ListView 的使用

一、 实验目的

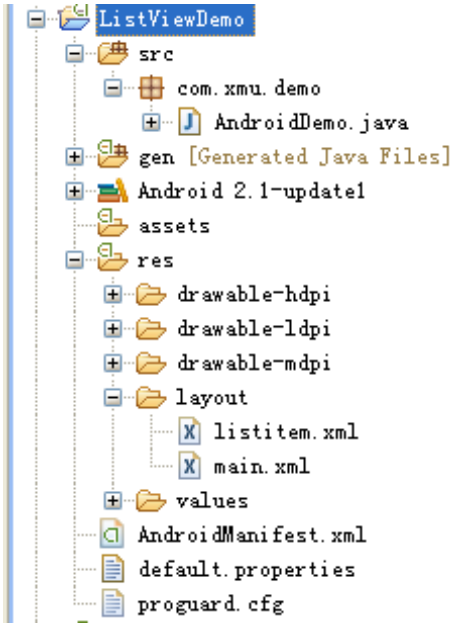
1、 了解 ListView 控件，
2、 学习使用 Adapter 存储 ListView 的数据

二、 实验条件

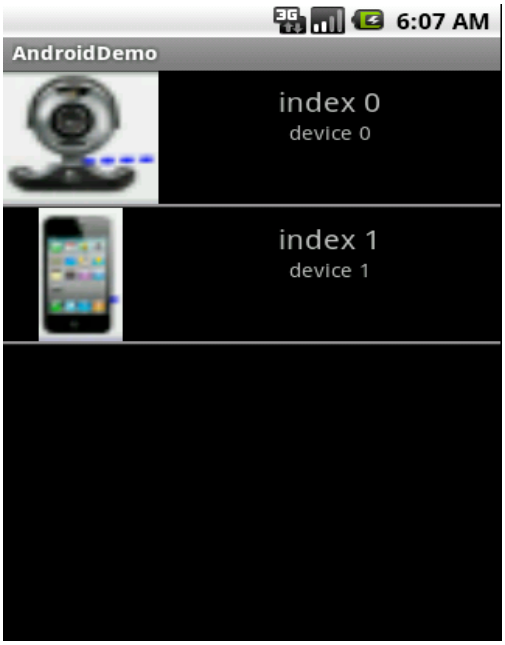
- ✓ IBM-PC 兼容机
- ✓ Windows、Ubuntu11.04 或其他兼容的 Linux 操作系统
- ✓ JDK（建议安装 JDK8 及其以上版本）、Android Studio 或 Eclipse with ADT
- ✓ INTEL ATOM 平板

三、 DEMO

若使用 Eclipse 开发工具，可以通过
File --> Import --->General --->Existing Projects into Workspace
将 ListViewDemo 工程文件夹导入 Eclipse 中。
在 Eclipse 里可以看到如下的工程文件



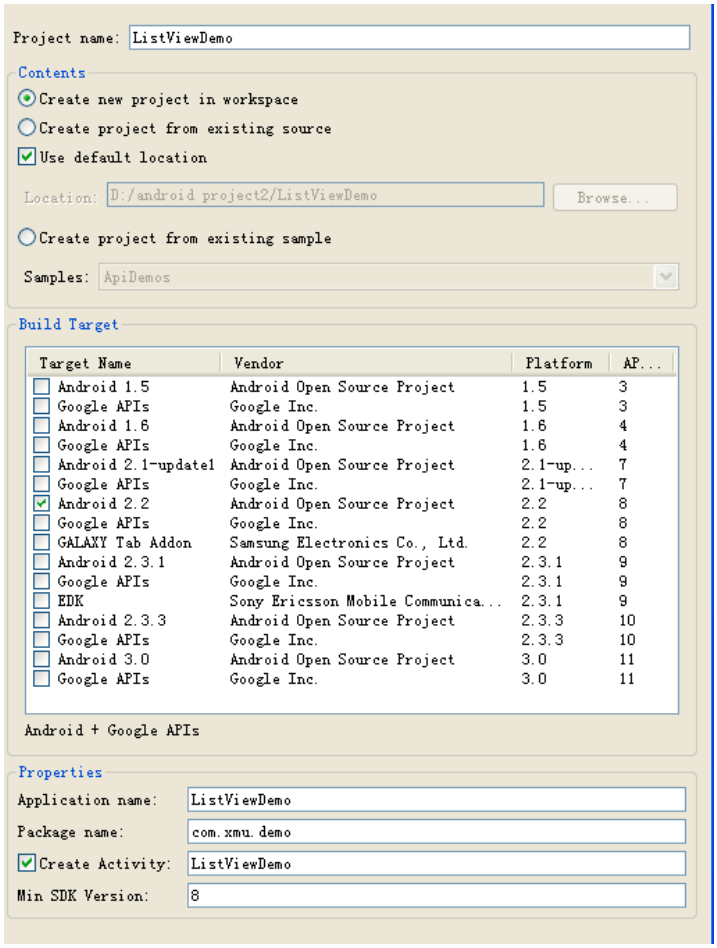
在这个例子里，主要有涉及到以下的几个文件，一个 AndroidDemo.java 文件和两个布局文件 listitem.xml 和 main.xml。
该实例的运行效果如下：



四、 实验步骤

1. 创建工程

首先，建立一个 ListViewDemo 的 android 工程，设置如下



在/res/layout 文件夹下添加布局文件 Listitem.xml

2. 布局文件 main.xml

在 mian.xml 文件中添加如下代码：

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    >
   <ListView android:id="@+id/listview"
        android:layout_width="fill_parent"
        android:layout_height="fill_parent"/>
</LinearLayout>
```

代码解释：

这是一个 LinearLayout 布局，通过设置 android:orientation 为 vertical，使其为竖直方向的布局。

在这个布局文件当中，定义了一个 ListView,并且这个 ListView 的 id 是“*@+id/listview*”，则我们可以通过 R.id.listview 来获取该 ListView 元素，并对该元素进行操作。
且在这里设置了 ListView 的属性 android:layout_width="fill_parent" 与 android:layout_height="fill_parent"

这两个的属性设置，会让该 ListView 撑起来

3. 布局文件 listview.xml

在 listview.xml 文件中添加如下代码。

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
  xmlns:android="http://schemas.android.com/apk/res/android"
  android:layout_width="fill_parent"
  android:layout_height="fill_parent"
  android:orientation="vertical">
<LinearLayout
  xmlns:android="http://schemas.android.com/apk/res/android"
  android:layout_width="fill_parent"
  android:layout_height="wrap_content"
  android:orientation="horizontal">
<ImageView
android:id="@+id_include_newworkmap_listitem/include_networkmap_listitem_image"
android:layout_width="100dip"
android:layout_height="100dip"/>
<LinearLayout
 xmlns:android="http://schemas.android.com/apk/res/android"
 android:layout_width="fill_parent"
 android:layout_height="wrap_content"
 android:orientation="vertical">
<TextView
android:id="@+id_include_newworkmap_listitem/include_networkmap_listitem_index"
android:text="设备"
android:textSize="20dip"
android:layout_width="fill_parent"
android:layout_height="wrap_content"
android:gravity="center_horizontal"
android:layout_marginTop="10dip" />
<TextView
android:id="@+id_include_newworkmap_listitem/include_networkmap_listitem_name"
android:text="hello world"
```

```
android:layout_width="fill_parent"
android:gravity="center_horizontal"
android:layout_height="wrap_content"/>
</LinearLayout>
</LinearLayout>
<View
 android:layout_height="2dip"
 android:background="#FF909090"
 android:layout_width="fill_parent" />
</LinearLayout>
```

代码解释：
该布局文件定义了在 ListView 每一项的布局。这个布局文件定义了这样的布局。
先把界面分为左右两个布局。左边放置一张图片，然后右边又进一步布局，分为上下两个布局，上下各放置一个 TextView。
这样的布局可以通过嵌套 LinearLayout，来实现如此布局。

4. AndroidDemo.java

往 AndroidDemo.java 文件添加如下代码。

```
public class AndroidDemo extends Activity {
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        // ///组织好数据
        List<Map<String, Object>> data = new ArrayList<Map<String, Object>>();
        Map<String, Object> item = new HashMap<String, Object>();
        item.put("type", R.drawable.camera);
        item.put("index", "index 0");
        item.put("name", "device 0");
        data.add(item);
        item = new HashMap<String, Object>();
        item.put("type", R.drawable.iphone);
        item.put("index", "index 1");
        item.put("name", "device 1");
        data.add(item);
        ListView listview = (ListView) this.findViewById(R.id.listview);
        // //使用适配器，将前面填充好的数据填充进来
```

```
SimpleAdapter adapter = new SimpleAdapter(
        this,
        data,
        R.layout.listitem,
        new String[] { "type", "index", "name" },
        new int[] {
        R.id_include_newworkmap_listitem.include_networkmap_listitem_image,
        R.id_include_newworkmap_listitem.include_networkmap_listitem_index,
        R.id_include_newworkmap_listitem.include_networkmap_listitem_name });
listview.setAdapter(adapter);}}
```

代码解释：

代码里构造了一个 List<Map<String, Object>> data，这是一个列表，列表里每一个元素是一个 Map 集合。在上面构造的每一个 Map 集合主要包含了三个元素，type,index 和 name.

```
SimpleAdapter adapter = new SimpleAdapter(
        this,
        data,
        R.layout.listitem,
        new String[] { "type", "index", "name" },
        new int[] {
        R.id_include_newworkmap_listitem.include_networkmap_listitem_image,
        R.id_include_newworkmap_listitem.include_networkmap_listitem_index,
        R.id_include_newworkmap_listitem.include_networkmap_listitem_name });
```

这一句代码，则是负责生成一个 SimpleAdapter 实例，下面具体看一下具体的参数。

.this 这个是当前上下文的 Context 引用

.R.layout.listitem 关联在 listitem.xml 文件当中定义的 Layout,这个 layout 规定了 ListView 当中每一项的布局。

.New String[]{"type","index","name"},里面的每一个字符串用于访问得到 Map 对象当中的值。例如在这个例子里，有执行 item.put("index", "index 0"),那么在 ListView 当中就可以通过 item.get("index")将"index 0"这个值取出来，并且在 ListView 中的项进行显示。

```
.new int[] {
        R.id_include_newworkmap_listitem.include_networkmap_listitem_image,
        R.id_include_newworkmap_listitem.include_networkmap_listitem_index,
        R.id_include_newworkmap_listitem.include_networkmap_listitem_name })
```

里面的值就是在 listitem.xml 文件里面定义的一个 ImageView，两个 TextView 的 ID。这个数组就是与前面的 string 数组相一一对应的，如前面举的例子，有设置了 item.get("index"),这就会把 ID 为 include_networkmap_listitem_index 的 TextView 的 Text 设为“index 0”。

5. 事件绑定

可以通过对 ListView 添加事件

```
listview.setOnItemClickListener(new OnItemClickListener() {

    @Override
    public void onItemClick(AdapterView<?> arg0, View arg1, int arg2,
            long arg3) {
        // 在这里可以写上要处理的事情

    }
});
```

该事件在用户单击 ListView 上的任意一项的时候会被触发。

五、 附加实验

为 ListView 添加 OnItemClick 事件的处理，当点击 Item 时将当前项的数据通过 Logcat 输出。

六、 参考资料

docs/reference/android/widget/ListView.html
docs/reference/android/widget/SimpleAdapter.html

七、 实验报告要求

实验报告中要包含以下几个部分：

1、实验目的
2、实验条件
3、实验原理
4、实验步骤分析
5、实验结果与总结
6、实验思考题

实验步骤要详细，关键步骤要有截图，运行结果也要有截图。