[ListView Использование ArrayAdapter. Ч1](http://codeandlife.ru/android/7-list-view-with-array-adapter-p1.html" \o "ListView. Использование ArrayAdapter. Ч1)

Введение

Одна из частых задач, возникающих при программировании Android приложений, - это  отображение на дисплее набора данных. Например, списка пользователей, содержимого папки, набора заметок и так далее. Для решения данной задачи в составе Android SDK есть такие компоненты как  ListView, GridView, RecyclerView. В этой статье речь пойдет об использовании ListView. Этот компонент на данный момент является самым распространенным.

Итак, ListView предназначен для отображения набора данных в виде одномерного (в один столбец) списка. Принцип работы данного компонента состоит в том, что он создает несколько View элементов, достаточных для отображения видимой части списка и затем переиспользует их (то есть заполняет новыми данными), когда пользователь скролит список.

Созданием View элементов списка и заполнением их данными занимается адаптер - объект, который реализует интерфейс Adapter.  Он создается разработчиком на этапе конфигурирования ListView. В составе Android SDK уже есть несколько готовых адаптеров, работающих «из коробки», например ArrayAdapter, SimpleAdapter, CursorAdapter. Также разработчик может написать свой, так называемый custom адаптер, что обычно и приходится делать.

**Базовая схема использования ListView выглядит так:**

* прописываем ListView в требуемый layout файл Activity или Fragment
* создаем объект для хранения данных и заполняем его данными
* создаем layout для одного элемента списка
* создаем адаптер, передав ему набор данных и layout одного элемента списка
* получаем ссылку на ListView и передаем ему адаптер
* добавляем обработчики событий, например OnItemClickListener

Попробуем написать простое приложение с ListView и ArrayAdapter.

ListView и ArrayAdapter

**1. Создаем в Android Studio проект на базе шаблона EmptyActivity.**

File > New > New Project..., далее следуем указаниям визарда.

Я создал проект с такими параметрами:

* Application name: ListViewTest1
* Company Domain: codeandlife.ru
* Platforms: Phone and Tablet -  API15:Android4.0.3 (IceCreamSandwich)
* Template: EmptyActivity

**2. В layout файл Activity добавляем ListView.**

По умолчанию AndroidStudio создает файл разметки с RelativeLayout и TextView. Меняем  RelativeLayout на FrameLayout, а TextView на ListView. Должен получиться примерно такой код:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14 | <?xml version="1.0" encoding="utf-8"?>  <FrameLayout      xmlns:android="http://schemas.android.com/apk/res/android"      xmlns:tools="http://schemas.android.com/tools"      android:layout\_width="match\_parent"      android:layout\_height="match\_parent"      tools:context="ru.codeandlife.listviewtest1.MainActivity">         <ListView          android:id="@+id/list\_view"          android:layout\_width="match\_parent"          android:layout\_height="match\_parent"/>    </FrameLayout> |

**id** нужен для того чтобы получить ссылку на ListView в коде.

**layout\_width** и**layout\_height** задают размеры Listview. В данном случае ListView займет все пространство, которое ему предоставит FrameLayout.

**3. Создаем объект для хранения данных и заполняем его.**

Тип объекта, используемого для хранения данных, определяется типом используемого адаптера. ArrayAdapter может работать с массивами или коллекциями, реализующими List интерфейс. Если разработчик создает свой адаптер, он может использовать произвольные  объекты, например xml файл.

Мы будем использовать ArrayList и набор строк. Код будет выглядеть так:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16 | //MainActivity.java        private List<String> list;      private ArrayAdapter<String> adapter;        @Override      protected void onCreate(Bundle savedInstanceState) {          super.onCreate(savedInstanceState);          setContentView(R.layout.activity\_main);            //Creates ArrayList and fake data          list = new ArrayList<>(20);          for(int i = 0; i < 20; i++){              list.add("item " + Integer.toString(i));          }     } |

**Замечание!**  На этапе создания адаптера и конфигурирования ListView, данные для отображения могут отсутствовать, например, если они подгружаются из базы данных или удаленного веб-сервиса. В этом нет ничего необычного, их можно добавить и потом. Этот момент будет разобран в следующих статьях. 

**4. Создаем layout одного элемента списка.**

Для этого в AndroidStudio нажимаем правой кнопкой мыши на папке res > layout и создаем новый Layout resource file.

ArrayAdapter умеет работать только с макетами, в которых есть только одно текстовое поле для заполнения. Обычно это TextView. В макете могут присутствовать другие View элементы, но ArrayAdapter не сможет ничего с ними сделать.

Наш макет будет выглядеть так:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9 | <?xml version="1.0" encoding="utf-8"?>  <TextView      android:id="@+id/title"      xmlns:android="http://schemas.android.com/apk/res/android"      android:layout\_width="match\_parent"      android:layout\_height="wrap\_content"      android:gravity="center\_vertical"      android:padding="16dp"      android:textSize="16sp"/> |

**id** нужен для того, чтобы адаптер мог получить ссылку на TextView (хотя если макет состоит из одного TextView, то этот параметр можно опустить)

**layout\_with** и**layout\_height** задают размеры TextView. В данном случае он растянется по горизонтали на всю ширину, предоставленную родительским компонентом (ListView), а по высоте будет равен высоте текста + padding параметр.

**padding** – задается в dp (density-independent pixel) единицах и представляет собой пространство между текстом и внешней границей TextView

**textSize** – определяет размер текста и задается в sp (scale-independent pixel) единицах 

**Замечание!** Android SDK имеет собственный набор ресурсов (картинки, макеты, наборы цветов и так далее), которые разработчик может использовать в своих приложениях. В данном случае мы могли бы применить готовый макет android.R.layout.simple\_list\_item\_1, который имеет такую же структуру — один TextView.

**5. Создаем адаптер и передаем его ListView.**

Адаптер - это объект, создающий View объект на основе переданного ему layout файла и заполняющий этот объект данными. В этой роли может выступать класс, который реализует  интерфейс Adapter. Для упрощения задачи, мы используем готовый класс адаптера — ArrayAdapter. Это самый простой адаптер в Android SDK, поэтому с него и стоит начать знакомство.

Класс ArrayAdapter имеет несколько конструкторов, нам подойдут вот такие:

|  |  |
| --- | --- |
| 1  2 | public ArrayAdapter(Context context, int resource, List<T> objects)  public ArrayAdapter(Context context, int resource, int textViewResourceId, List<T> objects) |

**context** – это объект, который используется адаптером для создания View объектов на основе переданного макета. Обычно здесь передается ссылка на Activity.

**resource** – это id макета элемента списка, то есть id layout файла.

**textViewResouceId** – id элемента, отображающего текст. В нашем случае это id TextView.

**objects** – ссылка на коллекцию, содержащую данные

Если layout элемента списка состоит только из TextView (как в нашем случае), достаточно первого конструктора. Если layout более сложный (из нескольких View объектов), нужно использовать второй конструктор и передавать адаптеру id текстового поля.

В нашем случае код будет выглядеть так:

|  |  |
| --- | --- |
| 1  2 | //MainActivity.java > onCreate method  adapter = new ArrayAdapter<>(this, R.layout.list\_item, list); |

**6. Получаем ссылку на ListView и передаем ему адаптер.**

Тут все просто, ссылку на View объект в Activity мы получаем с помощью метода findViewById(int id).

|  |  |
| --- | --- |
| 1  2  3  4 | //MainActivity.java > onCreate method    ListView listView = (ListView)findViewById(R.id.list\_view);  listView.setAdapter(adapter); |

Все. С этого момента ListView уже будет работать. Можно загрузить приложение в телефон или эмулятор и убедиться в этом. 

**7. Добавляем обработчики нажатий — listener объекты.**

Имея ссылку на ListView, разработчик может назначить ему обработчики событий. Наиболее часто используемые события — это кратковременное нажатие на элемент списка и длительное нажатие. Для регистрации обработчиков этих событий используются следующие методы:

|  |  |
| --- | --- |
| 1  2 | public void setOnItemClickListener(OnItemClickListener listener)  public void setOnItemLongClickListener(OnItemLongClickListener listener) |

Эти методы принимают ссылки на объекты, реализующие интерфейсы OnItemClickListener и OnItemLongClickListener. Каждый интерфейс содержит по одному методы, которые будут вызываться ListView при наступлении соответствующего события. Для интерфейса OnItemClickListener метод выглядит так:

|  |  |
| --- | --- |
| 1 | void onItemClick(AdapterView<?> parent, View view, int position, long id); |

**parent** – ссылка на ListVew

**view** – ссылка на view объект, на который было произведено нажатие.

**position** – позиция view  в адаптере

**id** – идентификационный номер элемента списка, обычно совпадает с position.

Самый простой вариант добавления обработчика состоит в использовании анонимного класса. Вот так:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | listView.setOnItemClickListener(new AdapterView.OnItemClickListener() {                @Override              public void onItemClick(AdapterView<?> parent, View view, int position, long id) {                    //do something              }          }); |

Многие разработчики не любят подобное нагромождение кода и предпочитают реализовать требуемый интерфейс в Activity (или Fragment) и передавать ссылку на нее. Вот так:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16 | public class MainActivity extends AppCompatActivity implements AdapterView.OnItemLongClickListener{        @Override      protected void onCreate(Bundle savedInstanceState) {          .....            //Adds listener to process events          listView.setOnItemLongClickListener(this);      }        @Override      public void onItemLongClick(AdapterView<?> parent, View view, int position, long id) {          //do something      }    } |

В обработчике события обычно выполняется какая-либо работа с нажатым View или данными, соответствующими этому элементу списка. Получить данные из обработчика можно несколькими способами. Самый простой способ состоит в использовании ссылки на объект, в котором хранит данные. В нашем случае это ссылка на ArrayList.

|  |  |
| --- | --- |
| 1  2  3  4 | public void onItemClick(AdapterView<?> parent, View view, int position, long id) {      String data = list.get(position);      ….  } |

Также можно получить данные, использую объект parent, который передается обработчику. На мой взгляд - это более правильный способ доступа к данным.

|  |  |
| --- | --- |
| 1  2  3  4 | public void onItemClick(AdapterView<?> parent, View view, int position, long id) {      String data = (String)parent.getItemAtPosition(position);      ….  } |

Полный код Activity с 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  62  63  64  65  66  67  68  69 | package ru.codeandlife.listviewtest1;    import android.support.v7.app.AppCompatActivity;  import android.os.Bundle;  import android.view.View;  import android.widget.AdapterView;  import android.widget.ArrayAdapter;  import android.widget.ListView;  import android.widget.SimpleAdapter;  import android.widget.Toast;    import java.util.ArrayList;  import java.util.List;     /\*\* It's a simple example how to use ListView with ArrayAdapter   \*  @Author Pavel Bobkov   \* \*/    public class MainActivity extends AppCompatActivity implements          AdapterView.OnItemLongClickListener{        private List<String> list;      private ArrayAdapter<String> adapter;        @Override      protected void onCreate(Bundle savedInstanceState) {          super.onCreate(savedInstanceState);          setContentView(R.layout.activity\_main);            //Creates ArrayList and fake data          list = new ArrayList<>(20);          for(int i = 0; i < 20; i++){              list.add("item " + Integer.toString(i));          }            //Creates ArrayAdapter and configures ListView          adapter = new ArrayAdapter<>(this, R.layout.list\_item, list);          ListView listView = (ListView)findViewById(R.id.list\_view);          listView.setAdapter(adapter);            //Adds listener to process events          listView.setOnItemClickListener(new AdapterView.OnItemClickListener() {              @Override              public void onItemClick(AdapterView<?> parent, View view, int position, long id) {                  String data;                    //There are a few variants how to get data                  data = list.get(position);                  //data = (String)parent.getItemAtPosition(position);                  //data = (String)parent.getAdapter().getItem(position);                    //Shows a simple message                  Toast.makeText(MainActivity.this,                          "OnItemClickListener\n" + data, Toast.LENGTH\_SHORT).show();              }          });            listView.setOnItemLongClickListener(this);      }        @Override      public boolean onItemLongClick(AdapterView<?> parent, View view, int position, long id) {          String data = (String)parent.getItemAtPosition(position);          Toast.makeText(MainActivity.this,                  "OnItemLongClickListener\n" + data, Toast.LENGTH\_SHORT).show();          return true;      }    } |