Файловый менеджер

2-й курс/Закрытая зона

[Получение списка файлов](http://developer.alexanderklimov.ru/android/fileexplorer.php#filelist)  
[Извлекаем информацию о EXIF](http://developer.alexanderklimov.ru/android/fileexplorer.php#exif)  
[Файловый менеджер для реального телефона](http://developer.alexanderklimov.ru/android/fileexplorer.php#realexplorer)  
[Фильтруем типы файлов](http://developer.alexanderklimov.ru/android/fileexplorer.php#filter)

Попробуем написать собственный файловый менеджер для просмотра файлов на своём устройстве. В предыдущем уроке про [файловую систему](http://developer.alexanderklimov.ru/android/filesystem.php) мы делали упор на файлах, которые относятся к самому приложению.

Получение списка файлов

Для работы с общими файлами на устройстве используются стандартные функции ввода/вывода из Java (класс **java.io.File**):

File file = new File(dirPath);

File[] filesArray = file.listFiles();

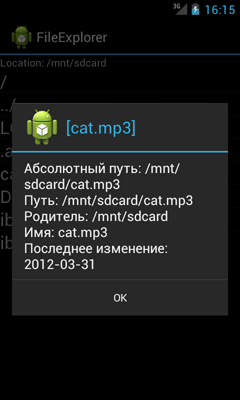
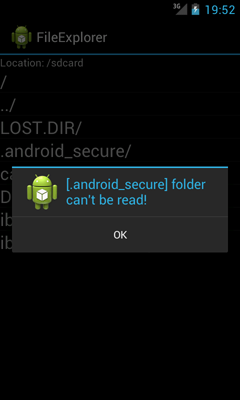
Существует также перегруженная версия метода **listFiles(FilenameFilter filter)**, где можно, например, указать фильтр для расширения файла, чтобы выводить только определённые типы файлов. Смотри пример [Фильтруем типы файлов](http://developer.alexanderklimov.ru/android/fileexplorer.php#filter).

Напишем пример просмотра файлов на устройстве.

Файловый менеджер будет состоять из прокручиваемого списка, а имена файлов и папок будем заносить в **ArrayList**. Если мы находимся не в корневой папке устройства, то добавим два элемента (корень и родительскую папку) в начало **ArrayList**, чтобы у нас была возможность быстро перемещаться на уровень выше.

Затем готовый **ArrayList** мы передадим в адаптер **ArrayAdapter** для окончательного вывода списка на экран.

Элементы списка будут кликабельны. Если щелкнуть по названию файлу, то в диалоговом окне выводится имя файла. Если щелкнуть по папке (проверка через **isDirectory()**), то открываем её в случае доступности (**canRead()**), в противном случае выводим сообщение, что папка закрыта для чтения.

Приступим к написанию программы. Добавим на форму **ListView** и два **TextView**. Вторая текстовая метка носит декоративный характер на случай, если список будет пуст.

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

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

android:layout\_width="match\_parent"

android:layout\_height="match\_parent"

android:orientation="vertical" >

<TextView

android:id="@+id/textViewPath"

android:layout\_width="match\_parent"

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

<ListView

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

android:layout\_width="match\_parent"

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

<TextView

android:id="@android:id/empty"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:text="Нет данных" />

</LinearLayout>

Создадим собственную разметку для элементов списка. Это решение вам пригодится, если вы захотите сделать навороченный менеджер со значками, подсказками и т.д. Вы сможете позже видоизменить шаблон для списка под свои задачи. Пока создадим в папке **res/layout** файл **list\_item.xml**:

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

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

android:id="@+id/textViewContent"

android:layout\_width="match\_parent"

android:layout\_height="30dp"

android:textSize="23sp" />

Для примера используем **ListActivity**:

package ru.alexanderklimov.testapplication;

import android.app.AlertDialog;

import android.app.ListActivity;

import android.content.DialogInterface;

import android.os.Bundle;

import android.view.View;

import android.widget.ArrayAdapter;

import android.widget.ListView;

import android.widget.TextView;

import java.io.File;

import java.util.ArrayList;

import java.util.Date;

import java.util.List;

public class MainActivity extends ListActivity {

private List<String> mPathList = null;

private String root = "/"; // символ для корневого элемента

private TextView mPathTextView;

public void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_main);

mPathTextView = (TextView) findViewById(R.id.textViewPath); // здесь будем выводить текущее местоположение

getDir(root); // выводим список файлов и папок в корневой папке системы

}

private void getDir(String dirPath) {

mPathTextView.setText("Путь: " + dirPath); // где мы сейчас

List<String> itemList = new ArrayList<>();

mPathList = new ArrayList<>();

File file = new File(dirPath);

File[] filesArray = file.listFiles(); // получаем список файлов

// если мы не в корневой папке

if (!dirPath.equals(root)) {

itemList.add(root);

mPathList.add(root);

itemList.add("../");

mPathList.add(file.getParent());

}

// формируем список папок и файлов для передачи адаптеру

for (File aFilesArray : filesArray) {

file = aFilesArray;

mPathList.add(file.getPath());

if (file.isDirectory()) // Это папка

itemList.add(file.getName() + "/");

else

itemList.add(file.getName());

}

// Можно выводить на экран список

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

R.layout.list\_item, itemList);

setListAdapter(adapter);

}

@Override

protected void onListItemClick(ListView l, View v, int position, long id) {

super.onListItemClick(l, v, position, id);

// обработка нажатий на элементах списка

File file = new File(mPathList.get(position));

// если это папка

if (file.isDirectory()) {

if (file.canRead()) // если она доступна для просмотра, то заходим в неё

getDir(mPathList.get(position));

else { // если папка закрыта, то сообщаем об этом

new AlertDialog.Builder(this)

.setIcon(R.drawable.ic\_launcher)

.setTitle(

"[" + file.getName()

+ "] папка не доступна!")

.setPositiveButton("OK",

new DialogInterface.OnClickListener() {

@Override

public void onClick(DialogInterface dialog,

int which) {

}

}).show();

}

} else { // если элемент списка является файлом, то выводим его имя

String fileInfo = "Абсолютный путь: " + file.getAbsolutePath()

+ "\n" + "Путь: " + file.getPath() + "\n" + "Родитель: "

+ file.getParent() + "\n" + "Имя: " + file.getName() + "\n"

+ "Последнее изменение: " + new Date(file.lastModified());

new AlertDialog.Builder(this)

.setIcon(R.drawable.ic\_launcher)

.setTitle("[" + file.getName() + "]")

.setMessage(fileInfo)

.setPositiveButton("OK",

new DialogInterface.OnClickListener() {

@Override

public void onClick(DialogInterface dialog,

int which) {

}

}).show();

}

}

}

Запустите проект и просмотрите внутренности вашей файловой системы.

Извлекаем информацию о EXIF

Фотографии, сделанные камерой устройства, содержат различную информацию о снимке, которая записывается в специальный формат EXIF в самом файле.

Формат EXIF предназначен для хранения различного рода метаданных, включая дату и время, свойства камеры (марку и модель), настройки изображения (апертуру и выдержку), а также описание снимка и информацию о местоположении.

Класс **ExifInterface** предоставляет механизм для чтения и изменения данных EXIF (Exchangeable Image File Format) внутри файла в формате JPEG. Создайте новый экземпляр этого класса, передав в его конструктор полное имя файла.

ExifInterface exif = new ExifInterface(filename);

Чтобы прочитать определенный атрибут из метаданных EXIF, вызовите метод **getAttribute()** из объекта **ExifInterface**, передав ему имя нужного атрибута. Класс **Exifinterface** содержит статические константы вида **TAG\_\***, которые можно использовать для доступа к стандартным данным EXIF.

Чтобы изменить метаданные, примените метод **setAttribute()**, передав ему в качестве параметров имя атрибута и соответствующее значение.

Расширим функциональность файлового менеджера и при щелчке на файле с расширением JPG будем выводить не только стандартную информацию о файле, но и EXIF-информацию. Добавим два вспомогательных метода и дополним блок **else**, в которой выводится информация о файле:

private String getExif(ExifInterface exif) {

String attribute = null;

attribute += getTagString(ExifInterface.TAG\_DATETIME, exif);

attribute += getTagString(ExifInterface.TAG\_FLASH, exif);

attribute += getTagString(ExifInterface.TAG\_GPS\_LATITUDE, exif);

attribute += getTagString(ExifInterface.TAG\_GPS\_LATITUDE\_REF, exif);

attribute += getTagString(ExifInterface.TAG\_GPS\_LONGITUDE, exif);

attribute += getTagString(ExifInterface.TAG\_GPS\_LONGITUDE\_REF, exif);

attribute += getTagString(ExifInterface.TAG\_IMAGE\_LENGTH, exif);

attribute += getTagString(ExifInterface.TAG\_IMAGE\_WIDTH, exif);

attribute += getTagString(ExifInterface.TAG\_MAKE, exif);

attribute += getTagString(ExifInterface.TAG\_MODEL, exif);

attribute += getTagString(ExifInterface.TAG\_ORIENTATION, exif);

attribute += getTagString(ExifInterface.TAG\_WHITE\_BALANCE, exif);

return attribute;

}

private String getTagString(String tag, ExifInterface exif) {

return (tag + " : " + exif.getAttribute(tag) + " ");

}

...

} else { // если элемент списка является файлом, то выводим его имя

**String exifAttribute = null;**

**String filename = file.getName();**

**String ext = filename.substring(filename.lastIndexOf('.') + 1,**

**filename.length());**

**if (ext.equalsIgnoreCase("JPG")) {**

**try {**

**ExifInterface exif = new ExifInterface(file.toString());**

**exifAttribute = getExif(exif);**

**} catch (IOException e) {**

**}**

**}**

String fileInfo = "Абсолютный путь: " + file.getAbsolutePath()

+ "\n" + "Путь: " + file.getPath() + "\n" + "Родитель: "

+ file.getParent() + "\n" + "Имя: " + file.getName() + "\n"

+ "Последнее изменение: " + new Date(file.lastModified());

new AlertDialog.Builder(this)

.setIcon(R.drawable.ic\_launcher)

.setTitle("[" + file.getName() + "]")

.setMessage(fileInfo **+ " " + exifAttribute**) // информация о файле и EXIF, если JPG

.setPositiveButton("OK",

new DialogInterface.OnClickListener() {

@Override

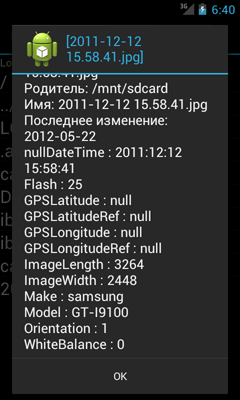
public void onClick(DialogInterface dialog,

int which) {

}

}).show();

}



На скриншоте показана информация о фотографии с моего телефона. Так как у меня стоит запрет на получении информации о моём местонахождении, то часть тегов, связанных с GPS-координатами, содержит пустые данные. Но при этом можно определить производителя, модель телефона и другую информацию.

Файловый менеджер для реального телефона

На самом деле созданный нами файловый менеджер годится для эмулятора и рутованных устройств. В обычных телефонах нет доступа к системным папками и файлам. Поэтому следует переписать проект, который будет работать в реальных условиях.

Во-первых, нужно изменить корневой элемент для исходной точки. Вместо корня "/" будем использовать папку внешнего хранилища, которую можно получить через метод**Environment.getExternalStorageDirectory()**. А скрытые или недоступные для чтения каталоги не будем выводим в списке.

package ru.alexanderklimov.testapplication;

import ...

public class MainActivity extends ListActivity {

private List mPathList = null;

//private String root = "/"; // символ для корневого элемента

private String root;

private TextView mPathTextView;

public void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_main);

mPathTextView = (TextView) findViewById(R.id.textViewPath);

// путь внешнего хранилища

root = Environment.getExternalStorageDirectory().getPath();

getDir(root); // выводим список файлов и папок в корневой папке системы

}

private void getDir(String dirPath) {

mPathTextView.setText("Путь: " + dirPath); // где мы сейчас

List<String> itemList = new ArrayList<>();

mPathList = new ArrayList<>();

File file = new File(dirPath);

File[] filesArray = file.listFiles(); // получаем список файлов

// если мы не в корневой папке

if (!dirPath.equals(root)) {

itemList.add(root);

mPathList.add(root);

itemList.add("../");

mPathList.add(file.getParent());

}

// формируем список папок и файлов для передачи адаптеру

for (File aFilesArray : filesArray) {

file = aFilesArray;

// Работаем только с доступными папками и файлами

if (!file.isHidden() && file.canRead()) {

mPathList.add(file.getPath());

if (file.isDirectory()) {

itemList.add(file.getName() + "/");

} else {

itemList.add(file.getName());

}

}

}

// Можно выводить на экран список

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

R.layout.list\_item, itemList);

setListAdapter(adapter);

}

@Override

protected void onListItemClick(ListView l, View v, int position, long id) {

super.onListItemClick(l, v, position, id);

// обработка нажатий на элементах списка

File file = new File(mPathList.get(position));

// если это папка

if (file.isDirectory()) {

if (file.canRead()) // если она доступна для просмотра, то заходим в неё

getDir(mPathList.get(position));

} else { // если элемент списка является файлом, то выводим его имя

String exifAttribute = null;

String filename = file.getName();

String ext = filename.substring(filename.lastIndexOf('.') + 1,

filename.length());

if (ext.equalsIgnoreCase("JPG")) {

try {

ExifInterface exif = new ExifInterface(file.toString());

exifAttribute = getExif(exif);

} catch (IOException e) {

}

}

String fileInfo = "Абсолютный путь: " + file.getAbsolutePath()

+ "\n" + "Путь: " + file.getPath() + "\n" + "Родитель: "

+ file.getParent() + "\n" + "Имя: " + file.getName() + "\n"

+ "Последнее изменение: " + new Date(file.lastModified());

new AlertDialog.Builder(this)

.setIcon(R.drawable.ic\_launcher)

.setTitle("[" + file.getName() + "]")

.setMessage(fileInfo + " " + exifAttribute) // информация о файле и EXIF, если JPG

.setPositiveButton("OK",

new DialogInterface.OnClickListener() {

@Override

public void onClick(DialogInterface dialog,

int which) {

}

}).show();

}

}

private String getExif(ExifInterface exif) {

// без изменений

}

private String getTagString(String tag, ExifInterface exif) {

return (tag + " : " + exif.getAttribute(tag) + " ");

}

}

Сортировка по имени или по дате изменения

Приложение получилось вполне рабочим, но есть один недостаток - имена папок и файлов выводятся не в алфавитном порядке. При большом количестве файлов это может стать головной болью для пользователя.

Решение очень простое. У класса **Array** есть метод **sort()**, позволяющий сортировать элементы массива с использованием интерфейса **Comparator**. Необходимо реализовать интерфейс**Comparator**, в котором имена файлов приведём в нижний регистр.

В методе **getDir()** добавляем одну строчку кода и добавляем новый блок кода для интерфейса.

private void getDir(String dirPath) {

mPathTextView.setText("Путь: " + dirPath); // где мы сейчас

List<String> itemList = new ArrayList<>();

mPathList = new ArrayList<>();

File file = new File(dirPath);

File[] filesArray = file.listFiles(); // получаем список файлов

// если мы не в корневой папке

if (!dirPath.equals(root)) {

itemList.add(root);

mPathList.add(root);

itemList.add("../");

mPathList.add(file.getParent());

}

Arrays.sort(filesArray, fileComparator);

// формируем список папок и файлов для передачи адаптеру

for (File aFilesArray : filesArray) {

file = aFilesArray;

// Работаем только с доступными папками и файлами

if (!file.isHidden() && file.canRead()) {

mPathList.add(file.getPath());

if (file.isDirectory()) {

itemList.add(file.getName() + "/");

} else {

itemList.add(file.getName());

}

}

}

// Можно выводить на экран список

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

R.layout.list\_item, itemList);

setListAdapter(adapter);

}

Comparator<? super File> fileComparator = new Comparator<File>() {

public int compare(File file1, File file2) {

if (file1.isDirectory()) {

if (file2.isDirectory()) {

return String.valueOf(file1.getName().toLowerCase())

.compareTo(file2.getName().toLowerCase());

} else {

return -1;

}

} else {

if (file2.isDirectory()) {

return 1;

} else {

return String.valueOf(file1.getName().toLowerCase())

.compareTo(file2.getName().toLowerCase());

}

}

}

};

Теперь все папки и файлы будут выводится в алфавитном порядке.

Если хочется выводить список файлов по дате изменения, то следует реализовать другой интерфейс:

Comparator<? super File> fileComparatorByLastModified = new Comparator<File>() {

public int compare(File file1, File file2) {

if (file1.isDirectory()) {

if (file2.isDirectory()) {

return Long.valueOf(file1.lastModified()).compareTo(

file2.lastModified());

} else {

return -1;

}

} else {

if (file2.isDirectory()) {

return 1;

} else {

return Long.valueOf(file1.lastModified()).compareTo(

file2.lastModified());

}

}

}

};

Чтобы переключаться между двумя способами сортировки, добавим две кнопки в основную разметку:

<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" >

**<LinearLayout**

**android:layout\_width="match\_parent"**

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

**android:orientation="horizontal" >**

**<Button**

**android:id="@+id/button\_alphabetically"**

**android:layout\_width="0dp"**

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

**android:layout\_weight="1"**

**android:text="По алфавиту"**

**android:textSize="16sp"**

**android:onClick="onAlphabetClick"/>**

**<Button**

**android:id="@+id/button\_lastDateModified"**

**android:layout\_width="0dp"**

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

**android:layout\_weight="1"**

**android:text="По дате изм."**

**android:textSize="16sp"**

**android:onClick="onDateModClick"/>**

**</LinearLayout>**

<TextView

android:id="@+id/textViewPath"

android:layout\_width="match\_parent"

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

<ListView

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

android:layout\_width="match\_parent"

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

<TextView

android:id="@android:id/empty"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:text="Нет данных" />

</LinearLayout>

Внесём косметические изменения в коде. Окончательный код приложения:

package ru.alexanderklimov.testapplication;

import android.app.AlertDialog;

import android.app.ListActivity;

import android.content.DialogInterface;

import android.media.ExifInterface;

import android.os.Bundle;

import android.os.Environment;

import android.view.View;

import android.widget.ArrayAdapter;

import android.widget.ListView;

import android.widget.TextView;

import java.io.File;

import java.io.IOException;

import java.util.ArrayList;

import java.util.Arrays;

import java.util.Comparator;

import java.util.Date;

import java.util.List;

public class MainActivity extends ListActivity {

private List<String> mPathList = null;

private String root;

private TextView mPathTextView;

private String mCurrentPath;

private Comparator<? super File> mComparator;

public void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_main);

mPathTextView = (TextView) findViewById(R.id.textViewPath);

mComparator = fileComparatorByAlphabetically;

// путь внешнего хранилища

root = Environment.getExternalStorageDirectory().getPath();

getDir(root); // выводим список файлов и папок в корневой папке системы

}

private void getDir(String dirPath) {

mCurrentPath = dirPath;

mPathTextView.setText("Путь: " + dirPath); // где мы сейчас

List<String> itemList = new ArrayList<>();

mPathList = new ArrayList<>();

File file = new File(dirPath);

File[] filesArray = file.listFiles(); // получаем список файлов

// если мы не в корневой папке

if (!dirPath.equals(root)) {

itemList.add(root);

mPathList.add(root);

itemList.add("../");

mPathList.add(file.getParent());

}

Arrays.sort(filesArray, mComparator);

// формируем список папок и файлов для передачи адаптеру

for (File aFilesArray : filesArray) {

file = aFilesArray;

// Работаем только с доступными папками и файлами

if (!file.isHidden() && file.canRead()) {

mPathList.add(file.getPath());

if (file.isDirectory()) {

itemList.add(file.getName() + "/");

} else {

itemList.add(file.getName());

}

}

}

// Можно выводить на экран список

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

R.layout.list\_item, itemList);

setListAdapter(adapter);

}

Comparator<? super File> fileComparatorByAlphabetically = new Comparator<File>() {

public int compare(File file1, File file2) {

if (file1.isDirectory()) {

if (file2.isDirectory()) {

return String.valueOf(file1.getName().toLowerCase())

.compareTo(file2.getName().toLowerCase());

} else {

return -1;

}

} else {

if (file2.isDirectory()) {

return 1;

} else {

return String.valueOf(file1.getName().toLowerCase())

.compareTo(file2.getName().toLowerCase());

}

}

}

};

Comparator<? super File> fileComparatorByLastModified = new Comparator<File>() {

public int compare(File file1, File file2) {

if (file1.isDirectory()) {

if (file2.isDirectory()) {

return Long.valueOf(file1.lastModified()).compareTo(

file2.lastModified());

} else {

return -1;

}

} else {

if (file2.isDirectory()) {

return 1;

} else {

return Long.valueOf(file1.lastModified()).compareTo(

file2.lastModified());

}

}

}

};

@Override

protected void onListItemClick(ListView l, View v, int position, long id) {

super.onListItemClick(l, v, position, id);

// обработка нажатий на элементах списка

File file = new File(mPathList.get(position));

// если это папка

if (file.isDirectory()) {

if (file.canRead()) // если она доступна для просмотра, то заходим в неё

getDir(mPathList.get(position));

} else { // если элемент списка является файлом, то выводим его имя

String exifAttribute = null;

String filename = file.getName();

String ext = filename.substring(filename.lastIndexOf('.') + 1,

filename.length());

if (ext.equalsIgnoreCase("JPG")) {

try {

ExifInterface exif = new ExifInterface(file.toString());

exifAttribute = getExif(exif);

} catch (IOException e) {

}

}

String fileInfo = "Абсолютный путь: " + file.getAbsolutePath()

+ "\n" + "Путь: " + file.getPath() + "\n" + "Родитель: "

+ file.getParent() + "\n" + "Имя: " + file.getName() + "\n"

+ "Последнее изменение: " + new Date(file.lastModified());

new AlertDialog.Builder(this)

.setIcon(R.drawable.ic\_launcher)

.setTitle("[" + file.getName() + "]")

.setMessage(fileInfo + " " + exifAttribute) // информация о файле и EXIF, если JPG

.setPositiveButton("OK",

new DialogInterface.OnClickListener() {

@Override

public void onClick(DialogInterface dialog,

int which) {

}

}).show();

}

}

private String getExif(ExifInterface exif) {

String attribute = null;

attribute += getTagString(ExifInterface.TAG\_DATETIME, exif);

attribute += getTagString(ExifInterface.TAG\_FLASH, exif);

attribute += getTagString(ExifInterface.TAG\_GPS\_LATITUDE, exif);

attribute += getTagString(ExifInterface.TAG\_GPS\_LATITUDE\_REF, exif);

attribute += getTagString(ExifInterface.TAG\_GPS\_LONGITUDE, exif);

attribute += getTagString(ExifInterface.TAG\_GPS\_LONGITUDE\_REF, exif);

attribute += getTagString(ExifInterface.TAG\_IMAGE\_LENGTH, exif);

attribute += getTagString(ExifInterface.TAG\_IMAGE\_WIDTH, exif);

attribute += getTagString(ExifInterface.TAG\_MAKE, exif);

attribute += getTagString(ExifInterface.TAG\_MODEL, exif);

attribute += getTagString(ExifInterface.TAG\_ORIENTATION, exif);

attribute += getTagString(ExifInterface.TAG\_WHITE\_BALANCE, exif);

return attribute;

}

private String getTagString(String tag, ExifInterface exif) {

return (tag + " : " + exif.getAttribute(tag) + " ");

}

public void onAlphabetClick(View v) {

mComparator = fileComparatorByAlphabetically;

getDir(mCurrentPath);

}

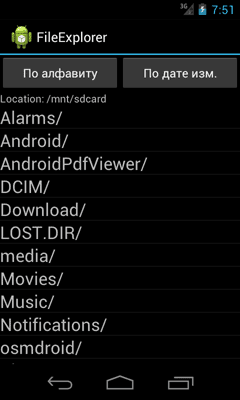
public void onDateModClick(View v) {

mComparator = fileComparatorByLastModified;

getDir(mCurrentPath);

}

}



Фильтруем типы файлов

Рассмотрим простейший пример вывода списка файлов определённого формата на SD-карте. Добавьте в разметку компонент **TextView**.

package ru.alexanderklimov.filter;

import ...

public class MainActivity extends Activity {

File root;

File[] fileArray;

@Override

public void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_main);

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

root = new File(Environment.getExternalStorageDirectory()

.getAbsolutePath());

fileArray = root.listFiles(new FilenameFilter() {

@Override

public boolean accept(File dir, String filename) {

return filename.toLowerCase().endsWith(".jpg");

}

});

// сначала выводим путь к SD-карте

String f = root.getAbsolutePath() + "\n\n";

// затем список файлов с расширением JPG

for (int i = 0; i < fileArray.length; i++) {

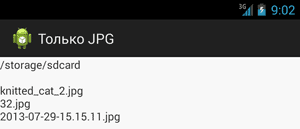
f += fileArray[i].getName() + "\n";

}

textInfo.setText(f);

}

}



Р