Java

Kotlin

Дизайн

Отладка

Open Source

Полезные ресурсы

Пишем справочник про котов

В этой статье я покажу как сделать простейшую программу **Справочник про котов**. На этом примере можно сделать огромное количество полезных приложений — например, небольшой сборник рецептов или набор схем оригами, если использовать **ListView** с миниатюрами.

Что мы узнаем:

- Как сделать простой список из массива, используя ListView
- Как загрузить текст из ресурсов
- Как загрузить html-текст в WebView
- Как передать данные из одной активности в другую

Создаём новый проект **Manual** (не путать с манулом). Начнем с интерфейса программы. Программа будет состоять из двух активностей. В первой выводится список тем, а во второй - полное описание выбранной темы. Откроем разметку первой активности **res/layout/activity_main.xml** и добавим компонент **ListView** для отображения списка тем:

```
<LinearLayout 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"
    android:orientation="vertical"
    tools:context=".MainActivity">

    <ListView
        android:id="@+id/listView"
        android:layout_width="wrap_content"
        android:layout_height="match_parent" />

</LinearLayout>
```

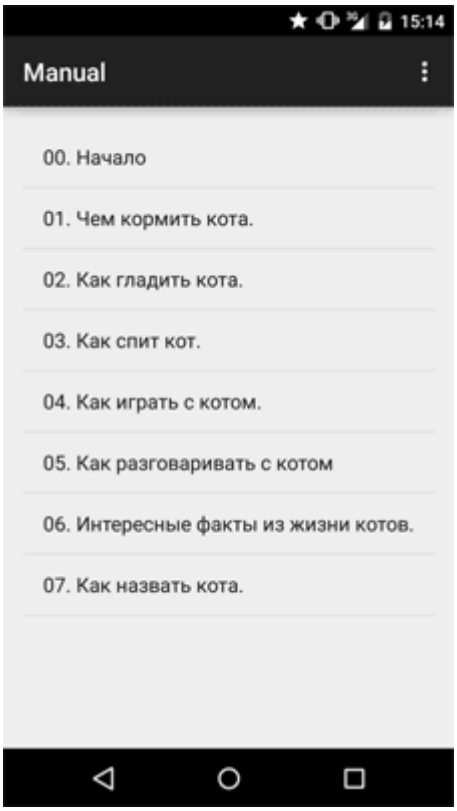
Сразу же создадим вторую активность **DetailActivity** (правой кнопкой мыши по имени пакета и выбираем **New | Activity | Empty Activity**). Создадим разметку для второй активности в файле **res/layout/activity_detail.xml**. Сюда мы добавим только компонент **WebView**. Скрытие строки состояния и заголовка сделаем в Java-коде.

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <WebView
        android:id="@+id/webView"
        android:layout_width="match_parent"
        android:layout_height="match_parent" />
</LinearLayout>
```

Дизайн приложения готов. Осталось написать код. Открываем файл класса **MainActivity**. В нём программно создадим список заголовков тем справочника и через адаптер добавим в список. Когда пользователь выбирает элемент списка, то получаем позицию выбранного элемента и запоминаем его. А затем запускаем вторую активность, в которую передаём номер позиции. Мы проходили подобные вещи раньше, поэтому просто освежите свою память.

```
import android.content.Intent;
import android.os.Bundle;
import android.support.v7.app.AppCompatActivity;
import android.view.View;
import android.widget.AdapterView;
import android.widget.ArrayAdapter;
import android.widget.ListView;


public class MainActivity extends AppCompatActivity {

    //Создаем массив разделов:
    private String titles[] = {
            "00. Начало",
            "01. Чем кормить кота.",
            "02. Как гладить кота.",
            "03. Как спит кот.",
            "04. Как играть с котом.",
            "05. Как разговаривать с котом",
            "06. Интересные факты из жизни котов.",
            "07. Как назвать кота.",
    };

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        // Получим идентификатор ListView
        ListView listView = findViewById(R.id.listView);
        //устанавливаем массив в ListView
        listView.setAdapter(
                new ArrayAdapter<>(this, android.R.layout.simple_list_item_1,
titles));
        listView.setTextFilterEnabled(true);

        //Обрабатываем щелчки на элементах ListView:
        listView.setOnItemClickListener(new AdapterView.OnItemClickListener() {
            public void onItemClick(AdapterView<?> a, View v, int position, long id) {
                Intent intent = new Intent();
                intent.setClass(MainActivity.this, DetailActivity.class);

                intent.putExtra("title", position);

                //запускаем вторую активность
                startActivity(intent);
```

```
}
```



Вторая активность

Запустив пример, вы можете щёлкнуть по любому элементу списка, чтобы открыть вторую активность. Сейчас вторая активность пуста, так как мы не написали никакого кода. Но при этом она получает номер позиции выбранного элемента через метод **putExtra()**. В зависимости от полученного номера мы формируем содержание веб-страницы.

Для справочника удобнее держать заранее подготовленные локальные файлы, чтобы не зависеть от интернета. Создадим новую папку - выбираем **res | New | Directory** и в диалоговом окне вводим имя папки **raw**.

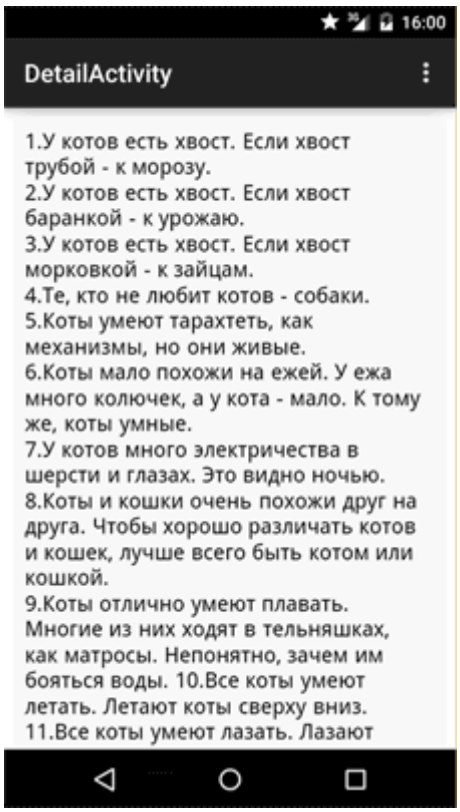
Самостоятельно подготовьте текстовые файлы с именами n0.txt, n1.txt, n2.txt и т.д. Символ **n** в начале имён файлов понадобился, чтобы избежать конфликта. Файлы ресурсов не должные начинаться на цифру.

Напишем код для второй активности. Во-первых, получим номер позиции, которую нам прислала первая активность. Во-вторых, открываем нужный файл для чтения и помещаем его содержимое в **WebView**.

```
import android.content.Context;
import android.content.Intent;
import android.os.Bundle;
import android.support.v7.app.AppCompatActivity;
import android.util.Log;
import android.webkit.WebView;

import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStream;
import java.io.InputStreamReader;

public class DetailActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_detail);

        WebView webView = findViewById(R.id.webView);

        Intent intent = getIntent();
        //получаем строку и формируем имя ресурса
        String resName = "n" + intent.getIntExtra("title", 0);
        Log.i("name", resName);
        Context context = getBaseContext(); //получаем контекст

        //читаем текстовый файл из ресурсов по имени
        String text = readRawTextFile(context, getResources().getIdentifier(resName,
                "raw", "ru.alexanderklimov.manual"));

        webView.loadDataWithBaseURL(null, text, "text/html", "en_US", null);
    }

    //читаем текст из raw-ресурсов
    private String readRawTextFile(Context context, int resId)
    {
        InputStream inputStream = context.getResources().openRawResource(resId);

        InputStreamReader inputReader = new InputStreamReader(inputStream);
        BufferedReader buffReader = new BufferedReader(inputReader);
        String line;
        StringBuilder builder = new StringBuilder();

        try {
            while (( line = buffReader.readLine()) != null) {
```

```
        } catch (IOException e) {
            return null;
        }
        return builder.toString();
    }
}
```

Теперь можно запустить проект и убедиться, что все работает.



Эту программу также можно скачать в Google Play.

В примере показаны базовые функции, достаточные для понимания. Вы можете усложнить пример, добавив поддержку фрагментов. Также вы можете самостоятельно доработать пример. Например, добавить картинки к элементам списка, загружать готовые html-документы, а также загружать веб-страницы из интернета.

Дополнительное чтение

Обсуждение статьи на форуме.