Освой программирование играючи
Сайт Александра Климова

(http://developer.alexanderklimov.ru/)

/ Моя кошка замечательно разбирается в программировании. Стоит мне объяснить проблему ей - и все становится ясно. */*
John Robbins, Debugging Applications, Microsoft Press, 2000

(http://feeds.feedburner.com/alexanderklimov/VJcl)

- Android (http://developer.alexanderklimov.ru/android)
- C#/Visual Basic (http://developer.alexanderklimov.ru/dotnet/)
- Windows Phone (http://developer.alexanderklimov.ru/windowsphone/wp.php)
- WPF (http://developer.alexanderklimov.ru/wpf/wpf.php)
- PHP (http://developer.alexanderklimov.ru/php)
- Arduino (http://developer.alexanderklimov.ru/arduino)

Главная (http://developer.alexanderklimov.ru/android/index.php)
Теория (http://developer.alexanderklimov.ru/android/theory/)
Palette (http://developer.alexanderklimov.ru/android/views.php)
ListView (http://developer.alexanderklimov.ru/android/listview/)
Котошоп (http://developer.alexanderklimov.ru/android/catshop/)
Анимация (http://developer.alexanderklimov.ru/android/animation/)
SQLite (http://developer.alexanderklimov.ru/android/sqlite/)
OpenGL ES (http://developer.alexanderklimov.ru/android/opengles/)
Библиотеки (http://developer.alexanderklimov.ru/android/library/)
Игры (http://developer.alexanderklimov.ru/android/games/)
Wear (http://developer.alexanderklimov.ru/android/wear/)
Эмулятор (http://developer.alexanderklimov.ru/android/emulator/)
Советы (http://developer.alexanderklimov.ru/android/tips-android.php)
Статьи (http://developer.alexanderklimov.ru/android/articles-android.php)
Книги (http://developer.alexanderklimov.ru/android/books.php)

Java. Экспресс-курс (http://developer.alexanderklimov.ru/android/java/java.php)

Дизайн (http://developer.alexanderklimov.ru/android/design/)

Отладка (http://developer.alexanderklimov.ru/android/debug/)

Open Source (http://developer.alexanderklimov.ru/android/opensource.php)

Полезные ресурсы (http://developer.alexanderklimov.ru/android/links.php)

Стильный ListView

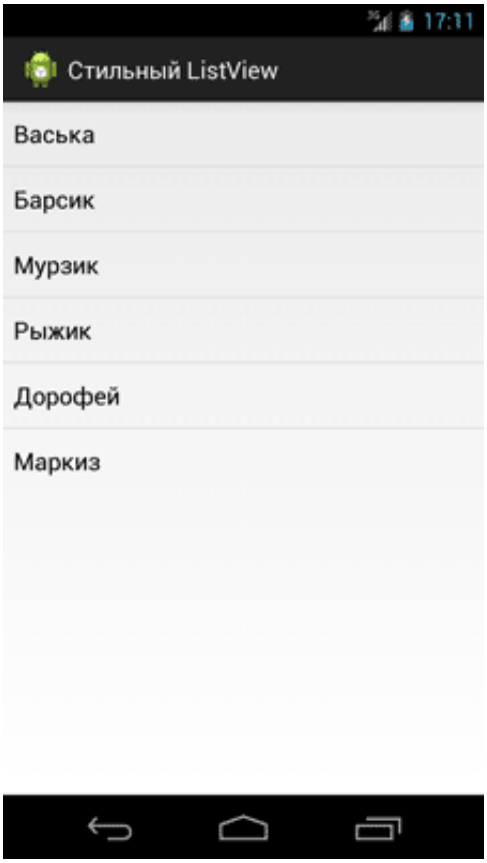
Попробуем сделать ListView красивым. Сначала создадим стандартную заготовку на основе ListActivity. Разметка будет стандартная:

```
<?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="@android:id/list"
        android:layout_width="match_parent"
        android:layout_height="wrap_content" >
    </ListView>

</LinearLayout>
```

Теперь создадим массив кошачьих имён и через адаптер добавим имена в список:

```
package ru.alexanderklimov.test;

import ...

public class TestActivity extends ListActivity {

        private static final String[] catnames = { "Васька", "Барсик", "Мурзик",
                        "Рыжик", "Дорофей", "Маркиз" };

        /** Called when the activity is first created. */
        @Override
        public void onCreate(Bundle savedInstanceState) {
                super.onCreate(savedInstanceState);

                setContentView(R.layout.activity_test);

                ListView listView = getListView();
                listView.setAdapter(new ArrayAdapter<String>(this,
                                android.R.layout.simple_list_item_1, catnames));
        }
}
```

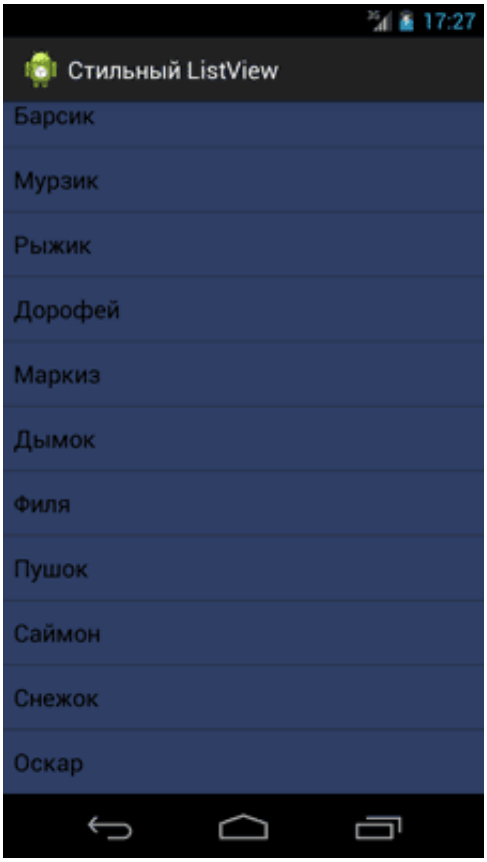
Получим стандартное приложение со списком:



Займёмся стилизацией списка. В файле **res/values/styles.xml** добавим пару стилей для списка:

```
<style name="CustomListView" parent="@android:style/Widget.ListView">
    <item name="android:background">#2F3F66</item>
    <item name="android:fastScrollEnabled">true</item>
</style>
```

Теперь нужно присоединить стиль к ListView через атрибут **style="@style/CustomListView"** и посмотреть, что у нас получилось:



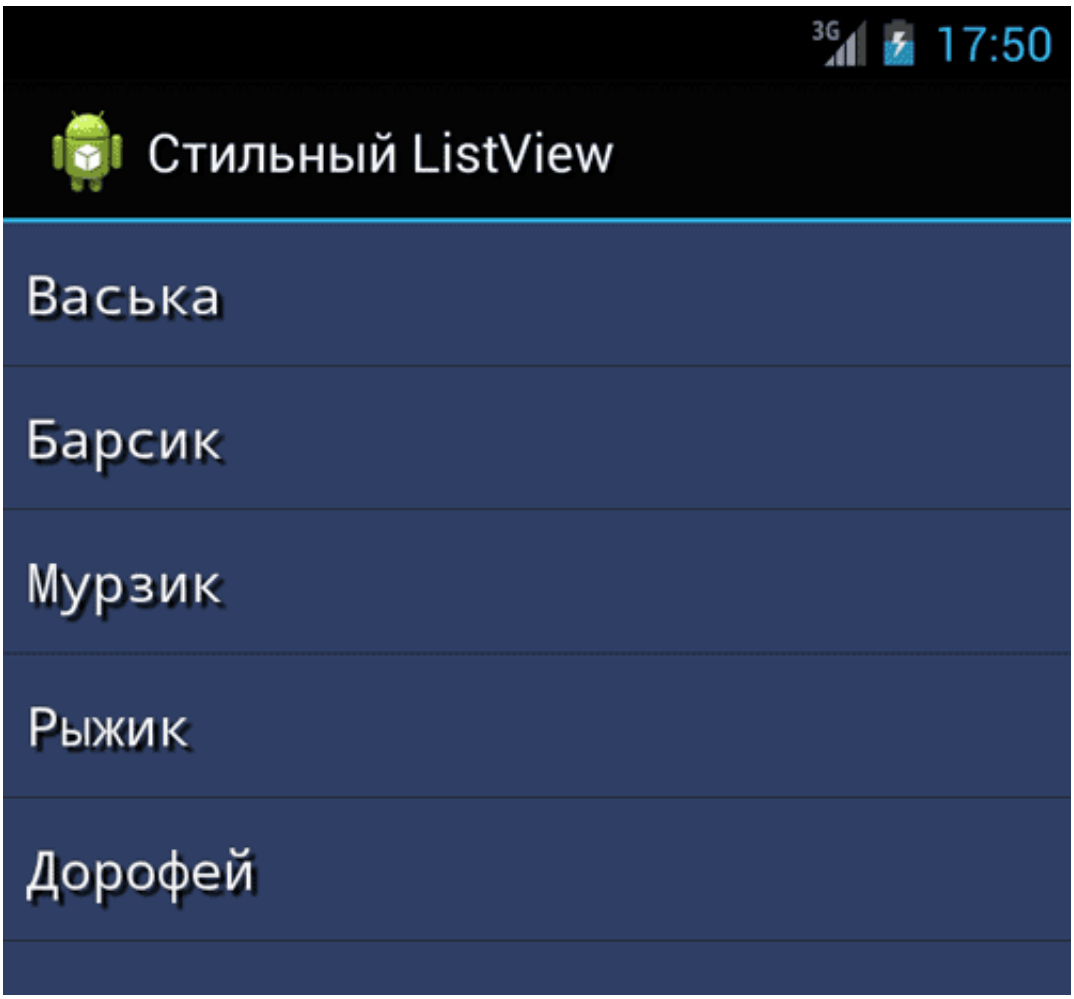
Мы поменяли фон для списка и добавили возможность быстрого скролирования.

Однако, продолжим.

Создадим отдельную тему для активности (или всего приложения):

```
<style name="CustomTheme" parent="@android:style/Theme.Holo">
    <item name="android:listViewStyle">@style/CustomListView</item>
    <item name="android:textViewStyle">@style/CustomTextView</item>
</style>

<style name="CustomListView" parent="@android:style/Widget.ListView">
    <item name="android:background">#2F3F66</item>
    <item name="android:fastScrollEnabled">true</item>
</style>

<style name="CustomTextView" parent="@android:style/Widget.TextView">
    <item name="android:typeface">monospace</item>
    <item name="android:shadowColor">#000000</item>
    <item name="android:shadowRadius">2.</item>
    <item name="android:shadowDx">4</item>
    <item name="android:shadowDy">4</item>
</style>
```

Кроме новой темы мы добавили ещё один стиль для TextView, который отвечает за отдельный элемент списка. Теперь текст будет выглядеть объёмным за счёт тени. Прописываем атрибут **android:theme="@style/CustomTheme"** в манифесте для активности или для всего приложения (application). Смотрим на результат:



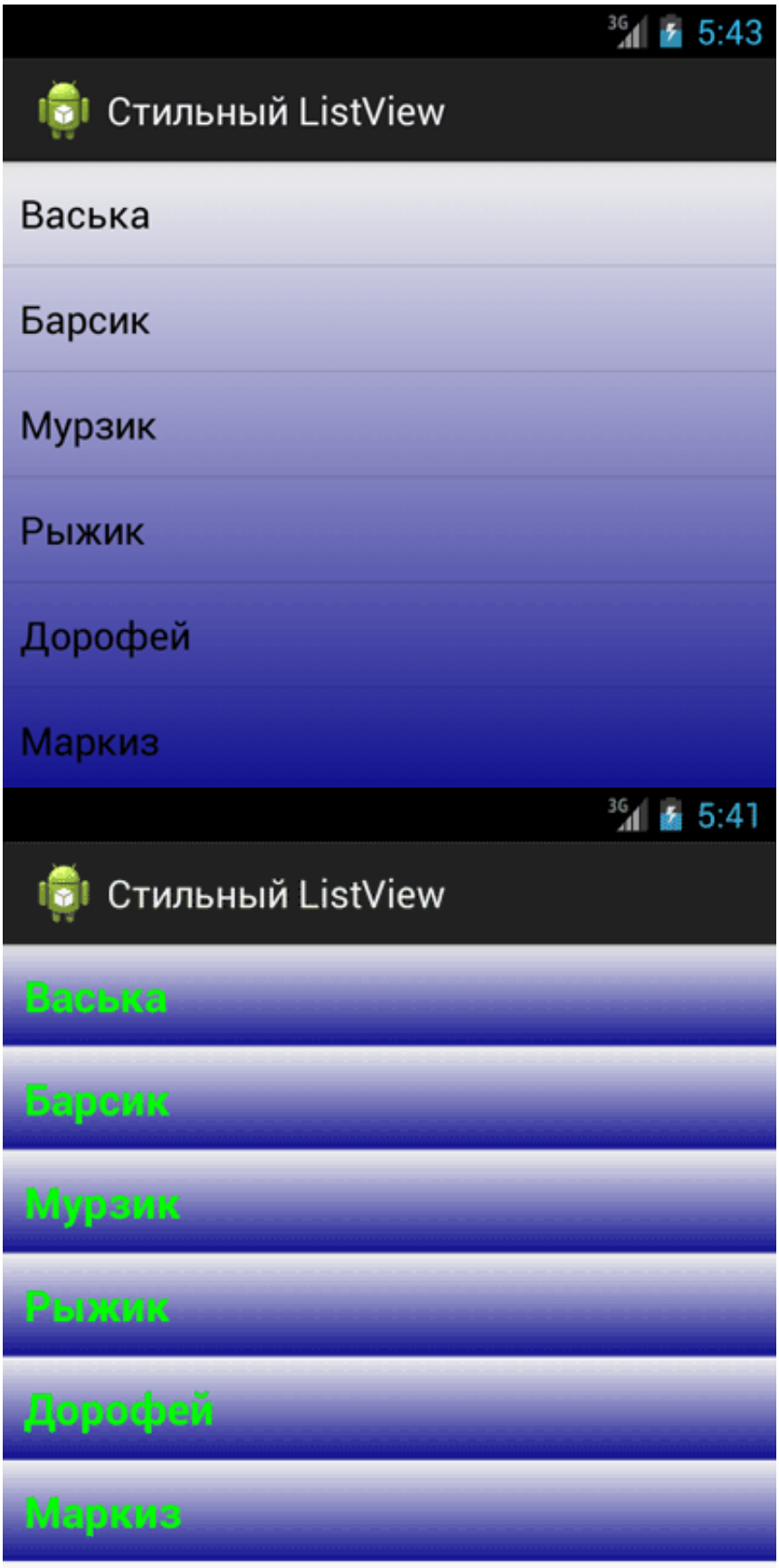
Также можете использовать градиент, который можно применить как ко всему ListView, так и отдельному элементу списка.

Создадим файл **res/drawable/gradient.xml**:

```
<?xml version="1.0" encoding="utf-8"?>
<shape xmlns:android="http://schemas.android.com/apk/res/android"
    android:shape="rectangle" >

    <gradient
        android:angle="270"
        android:endColor="#111191"
        android:startColor="#EFEFEF"
        android:type="linear" />

</shape>
```

При желании можете также использовать атрибут **android:middleColor**.

Теперь нужно прописать созданный градиент в атрибуте **android:background** либо у ListView, либо в отдельной разметке для элемента списка. Разница видна невооружённым взглядом:



Если через XML мы можем задать только три параметра для градиента, то программно через **GradientDrawable** можно задать больше опорных точек.

Нажатый элемент списка

В предыдущих примерах мы научились менять фон у списка, но при этом список утратил возможность выделять нажатый элемент. Это серьёзное упущение для профессиональных программ. Исправить ситуацию просто. Вы должны подготовить два файла - один для стандартного вида элемента списка, а второй для нажатого элемента.

drawable/row_default.xml

```
<?xml version="1.0" encoding="utf-8"?>
<shape xmlns:android="http://schemas.android.com/apk/res/android"
    android:shape="rectangle" >

    <gradient
        android:angle="270"
        android:endColor="#989898"
        android:startColor="#EFEFEF"
        android:type="linear" />

</shape>
```

drawable/row_pressed.xml

```
<?xml version="1.0" encoding="utf-8"?>
<shape xmlns:android="http://schemas.android.com/apk/res/android"
    android:shape="rectangle" >

    <gradient
        android:angle="270"
        android:endColor="#0661E5"
        android:startColor="#0B8CF2"
        android:type="linear" />

</shape>
```

Теперь нужно создать ещё один файл, в котором будут содержаться сведения о разных состояних элемента списка:

drawable/row_background.xml

```
<?xml version="1.0" encoding="utf-8"?>
<selector xmlns:android="http://schemas.android.com/apk/res/android">

    <item android:drawable="@drawable/row_pressed" android:state_pressed="true"/>
    <item android:drawable="@drawable/row_default"/>

</selector>
```

Данный файл теперь можно присвоить атрибуту **android:background** у отдельной разметки для элемента списка и список будет реагировать на нажатия пользователя.

Реклама