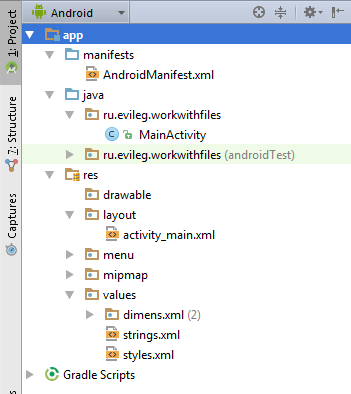
Файл в ОС Android. Операции чтения и записи

[Android](https://evileg.com/ru/tag/?q=Android), [read](https://evileg.com/ru/tag/?q=read), [write](https://evileg.com/ru/tag/?q=write), [file](https://evileg.com/ru/tag/?q=file)

Операции чтения и записи в файл являются стандартным функционалом любых приложений, которые ведут журналирование событий, работу с файлами, вплоть до передачи данных по сети. В данной статье рассмотрим методы записи в файлы информации и чтения из файла записанной строки.

Структура проекта



Эстетических изменений стандартных кнопок или ListView в данном уроке производиться не будет, поскольку мы будем работать с тем, что скрыто от глаз пользователя, а именно работать с файлами.

Вся структура проекта соответственно состоит на этот раз лишь из одного класса: **MainActivity**

Также в проекте присутствуют следующие файлы ресурсов:

1. activity\_main.xml
2. strings.xml
3. styles.xml - в данном файле отсутствуют какие-либо изменения относящиеся к проекту.

Кроме этого внесены изменения в файл **AndroidManifest.xml.**В файле необходимо добавить следующие две строки. Это разрешения для приложения - производить операции чтения и записи с внешним накопителем (то есть SD Card телефона) В современным смартфонах на базе ОС Android в большинстве случаев запись информации производится во внешний накопитель, хотя обычный пользователь считает этот накопитель внутренним, поскольку он является встроенным, но с точки зрения операционной систем данный накопитель (то есть SD Card) является внешним накопителем. В данной статье не будет рассматриваться вариант работы с истинно внутренним накопителем.

1. <manifest xmlns:android="http://schemas.android.com/apk/res/android"
2. package="ru.evileg.workwithfiles" >
4. <uses-permission android:name="android.permission.WRITE\_EXTERNAL\_STORAGE" />
5. <uses-permission android:name="android.permission.READ\_EXTERNAL\_STORAGE" />
7. ...
9. </manifest>

Формирование разметки приложения

activity\_main.xml

Разметка основной активити, в которой будет производиться работа нашего приложения. В данной разметке присутствует всего две кнопки (Button) и текстовое поле (TextView), в которое мы будем выводить информацию сохранённую в файле.

1. <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
2. xmlns:tools="http://schemas.android.com/tools" android:layout\_width="match\_parent"
3. android:layout\_height="match\_parent"
4. android:paddingLeft="@dimen/activity\_horizontal\_margin"
5. android:paddingRight="@dimen/activity\_horizontal\_margin"
6. android:paddingTop="@dimen/activity\_vertical\_margin"
7. android:paddingBottom="@dimen/activity\_vertical\_margin"
8. android:background="#ffffff"
9. android:orientation="vertical"
10. tools:context=".MainActivity">
12. <Button
13. android:layout\_width="match\_parent"
14. android:layout\_height="wrap\_content"
15. android:text="@string/write\_file"
16. android:id="@+id/buttonWrite"
17. android:layout\_gravity="center\_horizontal" />
19. <Button
20. android:layout\_width="match\_parent"
21. android:layout\_height="wrap\_content"
22. android:text="@string/read\_file"
23. android:id="@+id/buttonRead"
24. android:layout\_gravity="center\_horizontal" />
26. <TextView
27. android:layout\_width="match\_parent"
28. android:layout\_height="match\_parent"
29. android:id="@+id/textView"
30. android:textSize="26sp"
31. android:layout\_gravity="center\_horizontal" />
32. </LinearLayout>

strings.xml

Файл ресурсов текста в ОС Android. Подготовка всех строк, которые применяются в Вашем Приложении в данном файле, является не только хорошим тоном, но обязательным условием для разработки качественного Приложения. Поскольку, если вы будете дисциплинировать себя, сохранять всю подобную информацию в данном файле, то впоследствии это окупится, когда вы будете производить перевод Приложения на другие языки. Тем более, что в Android Studio имеется удобный функционал для этого.

1. <resources>
2. <string name="app\_name">Work With Files</string>
4. <string name="hello\_world">Hello world!</string>
5. <string name="action\_settings">Settings</string>
7. <string name="write\_done">Запись выполнена</string>
8. <string name="write\_file">Записать данные в файл</string>
9. <string name="read\_file">Считать данные из файла</string>
10. </resources>

styles.xml

В данном файле нет изменений относящихся к проекту. Но при создании проекта стандартная тема оформления не рендерится Android Studio. Выдается ошибки при предпросмотре и в режиме дизайна. Чтобы этого избежать, пропишите следующую информацию взамен старой.

1. <resources>
3. <!-- Base application theme. -->
4. <style name="AppTheme" parent="Base.Theme.AppCompat.Light"/>
6. </resources>

Основной класс проекта - MainActivity.java

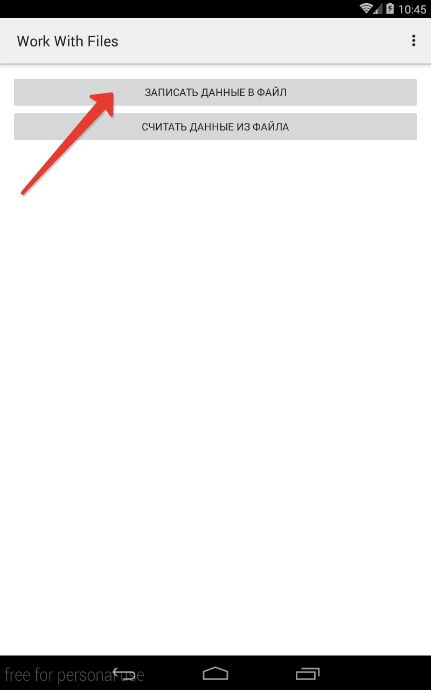
Сегодня в данном классе сконцентрирован весь программный код. В данном классе производится формирование внешнего вида главной активити, а также организуется работа с файлами.

1. package ru.evileg.workwithfiles;
3. import android.os.Bundle;
4. import android.os.Environment;
5. import android.support.v7.app.AppCompatActivity;
6. import android.view.Menu;
7. import android.view.MenuItem;
8. import android.view.View;
9. import android.widget.Button;
10. import android.widget.TextView;
11. import android.widget.Toast;
13. import java.io.BufferedReader;
14. import java.io.File;
15. import java.io.FileInputStream;
16. import java.io.FileNotFoundException;
17. import java.io.FileOutputStream;
18. import java.io.IOException;
19. import java.io.InputStreamReader;

22. public class MainActivity extends AppCompatActivity implements View.OnClickListener {
24. /\*
25. \* Создаем постоянные константы для удобства работы,
26. \* объявляем TextView, который должен быть доступен в нескольких методах класса
27. \*/
28. private static final String fileName = "hello.txt";
29. private static final String text = "Hello World";
30. private static TextView textView;
32. @Override
33. protected void onCreate(Bundle savedInstanceState) {
34. super.onCreate(savedInstanceState);
35. setContentView(R.layout.activity\_main);
37. /\*
38. \* Объявляем и инциализируем Кнопки (Button),
39. \* а также инициализируем TextView
40. \* Также в активити имплементирован метод слушателя событий нажатия,
41. \* то есть OnClickListener, который привязывается к кнопкам
42. \*/
43. textView = (TextView) this.findViewById(R.id.textView);
44. Button buttonWrite = (Button) this.findViewById(R.id.buttonWrite);
45. Button buttonRead = (Button) this.findViewById(R.id.buttonRead);
46. buttonWrite.setOnClickListener(this);
47. buttonRead.setOnClickListener(this);
49. }
51. @Override
52. public boolean onCreateOptionsMenu(Menu menu) {
53. // Inflate the menu; this adds items to the action bar if it is present.
54. getMenuInflater().inflate(R.menu.menu\_main, menu);
55. return true;
56. }
58. @Override
59. public boolean onOptionsItemSelected(MenuItem item) {
60. // Handle action bar item clicks here. The action bar will
61. // automatically handle clicks on the Home/Up button, so long
62. // as you specify a parent activity in AndroidManifest.xml.
63. int id = item.getItemId();
65. //noinspection SimplifiableIfStatement
66. if (id == R.id.action\_settings) {
67. return true;
68. }
70. return super.onOptionsItemSelected(item);
71. }
73. /\*
74. \* Обработчик нажатий кнопок. Нажатая кнопка определяется по её ID через метод getID()
75. \*/
76. @Override
77. public void onClick(View v) {
79. switch (v.getId()){
80. case R.id.buttonWrite:
81. writeFile();
82. break;
83. case R.id.buttonRead:
84. readFile();
85. break;
86. default:
87. break;
88. }
89. }
91. private void writeFile() {
92. try {
93. /\*
94. \* Создается объект файла, при этом путь к файлу находиться методом класcа Environment
95. \* Обращение идёт, как и было сказано выше к внешнему накопителю
96. \*/
97. File myFile = new File(Environment.getExternalStorageDirectory().toString() + "/" + fileName);
98. myFile.createNewFile(); // Создается файл, если он не был создан
99. FileOutputStream outputStream = new FileOutputStream(myFile); // После чего создаем поток для записи
100. outputStream.write(text.getBytes()); // и производим непосредственно запись
101. outputStream.close();
102. /\*
103. \* Вызов сообщения Toast не относится к теме.
104. \* Просто для удобства визуального контроля исполнения метода в приложении
105. \*/
106. Toast.makeText(this, R.string.write\_done, Toast.LENGTH\_SHORT).show();
107. } catch (Exception e) {
108. e.printStackTrace();
109. }
110. }
112. private void readFile() {
113. /\*
114. \* Аналогично создается объект файла
115. \*/
116. File myFile = new File(Environment.getExternalStorageDirectory().toString() + "/" + fileName);
117. try {
118. FileInputStream inputStream = new FileInputStream(myFile);
119. /\*
120. \* Буфферезируем данные из выходного потока файла
121. \*/
122. BufferedReader bufferedReader = new BufferedReader(new InputStreamReader(inputStream));
123. /\*
124. \* Класс для создания строк из последовательностей символов
125. \*/
126. StringBuilder stringBuilder = new StringBuilder();
127. String line;
128. try {
129. /\*
130. \* Производим построчное считывание данных из файла в конструктор строки,
131. \* Псоле того, как данные закончились, производим вывод текста в TextView
132. \*/
133. while ((line = bufferedReader.readLine()) != null){
134. stringBuilder.append(line);
135. }
136. textView.setText(stringBuilder);
137. } catch (IOException e) {
138. e.printStackTrace();
139. }
140. } catch (FileNotFoundException e) {
141. e.printStackTrace();
142. }
143. }
144. }

Итог

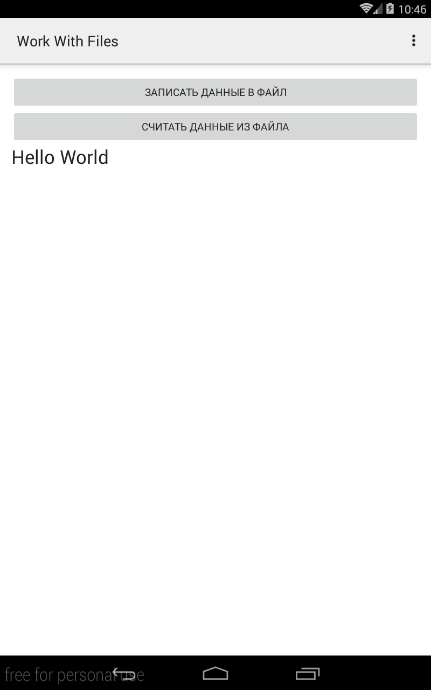
Если в процессе изучения материала не возникло никаких проблем и ошибок, то по нажатию кнопки записи в файл произойдет создание нового файла и будет произведена запись строки "Hello World". При нажатии кнопки чтения будет выведена информация сохранённая в текстовом файле. Процесс показан на скриншотах ниже.



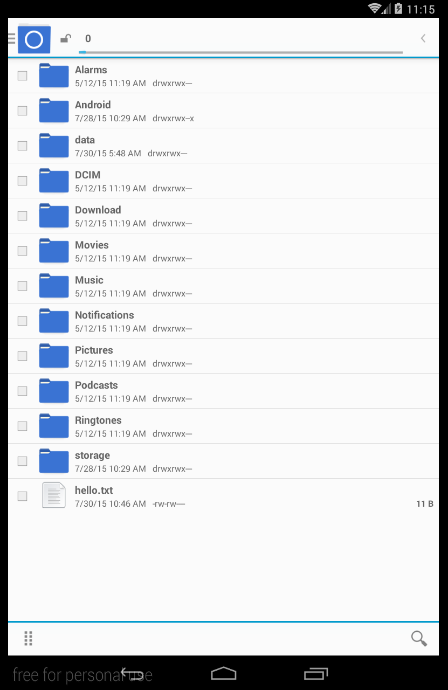
По нажатию кнопки производится запись в файл информации



Запись в файл



Чтение из файла производится по нажатию соответствующей кнопки. При этом в TextView выводится текст



Созданный файл hello.txt в файловом менеджере