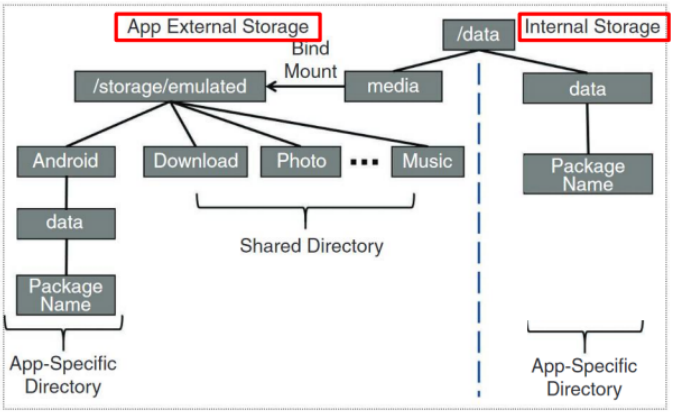
14. Lesson 02/05/23

Data Storage (II)

Android Storage Recap

- Files:
  - App-Specific storage - Internal or External
  - Shared Storage - Media & Doc
- Preferences
- Databases



App-specific files Storage

Both **Internal and External storage include a dedicated location** for:

- Storing persistent files.
- Storing cache data.

Files stored in these directories are meant for use only by your app (App-Specific).

Otherwise use Shared storage (Photo, Video, Docs, ecc…)

When **storing sensitive data** (data that shouldn't be accessible from any other app), use:

- Internal Storage **(Internal Storage —> data being hidden from users.)**
- Preferences
- Database

When the user uninstalls your app, the files saved in app-specific storage are removed.

External Storage for App-specific files

- If internal storage doesn't provide enough space to store app-specific files —> use **external storage.**
- The system provides directories within external storage where an app can organize files that provide value to the user only within your app.
- The files in these directories aren't guaranteed to be accessible, such as when a removable SD card is taken out of the device. If your app's functionality depends on these files —> **internal storage.**

	Type of content	Access method	Permissions needed	Can other apps access?	Files removed on app uninstall?
App-specific files	Files meant for your app's use only	From internal storage, getFilesDir() or getCacheDir() From external storage, getExternalFilesDir() or getExternalCacheDir()	Never needed for internal storage Not needed for external storage when your app is used on devices that run Android 4.4 (API level 19) or higher	No	Yes
Media	Shareable media files (images, audio files, videos)	MediaStore API	READ_EXTERNAL_STORAGE when accessing other apps' files on Android 11 (API level 30) or higher READ_EXTERNAL_STORAGE or WRITE_EXTERNAL_STORAGE when accessing other apps' files on Android 10 (API level 29) Permissions are required for **all** files on Android 9 (API level 28) or lower	Yes, though the other app needs the READ_EXTERNAL_STORAGE permission	No
Documents and other files	Other types of shareable content, including downloaded files	Storage Access Framework	None	Yes, through the system file picker	No
App preferences	Key-value pairs	Jetpack Preferences library	None	No	Yes

Always check availability of storage

The **external storage may be unavailable:**

- Such as when the user has mounted the storage to a PC or has removed the SD card that provides the external storage.
- —> You should always verify that the volume is available before accessing it:

```
/* Checks if external storage is available for read and write */
public boolean isExternalStorageWritable() {
  String state = Environment.getExternalStorageState();
  if (Environment.MEDIA_MOUNTED.equals(state)) {
    return true;
  }
  return false;
}

/*If the returned state is equal to MEDIA_MOUNTED, then you can read and write your files.*/
```

```
/* Checks if external storage is available to at least read */
public boolean isExternalStorageReadable() {
  String state = Environment.getExternalStorageState();
  if (Environment.MEDIA_MOUNTED.equals(state) ||
    Environment.MEDIA_MOUNTED_READ_ONLY.equals(state)) {
    return true;
```

```
  }
  return false;
}
```

Accessing External storage directories

1. Get a path using ***getExternalFilesDir()***
2. Create file

```
/*Example*/
File path = getExternalFilesDir(Environment.DIRECTORY_PICTURES);
File file = new File(path, "DemoPicture.jpg");
```

Select a physical storage location

A device that allocates a partition of its internal memory as external storage can also provide an SD card slot.

This means that the device has multiple physical volumes that could contain external storage, so you need to select which one to use for your app-specific storage.

```
File[] externalStorageVolumes =
ContextCompat.getExternalFilesDirs(getApplicationContext(), Environment.DIRECTORY_PICTURES);
// probably a partition of the device internal memory as external storage
File pathPrimaryExternalStorage = externalStorageVolumes[0];
// probably this is the SD card
File pathSecondaryExternalStorage = externalStorageVolumes[1];
```

The first element in the returned array (i.e., [0]) is considered the **primary external storage volume**.

How much storage left?

- If there is **not enough space**, throws ***IOException.***
- If you know the size of the file, check against space:
  - ***getFreeSpace()***
  - ***getTotalSpace()***
- If you do not know how much space is needed:
  - try/catch ***IOException***

Delete files no longer needed

- **External Storage —> *myFile.delete();***
- **Internal Storage —> *myContext.deleteFile(fileName);***

Do not delete the user's files!

When the user uninstalls your app, your app's private storage directory and all its contents are deleted.

—> **Do not use private storage for content that belongs to the user!**

For example:

- Photos captured or edited with your app.
- Music the user has purchased with your app.