ListView

[Знакомьтесь - ListView](http://developer.alexanderklimov.ru/android/views/listview.php#intro)  
[Адаптеры - заполнение списка данными](http://developer.alexanderklimov.ru/android/views/listview.php#listcontent)  
[Отступление. Использование ресурсов](http://developer.alexanderklimov.ru/android/views/listview.php#arraystring)  
[Собственная разметка](http://developer.alexanderklimov.ru/android/views/listview.php#customlayout)  
[Динамическое заполнение списка](http://developer.alexanderklimov.ru/android/views/listview.php#dynamiccontent)  
[Прослушивание событий элемента ListView](http://developer.alexanderklimov.ru/android/views/listview.php#listener)  
[Программное нажатие на элемент списка](http://developer.alexanderklimov.ru/android/views/listview.php#perform)  
[ListView не реагирует на нажатия](http://developer.alexanderklimov.ru/android/views/listview.php#problems)  
[Настраиваем внешний вид ListView](http://developer.alexanderklimov.ru/android/views/listview.php#design)  
[Пользовательский селектор](http://developer.alexanderklimov.ru/android/views/listview.php#customselector)  
[Множественный выбор](http://developer.alexanderklimov.ru/android/views/listview.php#multichoice)  
[Кнопка под списком](http://developer.alexanderklimov.ru/android/views/listview.php#underlist)  
[Плавная прокрутка в начало списка или любую позицию](http://developer.alexanderklimov.ru/android/views/listview.php#smoothscroll)  
[Настраиваем прокрутку](http://developer.alexanderklimov.ru/android/views/listview.php#customscroll)

Знакомьтесь - ListView

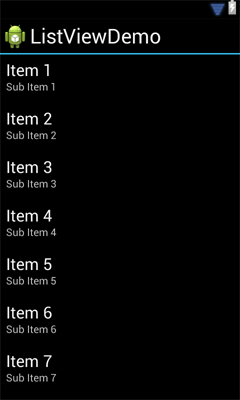
Виджет **ListView** представляет собой прокручиваемый список элементов. Очень популярен на мобильных устройства из-за своего удобства. Даже кот способен пользоваться этим элементом, проводя лапкой по экрану вашего телефона.



Находится в разделе **Containers**.

Компонент **ListView** более сложен в применении по сравнению с **TextView** и другим простыми элементами. Работа со списком состоит из двух частей. Сначала мы добавляем на форму сам **ListView**, а затем заполняем его элементами списка.

Рассмотрим для начала самый простой пример. Поместите на форму компонент **ListView**. Вы увидите, что список будет содержать несколько элементов **Item** и **Sub Item**.



Однако, если посмотрим XML-код, то там ничего не увидим.

<ListView

android:id="@+id/listView"

android:layout\_width="match\_parent"

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

</ListView>

Переходим в класс активности и пишем в методе **onCreate()** следующий код:

// получаем экземпляр элемента ListView

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

// определяем массив типа String

final String[] catNames = new String[] {

"Рыжик", "Барсик", "Мурзик", "Мурка", "Васька",

"Томасина", "Кристина", "Пушок", "Дымка", "Кузя",

"Китти", "Масяня", "Симба"

};

// используем адаптер данных

ArrayAdapter<String> adapter = new ArrayAdapter<>(this,

android.R.layout.simple\_list\_item\_1, catNames);

listView.setAdapter(adapter);

Вот и всё. Давайте разберёмся с кодом.

Адаптеры - заполнение списка данными

Компоненту **ListView** требуются данные для наполнения. Источником наполнения могут быть массивы, базы данных. Чтобы связать данные со списком, используется так называемый адаптер.

Адаптер обычно создаётся при помощи конструкции *new ArrayAdapter(Context context, int textViewResourceId, String[] objects)*.

* **context** - текущий контекст
* **textViewResourceId** - идентификатор ресурса с разметкой для каждой строки. Можно использовать системную разметку с идентификатором **android.R.layout.simple\_list\_item\_1** или создать собственную разметку
* **objects** - массив строк

Метод **setAdapter(ListAdapter)** связывает подготовленный список с адаптером.

Переходим к java-коду. Сначала мы получаем экземпляр элемента **ListView** в методе **onCreate()**. Далее мы определяем массив типа **String**. И, наконец, используем адаптер данных, чтобы сопоставить данные с шаблоном разметки. Выбор адаптера зависит от типа используемых данных. В нашем случае мы использовали класс **ArrayAdapter**.

Отступление

Если вы будете брать строки из ресурсов, то код будет таким:

final String[] catnames = {

getResources().getString(R.string.name1),

getResources().getString(R.string.name2),

getResources().getString(R.string.name3),

getResources().getString(R.string.name4),

getResources().getString(R.string.name5),

};

А будет еще лучше, если вы воспользуетесь специально предназначенным для этого случая типом ресурса **<string-array>**. В файле **res/values/strings.xml** добавьте следующее:

<string-array name="cat\_names">

<item>Рыжик</item>

<item>Барсик</item>

<item>Мурзик</item>

<item>Мурка</item>

<item>Васька</item>

<item>Томасина</item>

<item>Кристина</item>

<item>Пушок</item>

<item>Дымка</item>

<item>Кузя</item>

<item>Китти</item>

<item>Масяня</item>

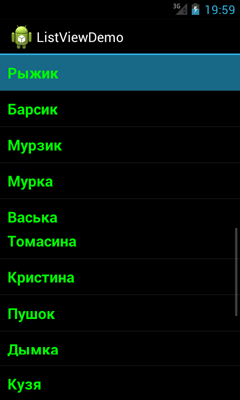
<item>Симба</item>

</string-array>

И тогда в коде используйте для объявления массива строк:

String[] catnames = getResources().getStringArray(R.array.cat\_names);

Запустив проект в эмуляторе, вы увидите работающий пример прокручиваемого списка. Правда, созданный список пока не реагирует на нажатия. Но при нажатии выбранный элемент выделяется цветным прямоугольником (в версии Android 2.3 был оранжевый, а в Android 4.0 - синий).



Собственная разметка

В примере мы используем готовую системную разметку **android.R.layout.simple\_list\_item\_1**, в которой настроены цвета, фон, высота пунктов и другие параметры. Но нет никаких препятствий самому создать собственную разметку под своё приложение.

Но для начала неплохо бы взглянуть на содержание системной разметки. Существуют различные утилиты и сайты для просмотра исходников Android. Наш *simple\_list\_item\_1* выглядит так (в одной из версий):

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

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

style="?android:attr/listItemFirstLineStyle"

android:paddingTop="2dip"

android:paddingBottom="3dip"

android:layout\_width="fill\_parent"

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

Мы видим, что в качестве разметки используется **TextView** с набором атрибутов.

Если говорить о системных разметках, то имеется несколько вариантов. Вкратце ознакомимся с ними.

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

Состоит из одного **TextView** (см. выше)

|  |
| --- |
| android.resource.id.text1 |

android.R.layout.simple\_list\_item\_2

Состоит из двух **TextView** - один побольше сверху и второй поменьше под ним.

|  |
| --- |
| android.resource.id.text1 |
| android.resource.id.text2 |

android.R.layout.simple\_list\_item\_checked

Справа от **CheckedTextView** будет находиться флажок

|  |
| --- |
| android.resource.id.text1 |

android.R.layout.activity\_list\_item

Слева от **TextView** находится значок **ImageView** с идентификатором **android.resource.id.Icon**.

|  |
| --- |
| android.resource.id.text1 |

Создадим свой шаблон для отдельного пункта списка. Для этого в папке **res/layout/** создадим новый файл **list\_item.xml**:

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

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

android:layout\_width="fill\_parent"

android:layout\_height="fill\_parent"

android:padding="10dp"

android:textColor="#00FF00"

android:textSize="20sp"

android:textStyle="bold" >

</TextView>

В некоторых случаях желательно установить атрибут **android:background="?android:attr/activatedBackgroundIndicator"** у родительского элемента, чтобы элементы списка реагировали на нажатие изменением цвета.

Вы можете настраивать все атрибуты у **TextView**, кроме свойства **Text**, так как текст будет автоматически заполняться элементом **ListView** программным путём. Ну, а дальше просто меняете в коде системную разметку на свою:

ArrayAdapter<String> adapter = new ArrayAdapter<String>(this, R.layout.list\_item, catnames);

Динамическое заполнение списка

Рассмотрим пример динамического заполнения списка, когда список изначально пуст и пользователь сам добавляет новые элементы. Разместим на экране текстовое поле, в котором пользователь будет вводить известные ему имена котов. Когда пользователь будет нажимать на клавишу Enter на клавиатуре, то введённое имя кота будет попадать в список.

@Override

public void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.main);

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

final EditText editText = (EditText) findViewById(R.id.editText);

// Создаём пустой массив для хранения имен котов

final ArrayList<String> catnames = new ArrayList<String>();

// Создаём адаптер ArrayAdapter, чтобы привязать массив к ListView

final ArrayAdapter<String> adapter;

adapter = new ArrayAdapter<String>(this,

android.R.layout.simple\_list\_item\_1, catnames);

// Привяжем массив через адаптер к ListView

listView.setAdapter(adapter);

// Прослушиваем нажатия клавиш

editText.setOnKeyListener(new OnKeyListener() {

public boolean onKey(View v, int keyCode, KeyEvent event) {

// TODO Auto-generated method stub

if (event.getAction() == KeyEvent.ACTION\_DOWN)

if (keyCode == KeyEvent.KEYCODE\_ENTER) {

catnames.add(0, editText.getText().toString());

adapter.notifyDataSetChanged();

editText.setText("");

return true;

}

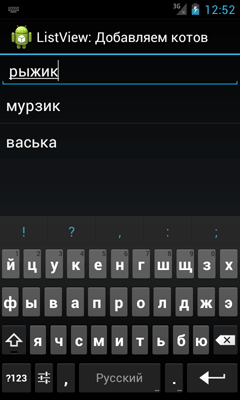
return false;

}

});

}

При нажатии на Enter мы получаем текст из текстового поля и заносим его в массив. А также оповещаем адаптер об изменении, чтобы список автоматически обновил своё содержание.



У нас получился каркас для чата, когда пользователь вводит текст и он попадает в список. Далее надо получить текст от другого пользователя и также добавить в список. К слову сказать, слово **chat** с французского означает "кошка". Но это уже совсем другая история.

Прослушивание событий элемента ListView

Нам нужно реагировать на определенные события, генерируемые элементом ListView, в частности, нас интересует событие, которое возникает, когда пользователь нажимает на один из пунктов списка.

В этом нам поможет метод **setOnItemClickListener** элемента ListView и метод **OnItemClick()** класса **AdapterView.OnItemClickListener**.

lv.setOnItemClickListener(new AdapterView.OnItemClickListener() {

@Override

public void onItemClick(AdapterView<?> parent, View itemClicked, int position,

long id) {

Toast.makeText(getApplicationContext(), ((TextView) itemClicked).getText(),

Toast.LENGTH\_SHORT).show();

}

});

Теперь при нажатии на любой элемент списка мы получим всплывающее сообщение, содержащее текст выбранного пункта.

Естественно, мы можем не только выводить сообщения, но и запускать новые активности и т.п.

lv.setOnItemClickListener(new AdapterView.OnItemClickListener() {

@Override

public void onItemClick(AdapterView<?> parent, View itemClicked, int position,

long id) {

TextView textView = (TextView) itemClicked;

String strText = textView.getText().toString(); // получаем текст нажатого элемента

if(strText.equalsIgnoreCase(getResources().getString(R.string.name1))) {

// Запускаем активность, связанную с определенным именем кота

startActivity(new Intent(this, BarsikActivity.class));

}

}

});

В метод **onItemClick()** передаётся вся необходимая информация, необходимая для определения нажатого пункта в списке. В приведенном выше примере использовался простой способ - приводим выбранный элемент к объекту TextView, так как известно, что все пункты являются элементами TextView (Для дополнительной проверки можете использовать оператор instanceOf). Мы извлекаем текст из выбранного пункта и сравниваем его со своей строкой.

Также можно проверять атрибут **id** для определения нажатия пункта списка.

Программное нажатие на элемент списка

Вдруг вам захочется программно нажать на элемент списка. Мы задали код, который будет выполняться при нажатии, в предыдущем примере. Теперь добавим кнопку и напишем код для щелчка.

public void onClick(View v) {

int activePosition = 0; // первый элемент списка

lv.performItemClick(lv.getAdapter().

getView(activePosition, null, null), activePosition, lv.getAdapter().

getItemId(activePosition));

}

Код громоздкий, но работоспособный.

ListView не реагирует на нажатия

В некоторых случаях нажатия на пунктах меню не срабатывают. Ниже приводятся несколько возможных причин.

Элемент списка содержит **CheckBox**, который также имеет свой слушатель нажатий. Попробуйте удалить фокус у него:

android:focusable="false"

android:focusableInTouchMode="false"

Попробуйте переместить **OnItemClickListener** перед установкой адаптера. Иногда помогает :-)

Элемент списка содержит **ImageButton**. Установите фокус в **false**:

ImageButton imageButton = (ImageButton) convertView.findViewById(R.id.imageButton);

imageButton.setFocusable(false);

Элемент списка содержит **TextView**. Если вы используете атрибут **android:inputType="textMultiLine"**, то замените его на **android:minLines/android:maxLines**.

Элемент списка содержит **TextView**, содержащий ссылку на веб-страницу или электронный адрес. Удалите атрибут **android:autoLink**.

Настраиваем внешний вид ListView

У **ListView** есть несколько полезных атрибутов, позволяющих сделать список более привлекательным. Например, у него есть атрибут **divider**, который отвечает за внешний вид разделителя, а также атрибут **dividerHeight**, отвечающий за высоту разделителя. Мы можем установить какой-нибудь цвет или даже картинку для разделителя. Например, создадим для разделителя цветовой ресурс с красным цветом, а также ресурс размера для его высоты:

<color name="reddivider">#FF0000</color>

<dimen name="twodp">2dp</dimen>

Далее присвоим созданный ресурс атрибуту **divider**, а также зададим его высоту в атрибуте **dividerHeight** у нашего элемента ListView:

<ListView

android:id="@+id/listView1"

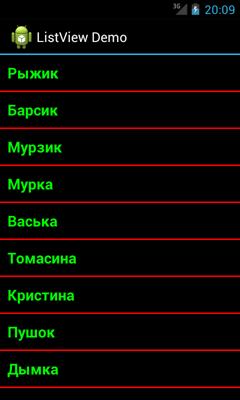
android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

**android:divider="@color/reddivider"**

**android:dividerHeight="@dimen/twodp" >**

</ListView>



Если вас не устраивает стандартный разделитель, что можете нарисовать какую-нибудь волнистую черту, сохранить ее в PNG-файле и использовать как drawable-ресурс. Проделайте это самостоятельно.

Можно работать с данными атрибутами программно:

ColorDrawable divcolor = new ColorDrawable(Color.DKGRAY);

listView.setDivider(divcolor);

listView.setDividerHeight(2);

Если хотите убрать разделители, то используйте прозрачный цвет.

listView.setDivider(getResources().getDrawable(android.R.color.transparent));

Заметил, что порядок вызова двух методов важен, если установку высоты вызвать перед установкой цвета разделителя, то метод затирает цвет и результат будет такой же, как с прозрачным цветом.

Обратите внимание, что по умолчанию разделитель не выводится перед первым и последним элементом списка. Если вы хотите изменить эти настройки, то используйте свойства **Footer dividers enabled** (атрибут **footerDividersEnabled**) и **Header dividers enabled** (атрибут **headerDividersEnabled**):

...

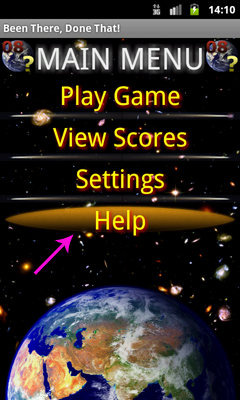
android:footerDividersEnabled="true"

android:headerDividersEnabled="true"

...

Пользовательский селектор

Мы уже видели, что по умолчанию выбранный элемент списка выделяется при помощи цветной полоски. Данный селектор также можно настроить через атрибут **android:listSelector**. Создайте какую-нибудь текстуру для селектора и привяжите его через ресурс. Вот образец текстурированного ореола желтого цвета для селектора.



Нужно подготовить сначала файл **res/drawable/selector.xml**:

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

<item android:state\_pressed="true" android:drawable="@drawable/bground"/>

</selector>

Если вам нужно сразу подсветить нужный элемент списка при запуске программы, то используйте связку двух методов:

listView.requestFocusFromTouch();

listView.setSelection(4); // выбираем 5 пункт списка

Множественный выбор

**ListView** позволяет выбирать не только один пункт, но и несколько. В этом случае нужно установить свойство **Choice Mode** в значение **multiplyChoice**, что соответствует атрибуту**android:choiceMode="multipleChoice"**.

Также множественный выбор можно установить программно при помощи метода **setChoiceMode(ListView.CHOICE\_MODE\_MULTIPLE)**.

Теперь, если создать массив строк, например список продуктов для кошачьего завтрака, то получим следующий результат.

<?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/textView1"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="Завтрак для кота" />

<ListView

android:id="@+id/listView1"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:choiceMode="multipleChoice" >

</ListView>

</LinearLayout>

public class MultiChoiceListViewActivity extends Activity {

ListView choiceList;

TextView selection;

String[] foods = { "Молоко", "Сметана", "Колбаска", "Сыр", "Мышка",

"Ананас", "Икра черная", "Икра кабачковая", "Яйцо" };

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

@Override

public void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.main);

selection = (TextView) findViewById(R.id.textView1);

choiceList = (ListView) findViewById(R.id.listView1);

ArrayAdapter<String> adapter = new ArrayAdapter<String>(this,

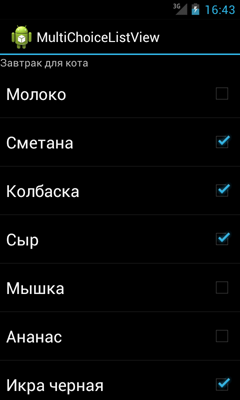
android.R.layout.simple\_list\_item\_multiple\_choice, foods);

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

choiceList.setAdapter(adapter);

}

}



Осталось только программно получить отмеченные пользователем элементы списка. Вот мой список продуктов, который я хочу предложить коту. Надеюсь, ему понравится мой выбор. Выбранные элементы будем помещать в TextView:

choiceList.setOnItemClickListener(new OnItemClickListener() {

@Override

public void onItemClick(AdapterView<?> parent, View v,

int position, long id) {

// TODO Auto-generated method stub

// Очистим TextView

selection.setText("");

// получим булев массив для каждой позиции списка

// Объект SparseBooleanArray содержит массив значений, к которым можно получить доступ

// через valueAt(index) и keyAt(index)

SparseBooleanArray chosen = ((ListView) parent).getCheckedItemPositions();

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

// если пользователь выбрал пункт списка,

// то выводим его в TextView.

if (chosen.valueAt(i)) {

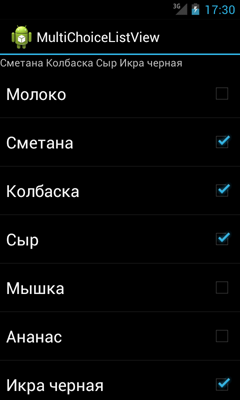
selection.append(foods[chosen.keyAt(i)] + " ");

}

}

}

});



Если нужно получить отдельно список выбранных и невыбранных элементов списка, то можно написать следующее:

choiceList.setOnItemClickListener(new OnItemClickListener() {

@Override

public void onItemClick(AdapterView<?> parent, View v,

int position, long id) {

// TODO Auto-generated method stub

// Clear the TextView before we assign the new content.

selection.setText("");

int cntChoice = choiceList.getCount();

String checked = "";

String unchecked = "";

SparseBooleanArray sparseBooleanArray = choiceList

.getCheckedItemPositions();

for (int i = 0; i < cntChoice; i++) {

if (sparseBooleanArray.get(i) == true) {

checked += choiceList.getItemAtPosition(i).toString()

+ "\\n";

// выводим список выбранных элементов

//selection.setText(checked);

} else if (sparseBooleanArray.get(i) == false) {

unchecked += choiceList.getItemAtPosition(i).toString()

+ "\\n";

// выводим список невыбранных элементов

selection.setText(unchecked);

}

}

}

});

Переменная *checked* будет содержать список выбранных элементов, а переменная *unchecked* - список невыбранных элементов.

Следует отметить, что в примерах использовался старый метод **getCheckedItemPositions()**, доступный с Android 1. В Android 2.2 появился новый метод **getCheckedItemIds()**. Учтите, что с новым методом можно получить массив только выбранных элементов, хотя в большинстве случаев этого достаточно. Но данный метод требует своих заморочек и в данном моём примере он не заработал.

Кнопка под списком

Если вы хотите разместить кнопку под списком, которая бы не зависела от количества элементов в **ListView**, то воспользуйтесь весом (layout\_weight).

<?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" >

<ListView

android:id="@+id/listView1"

android:layout\_width="match\_parent"

android:layout\_height="0dp"

android:layout\_weight="1" >

</ListView>

<Button

android:id="@+id/button1"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:layout\_weight="0"

android:text="Button" />

</LinearLayout>

Плавная прокрутка в начало списка или любую позицию

У списка есть специальный метод **smoothScrollToPosition()**, позволяющий плавно прокрутить до нужного места. Достаточно в методе указать номер позиции для прокрутки:

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

int n = 0; // прокручиваем до начала

listView.smoothScrollToPosition(n);

Учтите, что если элементов в списке несколько сотен и вы запустите плавную прокрутку указанным способом, то процесс может растянуться надолго. Например, коты могут и заснуть, не дождавшись конца операции. Задумайтесь.

Настраиваем прокрутку

У **ListView** есть атрибуты для настройки внешнего вида полосы прокрутки

android:scrollbarTrackVertical="@drawable/scrool\_bg"

android:scrollbarThumbVertical="@drawable/scroll"

Аналогично это применимо к полосам прокрутки у **ScrollView**, **EditText** и т.д.

ListActivity

Если вам нужна форма, состоящая только из списка, то вам проще воспользоваться системным классом **ListActivity** вместо стандартного **Activity**. Именно такой подход описан в документации по ListView. Пример работы описан в статье [ListActivity - создаём прокручиваемый список](http://developer.alexanderklimov.ru/android/listactivity.php).

Дополнительное чтение

[Продвинутые примеры с ListView](http://developer.alexanderklimov.ru/android/views/listview-advanced.php)

[Отдельный раздел о ListView](http://developer.alexanderklimov.ru/android/listview/)