Persistent Storage Mechanisms

Subtitles are for chumps

So, you've got an idea

Persistent Storage Options

- Shared Preferences
- Internal Storage (private documents)
- External Storage (public documents)
- SQLite Database
- Network Connection

Shared Preferences

- Private dictionary
- Key value pairs
- All keys are Strings
- Stores primitive types (int, double, String)

Shared Preferences

```
SharedPreferences prefs =
   context.getSharedPreferences("prefs",
   Context.MODE_PRIVATE);
String value = prefs.getString(“key”);

SharedPreferences.Editor editor = prefs.edit();
editor.putString(“key”, “value”);
editor.commit();
```

Internal Storage

- Private documents folder, similar to iOS.
- Deleted when app is deleted.
- Files are accessed through API calls.

```
File toWrite = openFileOutput(…);
File toRead = openFileInput(…);
```

Internal Cache

- Private folder meant for cache files.
- Can be emptied by the system or user if running out of storage.
- Not “cleaned up” automatically

```
File cacheDir = context.getCacheDir();
```

External Storage

- Public file storage.
- Different on every device. Some are built in and some are removable. Some have both.
- Not always mounted.


```
getExternalFilesDir(); // API 8+
getExternalStorageDirectory(); // API 7-
```

File

- Represents files and directories.
- Can be read/written using FileInputStream and FileOutputStream.

```
String FILENAME = "hello_file";
String string = "hello world!";
FileOutputStream fos = openFileOutput(FILENAME,
   Context.MODE_PRIVATE);
fos.write(string.getBytes());
fos.close();
```

SQLite Database

- C based relational database. Popular in embedded systems.
- Supported by many languages/platforms.
- Single file, extremely portable.
- Runs in application process (is not a separate service).
- Implements *most* of the SQL standard.

SQLite Database

- Stored in internal storage. Accessed by subclassing SQLiteHelper class.
- Queries return a Cursor object (collection of rows).
- Must query using SQL. No ORM (object relational mapping).
- Third party ORMs do exist (greenDAO or ORMLite)

Network Connection

- Unlimited options.
- Requires serialization like json or xml.
- Often a database wrapped in a web service.
- Introduces connectivity and syncing issues.
- Need a fallback if connection is lost.

Which one do I use?

Shared Preferences

- Simple, fast, secure.
- Dictionary paradigm.
- Can only store primitives.

Internal Storage

- Private files. Secure.
- Deleted when app is deleted.

External Storage

- Public files. Allows sharing, or for user to bring their own files.
- Not deleted by system ever, including factory reset (in most cases).
- Can be altered by other programs.

Internal/External Cache

- System allowed to delete if needed.
- Otherwise same as internal/external.

SQLite Database

- Supports more complex data models.
- Allows for dynamic querying, filtering, sorting.
- Not as sophisticated as other databases.

Network Connection

- Can be anything you need.
- Allows integration with other platforms.
- Affected by connectivity issues.