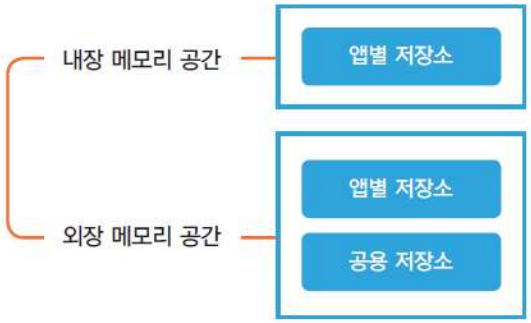
02

02-3. 파일

데이터 영속화

17-2 파일에 보관하기

- 안드로이드 앱에서 파일을 다룰 때는 대부분 java.io 패키지에서 제공하는 클래스를 이용
  - File: 파일 및 디렉터리를 지칭하는 클래스입니다.
  - FileInputStream / FileOutputStream: 파일에서 바이트 스트림으로 데이터를 읽거나 쓰는 클래스입니다.
  - FileReader / FileWriter: 파일에서 문자열 스트림으로 데이터를 읽거나 쓰는 클래스입니다.



17-2 파일에 보관하기

내장 메모리의 파일 이용하기

- 앱의 패키지명으로 디렉터리를 만들어 주는데, 이 디렉터리가 바로 앱의 내장 메모리 공간
- 파일을 내장 메모리에 저장하려면 java.io의 File 클래스를 이용

• 파일 객체 생성 후 데이터 쓰기

```
val file = File(filesDir, "test.txt")
val writeStream: OutputStreamWriter = file.writer()
writeStream.write("hello world")
writeStream.flush()
```

• 파일의 데이터 읽기

```
val readStream: BufferedReader = file.reader().buffered()
readStream.forEachLine {
    Log.d("kkang", "$it")
}
```

• Context 객체의 함수 사용

```
openFileOutput("test.txt", Context.MODE_PRIVATE).use {
    it.write("hello world!!".toByteArray())
}
openFileInput("test.txt").bufferedReader().forEachLine {
    Log.d("kkang", "$it")
}
```

17-2 파일에 보관하기

외장 메모리의 파일 이용하기

- Environment.getExternalStorageState() 함수로 외장 메모리를 사용할 수 있는지부터 확인

• 외장 메모리 사용 가능 여부 판단

```
if (Environment.getExternalStorageState() == Environment.MEDIA_MOUNTED) {
    Log.d("kkang", "ExternalStorageState MOUNTED")
} else {
    Log.d("kkang", "ExternalStorageState UNMOUNTED")
}
```

17-2 파일에 보관하기

- 앱별 저장소 이용
  - 외장 메모리 공간은 앱별 저장소와 공용 저장소로 구분
  - 앱별 저장소는 개별 앱에 할당된 공간
  - 앱별 저장소의 파일을 외부 앱에서 접근하게 하려면 파일 프로바이더로 공개해야 합니다.
  - 외장 메모리의 앱별 저장소 위치는 getExternalFilesDir() 함수로 구합니다.

• 앱별 저장소에 접근

```
val file: File? = getExternalFilesDir(null)
Log.d("kkang", "${file?.absolutePath}")
```

17-2 파일에 보관하기

- getExternalFilesDir() 함수를 이용할 때 매개변수는 파일의 종류를 나타내며 null이 아닌 다음과 같은 Environment의 상수를 전달
  - Environment.DIRECTORY_PICTURES
  - Environment.DIRECTORY_DOCUMENTS
  - Environment.DIRECTORY_MUSIC
  - Environment.DIRECTORY_MOVIES

• 앱별 저장소에 파일 쓰기와 읽기

```
// 파일 쓰기
val file: File = File(getExternalFilesDir(null), "test.txt")
val writeStream: OutputStreamWriter = file.writer()
writeStream.write("hello world")
writeStream.flush()
// 파일 읽기
val readStream: BufferedReader = file.reader().buffered()
readStream.forEachLine {
    Log.d("kkang", "$it")
}
```

17-2 파일에 보관하기

- 공용 저장소 이용

  - 모든 앱이 이용할 수 있는 저장소
  - 안드로이드 12(API Level 32)까지는 공용 공간을 이용하기 위해서는 android.permission.READ_EXTERNAL_STORAGE와 android.permission.WRITE_EXTERNAL_STORAGE를 퍼미션으로 추가 장소
  - 안드로이드 13(API Level 33)부터는 공용 공간의 파일 타입을 구분해서 이용하므로 미디어 파일도 이미지인지 비디오인지에 따라 다르게 퍼미션을 설정
  - 안드로이드 14부터는 사용자에게 앱에서 미디어 파일에 대한 접근을 다시 선택할 수 있게 하기 위한 android.permission.READ_MEDIA_VISUAL_USER_SELECTED 퍼미션이 추가

• 매니페스트 퍼미션 설정

```
<!-- API Level 32 까지 -->
<uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE"/>
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>
<!-- API Level 33 부터 -->
<uses-permission android:name="android.permission.READ_MEDIA_IMAGES" />
<uses-permission android:name="android.permission.READ_MEDIA_VIDEO" />
<!-- API level 34 -->
<uses-permission android:name="android.permission.READ_MEDIA_VISUAL_USER_SELECTED" />
```

17-2 파일에 보관하기

- 공용 공간의 미디어 파일에 대한 정보를 획득할

• 파일 정보 획득

```
val projection = arrayOf(
    MediaStore.Images.Media._ID,
    MediaStore.Images.Media.DISPLAY_NAME,
    MediaStore.Images.Media.SIZE,
    MediaStore.Images.Media.MIME_TYPE,
)

val collectionUri = if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.Q) {
    MediaStore.Images.Media.getContentUri(MediaStore.VOLUME_EXTERNAL)
} else {
    MediaStore.Images.Media.EXTERNAL_CONTENT_URI
}
```

```
contentResolver.query(
    collectionUri,
    projection,
    null,
    null,
    null
)?.use { cursor ->
    while (cursor.moveToNext()) {
        val uri = cursor.getLong(0)
        val name = cursor.getString(1)
        val size = cursor.getLong(2)
        val mimeType = cursor.getString(3)

        Log.d("kkang", "uri: $uri, name: $name, size: $size, mimeType: $mimeType")
    }
}
```

17-2 파일에 보관하기

- MediaStore.Images.Media._ID 데이터가 파일의 식별자, 식별자를 이용해 해당 파일의 내용을 읽기

• 파일 읽어서 화면 출력

```
val contentUri: Uri = ContentUris.withAppendedId(
    MediaStore.Images.Media.EXTERNAL_CONTENT_URI,
    uri
)

val resolver = applicationContext.contentResolver
resolver.openInputStream(contentUri).use { stream ->
    // stream 객체에서 작업 수행
    val option = BitmapFactory.Options()
    option.inSampleSize = 10
    val bitmap = BitmapFactory.decodeStream(stream, null, option)
    binding.imageView.setImageBitmap(bitmap)
}
```

감사합니다

단단히 마음먹고 떠난 사람은
산꼭대기에 도착할 수 있다.
산은 올라가는 사람에게만 정복된다.

윌리엄 셰익스피어
William Shakespeare