INTRODUCTION TO DATA PERSISTENCE IN ANDROID

TOPICS

- Introduction
- Reading and writing to files on the device
- Different storage options
  - Internal storage
  - External storage
- Examples

DATA PERSISTENCE

- Storing data "permanently " so that it can be retained if the device is shutdown. (a. k. a. non-volatile storage.)
- Preserving data as
  - files on the device
  - key-value pairs
  - Database

PERSISTENT DATA IN ANDROID

- Android provides several options for you to save your app data
- **1- Files I/O**
  - Internal file storage: Store app-private files on the device
  - External file storage: Store files on the shared external file system. This is usually for shared user files, such as photos.
- 2- Shared preferences: Store private primitive data in key-value pairs.
- 3- Databases: Store structured data in a private database.
- 4- Networking: store data on remote server

Storage type	Data type
Files IO	storing large data , images, music files
Shared preferences:	user preferences , name-value pair
Databases:	storing relational data

CATEGORIES OF STORAGE LOCATIONS

- Android provides two types of physical storage locations:
- *internal storage* (smaller in size , more reliable place)
- *external storage*. (could include Removable volumes, such as an SD card)

	Type of content	Access method	Permissions needed	Can other apps access?	Files removed on app uninstall?
		From internal storage, getFilesDir() or getCacheDir()	Never needed for internal storage		
App-specific files	Files meant for your app's use only	From external storage, getExternalFilesDir() or getExternalCacheDir()	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	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
Database	Structured data	Room persistence library	None	No	

INTERNAL STORAGE

- to store sensitive information that other apps shouldn't access.
- Stored in the application's sandbox
- Other applications cannot access files stored here
- Files are deleted along with your app when it is uninstalled
- No permissions required for reading/writing files here

- Get more information
  https://developer.android.com/guide/topics/data/data-storage.html

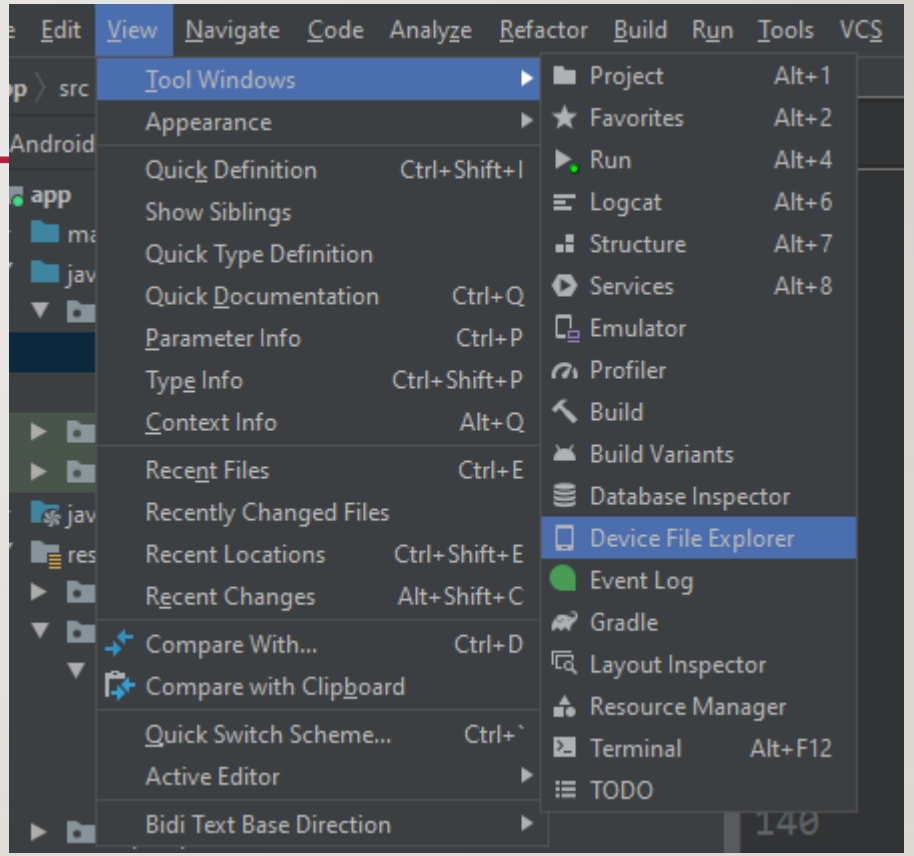
ACCESS METHOD

- openFileInput(String name)
  - Returns a FileInputStream for reading a file from the application's internal storage space
- openFileOutput(String name, int mode)
  - Returns a FileOutputStream for writing to a file in the application's internal storage space
- getFilesDir()
  - Returns the absolute path to the file system where your internal files are saved
- getDir()
  - Creates (or opens an existing) folder within your internal storage

WRITE TEXT TO A FILE:

```
public void writeToFile(View view) {
    String filename = "myfile.txt";
    String fileContents = "Hello world!";
    try {
        FileOutputStream fos = openFileOutput(filename, Context.MODE_PRIVATE);
        fos.write(fileContents.getBytes());
    }catch ( Exception e) {

        Log.d("Files" , "Error");
    }
}
```

Files are stored in your apps sandbox in the /data/data folder



https://developer.android.com/studio/debug/device-file-explorer

READING DATA FROM FILE

```
public void readFile(View view) {
    String filename = "myfile.txt";
    StringBuilder stringBuilder = new StringBuilder();
    try {
    FileInputStream fis = openFileInput(filename);
    Scanner sc  = new Scanner(fis);
        String line = "";
        while (sc.hasNextLine()) {
            line = sc.nextLine();
            stringBuilder.append(line).append('\n');
            Log.d("Line" , line);

        }
        sc.close();
        String contents = stringBuilder.toString();
        Toast.makeText(this, contents,Toast.LENGTH_SHORT).show();
        // tv.setText(contents)

    } catch (IOException e) {
        // Error occurred when opening raw file for reading.
    }
```

EXTERNAL STORAGE

- External to your application's sandbox.
- Historically was mapped to externally mounted storage devices (i.e. SD card), but not necessarily true now

storage-related permissions:

Android defines the following storage-related permissions:

READ_EXTERNAL_STORAGE,
WRITE_EXTERNAL_STORAGE,
and MANAGE_EXTERNAL_STORAGE.

On earlier versions of Android, apps needed to declare the READ_EXTERNAL_STORAGE permission to access any file outside the app-specific directories on external storage. Also, apps needed to declare the WRITE_EXTERNAL_STORAGE permission to write to any file outside the app-specific directory.

More recent versions of Android rely more on a file's purpose than its location for determining an app's ability to access, and write to, a given file.

EXTERNAL STORAGE :- WRITING TEXT TO FILE

```
public void  writeToFile( ){

     FileOutputStream fos ;
     File path;
      String data = "Welcome to android ";
     path = getExternalFilesDir("MyFolder");
     File file =  new File(path,"myFile.txt");


     try {
         fos = new FileOutputStream(file);
         fos.write(data.getBytes());
         fos.close();
         Log.d("Files" , "done");
     }catch (Exception e) {
         Log.d("Files" , e.toString());
         e.printStackTrace();

     }
}
```

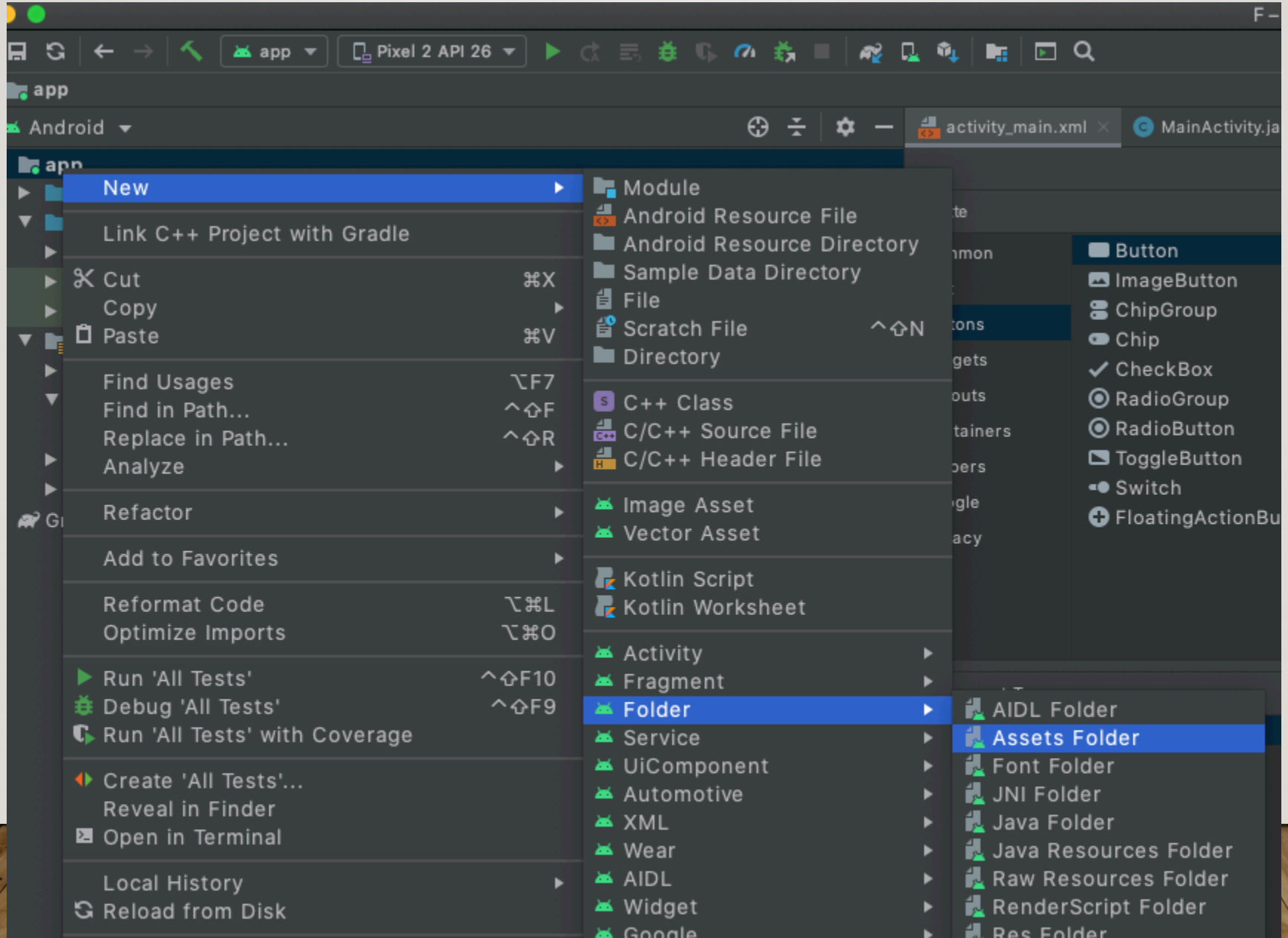
EXTERNAL STORAGE :- READ TEXT FROM FILE

```
public void  readToFile( ){

    FileInputStream fis ;
    File path = getExternalFilesDir("MyFolder");
    File file =  new File(path,"myFile.txt");
    StringBuilder stringBuilder = new StringBuilder();
    try {
        fis = new FileInputStream(file);
        Scanner sc  = new Scanner(fis);
        String line = "";
        while (sc.hasNextLine()) {
            line = sc.nextLine();
            stringBuilder.append(line).append('\n');
            Log.d("Line" , line);
        }
        sc.close();
        String contents = stringBuilder.toString();
        // tv.setText(contents)
        fis.close();
      }catch (Exception e) {
        Log.d("Files" , e.toString());
        e.printStackTrace();

    }
}
```

FILES BUNDLED WITH YOUR APPLICATION

- Files can be stored in the /res/raw folder to include these files with your Android distribution.
- These files are only visible to your app and will be deleted when the app is uninstalled
- These files are not writable
- Examples of information you might want to store there
  - Long textual information
  - An initial configuration file

CREATING ASSETS FOLDER



READING A FILE BUNDLED IN ASSETS

```
private void readData(){
    BufferedReader reader = null;
  try {
        reader = new BufferedReader(
                new InputStreamReader(getAssets().open("file.txt")));

     String line;
     while ((line = reader.readLine()) != null) {
           Log.d("data" , line );
     }
    } catch (IOException e) {
        Log.d("error" ,e.getMessage() );
  }
}
```

REFERENCES

- https://developer.android.com/training/data-storage
- https://developer.android.com/training/data-storage/app-specific