Data Storage: Internal/External Storage, Static Files and Storage Access Framework

Sensor Based Mobile Applications

Patrick Ausderau, Ulla Sederlöf, Jarkko Vuori

Helsinki Metropolia University of Applied Science

2022

Metropolia

Outline

Doc: `https://developer.android.com/guide/topics/data`

Data Storage on Android

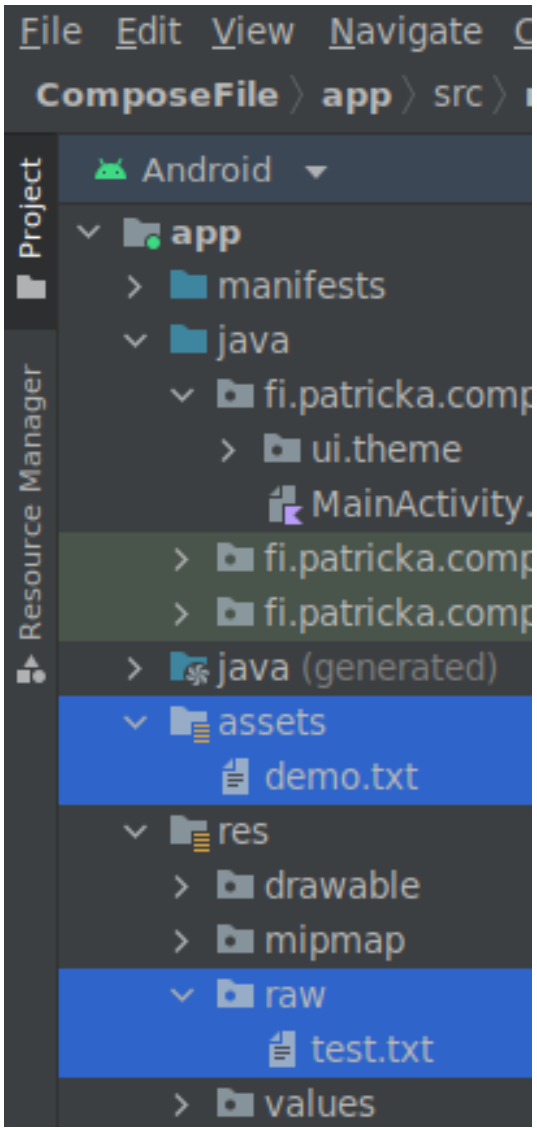
- Shared Preferences
  - Store private primitive data in key-value pairs
- Internal Storage
  - Store private data on the device memory
- External Storage
  - Store (public (deprecated)) or half-private data on the shared external storage
- SQLite Databases / Room
  - Store structured data in a private database
- Media
  - Shareable media files (images, audio files, videos)

Data Storage on Android

- Storage Access Framework
  - allows users to interact with a system picker to choose a documents provider and select specific files and directories for your app to create, open, or modify
- Content providers
  - Manages access to a central repository of data
- Network Connection
  - Store data on the web with your own network server
- Data Backup
  - Store data on user's Google Drive or Android Backup Service in order to restore user data on new devices
- FileProvider API
  - extension of content provider to securely share files from your app to another app using content URIs (with temporary permissions granted only to the receiving app)

Static Files in Application at Compile Time

- If you want to save a static file in your application at compile time, save the file in your project `res/raw/` or `assets/` directory.
- Note: once your application is compiled into .apk, it is immutable! i.e. impossible to modify the app at runtime. So you can not modify files stored in your application.



Static Files in Application at Compile Time

- In `res/raw/`, filename must be a valid kotlin identifier. Open it with `application.resources.openRawResource()`, passing the `R.raw.<filename>` resource ID.
  - If that folder do not exist: In Android Studio, select your project, then File ⇒ New ⇒ Android resource directory. Choose Resource type: raw.
- In `assets/`, you have more freedom for filenames and subfolders. You can open it with `application.assets.open("<filename>")`.
  - If that folder do not exist: In Android Studio, select your project, then File ⇒ New ⇒ Folder ⇒ Assets Folder
- These methods return an `InputStream` that you can use to read the file (e.g. chain it with buffered Reader).

Static Files in Application at Compile Time - Example

```
1  @Composable
2  fun MainView(application: Application) {
3    ComposeFileTheme {
4      Column(Modifier.padding(16.dp)) {
5        Text(
6          text =
               application.resources.openRawResource(R.raw.test)
               .bufferedReader().readText()
7        )
8        Text(
9          text =
               application.assets.open("demo.txt").bufferedReader()
               .readText()
10       )
11     }
12   }
13 }
```

Shared Storage

- shared storage can be used for user data that can or should be accessible to other apps and saved even if the user uninstalls your app
  - `MediaStore` API provides standard public directories to store and share audio, video, pictures,...files[1]
  - `Storage Access Framework` has special directory for containing other file types, such as PDF, EPUB,...

[1] `https://developer.android.com/training/data-storage/shared/media`

Storage Access Framework

- with Android >= 4.4 (API 19), your app can interact with a documents provider, including external storage volumes and cloud-based storage
- allows users to interact with a system picker to choose a documents provider and select specific documents and other files for your app to create, open, or modify
- because the user is involved in selecting the files or directories that your app can access, this mechanism doesn't require any system permissions
- the files are stored outside of an app-specific directory and outside of the media store, so they remain on the device after your app is uninstalled

Storage Access Framework

The Storage Access Framework supports the following use cases for accessing files and other documents:

- `ACTION_CREATE_DOCUMENT` intent action allows users to save a file in a specific location
  - cannot overwrite an existing file! If a file with same filename exist, will append (#) at the end. E.g. if file "demo.pdf" exist, will create "demo.pdf (1)"
- `ACTION_OPEN_DOCUMENT` intent action allows users to select a specific document or file to open
- `ACTION_OPEN_DOCUMENT_TREE` intent action, available on Android $>=$ 5.0 (API 21), allows users to select a specific directory, granting your app access to all of the files and sub-directories within that directory

Create/Open shared file for modification - Example

```
1  // inside Activity (or fragment) class
2  companion object{
3      private lateinit var uri: Uri
4      fun isInitialized():Boolean = ::uri.isInitialized
5  }
6
7  private fun openDocument(create: Boolean = false) {
8    val intent = Intent(if (create) Intent.ACTION_CREATE_DOCUMENT
         else Intent.ACTION_OPEN_DOCUMENT).apply {
9      addCategory(Intent.CATEGORY_OPENABLE)
10     if(Build.VERSION.SDK_INT >= Build.VERSION_CODES.R){
11       type = "*/*"
12       putExtra(Intent.EXTRA_MIME_TYPES, arrayOf("text/plain"))
13     } else {
14       type = "plain/text"
15     }
16     if (create) putExtra(Intent.EXTRA_TITLE, "sharedFile.txt")
17   }
18   startForResult.launch(intent)
19 }
```

Create/Open shared file for modification - Example continued

```
1  private val startForResult = registerForActivityResult(
       ActivityResultContracts.StartActivityForResult()) {
2    if (it.resultCode == Activity.RESULT_OK) {
3      it.data?.also {data ->
4        uri = data.data ?: return@registerForActivityResult
5        contentResolver.takePersistableUriPermission(uri,
             Intent.FLAG_GRANT_WRITE_URI_PERMISSION)
6        Log.d("FILE","file: $uri")
7      }
8    }
9  }
```

Create/Open shared file for modification - Example continued

```
1  // read content of file from Uri
2  private fun readTextFromUri(): String {
3    if(!isInitialized()) return ""
4    val stringBuilder = StringBuilder()
5    contentResolver.openInputStream(uri)?.use { inputStream ->
6      BufferedReader(InputStreamReader(inputStream)).use { reader
           ->
7        var line: String? = reader.readLine()
8        while (line != null) {
9          stringBuilder.append("$line\n")
10         line = reader.readLine()
11       }
12     }
13   }
14   return stringBuilder.toString()
15 }
```

Create/Open shared file for modification - Example continued

```
1  // append text at the end of the file
2  private fun alterFile(text: String) {
3    if (!isInitialized()) return
4    contentResolver.openFileDescriptor(uri, "rw")?.use {
5      FileWriter(it.fileDescriptor).use {
6        //append overwrite existing content :(
7        // FileWriter constructor with Uri parameter does not
             accept append flag :(
8        it.append(readTextFromUri())
9        it.append("${text}\n")
10       it.flush()
11     }
12   }
13 }
```

Create/Open shared file for modification - Example end

```
1  override fun onCreate(savedInstanceState: Bundle?) {
2    // super, setContent, Column, remember text, TextField...
3    Button(onClick = { openDocument(true) }) {
4      Text(stringResource(R.string.create))
5    }
6    Button(onClick = { openDocument() }) {
7      Text(stringResource(R.string.open))
8    }
9    Button(onClick = { alterFile(text) }) {
10     Text(stringResource(R.string.save))
11   }
12 }
```

Lab_w3_d5 Shared folder

- Create an application that will show the list of files and folders (recursively) from a location chosen by the user `Intent.ACTION_OPEN_DOCUMENT_TREE`
- hint:

```
val documentsTree =
DocumentFile.fromTreeUri(getApplication(),
directoryUri)
val childDocuments = documentsTree.listFiles()
```

  check methods/attributes `.isDirectory` `.name` and `.type` form DocumentFile class[2]
- As you might iterate through a large number of files/sub-folders within the directory accessed, your app's performance might be reduced ⇒ use async (e.g. coroutines)

[2] `https://developer.android.com/reference/androidx/documentfile/provider/DocumentFile`