**Прокручиваемый список ListView**

[**Пример множественного выбора**](http://java-online.ru/android-listview.xhtml#multiplychoice)  
[**Наборы данных ArrayMap, SparseArray**](http://java-online.ru/android-listview.xhtml#arraymap)  
[**Системные разметки ListView**](http://java-online.ru/android-listview.xhtml#listview-layout)

Компонент *ListView* представляет собой прокручиваемый список элементов. По сравнению с *TextView* и другими простыми элементами компонент *ListView* намного сложнее в использовании. Сложности связаны с тем, что программирование прокручиваемого списка включает определение интерфейса и формирование набора данных. За набор данных отвечает *Adapter*.

**Адаптеры списков**

Прежде чем говорить о *ListView* необходимо пару слов сказать об адаптерах, которые формируют данные для прокручиваемого списка. Адаптер выбирает по порядку предоставленные данные и размещает их в списке. При этом для списка *ListView* адаптер на лету создаёт компоненты *TextView* и помещает в них информацию в нужном виде. Исходные данные для списка могут быть определены как в массиве, так и в базе данных; для второго случая будет использоваться адаптер взаимодействия с сервером базы данных. Разработчик может создать свой собственный адаптер.

В системе существуют уже готовые адаптеры. Например, *ArrayAdapter* использует массив значений, а *CursorAdapter* работает с объектом типа *Cursor*, используемый в сервере базы данных. Ниже представлены готовые адаптеры :

|  |  |
| --- | --- |
| • ArrayAdapter<T> | адаптер из массива данных формирует массив компонентов TextView для списка ListView; |
| • SimpleAdapter | адаптер формирует данные более сложной структуры; |
| • ListAdapter | адаптер формирования данных для ListView; строго говоря, это класс-интерфейс, который можно использовать как в ArrayAdapter, так и в SimpleAdapter; |
| • SpinnerAdapter | адаптер формирования данных для компонента Spinner; |
| • CursorAdapter | адаптер формирует данные для ListView с использованием курсора (данные должны иметь колонку с наименованием "\_id"); |
| • ResourceCursorAdapter | адаптер дополняет свойства CursorAdapter и может создавать "виды" из ресурсов; |
| • SimpleCursorAdapter | расширяет свойства ResourceCursorAdapter и формирует компоненты TextView/ImageView из содержащихся в курсоре столбцов; |
| • HeaderViewListAdapter | расширенный вариант ListAdapter, когда в ListView определяются заголовки. |

Адаптер стандартного списка ArrayAdapter имеет конструктор следующего вида :

ArrayAdapter(Context context, int resourceId, String[] data);

где :

* **context** - текущий контекст;
* **resourceId** - идентификатор ресурса с разметкой для каждой строки; можно использовать системную разметку, например [android.R.layout.simple\_list\_item\_1](http://java-online.ru/android-listview.xhtml#listview-layout), или собственную разметку;
* **data** - массив строк.

**BaseAdapter**

Основой представленных выше адаптеров является базовый абстрактный класс *BaseAdapter*. Если необходимо создать собственный адаптер, например, для специального управления данными или для кэширования с целью повышения производительности работы приложения, то можно использовать **BaseAdapter**, свойства которого можно расширить.

*BaseAdapter* содержит несколько методов, которые необходимо переопределить. Например, метод *getCount()* позволяет определить количество отображаемых объектов. Второй, и самый важный метод адаптера — это метод *getView()*. Данный метод вызывается для каждого элемента списка, чтобы получить информацию для отображения. Метод *getVew()* содержит параметр *convertView*, позволяющий использовать заново уже существующий элемент списка, который вышел из видимой области экрана. Если *convertView* не пустой, то он может быть использован заново, чтобы не грузить разметку списка, что способствует росту производительности.

Метод *getView()* возвращает View компонент, который фактически является контейнером *ViewGroup*, содержащий в себе другие компоненты, например, *ImageView* или *TextView*.

**Класс ListActivity**

Если основная функция активности связана с представлением списка и выбором из него определенного элемента, то можно класс активности наследовать от *ListActivity*, расширяющий (extends) свойства *Activity*. Это значительно упростит использование списка *ListView*.

Класс **ListActivity** содержит *ListAdapter*, ответственный за управление набором данных. Адаптер с набором данных должен быть определен в методе активности *onCreate()* вызовом метода *setListAdapter(ListAdapter)*, получающий в качестве параметра объект адаптера.

Класс *ListActivity* имеет метод *onListItemClick()*, вызываемый при выборе одного из элементов списка. Этот метод позволяет получить доступ к выбранному элементу.

**Пример использования ListActivity**

Рассмотрим пример использования класса *ListActivity*. Для этого создадим новый модуль ([Module](http://java-online.ru/android-module.xhtml)) и назовем его *p07listview*. В этом модуле создадим список из текстовых записей. При выборе одного из элементов списка выведем сообщение [Toast](http://java-online.ru/android-toast.xhtml) с соответствующим названием элемента. Ниже представлен листинг активности MainActivity.

**Листинг активности MainActivity**

import android.app.Activity;

import android.app.ListActivity;

import android.os.Bundle;

import android.view.View;

import android.widget.AdapterView;

import android.widget.ArrayAdapter;

import android.widget.ListView;

import android.widget.TextView;

import android.widget.Toast;

public class MainActivity extends ListActivity

{

String[] data = new String[] {"Алексей", "Иван" ,

"Сергей" , "Николай",

"Михаил" , "Виктор"};

@Override

protected void **onCreate**(Bundle savedInstanceState)

{

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_main);

ArrayAdapter<String> adapter;

adapter = new ArrayAdapter<String>(this,

android.R.layout.simple\_list\_item\_1,

data);

setListAdapter(adapter);

}

@Override

protected void **onListItemClick**(ListView l, View view,

int position, long id) {

String item;

item = (String) getListAdapter().getItem(position);

Toast.makeText(this, item + " выбран",

Toast.LENGTH\_SHORT).show();

}

}

В методе активности *onCreate* создается адаптер массива текстовых строк, который подключается к списку. Метод *onListItemClick()*, вызываемый при выборе элемента из списка, формирует и показывает сообщение *Toast*.

Android имеет некоторые предопределенные разметки адаптера. В активности примера применена разметка [android.R.layout.simple\_list\_item1](http://java-online.ru/android-listview.xhtml#listview-layout).

При использовании предопределенной разметки необходимо соответствующим образом определять идентификатор компонента ListView; он должен иметь значение "@android:id/list". В противном случае Android Studio выбросит Exception в режиме run-time с сообщением о несоответствии значения идентификатора компонента. Ниже представлен листинг описания интерфейса активности **res/layout/activity\_main.xml**; интерфейс примера включает только один компонент типа *ListView*.

**Листинг activity\_main.xml**

<?xml version="1.0" encoding="utf-8"?>

<LinearLayout

xmlns:android="http://schemas.android.com/apk/res/android"

xmlns:app="http://schemas.android.com/apk/res-auto"

xmlns:tools="http://schemas.android.com/tools"

android:layout\_width="match\_parent"

android:layout\_height="match\_parent"

tools:context=".MainActivity">

<ListView

android:id="@android:id/list"

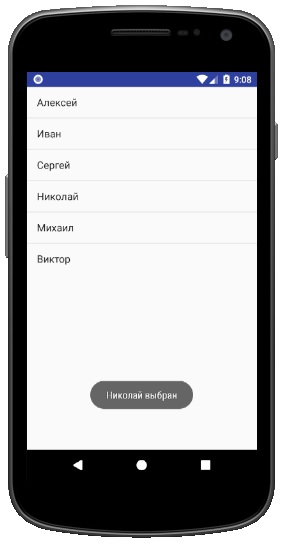
android:layout\_width="match\_parent"

android:layout\_height="wrap\_content" >

</ListView>

</LinearLayout>

На следующем скриншоте представлен интерфейс стартованного примера после выбора одного из элементов списка. В нижней части отображается соответствующее сообщение *Toast*.



**Системные разметки**

В примере была использована готовая системная разметка **android.R.layout.simple\_list\_item\_1**, состоящая из одного *TextView*. В этой разметке определен интерфейс компонента. Android Studio позволяет просмотреть исходный код этой разметки. Для этого следует установить курсор на **simple\_list\_item\_1** и нажать клавиши **Ctrl+B**. Ниже представлен листинг разметки :

<?xml version="1.0" encoding="utf-8"?>

<!--

Copyright (C) 2006 The Android Open Source Project

Licensed under the Apache License, Version 2.0

(the "License"); you may not use this file except

in compliance with the License. You may obtain a

copy of the License at

http://www.apache.org/licenses/LICENSE-2.0

Unless required by applicable law or agreed to in writing,

software distributed under the License is distributed on

an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY

KIND, either express or implied. See the License for the

specific language governing permissions and limitations

under the License.

-->

<TextView

xmlns:android="http://schemas.android.com/apk/res/android"

android:id="@android:id/text1"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:textAppearance=

"?android:attr/textAppearanceListItemSmall"

android:gravity="center\_vertical"

android:paddingStart=

"?android:attr/listPreferredItemPaddingStart"

android:paddingEnd=

"?android:attr/listPreferredItemPaddingEnd"

android:minHeight=

"?android:attr/listPreferredItemHeightSmall" />

В листинге в качестве разметки представлен один компонент TextView с набором атрибутов. Имеется несколько вариантов системных разметок :

**android.R.layout.simple\_list\_item\_2**  
 состоит из двух компонентов TextView, расположенных друг над другом. Верхний играет роль заголовка, нижний – роль подтекста.

**android.R.layout.simple\_list\_item\_checked**  
 используется компонент CheckedTextView, справа от которого будет находиться флажок выбора элемента.

**android.R.layout.activity\_list\_item**  
 используется компонент TextView, слева от которого располагается иконка ImageView с идентификатором android.resource.id.Icon.

**Пример множественного выбора**

В рассмотренном выше примере выбирался только один элемент из списка. Компонент **ListView** позволяет сделать не только единичный выбор, но и множественный. В этом случае необходимо использовать соответствующую разметку (*simple\_list\_item\_checked*) и установить свойство *ChoiceMode* в описании интерфейса компонента равным *multiplyChoice*. Множественный выбор также можно установить программно при помощи метода *setChoiceMode(ListView.CHOICE\_MODE\_MULTIPLE)*.

В основе примера множественного выбора элементов из списка будем использовать предыдущий пример *p07listview*. Необходимо внести изменения как в описание интерфейса, так и в описание активности. Теперь листинг описания интерфейса активности **res/layout/activity\_main.xml** примет следующий вид :

**Листинг activity\_main.xml**

<?xml version="1.0" encoding="utf-8"?>

<LinearLayout

xmlns:android="http://schemas.android.com/apk/res/android"

xmlns:app="http://schemas.android.com/apk/res-auto"

xmlns:tools="http://schemas.android.com/tools"

android:layout\_width="match\_parent"

android:layout\_height="match\_parent"

tools:context=".MainActivity">

<ListView

android:id="@+id/list"

android:choiceMode="multipleChoice"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content" >

</ListView>

</LinearLayout>

В чем отличие данного описания интерфейса от предыдущего? В первую очередь изменение коснулось значения идентификатора объекта *android:id*, связанное с тем, что в этом примере активность будет наследоваться от обычного класса *Activity*, а не от *ListActivity*, как это было ранее. Кроме этого, добавлен атрибут множественного выбора *android:choiceMode*. Если не определить данный атрибут, и использовать разметку*simple\_list\_item\_checked*, то невозможно будет выделить ни одного элемента из списка : перехватить нажатие на элементе списка можно, но установить флажок нельзя. При значении атрибута *android:choiceMode* равным "singleChoice" можно будет выбрать/выделить только один элемент из списка.

**Ресурсный файл**

Второе изменение в примере коснётся описания массива строк. Нехорошо описывать элементы массива в java-коде, как это было выполнено в предыдущем примере. Правильнее хранить их в ресурсных файлах. Тем более, что это удобнее с точки зрения [локализации](http://java-online.ru/android-i18n.xhtml) интерфейса. Найдем в проекте файл с ресурсами **res/values/strings.xml** и добавим туда массив строк с именами. В итоге должен получиться файл с таким содержимым :

<resources>

<string name="app\_name">p07listview</string>

<string-array name="names">

<item>Алексей</item>

<item>Иван</item>

<item>Сергей</item>

<item>Николай</item>

<item>Михаил</item>

<item>Виктор</item>

</string-array>

</resources>

Для чтения данного массива значений из ресурсного файла необходимо использовать следующее выражение :

String[] data = getResources().getStringArray(R.array.names);

**Наборы данных ArrayMap, SparseArray**

Прежде чем переходить к классу «активности» необходимо рассмотреть объекты формирования массивов элементов в Android. Поскольку мы будем использовать множественный выбор элементов, то набор выделенных записей необходимо где-то сохранить. Для этого будем использовать класс типа **ArrayMap**.

Android имеет собственную реализацию набора данных типа [HashMap](http://java-online.ru/java-map.xhtml), это класс [ArrayMap](https://developer.android.com/reference/android/support/v4/util/ArrayMap), использующий принцип "ключ-значение". В отличие от *HashMap* класс *ArrayMap* внутри содержит два массива : в первом массиве в отсортированном виде хранятся хэш-коды ключей, во втором массиве – ключи и значения. Вторым важным отличием *ArrayMap* является возможность использования прохода по индексам в цикле (это обеспечивается вторым внутренним массивом). Рекомендуется использовать *ArrayMap* при ограниченном количестве объектов в наборе, когда количество операций добавления элементов несущественно. Пример использования класса *ArrayMap* :

ArrayMap<String, String> array = new ArrayMap<>();

array.put("Администратор", "admin" );

array.put("Разработчик" , "developer");

array.put("Менеджер" , "manager" );

for (int i = 0; i < array.size(); i++) {

String key = array.keyAt(i);

String value = array.valueAt(i);

Log.i(key, value);

}

Помимо *ArrayMap* android включает в свою библиотеку классы типа *SparseArray*. Класс **SparseArray** подобен *ArrayMap* с той лишь разницей, что в качестве ключа используются примитивные типы. Библиотека android содержит несколько разновидностей класса *ArrayMap* для различных типов : LongSparseArray, SparseIntArray, SparseBooleanArray. В следующей таблице представлены соответствия классов *ArrayMap* и *SparseArray*объектам *HashMap* :

|  |  |
| --- | --- |
| ArrayMap<K,V> | HashMap<K,V> |
| SparseArray<Object> | HashMap<Integer, Object> |
| SparseBooleanArray | HashMap<Integer, Boolean> |
| SparseIntArray | HashMap<Integer, Integer> |
| SparseLongArray | HashMap<Integer, Long> |
| LongSparseArray<Object> | HashMap<Long, Object> |

**Листинг активности**

Класс активности *MainActivity* является наследником *Activity*. В активности набор записей из ресурсного файла сначала заносится в массив *data*. После этого массив используется при создании адаптера типа *ArrayAdapter*, который регистрируется в прокручиваемом списке *listView* вызовом метода *setAdapter*. К компоненту списка *listView* подключен слушатель выделения элемента. При выделении любой записи набора слушатель получает массив логических значений *SparseBooleanArray* и в цикле формирует строку сообщения с перечнем выбранных записей, которые отображает в *Toast*.

import android.app.Activity;

import android.os.Bundle;

import android.view.View;

import android.widget.Toast;

import android.widget.ListView;

import android.widget.TextView;

import android.widget.AdapterView;

import android.widget.ArrayAdapter;

import android.util.SparseBooleanArray;

public class MainActivity extends Activity

{

private String[] data;

@Override

protected void onCreate(Bundle savedInstanceState)

{

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_main);

data = getResources().getStringArray(R.array.names);

ArrayAdapter<String> adapter;

adapter = new ArrayAdapter(this,

android.R.layout

.simple\_list\_item\_checked,

data);

ListView listView = (ListView)findViewById(R.id.list);

// listView.setChoiceMode(ListView.CHOICE\_MODE\_MULTIPLE);

listView.setAdapter(adapter);

listView.setOnItemClickListener(

new AdapterView.OnItemClickListener(){

@Override

public void onItemClick(AdapterView<?> parent,

View view, int position,

long id) {

String choosen = "";

SparseBooleanArray checked;

checked = parent.getCheckedItemPositions();

for (int i = 0; i < checked.size(); i++) {

if (checked.valueAt(i)) {

choosen += data[checked.keyAt(i)]

+ " ";

}

}

Toast.makeText(getApplicationContext(),

"выбраны : " + choosen,

Toast.LENGTH\_SHORT).show();

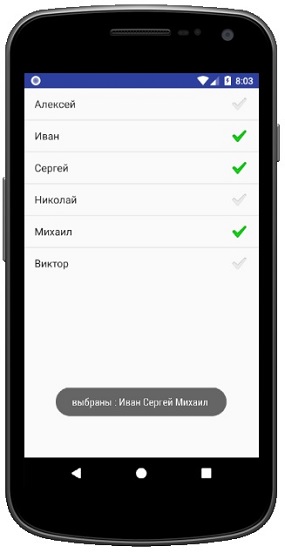
}

});

}

}

На следующем скриншоте представлен интерфейс примера с тремя выбранными записями, отображаемыми в сообщении *Toast*.



Продолжение статьи связано с [настройкой интерфейса](http://java-online.ru/android-listview-custom.xhtml) прокручиваемого списка ListView.