ListView是为了实现滑动项目的空间，与前面所学的控件有很到的区别，变得更加复杂了。它有3种形式，与之对应的有三种适配器。三者有相同点，三者都要适配器【适配器的作用相当于一个组装工人，将数据源和模板一项一项组装起来，提供给索要的ListView】，都要ListView布局【索要者】，和一个显示模板【显示者】。其中后两者是完全相同的。三者都必须具备的四要素：**ListVIew，模板，数据源，设置Adapter。**

**第一种：ArrayAdapter**，这种对应的适配器，只能适合实现一个item显示一种内容，是最简单的适配器了。

1. 创建一个工程，在工程中添加ListView控件【其中只要声明id号（这个id号特别重要），宽度和高度就行了】。在main\_activity中声明ListView控件，并寻找findViewById【供最后设置适配器用】

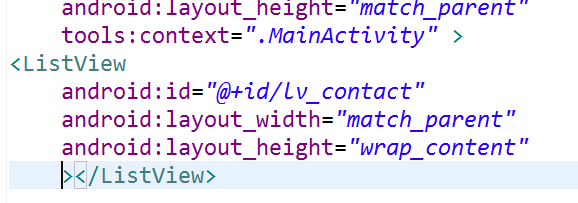


图1.1.1ListView声明

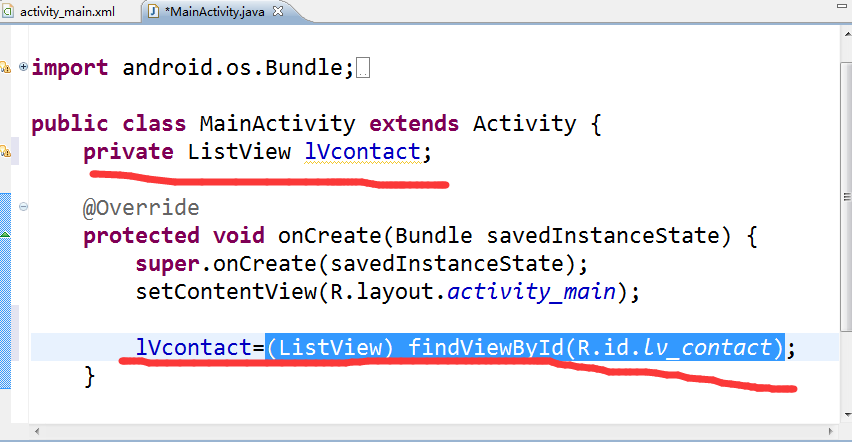


图1.1.2ListView控件的声明

1. 模板的建立。在layout中再建立一个显示的模板，由于每一个item中只有一项内容，用textview就行了。Padding中设置item中的内容间隔，注意只能在width和height中插入，再两者后面插入时无法插入的。

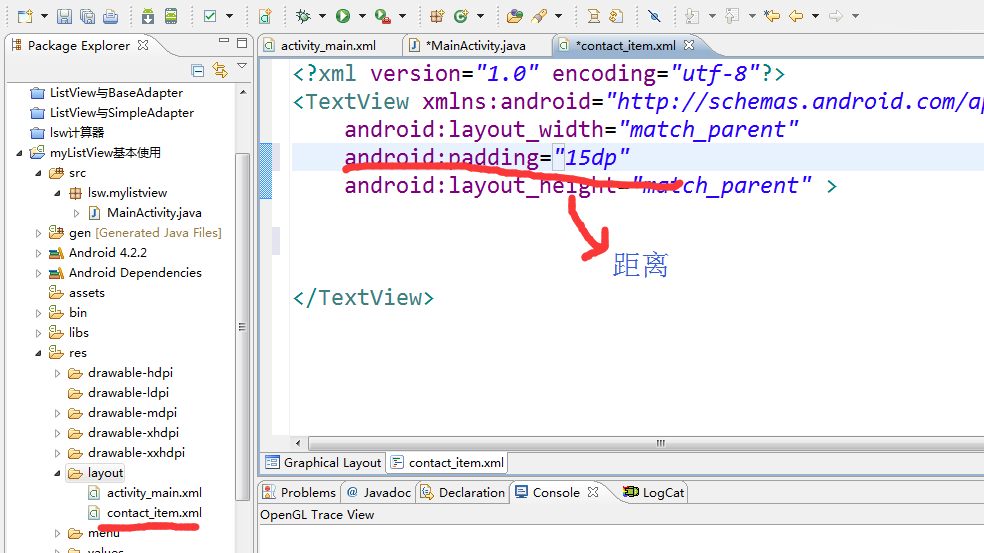


图1.2.1模板的设置

1. 准备数据源。首先在main\_activity中声明List<string > data.再对data进行内存分配，这是用的是ArrayList

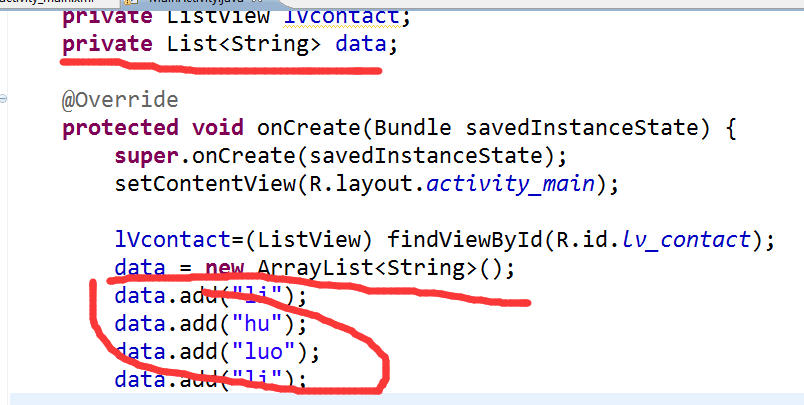


图1.3.1 数据源的设置

1. 声明相应Adapter,并设置相应的Adapter。

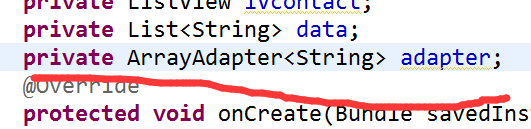


图1.4.1声明ArrayAdapter适配器

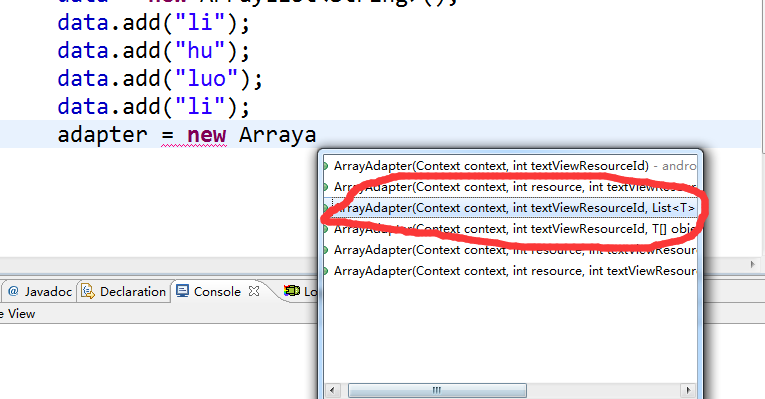


图1.4.2ArrayApadter适配器的选择

注意三个参数填的内容，context一般都填为this，textViewResourceId填的是模板的布局id。Objects填的是数据源data。

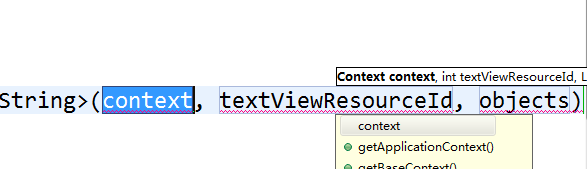
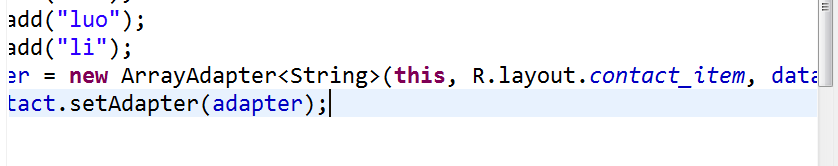


图1.4.3ArrayApadter适配器的参数配置

图1.4.4ArrayApadter适配器的参数配置的结果以及Adapter的设置

**第二种，SimpleAdapter**

1，在ListVIew，模板和第一种是相同的步骤【只是由于SimpleAdapter一个item中可以显示多项内容，模板就要使用RelativeLayout，在里面声明要显示的一个item中的多项内容】。与第一种的区别是数据源的设置和适配器的声明。

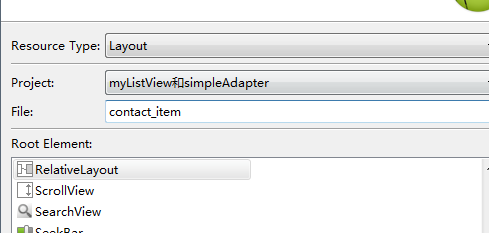


图2.1.1模板的建立

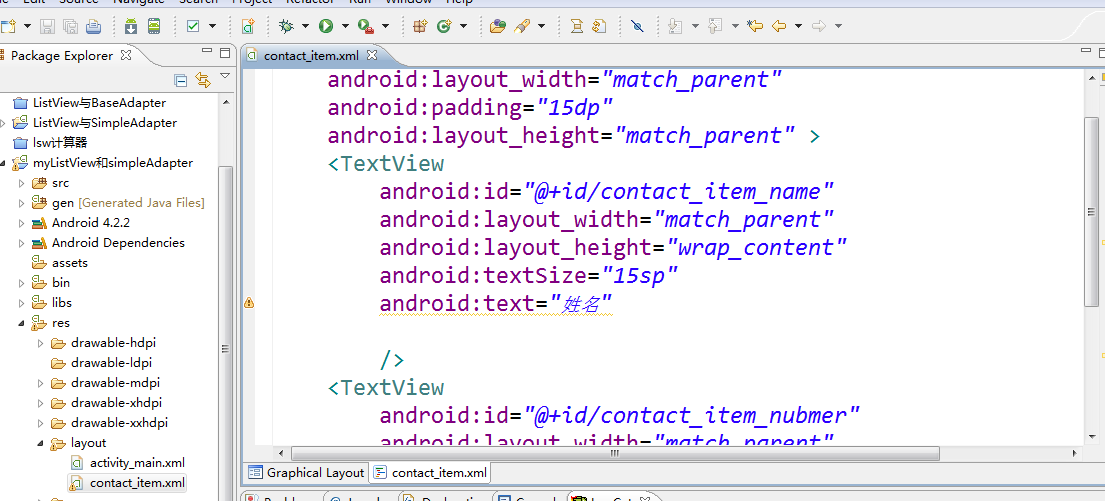


图2.1.2 模板的布局

2，先设置适配器，注意选择正确的适配器和对其中的五个参数进行配置，这是simpleAdapter的难点，首先将context改为this，剩余的4个，利用系统提供的更改意见，填出相应的数据结构。先从简单的后三个开始。

Resource变为int resource，将其补全，即模板的地址。

From变为String[] from，将其补全，即数据源中item中显示的内容。这个是在数据源中定义的

To变为int to[],将其补全，即模板中的item中要显示的内容的id号，注意是引号的，直接写出id号就行，中间用“，”隔开就行，这一个From是不同的。

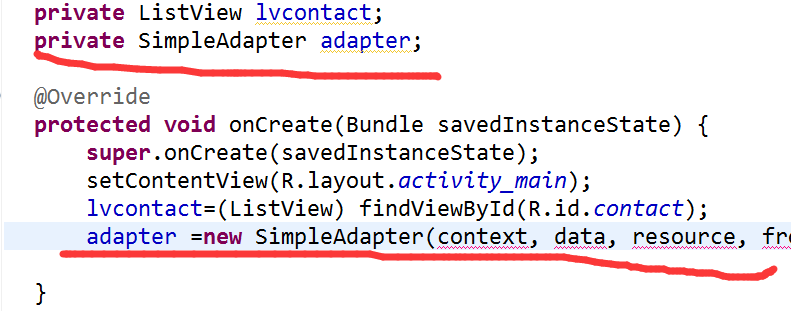


图2.2.1适配的选择

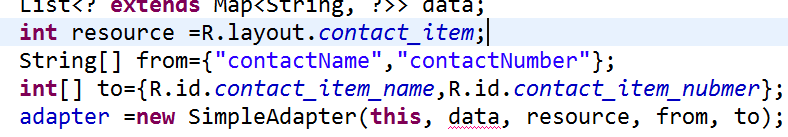


图2.2.2三个比较简单的属性的设置

对data的设置。Data补全为：List<? extends Map<String, ?>> data，这里的data的类型，有点陌生，把?? Extends 去掉，把后面的一个？改为object。在开始进行data的声明，

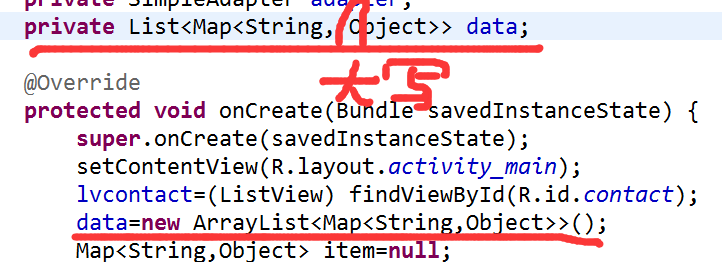


图2.2.3

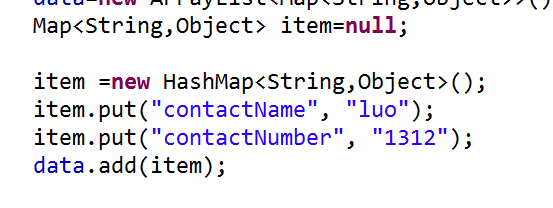
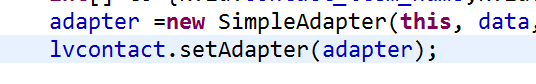


图2.2.4添加数据源



设置适配器

第三种，BaseAdapter，这是最难的一种LIstView了。

在模板，ListView方面和第二种一模一样。

1数据源的设置，注意输入的数据要和Context的参数格式一直对应。不要弄乱了。首先声明private List<Contact> data;这时由于List非String，里面是自己定义的Contact变量，所以根据开发工具的提示，创建Contact类，并在该类中，声明自己将要添加的数据，并且还有一步，点击Source--->选中其中的Generate Constructor using fields选项。

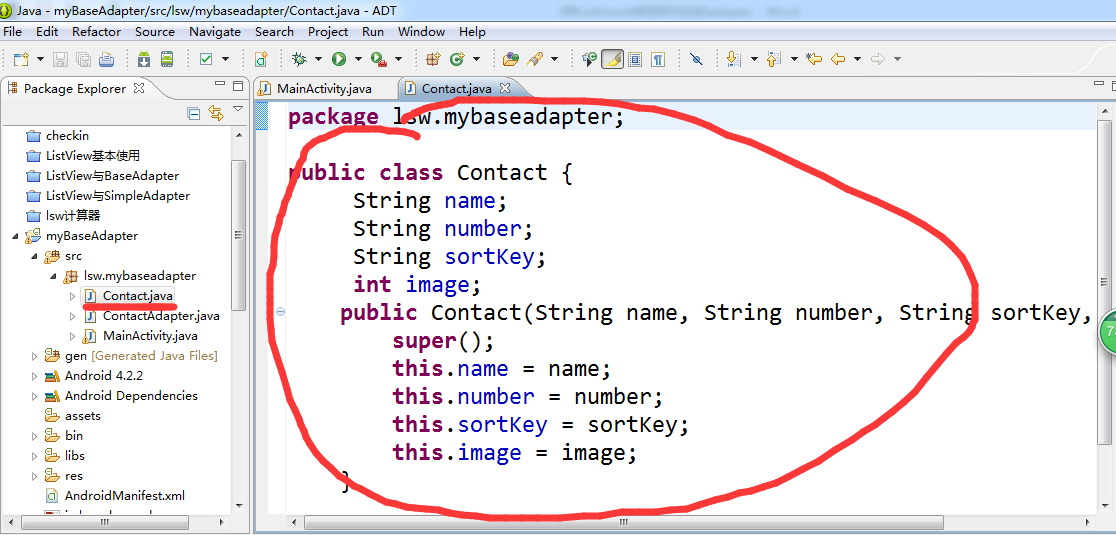


图3.1.1生成Contact类，并声明变量

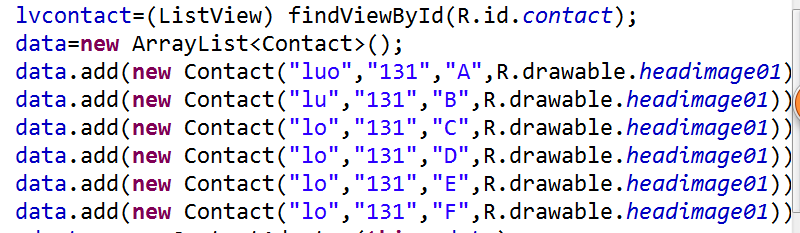


图3.1.2 添加数据

2相应的Adapter的设置。这是BaseAdapter中最难的，

2,1声明Adapter ，这时会报错，需要创建一个ContactAdapter类，这个类继承与BaseAdapter，这时还好出错，要生成4中方法，其种有两种方法是没有必要更改的。另外两种，就要加入代码了。首先是getCount（）【数据的长度】在返回值中加入data.size()，集合，因此在开始还要和在main\_Acticity中一样声明一下data。



图3.2.1.1适配器的声明



图3.2.1.2 继承BaseAdapter

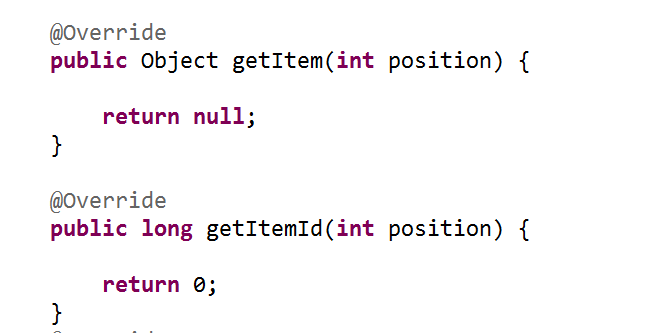


图3.2.1.3 两种无需填入的方法

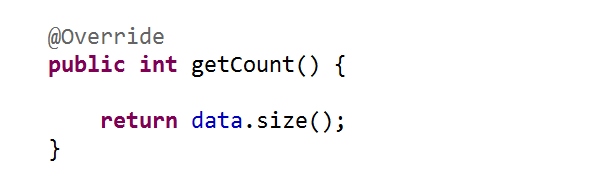
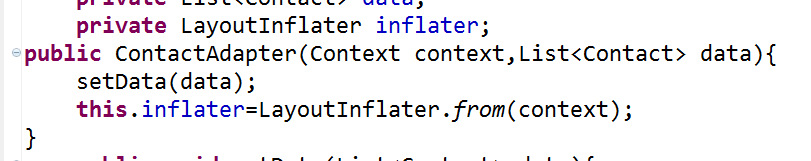
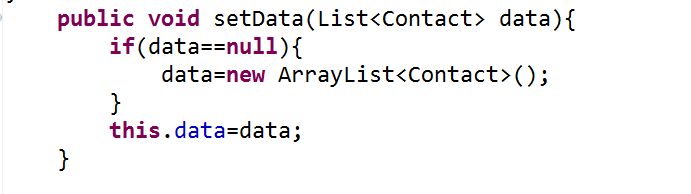


图3.2.1.4对getCount方法的更改

接下来时最难的，对public View getView(int position, View convertView, ViewGroup parent)进行设置，这里有很多相关的知识。

1. 要形成一个ContactAdapter的构造方法，并且在里面设置两个参数，还要形成一个setData的方法，以免data为空引起错误。



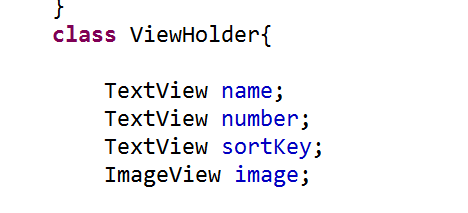


2关于BaseAdapter的优化--使用ViewHolder

------------------------------------------

推荐在ContactAdapter中定义内部类ViewHolder，用于封装模板中的各个控件，在getView()中，如果convertView为null，则创建新的ViewHolder，为ViewHolder中的各个控件赋值，并调用convertView的setTag()方法将ViewHolder对象封装起来，如果convertView()不为null，则直接调用convertView的getTag()方法获取ViewHolder对象。

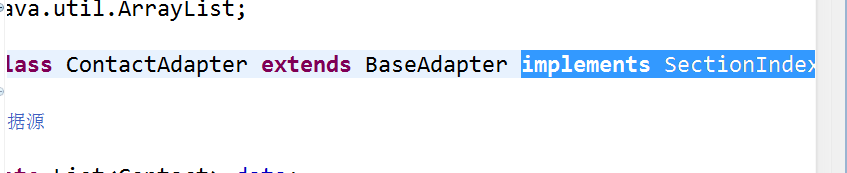
为了优化，引入ViewHolder类。在里面定义相关的变量数据



接下了的步骤，见老师的工程，自己在这里也不好解释。

对关键字按字母进行排序，

1. 调用Collections.sort(数据源);会自动进行排序
2. implements SectionIndexer。



1. 在调出其中的3个方法。其中有一个是无需填写的。

getPositionForSection（）根据标签得到位置。

getSectionForPosition（） 根据位置得到标签，这是比较容易得到的。

