데이터 저장을 위해서 파일을 이용하는 방법. 파일에 데이터를 write하고, 파일에서 데이터를 read 해온다.

● 안드로이드에서 파일 관련 프로그램은 대부분 자바 API를 그대로 사용하므로, java.io 패키지의 클래스들을 이용해서 작성해야 한다.

File	파일 및 디렉터리를 지칭하는 클래스
FileInputStream	파일에서 바이트 데이터를 읽기 위한 함수 제공
FileOutputStream	파일에 바이트 데이터를 쓰기 위한 함수 제공
FileReader	파일에서 문자열 데이터를 읽기 위한 함수 제공
FileWriter	파일에서 문자열 데이터를 쓰기 위한 함수 제공

안드로이드 파일 저장 공간: 1. 내장 메모리: 앱내 저장소 2. 외장 메모리: 1) 앱 저장 공간: 다든 앱에서 접근할 수 없다 2) 공용 저장 공간: 모든 앱이 접근할 수 있다. (ex. 내장 메모리 공간이 작아서 파일을 많이 저장할 수 없을 경우 외장 메모리의 앱 저장 공간을 이용해서 저장하면서 동시에 다든 외부 앱이 접근하지 못하도록 할 수도 있다). 내장 메모리나 외장 메모리의 앱 저장소는 개별 앱을 위한 공간이므로 앱이 삭제되면 파일도 모두 삭제되지만, 공용 저장소는 모든 앱을 위한 공간이므로 파일을 만든 앱을 삭제해도 파일은 삭제되지 않는다.

● 저장소와 관련된 각종 정보는 Environmnet 클래스로 얻을 수 있다. 메모리 공간의 경로는 스마트폰마다 다를 수 있기 때문에, 파일을 이용할 때 경로를 문자열로 지정하는 것이 아니라 아래의 함수를 이용하여 스마트폰에 따라 다르게 대응해야 한다.

Environment.getExternalStorageState()	외부 저장 공간 상태
Environment.getExternalStorageDirectory().getAbsolutePath()	외부 저장 공간 경로
Environment.getDataDirectory().getAbsolutePath()	내부 저장 공간 경로

● 외부 저장 공간 이용: 외부 저장 공간을 이용하려면 먼저 스마트폰에서 이를 제공하고 있는지 판단해야 한다. Environment.getExternalStorageState() 함수를 통해 얻은 상태 값이 Envirionment.MEDIA_MOUNTED이면 외부 저장 공간이 제공된다는 의미이다. 그럴 경우 상태 값이 Environment.MEDIA_MOUNTED_READ_ONLY인지를 판단해서 그럴 경우 읽기만 할 수 있고 ,아니라면 읽고 쓰기가 모두 가능하다.

```
String state = Environment.getExternalStorageState();
if (state.equals(Environment.MEDIA_MOUNTED)) {
    if (state.equals(Environment.MEDIA_MOUNTED_READ_ONLY)) {
        externalStorageReadable = true;
        externalStorageWritable = false;
    } else {
        externalStorageReadable = true;
        externalStorageWritable = true;
} else {
    externalStorageReadble = externalStroageWritable = false;
}
```

외부 저장 공간에 파일을 읽거나 쓰는 작업을 할 때, AndroidManifest.xml 파일에서 android.permission.READ_EXTERNAL_STORAGE 또는/와 android.permission.WRITE_EXTERNAL_STORAGE 퍼미션을 설정해 주어야 한다. 또한 만약 File API를 사용한다면 Android 10버전부터는 2가지의 퍼미션 설정과 함께 requestLegacyExternalStorage값 설정도 해 주어야 한다. 파일을 이용하는 방식은 안드로이드 버전에 따라 다르고 API 호환성까지 고려한다면 결과적으로 외장 메모리를 사용할 때는 다음처럼 선언하는 것이 좋다. (예외: 외장 메모리 공간에 리드와 라이트를 한다고 해도, ContentResolver에서 제공하는 InputStream 등을 이용해 스트림을 받아서 이용하는 것이라면 선언할 필요가 없다. 하지만 File API를 사용해 직접 리드 함수, 롸이트 함수를 호출한다고 하면 퍼미션을 설정해야 한다).

```
// AndroidManifest.xml

<manifest.........>
    <uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE" />
    <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
    <application>
        .........
        android:requestLegacyExternalStorage="true">
        ........
    </application>
</manifest>
```

▶ 앱 저장소 이용: 각 앱의 개별 공간이 할당되어 있으며, 기본적으로 각각의 앱에서 할당받은 공간에만 접근할 수 있다.

☞ 외장 메모리의 앱 저장소 위치 구하기: getExternalFilesDir() 함수로 구한다. getExternalFilesDir(null) 함수가 반환하는 위치는 기기에 따라 다르지만, 보통 외장 메모리 Android 아래에 패키지명(앱 식별자)으로 디렉터리가 생기고, 그 아래에 있는 files 디렉터리에 각각의 앱별 파일이 저장된다.

```
File file = getExternalFilesDir(null);
Log.d("kkang", file.getAbsolutePath());
```

```
/storage/emulated/0/Android/data/패키지명/files
    외장메모리      /        앱별 저장소
```

getExternalFilesDir() 함수의 매개변수로 파일의 종류를 지정하며, null이 아닌 다음과 같은 Environment 상수를 전달할 수도 있다.

Environment.DIRECTORY_PICTURES	이미지 파일 저장 폴더

Environment.DIRECTORY_MUSIC	음악 파일 저장 폴더
Environment.DIRECTORY_MOVIES	영상 파일 저장 폴더
Environment.DIRECTORY_DOWNLOADS	다운로드한 파일 저장 폴더
Environment.DIRECTORY_DCIM	카메라로 촬영한 사진 저장 폴더
Environment.DIRECTORY_ALARMS	알람으로 사용할 오디오 파일 저장 폴더
Environment.DIRECTORY_NOTIFICATIONS	알림음으로 사용할 오디오 파일 저장 폴더

```
// getExternalFilesDir(Environment.DIRECTORY_ALARMS)일 경우, 파일이 저장되는 위치는 다음과
/storage/emulated/0/Android/data/패키지명/files/Alarms
```

☞ 외장 메모리 앱 저장소에 파일을 쓰기

```
// Environment.DIRECTORY_DOCUMENTS 타입의 앱 저장소에 text.txt라는 파일을 만들어서
// 문자열 데이터를 저장하는 코드

try {
    File file = new File(getExternalFilesDir(Environment.DIRECTORY_DOCUMENTS), "test.txt
    // 파일이 없다면 새로 만들어 준다
    if (!file.exists()) {
        file.createNewFile();
    }

    FileWriter writer = new FileWriter(file, true);
    // 문자열 데이터를 저장하기 위해 FileWriter 사용. 이미지 등의 바이트 파일을 저장하려면 Out
    writer.write("hello world");
    writer.flush();
    writer.close();
} catch (Exception e) {
    e.printStackTrace();
}
```

☞ 외장 메모리 앱 저장소의 파일을 읽기

```
// Environment.DIRECTORY_DOCUMENTS 타입의 앱 저장소에 text.txt라는 파일을 찾아서 읽고,
// StringBuffer 객체에 저장하는 코드

try {
    File file = new File(getExternalFilesDir(Environment.DIRECTORY_DOCUMENTS), "test.txt
    BufferedReader reader = new BufferedReader(new FileReader(file));
    StringBuffer buffer = new StringBuffer();
    String line;

    while ((line = reader.readline()) != null) {
        buffer.append(line);
    }

    reader.close();
    Log.d("kkang", buffer.toString());
} catch (Exception e) {
    e.printStackTrace();
}
```

만약 각 앱 저장소에 있는 파일에 외부 앱이 접근되도록 하려면 FileProvider을 이용하면 된다 (외부에서 이용할 수 있는 파일명이나 경로를 지정할 수 있다).

▶ 공용 저장소 이용: 앱에서 만든 파일을 모든 앱에서 이용할 수 있게 해야 하는 경우 공용 저장소를 이용한다. 공용 저장소는 모든 앱을 위한 공간이므로 파일을 만든 앱을 삭제해도 파일은 삭제되지 않는다. 공용 저장소의 폴더는 파일 종듀로 구분되어 있다,. 사진이나 음원, 또는 문서 등 그 종듀에 따라 해당 파일이 저정되는 폴더가 지정되어 있다. 이 공용 저장소에는 파일 경로로 직접 접근하지 않고, 시스템이 제공하는 API를 이용한다.

```
// 공용 저장소에 저장된 이미지 파일의 정보를 가져와 로그로 출력하는 코드
// 외장 메모리의 파일 정보를 이용하지만 파일 경로를 직접 사용하지는 않았다
try {
    String[] projection = {
        MediaStore.Images.Media._ID,
        MediaStore.Images.Media.DISPLAY_NAME
    };
    Cursor cursor = getContentResolver().query(
        // 공용 저장소 중에서 이미지가 저장된 곳
        MediaStore.Images.Media.EXTERNAL_CONTENT_URI,
        projection,
        null, null, null
    );

    while (cursor.moveToNext()) {
        Log.d("kkang", "_id : " + cursor.getLong(0) + ", name : " + cursor.getString(1))
    }
} catch (Exception e) {
    e.printStackTrace();
}
```

getContentResolver().query() 함수의 첫 번째 매개변수에 Uri 값을 지정할 때 MediaStore.Images를 이용했는데, 이는 안드로이드폰 이미지 파일이 저장되는 공용 저장소인 DCIM과 Pictures 디렉토리를 의미한다. MediaStore.Video는 DCIM과 Movies, 그리고 Pictures 디렉터리이다. 그리고 MediaStore.Audio는 Alarms, Audiobooks, Music, Notifications, Podcasts, Ringtones 디렉터리이다.

```
// 공용 저장소의 이미지 파일 정보를 이용해 이미지 데이터를 가져오고, 화면에 출력하는 코드

Uri contentUri = ContentUris.withAppendedId(
    MediaStore.Images.Media.EXTERNAL_CONTENT_URI,
    cursor.getLong(0)
);

InputStream inputStream = getContentResolver().openInputStream(contentUri);
// stream 객체에서 작업 수행
BitmapFactory.Options option = new BitmapFactory.Options();
option.inSampleSize = 10;
Bitmap bitmap = BitmapFactory.decodeSteram(inputStream, null, option);
binding.resultImageView.setImageBitmap(bitmap);
```

ContentUris.withAppendedId() 함수의 두 번째 매개변수가 가져온 이미지의 식별자이다. 이렇게 하면 해당 이미지 파일을 이용할 수 있는 Uri 값이 반환되고, Uri 값으로 이미지를 읽는 InputStream 객체를 얻는다. getContentResolver().openInputSTream(contentUri)로 파일을 읽는 Stream 객체를 얻고, 이 객체로 이미지 데이터를 획득한다.

● 내부 저장 공간 이용: 내부 저장 공간 경로는 getFileDir() 함수로 얻을 수 있다.

```
// getFileDir() 함수로 앱의 내부 저장 공간에 myfile.txt 파일을 만들고, 그 파일에 문자열 데이터
// 저장 경로를 지칭하는 방법만 외부 저장 공간과 차이가 있다.
String filename = "myfile.txt"
FileWriter writer;

try {
    File file = new File(getFilesDir(), filename);
    if (!file.exists()) {
        file.createNewFile();
    }
    writer = newFileWriter(file, true);
    writer.write(content);
    writer.flush();
    writer.close();
} catch (Exception e) {
    e.printStackTrace();
}
```

```
// 내부 저장 공간에서 myfile.txt 파일의 내용을 읽어 StringBuffer에 저장한 예
// getFileDir() 함수를 이용하여 파일의 경로를 내부 저장 공간으로 지정해 준다.
file = new File(getFilesDir(), "myfile.txt");

try {
    BufferedReader reader = new BufferedReader(new FileReader(file));
    StringBuffer buffer = new StringBuffer();
    String line;
    while ((line == reader.readLine()) != null) {
        buffer.append(line);
    }
    reader.close();
} catch (Exception e) {
    e.printStackTrace();
}
```

```java
// MainActivity.java - 파일 쓰기
package com.android.practicehandlingfile;

import androidx.appcompat.app.AppCompatActivity;

import android.content.Intent;
import android.os.Bundle;
import android.os.Environment;
import android.widget.Toast;

import com.android.practicehandlingfile.databinding.ActivityMainBinding;
import java.io.File;
import java.io.FileWriter;

public class MainActivity extends AppCompatActivity {
    ActivityMainBinding binding;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        binding = ActivityMainBinding.inflate(getLayoutInflater());
        setContentView(binding.getRoot());

        binding.saveButton.setOnClickListener(view -> {
            // 외장, 앱별 read/write
            try {
                File file = new File(getExternalFilesDir(Environment.DIRECTORY_DOCUMENTS

                // 파일이 없다면 새로 만들어 주기
                if (!file.exists()) {
                    file.createNewFile();
                }
                FileWriter writer = new FileWriter(file, true);
                writer.write(binding.content.getText().toString());
                writer.flush();
                writer.close();
                Toast.makeText(this, "file saved", Toast.LENGTH_SHORT).show();

                Intent intent = new Intent(this, ReadFileActivity.class);
                startActivity(intent);

            } catch (Exception e) {
                e.printStackTrace();
            }
        });
    }
}
```

```java
// ReadFileActivity.java - 파일에 데이터 읽어와서 화면에 뿌리기
package com.android.practicehandlingfile;

import androidx.appcompat.app.AppCompatActivity;

import android.os.Bundle;
import android.os.Environment;

import com.android.practicehandlingfile.databinding.ActivityReadFileBinding;

import java.io.BufferedReader;
import java.io.File;
import java.io.FileReader;

public class ReadFileActivity extends AppCompatActivity {

    ActivityReadFileBinding binding;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        binding = ActivityReadFileBinding.inflate(getLayoutInflater());
        setContentView(binding.getRoot());

        try {
            File file = new File(getExternalFilesDir(Environment.DIRECTORY_DOCUMENTS), "

            BufferedReader br = new BufferedReader(new FileReader(file));
            StringBuffer buffer = new StringBuffer();
            String line;

            while ((line = br.readLine()) != null) {
                buffer.append(line);
            }
            br.close();

            binding.fileResult.setText(buffer.toString());
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

Resources: 깡쌤의 안드로이드 프로그래밍