级4：修改getView()部分的内容

```
@Override
public View getView(int position, View convertView, ViewGroup parent) {
    ViewHolder holder = ViewHolder.bind(parent.getContext(), convertView, parent, mLayoutRes, position);
    bindView(holder, getItem(position));
    return holder.getItemView();
}
```

2.升级完毕，我们写代码来体验下：

我们要实现的效果图：



就是上面有两个列表，布局不一样，但是我只使用一个BaseAdapter类来完成上述效果！

关键代码如下：

MainActivity.java：

```
public class MainActivity extends AppCompatActivity {

    private Context mContext;
    private ListView list_book;
    private ListView list_app;

    private MyAdapter<App> myAdapter1 = null;
    private MyAdapter<Book> myAdapter2 = null;
    private List<App> mData1 = null;
    private List<Book> mData2 = null;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        mContext = MainActivity.this;
        init();
    }

    private void init() {
```

```

list_book = (ListView) findViewById(R.id.list_book);
list_app = (ListView) findViewById(R.id.list_app);

//数据初始化
mData1 = new ArrayList<App>();
mData1.add(new App(R.mipmap.iv_icon_baidu,"百度"));
mData1.add(new App(R.mipmap.iv_icon_douban,"豆瓣"));
mData1.add(new App(R.mipmap.iv_icon_zhifubao,"支付宝"));

;

mData2 = new ArrayList<Book>();
mData2.add(new Book("《第一行代码Android》","郭霖"));
mData2.add(new Book("《Android群英传》","徐宜生"));
mData2.add(new Book("《Android开发艺术探索》","任玉刚"));

//Adapter初始化
myAdapter1 = new MyAdapter<App>((ArrayList)mData1,R.lay
out.item_one) {
    @Override
    public void bindView(ViewHolder holder, App obj) {
        holder.setImageResource(R.id.img_icon,obj.getIconResId());
        holder.setText(R.id.txt_aname,obj.getName());
    }
};
myAdapter2 = new MyAdapter<Book>((ArrayList)mData2,R.layout.item_two) {
    @Override
    public void bindView(ViewHolder holder, Book obj) {
        holder.setText(R.id.txt_bname,obj.getbName());
        holder.setText(R.id.txt_bauthor,obj.getbAuthor());
    }
};

//ListView设置下Adapter:
list_book.setAdapter(myAdapter2);
list_app.setAdapter(myAdapter1);

}

}

```

我们写的可复用的BaseAdapter的使用就如上面所述~

3.代码示例下载：

[ListViewDemo4.zip](#)

贴下最后写好的MyAdapter类吧，可根据自己的需求进行扩展：

MyAdapter.java：

```
/**
 * Created by Jay on 2015/9/22 0022.
 */
public abstract class MyAdapter<T> extends BaseAdapter {

    private ArrayList<T> mData;
    private int mLayoutRes;          //布局id

    public MyAdapter() {
    }

    public MyAdapter(ArrayList<T> mData, int mLayoutRes) {
        this.mData = mData;
        this.mLayoutRes = mLayoutRes;
    }

    @Override
    public int getCount() {
        return mData != null ? mData.size() : 0;
    }

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

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

    @Override
    public View getView(int position, View convertView, ViewGroup
up parent) {
        ViewHolder holder = ViewHolder.bind(parent.getContext()
, convertView, parent, mLayoutRes
        , position);
        bindView(holder, getItem(position));
        return holder.getItemView();
    }

    public abstract void bindView(ViewHolder holder, T obj);

    //添加一个元素
    public void add(T data) {
        if (mData == null) {
            mData = new ArrayList<>();
        }
        mData.add(data);
        notifyDataSetChanged();
    }

    //往特定位置，添加一个元素
```

```

public void add(int position, T data) {
    if (mData == null) {
        mData = new ArrayList<>();
    }
    mData.add(position, data);
    notifyDataSetChanged();
}

public void remove(T data) {
    if (mData != null) {
        mData.remove(data);
    }
    notifyDataSetChanged();
}

public void remove(int position) {
    if (mData != null) {
        mData.remove(position);
    }
    notifyDataSetChanged();
}

public void clear() {
    if (mData != null) {
        mData.clear();
    }
    notifyDataSetChanged();
}

public static class ViewHolder {

    private SparseArray<View> mViews;    //存储ListView 的 item中的View
    private View item;                  //存放convertView
    private int position;                //游标
    private Context context;            //Context上下文

    //构造方法，完成相关初始化
    private ViewHolder(Context context, ViewGroup parent, int layoutRes) {
        mViews = new SparseArray<>();
        this.context = context;
        View convertView = LayoutInflater.from(context).inflate(layoutRes, parent, false);
        convertView.setTag(this);
        item = convertView;
    }

    //绑定ViewHolder与item
    public static ViewHolder bind(Context context, View convertView, ViewGroup parent,
                                   int layoutRes, int position) {

```

```
ViewHolder holder;
if (convertView == null) {
    holder = new ViewHolder(context, parent, layout
Res);
} else {
    holder = (ViewHolder) convertView.getTag();
    holder.item = convertView;
}
holder.position = position;
return holder;
}

@SuppressWarnings("unchecked")
public <T extends View> T getView(int id) {
    T t = (T) mViews.get(id);
    if (t == null) {
        t = (T) item.findViewById(id);
        mViews.put(id, t);
    }
    return t;
}

/**
 * 获取当前条目
 */
public View getItemView() {
    return item;
}

/**
 * 获取条目位置
 */
public int getItemPosition() {
    return position;
}

/**
 * 设置文字
 */
public ViewHolder setText(int id, CharSequence text) {
    View view = getView(id);
    if (view instanceof TextView) {
        ((TextView) view).setText(text);
    }
    return this;
}

/**
 * 设置图片
 */
public ViewHolder setImageResource(int id, int drawable
Res) {
    View view = getView(id);
```



反馈

```
        if (view instanceof ImageView) {
            ((ImageView) view).setImageResource(drawableRes);
        } else {
            view.setBackgroundResource(drawableRes);
        }
        return this;
    }

    /**
     * 设置点击监听
     */
    public ViewHolder setOnClickListener(int id, View.OnClickListener listener) {
        getView(id).setOnClickListener(listener);
        return this;
    }

    /**
     * 设置可见
     */
    public ViewHolder setVisibility(int id, int visible) {
        getView(id).setVisibility(visible);
        return this;
    }

    /**
     * 设置标签
     */
    public ViewHolder setTag(int id, Object obj) {
        getView(id).setTag(obj);
        return this;
    }

    //其他方法可自行扩展
}
}
```

本节小结：

本节给大家介绍了如何来实现一个可供复用的BaseAdapter，当然大家可以在这个的基础上根据自己的需求进行修改，比如通过异步设置网络图片等~改代码是参考鸿洋大神的视频写的：视频链接：

[Android-打造万能适配器](#) 另外，实际编写中遇到一些问题，非常感

谢Berial(B神)的耐心点拨~



ありがとうございます~

卡塔维基catawiki

劳力士手表拍卖



在线实例

- HTML 实例
- CSS 实例
- JavaScript 实例
- Ajax 实例
- jQuery 实例
- XML 实例
- Java 实例

字符集&工具

- HTML 字符集设置
- HTML ASCII 字符集
- HTML ISO-8859-1
- HTML 实体符号
- HTML 拾色器
- JSON 格式化工具

最新更新

- PHP接收json，并...
- Foundation CSS ...
- Foundation CSS ...
- Foundation 图标...
- Foundation 网...
- Foundation 块状...
- Foundation 网...

站点信息

- 意见反馈
- 免责声明
- 关于我们
- 文章归档

关注微信



Copyright © 2013-2015 菜鸟教程 runoob.com All Rights Reserved. 备案号：闽ICP备15012807号-1