12. Lesson 17/04/23

Data Storage

Contents

- How to **store data** in Android
- **Internal Storage**

Storing Data

Android provides several **options for saving persistent app data**.

Possible solution include:

- **Internal Storage** —> Private data on device memory
- **External Storage** —> App-Specific files or Public data on device or External Storage
- **Shared Preferences** —> Private primitive data key-value pairs
- **SQLite Databases** —> Structured data in private databases

The best solution depends on the app specific needs.

- Data should be **private** to the app OP accessible to other apps/user?
- How much **space** the data requires? Complex structures needed?

Storing data beyond Android

- **Network Connection** —> On the web with your own server.
- **Cloud Backup** —> Back up app and user data in the cloud.
- **Firebase Real Time Database** —> Store and sync data with NoSQL cloud database across clients in real time.

Files - Internal and External Storage

Android File System

Android uses a **file system** that is similar to disk-based file systems on other platforms such as Linux.

All **Android devices** have **two file storage areas**:

- **Internal storage:** Private directories for just your app.
- **External storage:** Public directories.

The App can browse the directory structure.

Internal Storage

- Always available in the App.
- Uses device's file system.
- Only your app can access file.
- On app uninstall, system removes all app's files from internal storage.

External Storage

- Uses device's file system or physically external storage like SD card.
- Not always available , can be removed (SD cards).
- World-readable, so any app can read it.
- On uninstall, the system does not remove files.

When to use internal/external storage

Internal is best when:

- Want to be sure that neither the user not other apps can access your files.

External is best when:

- Don't require access restrictions.
- You want to share with other apps.
- You allow the user to access with a computer.

Internal Storage

Your app **always has permission to read and write files** in its internal storage directory.

- **Permanent** storage directory —> *getFilesDir()*
- **Temporary** storage directory —> *getCacheDir()*
  This is recommended for small, temporary files totaling less than 1MB.
  The system may delete temporary files if it runs low on memory.

Creating a file

To create a new file in one of these directories, ise the *File()* constructor, passing the File provided by one of the methods (*getFilesDir(), getCacheDir()*) that specifies your internal storage directory.

```
File file = new File(context.getFilesDir(), filename);
```

Then use standard java.io file operators or streams to interact with files.

Writing Text in an Internal File

- First get a file object.
  You will need the storage Path. For the internal storage use:

  ```
  File path = context.getFilesDir();
  ```

- Create your file object:

  ```
  File file = new File(path, "my_file.txt");
  ```

- Write a string to the file:

  ```
  FileOutputStream stream = new FileOutputStream(file);
  try {
    stream.write("text-to-write".getBytes());
  } finally {
    stream.close();
  }
  ```

Read Text from an Internal File

```
File file = new File(path, "my_file.txt");

int length = (int) file.length();
```

```java
byte[] bytes = new byte[length];

FileInputStream in = new FileInputStream(file);
try {
  in.read(bytes);
} finally {
  in.close();
}
String contents = new String(bytes);
```