Запись и чтение файла (внутренняя память устройства)

Приложение может сохранять какую-то информацию в файлах, которые могут храниться как на SD-карте, так и во «внутренней» памяти устройства. Ниже мы рассмотрим работу с файлами в случае второго варианта, то есть когда файл хранится в памяти смартфона.

Мы создадим приложение с одним экраном, на котором будут располагаться две кнопки. Одна кнопка будет запускать запись в файл, а вторая — его чтение. Для простоты примера прочитанные данные будут выводиться в лог. Также мы будем писать в файл заранее заданную строку, чтобы не нагромождать лишний код и сохранить приложение максимально понятным.

В примере будет использоваться стандартные классы Java из библиотеки IO. Если вы еще не знакомы с этой библиотекой, то рекомендую что-то почитать, т.к. разбирать это API мы тут не будем.

Итак, приступим.

1. Создадим layout — **res/layout/activity\_main.xml**

|  |
| --- |
| <RelativeLayout      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:paddingLeft="@dimen/activity\_horizontal\_margin"      android:paddingRight="@dimen/activity\_horizontal\_margin"      android:paddingTop="@dimen/activity\_vertical\_margin"      android:paddingBottom="@dimen/activity\_vertical\_margin"      tools:context=".MainActivity">        <Button          android:layout\_width="wrap\_content"          android:layout\_height="wrap\_content"          android:text="Write"          android:id="@+id/button"          android:layout\_alignParentLeft="true"          android:layout\_alignParentStart="true"          android:onClick="writeToFile"          android:layout\_marginTop="39dp" />        <Button          android:layout\_width="wrap\_content"          android:layout\_height="wrap\_content"          android:text="Read"          android:id="@+id/button2"          android:layout\_below="@+id/button"          android:layout\_alignParentLeft="true"          android:layout\_alignParentStart="true"          android:onClick="readFile"          android:layout\_marginTop="43dp" />    </RelativeLayout> |

В верстке использовались отступы — **res/values/dimens.xml**

|  |
| --- |
| <resources>      <!-- Default screen margins, per the Android Design guidelines. -->      <dimen name="activity\_horizontal\_margin">16dp</dimen>      <dimen name="activity\_vertical\_margin">16dp</dimen>  </resources> |

2. Класс Активити — **MainActivity.java**

|  |
| --- |
| package ru.androiddocs.fileworker;    import android.support.v7.app.ActionBarActivity;  import android.os.Bundle;  import android.util.Log;  import android.view.View;    import java.io.BufferedReader;  import java.io.FileInputStream;  import java.io.FileOutputStream;  import java.io.InputStreamReader;      public class MainActivity extends ActionBarActivity {        private static final String LOG\_TAG = "my\_tag";      private String mFileName = "myfile";          @Override      protected void onCreate(Bundle savedInstanceState) {          super.onCreate(savedInstanceState);          setContentView(R.layout.activity\_main);      }          public void readFile(View v) {            Log.d(LOG\_TAG, "readFile");            FileInputStream stream = null;          StringBuilder sb = new StringBuilder();          String line;            try {              stream = openFileInput(mFileName);                try {                  BufferedReader reader = new BufferedReader(new InputStreamReader(stream, "UTF-8"));                  while ((line = reader.readLine()) != null) {                      sb.append(line);                  }              } finally {                  stream.close();              }                Log.d(LOG\_TAG, "Data from file: " + sb.toString());            } catch (Exception e) {              Log.d(LOG\_TAG, "Файла нет или произошла ошибка при чтении");          }      }        public void writeToFile(View v) {            Log.d(LOG\_TAG, "writeToFile");            String string = "My test string";            try {              FileOutputStream outputStream = openFileOutput(mFileName, MODE\_PRIVATE);              outputStream.write(string.getBytes());              outputStream.close();            } catch (Exception e) {              Log.d(LOG\_TAG, "Произошла ошибка при записи");          }      }  } |

Мы имеем тут два собственных метода, срабатывающих при нажатии на соответствующую кнопку:

**readFile()** — читает файл  
**writeToFile()** — пишет в файл

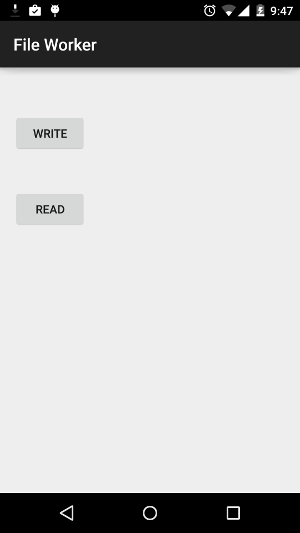
При чтении мы сначала открываем файл для чтения, создавая поток. Делаем это с помощью метода**openFileInput()** (этот метод является частью Android SDK). Затем мы считываем данные в буфер и построчно читаем их уже оттуда, добавляя их к результирующей строке с помощью **StringBuilder**. По окончанию чтения выводим данные в лог.

Если указанный файл не найден (мы задаем его имя в **mFileName**), то возникает **exception**, которое ловится блоком **catch** (тут мы просто пишем об ошибке в лог).

Метод для записи покороче. Создаем поток для записи, используя метод **openFileOutput()** (тоже входит в Android SDK). Данный метод на вход принимает два параметра: имя файла и константу-режим записи (мы используем **MODE\_PRIVATE**, т.е. файл будет доступен только нашему приложению). Затем пишем данные в поток и закрываем его. Блок **catch** ловит возможные исключения.

Если указанный файл не будет найден, то система создаст его самостоятельно.

Запускаем приложение. Жмем на кнопочки и смотрим в логи.



Запись и чтение файла

Для удаления файла можно использовать метод:

|  |  |
| --- | --- |
| 1 | deleteFile("myfile"); |

который также входит в SDK.