Хранение файлов

В ДАННОМ УРОКЕ МЫ РАССМОТРИМ

* [Выбор между внутренним и внешним хранилищем](http://easyandroid.ru/xranenie-fajlov/#ChooseStorage)
* [Получение прав для внешнего хранилища](http://easyandroid.ru/xranenie-fajlov/#PermissionExternal)
* [Сохранение файлов во внутреннем хранилище](http://easyandroid.ru/xranenie-fajlov/#SaveInternal)
* [Сохранение файлов во внешнем хранилище](http://easyandroid.ru/xranenie-fajlov/#SaveExternal)
* [Запрос свободного пространства](http://easyandroid.ru/xranenie-fajlov/#FreeSpace)
* [Удаление файла](http://easyandroid.ru/xranenie-fajlov/#Delete)

ТАКЖЕ РЕКОМЕНДУЕМ ПРОЧИТАТЬ

* [Использование внутреннего хранилища](http://developer.android.com/guide/topics/data/data-storage.html#filesInternal)
* [Использование внешнего хранилища](http://developer.android.com/guide/topics/data/data-storage.html#filesExternal)

Файловая система Android очень похожа на файловые системы, основанные на дисках, используемые в других системах. В данном уроке рассказывается как работать с файловой системой Android, читать и записывать файлы, используя класс [File](http://developer.android.com/reference/java/io/File.html).

Объект типа [File](http://developer.android.com/reference/java/io/File.html) отлично подходит для чтения и записи больших объемов данных, от начала до конца. Это могут быть например картинки или другие файлы, переданные по сети.

Сейчас мы рассмотрим выполнение базовых операций, связанных с использованием файлов в вашем приложении. Мы предполагаем, что вы уже знакомы с базовыми возможностями файловой системой Linux, а также с API ввода/вывода [java.io](http://developer.android.com/reference/java/io/package-summary.html)

Выбор между внутренним и внешним хранилищем

Все устройства под Android имеют два пространства для хранения файлов: **внешнее** и **внутреннее** хранилища. Эти названия остались от первых устройств на Android. На большинстве устройств применялась энергозависимая память (**внутреннее хранилище**) и съемные устройства, вроде microSD карт (**внешнее хранилище**). На некоторых устройствах постоянное место хранения делится между внутренним и внешним разделами, поэтому даже без SD карт у нас будет и “внутреннее” и “внешнее” пространство. При этом для API неважно располагается ли внешнее пространство на съемном устройстве или нет. В списке приведены общие факты об обоих типах пространства:

* **Внутреннее хранилище:**
* Всегда доступно.
* По умолчанию доступ к файлам разрешен только для приложений, создавших эти файлы.
* При удалении приложения, из внутреннего хранилища удаляются все файлы приложения.
* Внутреннее хранилище лучше подходит, если вы хотите быть уверены, что другие приложения не получат доступ к данным.
* **Внешнее хранилище:**
* Не всегда доступно, поскольку пользователь может вытащить съемную память из устройства.
* Файлы могут быть прочитаны любым приложением.
* При удалении приложения файлы будут удалены только в том случае, если вы сохранили их в директории, полученной с помощью метода[getExternalFilesDir()](http://developer.android.com/reference/android/content/Context.html#getExternalFilesDir(java.lang.String)).
* Внешнее хранилище – это отличное место для файлов, которые не нуждаются в особой секретности. Также во внешних хранилищах могут храниться файлы, которые могут быть открыты на компьютере пользователя или те, которые вы хотите использовать сразу в нескольких приложениях.

***Совет:****хотя приложения устанавливаются по умолчанию во внутреннее хранилище, вы можете задать место хранения с помощью атрибута*[*android:installLocation*](http://developer.android.com/guide/topics/manifest/manifest-element.html#install)*в файле манифеста. Пользователям понравится, что вы позаботились о них и установили эту опцию, в случаях, когда размер файла APK вашего приложения будет больше свободного пространства их внутреннего хранилища. Больше информации о данной опции смотрите в разделе*[*Место установки приложения*](http://developer.android.com/guide/topics/data/install-location.html)*.*

Получение прав для внешнего хранилища

Для записи во внешнее хранилище, вы должны запросить права [WRITE\_EXTERNAL\_STORAGE](http://developer.android.com/reference/android/Manifest.permission.html#WRITE_EXTERNAL_STORAGE) в [файле манифеста](http://developer.android.com/guide/topics/manifest/manifest-intro.html):

|  |  |
| --- | --- |
| 1  2  3  4 | <manifest ...>      <uses-permission android:name="android.permission.WRITE\_EXTERNAL\_STORAGE" />      ...  </manifest> |

***Внимание:****в настоящее время, все приложения имеют права на чтение файлов из внешнего хранилища без получения специального разрешения. Однако это изменится в будущих версиях. Если ваше приложение должно читать файлы из внешнего хранилища (но не писать), необходимо получить право*[*READ\_EXTERNAL\_STORAGE*](http://developer.android.com/reference/android/Manifest.permission.html#READ_EXTERNAL_STORAGE)*. Чтобы гарантировать, что ваше приложение всегда будет работать так, как вы ожидаете, объявляйте права на чтение уже сейчас, чтобы не беспокоиться об этом в будущем:*

|  |  |
| --- | --- |
| 1  2  3  4 | <manifest ...>      <uses-permission android:name="android.permission.READ\_EXTERNAL\_STORAGE" />      ...  </manifest> |

Однако если ваше приложение использует право на запись WRITE\_EXTERNAL\_STORAGE, права на чтение будут неявно заданы.

Вам не нужно получать какие-либо дополнительные разрешения для сохранения файлов во внутреннем хранилище. Приложения всегда имеют права на запись и чтение из директорий внутреннего хранилища.

Сохранение файлов во внутреннем хранилище

При сохранении файлов во внутреннем хранилище, можно получить соответствующую директорию как объект типа [File](http://developer.android.com/reference/java/io/File.html), вызвав один из методов:

[getFilesDir()](http://developer.android.com/reference/android/content/Context.html#getFilesDir())

Возвращает объект типа File представляющий директорию внутреннего хранилища для вашего приложения.

[getCacheDir()](http://developer.android.com/reference/android/content/Context.html#getCacheDir())

Возвращает объект типа File представляющий директорию внутреннего хранилища для хранения временных кэш файлов. Обязательно удаляйте все файлы, как только вы перестаете в них нуждаться и установите разумное ограничение на количество памяти, которое вы используете в определенный момент времени, например 1Мб. Если в системе останется мало места для хранения, она может удалить ваши временные файлы без предупреждения.

Чтобы создать новый файл в одной из директорий, вы можете использовать конструктор [File()](http://developer.android.com/reference/java/io/File.html" \l "File(java.io.File,%20java.lang.String)), передав в него объект типа File, полученный с помощью соответствующего метода:

|  |  |
| --- | --- |
| 1 | File file = new File(context.getFilesDir(), filename); |

Можно также использовать метод [openFileOutput()](http://developer.android.com/reference/android/content/Context.html" \l "openFileOutput(java.lang.String,%20int)) для получения объекта [FileOutputStream](http://developer.android.com/reference/java/io/FileOutputStream.html), который записывает файл во внутренний каталог. Пример записи некоторого текста в файл:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11 | String filename = "myfile";  String string = "Hello world!";  FileOutputStream outputStream;    try {    outputStream = openFileOutput(filename, Context.MODE\_PRIVATE);    outputStream.write(string.getBytes());    outputStream.close();  } catch (Exception e) {    e.printStackTrace();  } |

Или если необходимо создать временный файл, используйте [createTempFile()](http://developer.android.com/reference/java/io/File.html" \l "createTempFile(java.lang.String,%20java.lang.String)). В следующем примере метод получает имя файла из [URL](http://developer.android.com/reference/java/net/URL.html) и создает файл с указанным именем во внутренней временной директории:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10 | public File getTempFile(Context context, String url) {      File file;      try {          String fileName = Uri.parse(url).getLastPathSegment();          file = File.createTempFile(fileName, null, context.getCacheDir());      catch (IOException e) {          // Ошибка при создании файла      }      return file;  } |

***Примечание:****расположение каталога внутреннего хранилища вашего приложения в файловой системе Android зависит от имени пакета. Технически, другое приложение может прочитать ваши внутренние файлы, если вы установите режим доступа для чтения. Однако другому приложению понадобится также знать название пакета вашего приложения и имя самого файла. Другие приложения не могут просматривать содержимое внутреннего хранилища и не имеют права чтения и записи, пока явно не указано что файл может быть прочитан или записан. Так что пока вы используете флаг*[*MODE\_PRIVATE*](http://developer.android.com/reference/android/content/Context.html#MODE_PRIVATE)*для ваших файлов во внутреннем хранилище, они будут недоступны для других приложений.*

Сохранение файлов во внешнем хранилище

Поскольку внешнее хранилище может быть недоступно – например при подключении устройства к компьютеру или при удалении SD карты, вы должны всегда проверять раздел на доступность, прежде чем попытаться его использовать. Вы можете запросить состояние внешнего хранилища с помощью метода [getExternalStorageState()](http://developer.android.com/reference/android/os/Environment.html" \l "getExternalStorageState()). Если метод вернул состояние, равное [MEDIA\_MOUNTED](http://developer.android.com/reference/android/os/Environment.html#MEDIA_MOUNTED), вы можете читать и записывать ваши файлы. Пример метода проверки внешнего хранилища на доступность:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18 | /\* Проверяем хранилище на доступность чтения и записи\*/  public boolean isExternalStorageWritable() {      String state = Environment.getExternalStorageState();      if (Environment.MEDIA\_MOUNTED.equals(state)) {          return true;      }      return false;  }    /\* Проверяем внешнее хранилище на доступность чтения \*/  public boolean isExternalStorageReadable() {      String state = Environment.getExternalStorageState();      if (Environment.MEDIA\_MOUNTED.equals(state) ||          Environment.MEDIA\_MOUNTED\_READ\_ONLY.equals(state)) {          return true;      }      return false;  } |

Хотя содержимое внешнего хранилища может быть изменено пользователем или другим приложением, есть две категории файлов, которые вы можете в нем хранить:

**Публичные файлы**

Файлы, которые должны быть доступны другим приложениям и пользователям. После удаления приложения эти файлы должны оставаться доступными пользователю. Это, например, фотографии, сделанные в вашем приложении, или скачанные файлы.

**Приватные файлы**

Файлы, по праву принадлежащие приложению, которые должны быть удалены вместе с приложением. Хотя эти файлы технически доступны для пользователя и других приложений, поскольку они находятся во внешнем хранилище, реально они не предоставляют свои данные за пределами приложения. При удалении приложения, все такие файлы будут также удалены. Это, к примеру, временные медиа-файлы.

Если вы хотите сохранить публичные файлы по внешнем хранилище, используйте метод [getExternalStoragePublicDirectory()](http://developer.android.com/reference/android/os/Environment.html" \l "getExternalStoragePublicDirectory(java.lang.String)) для получения объекта типа File соответствующего каталогу. В качестве аргумента метод принимает тип файла, который вы хотите сохранить, логически он может быть организован с другими публичными файлами, например каталоги музыки ([DIRECTORY\_MUSIC](http://developer.android.com/reference/android/os/Environment.html#DIRECTORY_MUSIC)) или картинок ([DIRECTORY\_PICTURE](http://developer.android.com/reference/android/os/Environment.html#DIRECTORY_PICTURES)). Например:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9 | public File getAlbumStorageDir(String albumName) {      // Получение публичного каталога картинок      File file = new File(Environment.getExternalStoragePublicDirectory(              Environment.DIRECTORY\_PICTURES), albumName);      if (!file.mkdirs()) {          Log.e(LOG\_TAG, "Directory not created");      }      return file;  } |

Если вы хотите сохранить приватный файл приложения, воспользуйтесь методом < href="http://developer.android.com/reference/android/content/Context.html#getExternalFilesDir(java.lang.String)" rel='nofollow'>getExternalFilesDir() и передайте имя типа директории. Каждая созданная таким образом директория добавляется к корневой директории вашего приложения и собирает в себе все внешние файлы, которые будут удалены вместе с приложением.

Пример создания директории собственного фотоальбома:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9 | public File getAlbumStorageDir(Context context, String albumName) {      // Получение приватной директории для фотоальбома приложения      File file = new File(context.getExternalFilesDir(              Environment.DIRECTORY\_PICTURES), albumName);      if (!file.mkdirs()) {          Log.e(LOG\_TAG, "Directory not created");      }      return file;  } |

Если ни одно из предопределенных имен каталогов не подходит, вы можете в метод getExternalFilesDir() передать null. Метод возвратит корневой приватный каталог вашего приложения, расположенный во внешнем хранилище.

Помните, что getExternalFilesDir() создает каталоги внутри директории, которая будет удалена вместе с приложением. Если файлы должны быть доступны после удаления приложения, например снимки с камеры, используйте метод getExternalStoragePublicDirectory().

Очень важно использовать имена директорий, представленных константами API, такие как [DIRECTORY\_PICTURES](http://developer.android.com/reference/android/os/Environment.html#DIRECTORY_PICTURES) независимо от того, использовался ли метод getExternalFilesDir() или getExternalStoragePublicDirectory(). Предопределенные имена директорий позволяют системе правильно обрабатывать ваши файлы. Например, файлы сохраненные в директории [DIRECTORY\_RINGTONES](http://developer.android.com/reference/android/os/Environment.html#DIRECTORY_RINGTONES) будут помечены системным медиа-сканером как мелодии звонка, а не как музыкальные файлы.

Запрос свободного пространства

Если вы заранее знаете как много данных вам надо сохранить, вы можете узнать достаточно ли места, не дожидаясь исключения [IOException](http://developer.android.com/reference/java/io/IOException.html), с помощью методов [getFreeSpace()](http://developer.android.com/reference/java/io/File.html" \l "getFreeSpace()) или [getTotalSpace()](http://developer.android.com/reference/java/io/File.html" \l "getTotalSpace()). Данные методы позволяют узнать количество свободного и общего пространства на разделе хранилища, соответственно. Данная информация также полезна, чтобы избежать заполнения раздела хранилища выше определенного порога.

Однако система не гарантирует, что вы можете записать столько же, сколько показывает метод getFreeSpace(). Если свободного места немного больше, чем вы хотите сохранить, или если файловая система заполнена меньше, чем на 90%, вероятно можете продолжать, иначе лучше воздержаться.

***Примечание:****вам не обязательно проверять количество свободного места перед записью вашего файла. Вы можете попробовать записать файл и обработать исключение IOException. Вы также можете не знать точно сколько места вам нужно. Например, если вы меняете кодировку файла перед сохранением, например конвертируете PNG в JPEG, вы можете не знать конечного размера файла.*

Удаление файла

Всегда удаляйте файлы, если они больше не используются. Самый простой способ удалить открытый файл – вызвать для него метод [delete()](http://developer.android.com/reference/java/io/File.html" \l "delete()):

|  |  |
| --- | --- |
| 1 | myFile.delete(); |

Если файл сохранен во внешнем хранилище, вы можете запросить [Контекст](http://developer.android.com/reference/android/content/Context.html) найти и удалить файл, используя метод [deleteFile()](http://developer.android.com/reference/android/content/Context.html" \l "deleteFile(java.lang.String)):

|  |  |
| --- | --- |
| 1 | myContext.deleteFile(fileName); |

***Примечание:****При удалении приложения, система удаляет следующие файлы:*

* Файлы, сохраненные во внутреннем хранилище
* Файлы внешнего хранилища, сохраненные с использованием метода getExternalFilesDir()

Однако необходимо регулярно удалять вручную все временные файлы, созданные с помощью getCacheDir() и другие файлы, которые больше не нужны.