**Создаем свой кастомный ListView**

Недавно я озадачился созданием кастомного ListView, такого, чтоб вместо одной строки в каждом элементе списка было бы несколько строк: ну например заголовок и подзаголовок. Порылся по инету и пришел к выводу, что адекватного русскоязычного туториала на эту тему к сожалению нет. Ниче страшного. Теперь будет! :). Как минимум русскоязычный и надеюсь адекватный :).

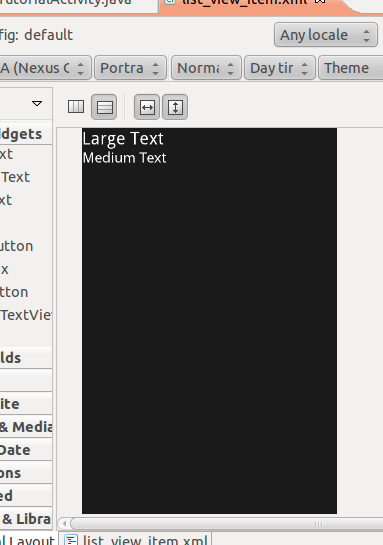
Итак, прежде всего попробую описать логику создания такого ListView, а затем приведу пример с исходниками.

Начнем с того, что для отображения любой информации в ListView нам необходим адаптер — прослойка между вашими данными и их графическим отображением в ListView. Если мы хотим обычный список — то ту все здорово — используем готовый класс StringAdapter и радуемся. Передаем ему любой строковый массив, затем передаем наш адаптер в ListView и получаем готовый ListView заполненный элементами строкового массива. Однако, все становится гораздо сложнее, когда каждый элемент ListView содержит что-то большее чем одну строку. Для такого случая нам с Вами придется создать отдельный файл c разметкой (Layout) в котором будет хранится содержимое каждого элемента ListView. Поскольку мне нужен список с заголовком и подзаголовком я создал вот такой вот файлик:

[?](http://davidmd.ru/2011/12/16/%D1%81%D0%BE%D0%B7%D0%B4%D0%B0%D0%B5%D0%BC-%D1%81%D0%B2%D0%BE%D0%B9-%D0%BA%D0%B0%D1%81%D1%82%D0%BE%D0%BC%D0%BD%D1%8B%D0%B9-listview/)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21 | <?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" >        <TextView          android:id="@+id/item\_headerText"          android:layout\_width="wrap\_content"          android:layout\_height="wrap\_content"          android:text="Large Text"          android:textAppearance="?android:attr/textAppearanceLarge" />        <TextView          android:id="@+id/item\_subHeaderText"          android:layout\_width="wrap\_content"          android:layout\_height="wrap\_content"          android:text="Medium Text"          android:textAppearance="?android:attr/textAppearanceMedium" />    </LinearLayout> |

В визуальном редакторе это выглядит вот так:

[](http://davidmd.ru/wp-content/uploads/2011/12/%D0%A1%D0%BD%D0%B8%D0%BC%D0%BE%D0%BA-1.png)

Внешний вид разметки

У нас в файле разметки имеется два текстовых поля: первое item\_headerText и второе item\_subHeaderText.

Следующим шагом создадим конструкцию с заголовком и подзаголовком. Я сделал это вот в таком небольшом классе:

[?](http://davidmd.ru/2011/12/16/%D1%81%D0%BE%D0%B7%D0%B4%D0%B0%D0%B5%D0%BC-%D1%81%D0%B2%D0%BE%D0%B9-%D0%BA%D0%B0%D1%81%D1%82%D0%BE%D0%BC%D0%BD%D1%8B%D0%B9-listview/)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39 | package ru.davidmd.titorials.ListView;    public class Item {      /\*\*       \* Заголовок       \*/      String header;        /\*\*       \* Подзаголовок       \*/      String subHeader;        /\*\*       \* Конструктор создает новый элемент в соответствии с передаваемыми       \* параметрами:       \* @param h - заголовок элемента       \* @param s - подзаголовок       \*/      Item(String h, String s){          this.header=h;          this.subHeader=s;      }        //Всякие гетеры и сеттеры      public String getHeader() {          return header;      }      public void setHeader(String header) {          this.header = header;      }      public String getSubHeader() {          return subHeader;      }      public void setSubHeader(String subHeader) {          this.subHeader = subHeader;      }    } |

Теперь приступим к реализации класса-адаптера. Фактически именно класс-адаптер создает каждый элемент в ListView а значит нашей задачей будет именно создание каждого элемента. Помимо этого на класс адаптер возлагаются еще некоторые другие функции. Лично я свой адаптер унаследовал от абстрактного класса BaseAdapter. Нам необходимо реализовать несколько его методов. Что я собственно и сделал:

[?](http://davidmd.ru/2011/12/16/%D1%81%D0%BE%D0%B7%D0%B4%D0%B0%D0%B5%D0%BC-%D1%81%D0%B2%D0%BE%D0%B9-%D0%BA%D0%B0%D1%81%D1%82%D0%BE%D0%BC%D0%BD%D1%8B%D0%B9-listview/)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61 | package ru.davidmd.titorials.ListView;    import java.util.ArrayList;  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 class MyAdapter extends BaseAdapter {        ArrayList<Item> data = new ArrayList<Item>();      Context context;        public MyAdapter(Context context, ArrayList<Item> arr) {          if (arr != null) {              data = arr;          }          this.context = context;      }        @Override      public int getCount() {          // TODO Auto-generated method stub          return data.size();      }        @Override      public Object getItem(int num) {          // TODO Auto-generated method stub          return data.get(num);      }        @Override      public long getItemId(int arg0) {          return arg0;      }        @Override      public View getView(int i, View someView, ViewGroup arg2) {          //Получение объекта inflater из контекста          LayoutInflater inflater = LayoutInflater.from(context);          //Если someView (View из ListView) вдруг оказался равен          //null тогда мы загружаем его с помошью inflater          if (someView == null) {              someView = inflater.inflate(R.layout.list\_view\_item, arg2, false);          }          //Обявляем наши текствьюшки и связываем их с разметкой          TextView header = (TextView) someView.findViewById(R.id.item\_headerText);          TextView subHeader = (TextView) someView.findViewById(R.id.item\_subHeaderText);            //Устанавливаем в каждую текствьюшку соответствующий текст          // сначала заголовок          header.setText(data.get(i).header);          // потом подзаголовок          subHeader.setText(data.get(i).subHeader);          return someView;      }    } |

Теперь давайте смотреть что тут к чему: конструктор MyAdapter() получает два параметра: первый — контекст, второй массив значений из которых мы будем строить наш ListView. Методы getCount() и getItem() думаю в пояснении не нуждаются. Самое интересное происходит в методе getView(). Этот метод вызывается из ListView тогда, когда листвью пытается заполнить саму себя с помощью адаптера. В качестве параметров сюда приходят номер элемента i, который в данный момент пытается добавить ListView, пустой объект someView — это как раз тот самый View который мы с вами будем заполнять и экземпляр ViewGroup — который есть ни что иное как сам заполняемый ListView.

В этом методе сначала мы получаем экземпляр класса LayoutInflater — это класс который используется для создания объектов View из их описания в xml. Затем в переданный нам someView мы закидываем объект View полученный из нашего файла разметки. И настраиваем его параметры — такие как заголовок и подзаголовок! Ничего сложного, ура!

Ну и последний элемент — это собственно сам класс Activity:

[?](http://davidmd.ru/2011/12/16/%D1%81%D0%BE%D0%B7%D0%B4%D0%B0%D0%B5%D0%BC-%D1%81%D0%B2%D0%BE%D0%B9-%D0%BA%D0%B0%D1%81%D1%82%D0%BE%D0%BC%D0%BD%D1%8B%D0%B9-listview/)

package ru.davidmd.titorials.ListView;

import java.util.ArrayList;

import android.app.Activity;

import android.os.Bundle;

import android.widget.ListView;

public class ListViewTutorialActivity extends Activity implements OnItemcClickListener {

    /\*\* Called when the activity is first created. \*/

    ListView lv;

    ArrayList<Item> data = new ArrayList<Item>();

    @Override

    public void onCreate(Bundle savedInstanceState) {

        super.onCreate(savedInstanceState);

        setContentView(R.layout.main);

        data.add(new Item("Заголовок1","Подзаголовок1"));

        data.add(new Item("Заголовок2","Подзаголовок2"));

        data.add(new Item("Заголовок3","Подзаголовок3"));

        data.add(new Item("Заголовок4","Подзаголовок4"));

        data.add(new Item("Заголовок5","Подзаголовок5"));

        lv = (ListView) this.findViewById(R.id.listView1);

        lv.setAdapter(new MyAdapter(this,data));

        lv.setOnItemClickListenr(this);

    }

    @Override

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

        Toast.MakeText(this, ((Item)((MyAdapter)parent).getItem(position)).header, Toast.LENGTH\_LONG).show();

    }

}

Здесь тоже никаких проблем возникнуть не должно. Формируем список объектов и закидываем его в создаваемый анонимный адаптер. Ура, все работает. Вот тут [ListViewTutorial](http://davidmd.ru/wp-content/uploads/2011/12/ListViewTutorial.zip) можно скачать проект для Eclipse.

Метод onItemClick добавлен по просьбе **Dmitrii.**Заметьте, что наша активити реализует интерфейс OnItemClickListenr.  Соответственно метод onItemClick вызывается когда произойдет клик на каком-то из элементов списка. При клике мы создаем всплывающе уведомление Toast, в котом показываем заголовок того элемента списка по которому тапнул пользователь.