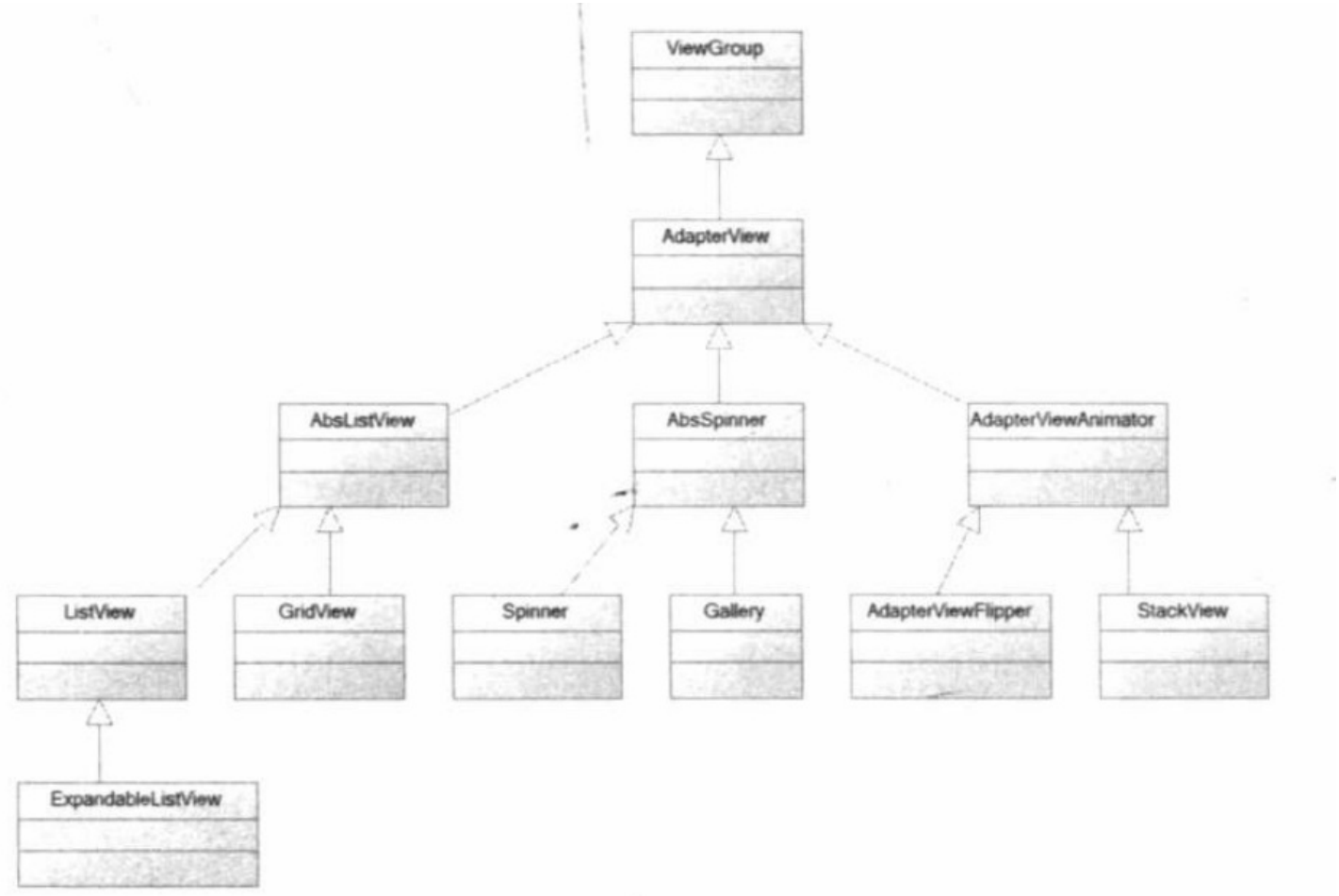
AdapterView 及其子类



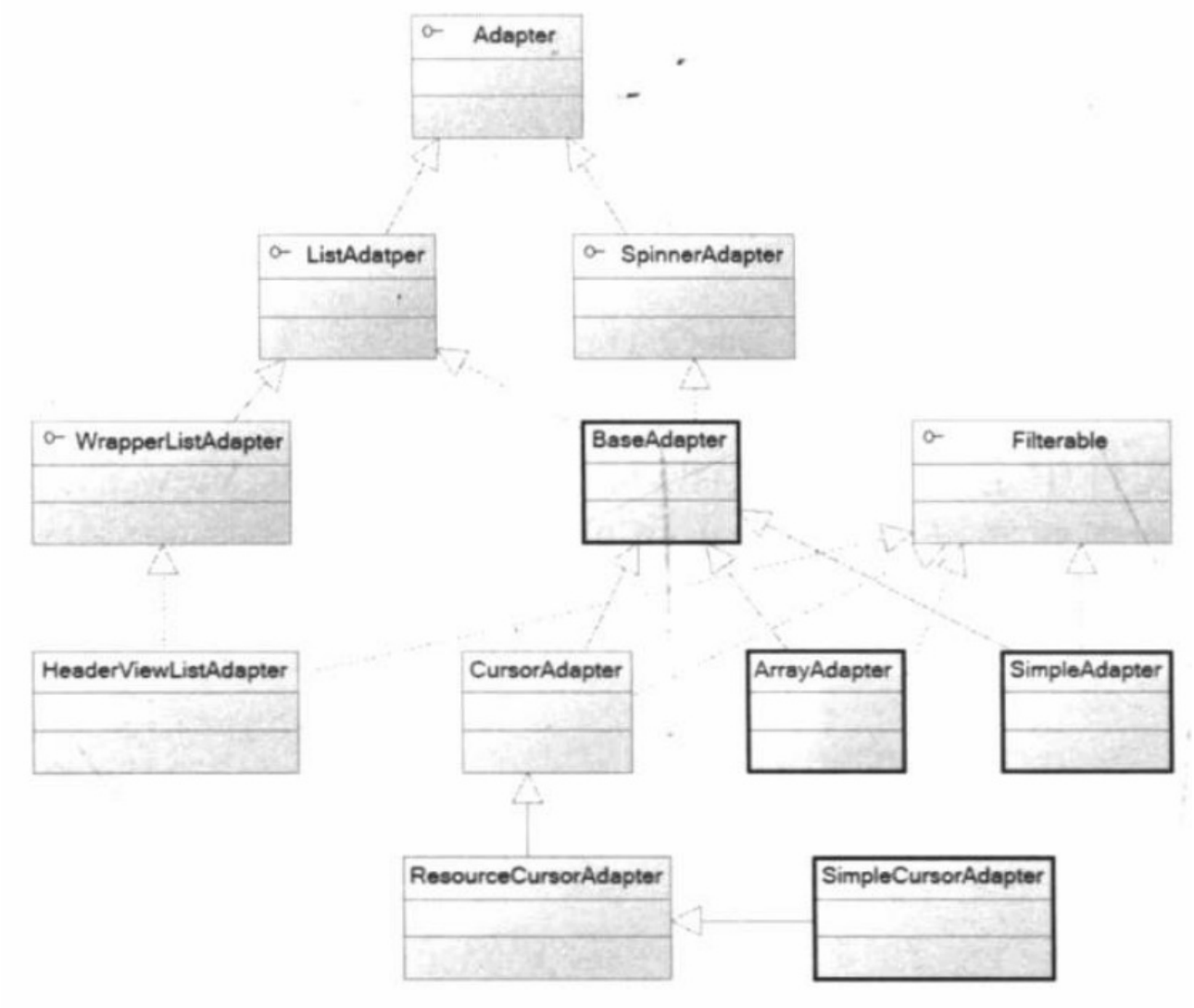
如上，Gallery已过时，Android推荐使用HorizontalScrollView代替它。

AdapterView只是容器，而Adapter负责提供每个"列表项"组件,AdapterView则负责采用合适的方式显示这些"列表项"。

AdapterView的特征

AdapterView继承ViewGroup，是一个容器。
AdapterView可以包含多个"列表项"，并将多个"列表项"以合适的形式显示出来。
adapterView显示的多个"列表项"由Adapter提供。调AdapterView.setAdapter(Adapter)方法设置Adapter即可

Adapter 接口及实现类



ListAdapter为AbsListView提供列表项

SpinnerAdapter为AbsSpinner提供列表项

BaseAdapter及其子类可以同时为AbsListView、AbsSpinner提供列表项

ArrayAdapter： 通常用于将数组或List集合的元素包装成列表项

SimpleAdapter： 可用于将List集合的对象元素包装成列表项

SimpleCursorAdapter： 跟SimpleAdapter类似，只是用于包装Cursor提供的数据

BaseAdapter： 通常用于被扩展。扩展BaseAdapter可对列表项进行最大限度的定制

AbsListView 常用XML属性

```
android:listSelector 指定被选中的列表项上绘制的Drawable

android:drawSelectorOnTop 设置为true时表示选中的列表项将会显示在上面

android:stackFromBottom

android:scrollingCache

android:textFilterEnabled

android:transcriptMode
```

```
android:cacheColorHint

android:fastScrollEnabled 设置为true，将会显示滚动图标，并允许用户拖动该滚动图标快速滚动

android:smoothScrollbar 设置为false，则不再header view之后绘制分隔条

android:choiceMode
```

ListView 和ListActivity

ListView常用的XML属性

```
android:entries  指定一个字符串数组作为列表项内容显示，设置了entries属性后，会为ListView设置
                 一个ArrayAdapter

        setAdapter(new ArrayAdapter<>(context,
                        R.layout.simple_list_item_1, entries));

android:divider  设置列表项的分隔条。可以指定一个@drawable，也可以指定16进制的颜色值"#f00"
                当指定为@null时，相当于分割线变成透明的了。

android:dividerHeight 设置分隔条的高度

android:headerDividersEnabled  设置为false则不在header view之后绘制分隔条
android:footerDividersEnabled  设置为false则不在footer view之前绘制分隔条
```

基于android:entries指定数组实现的ListView

在layout.xml文件中为ListView指定android:entries属性，设置一个字符串数组后，就可以将此数组元素作为列表项内容显示出来。

```
<ListView
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:entries="@arrays/books"/>
```

在ListView的构造方法中会判断enries属性是否指定了一个数组，如果指定了数组，就会调setAdapter方法设置此ListView的列表项：

```
setAdapter(new ArrayAdapter<>(context,  R.layout.simple_list_item_1, entries));
```

所以，在设置了entries属性后，不再需要另外调setAdapter属性为ListView配置列表项了。

此方式实现的ListView非常简单，不能对ListView的外观，行为进行定制。

基于ArrayAdapter实现的ListView

通过ArrayAdapter提供的构造方法，就可以对列表项进行配置

```
public ArrayAdapter(Context context, int resource, T[] objects)

public ArrayAdapter(Context context, int resource,
        int textViewResourceId, T[] objects)

public ArrayAdapter(Context context, int resource, List<T> objects)

public ArrayAdapter(Context context, int resource,
        int textViewResourceId, List<T> objects)
```

其中参数resource指定一个layout.xml布局文件；

参数textViewResourceId指定TextView在layout.xml中的id；
如果构造方法不指定textViewResourceId，则layout.xml的根布局控件就是TextView；
如果构造方法指定了textViewResourceId，则layout.xml中只要包含组件id为textViewResourceId的TextView就行。

objects表示的数组或集合中的数据元素类型为T，调TextView.setText(T.toString())方法给列表项设置内容。

ListActivity

如果当前Activity仅仅用来显示一个列表，则可以让我们的Activity继承ListActivity。

ListActivity的子类不用再调setContentView方法设置Activity的布局界面。

```
public class ListActivity extends Activity {
    ...
    public void setListAdapter(ListAdapter adapter) {
        synchronized (this) {
            ensureList();
            mAdapter = adapter;
            mList.setAdapter(adapter);
        }
    }
    ...
    private void ensureList() {
        if (mList != null) {
            return;
        }
        setContentView(com.android.internal.R.layout.list_content_simple);
    }
    ...
    @Override
    public void onContentChanged() {
        super.onContentChanged();
        View emptyView = findViewById(com.android.internal.R.id.empty);
        mList = (ListView)findViewById(com.android.internal.R.id.list);
        if (mList == null) {
            throw new RuntimeException(
                    "Your content must have a ListView whose id attribute is " +
                    "'android.R.id.list'");
        }
        if (emptyView != null) {
            mList.setEmptyView(emptyView);
        }
        mList.setOnItemClickListener(mOnClickListener);
        if (mFinishedStart) {
            setListAdapter(mAdapter);
        }
        mHandler.post(mRequestFocus);
        mFinishedStart = true;
    }
    ...
    protected void onListItemClick(ListView l, View v, int position, long id) {}
    ...
    private AdapterView.OnItemClickListener mOnClickListener =
        new AdapterView.OnItemClickListener() {

        public void onItemClick(AdapterView<?> parent, View v,
            int position, long id) {

            onListItemClick((ListView)parent, v, position, id);
        }
    };
}
```

如上，从ListActivity的源码可知，我们只需在onCreate中调用setListAdapter(ListAdapter)方法传入一个ListAdapter对象即可实现ListView的显示；如果想处理ListView的列表项的点击事件，重写onListItemClick方法即可。

基于SimpleAdapter实现的ListView

```
public SimpleAdapter(Context context, List<? extends Map<String, ?>> data,
```

```
        @LayoutRes int resource, String[] from, @IdRes int[] to)
```

参数resource： 表示列表项的layout.xml布局文件

参数to： 布局文件中控件id组成的数组，控件id指定的控件可以是TextView、ImageView、或实现了Checkable的控件（如CompoundButton的子类：CheckBox、RadioButton、Switch、ToggleButton）

参数data： 表示一个数据集合，该集合中元素的个数就是ListView的列表项的个数，每个列表项显示的数据内容由Map类型的元素指定。

```
当对应于数组参数to中的控件元素是Checkable的实现类时，类型？是Boolean类型。
如果此控件元素既实现了Checkable又继承了TextView，那么类型？可以是Boolean类型，也可以是其他类型。当类型？是Boolean类型时，作为控件元素的setChecked

当控件元素是TextView时，？类型的toString的返回结果为此TextView的显示内容；

当控件元素是ImageView时，类型？可以是Integer，此时Integer对象表示ImageView的图片资源ID；当类型？为其他类型时，那么其他类型的toString的返回结果要能
```

参数from： 表示集合data中每个元素类型 Map 中的索引String字符串

SimpleAdapter对列表项布局内容的处理方式就是：在处理每一个列表项时，遍历数组to得到需要处理的每个控件，然后把数组from中相同索引位置的字符串作为key值，去获取当前列表项对应的Map集合中的？类型的数据对象，最后用此数据对象处理当前遍历到的控件。

SimpleAdater.ViewBinder的作用

以上可知，SimpleAdapter本身只能绑定处理列表项布局文件中的TextView、ImageView和实现了Checkable的控件。如果想要处理列表项布局文件中的其他控件，或者要改变SimpleAdapter对支持控件的处理方式，可以实现SimpleAdapter.ViewBinder接口。

```
public static interface ViewBinder {
    boolean setViewValue(View view, Object data, String textRepresentation);
}
```

如上，重写setViewValue方法，在方法体中处理当前遍历到的控件view。返回true则SimpleAdapter不再对当前控件view进行默认处理，返回false则表示继续使用SimpleAdapter默认的方式处理此view控件。
setViewValue方法的参数data对应指定Map中的？类型对象，参数textRepresentation是data的toString方法返回结果

自定义BaseAdapter实现的ListView

BaseAdapter直接继承自ListAdapter和SpinnerAdapter，间接继承了Adapter。
BaseAdapter是一个抽象类，这是因为BaseAdapter并没有重写父接口Adapter提供的如下4个方法：

```
int getCount();  //返回值控制该Adapter将会包含多少个列表项

Object getItem(int position);  //返回值决定第position处的列表项的内容数据

long getItemId(int position);  //返回值决定第position处的列表项的ID

View getView(int position, View convertView, ViewGroup parent);  //返回值决定第position处的列表项的根控件。
```

所以自定义BaseAdapter时，需要重写上述的4个方法。ArrayAdapter和SimpleAdapter都继承BaseAdapter，并重写了上面的4个方法。

GridView 网格视图

GridView与ListView的区别是：GridView支持多列显示。
GridView和ListView都继承自AbsListView，所以GridView和ListView具有很高的相似性。

GridView常用XML属性

```
android:horizontalSpacing  设置各元素之间水平间距

android:verticalSpacing  设置各元素之间的垂直间距

android:stretchMode  设置元素的拉伸模式，使元素整体布局尽可能符合空间大小
```

```
android:columnWidth  设置列的宽度

android:numColumns  设置列数，该属性默认值为1，一般应指定大于1的值，
                        否则GridView就变成了ListView

android:gravity  设置水平对齐方式，默认是Gravity.LEFT
```

ExpandableListView 可展开的列表组件

ExpandableListView是ListView的子类，通过分组的方式把不同的列表分成不同的组，从而每组中都包含多个列表项组成的列表。

ExpandableListView的显示方式为：

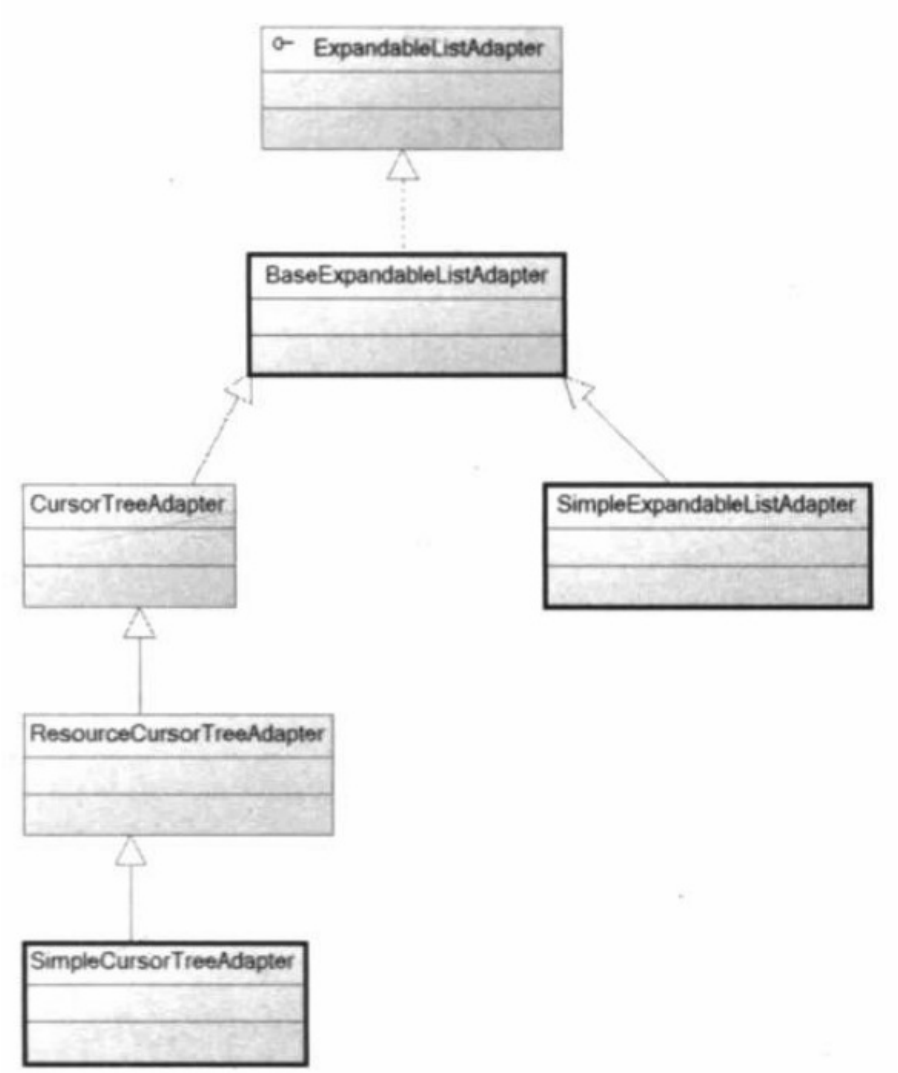
```
组列表项
...
该组包含的多个子列表项
...
组列表项
...
该组包含的多个子列表项
...

...

组列表项
...
该组包含的多个子列表项
...
```

ExpandableListView的XML属性

```
android:groupIndicator  显示在组列表旁边的Drawable对象
android:indicatorLeft
android:indicatorRight
android:childIndicator  显示在子列表项旁边的Drawable对象
android:childIndicatorLeft
android:childIndicatorRight
android:childDivider  指定各组内各子列表项之间的分隔条
android:indicatorStart
android:indicatorEnd
android:childIndicatorStart
android:childIndicatorEnd
```

ExpandableListAdapter

ExpandableListView显示的组和组中的多个列表项由ExpandableListAdapter提供，ExpandableListAdapter
是一个接口，继承关系如下图：



实现ExpandableListAdapter有如下三种常用的方式：

```
自定义BaseExpandableListAdapter子类实现ExpandableListAdapter

使用SimpleExpandableListAdapter将2个集合包装成ExpandableListAdapter

使用SimpleCuTreeAdapter将Cursor中的数据包装成SimpCursorTreeAdapter
```

自定义BaseExpandableListAdapter子类实现ExpandableListAdapter

此方式需要重写如下几个方法：

```
int getGroupCount();  //返回组列表项的个数

int getChildrenCount(int groupPosition); //返回groupPosition表示的组中的子列表项的个数

Object getGroup(int groupPosition); //返回第groupPosition处组列表项布局要显示的内容数据

Object getChild(int groupPosition, int childPosition); //返回groupPosition、     childPosition指定的子列表项的布局要显示的内容数据

long getGroupId(int groupPosition); //返回第groupPosition处组列表项的ID

long getChildId(int groupPosition, int childPosition);//返回groupPosition、     childPosition指定的子列表项的ID

View getGroupView(int groupPosition, boolean isExpanded, View convertView,
        ViewGroup parent);  //返回第groupPosition处组列表项的根布局控件

View getChildView(int groupPosition, int childPosition, boolean isLastChild,
```

```
        View convertView, ViewGroup parent);  //返回groupPosition、childPosition指定的子列表项的根布局控件

boolean isChildSelectable(int groupPosition, int childPosition); //重写时可返回true

boolean hasStableIds(); //重写时可返回true
```

可以参考SimpleExpandableListAdapter的实现方式

使用SimpleExpandableListAdapter

SimpleExpandableListAdapter提供如下三个构造方法用于创建一个配置组列表项和各组中子列表项的ExpandableListAdapter对象供ExpandableListView使用

```
public SimpleExpandableListAdapter(Context context,
        List<? extends Map<String, ?>> groupData, int groupLayout,
        String[] groupFrom, int[] groupTo,
        List<? extends List<? extends Map<String, ?>>> childData,
        int childLayout, String[] childFrom, int[] childTo)

public SimpleExpandableListAdapter(Context context,
        List<? extends Map<String, ?>> groupData, int expandedGroupLayout,
        int collapsedGroupLayout, String[] groupFrom, int[] groupTo,
        List<? extends List<? extends Map<String, ?>>> childData,
        int childLayout, String[] childFrom, int[] childTo)

public SimpleExpandableListAdapter(Context context,
        List<? extends Map<String, ?>> groupData, int expandedGroupLayout,
        int collapsedGroupLayout, String[] groupFrom, int[] groupTo,
        List<? extends List<? extends Map<String, ?>>> childData,
        int childLayout, int lastChildLayout, String[] childFrom,
        int[] childTo)
```

杂项组件

AutoCompleteTextView 自动完成文本框

自动完成文本框AutoCompleteTextView是EditText的子类，相比EditText，多了一个功能：当用户输入一定字符之后，自动显示一个下拉菜单，用户可以从下拉菜单中选择预填入的列表项内容。
提示内容列表项通过一个Adapter提供给AutoCompleteTextView

```
public class AutoCompleteTextView extends EditText implements Filter.FilterListener

private final ListPopupWindow mPopup;

public <T extends ListAdapter & Filterable> void setAdapter(T adapter) {
    ...
    mPopup.setAdapter(mAdapter);
}
```

ListPopupWindow

```
public class ListPopupWindow implements ShowableListMenu

private DropDownListView mDropDownList;  //DropDownListView extends ListView

public void setAdapter(@Nullable ListAdapter adapter) {
    ...
    mDropDownList.setAdapter(mAdapter);
}
```

android.R.styleable#AutoCompleteTextView_completionHint 设置下拉菜单的提示标题
android.R.styleable#AutoCompleteTextView_completionThreshold 显示下拉菜单的至少输入字符数

android.R.styleable#AutoCompleteTextView_completionHintView 下拉菜单的提示标题的视图
android.R.styleable#AutoCompleteTextView_dropDownSelector
android.R.styleable#AutoCompleteTextView_dropDownAnchor 下拉菜单的定位组件，默认是TextView本省
android.R.styleable#AutoCompleteTextView_dropDownWidth 下拉菜单的宽度
android.R.styleable#AutoCompleteTextView_dropDownHeight 下拉菜单的高度

android.R.styleable#ListPopupWindow_dropDownVerticalOffset 下拉菜单与文本框的水平偏移，默认左对齐
android.R.styleable#ListPopupWindow_dropDownHorizontalOffset 下拉菜单与文本框的垂直偏移，默认紧跟文本框

android.R.styleable#PopupWindow_popupBackground 下拉菜单的背景