Contents

1 Logical Data Storage on Android

File-based abstractions

- Shared Preferences: Simple key value pairs
- File-based storage:
  - Internal Data Storage: Soldered RAM. Internal APK resources, temporary files
  - External Data Storage: SD Card. Large media files
- SQLite Database: Structured data, small binary files

Network sync adapters:

- Shared contact lists, backups
- Synchronising local and remote files

2 Internal File Storage

Internal Data storage is private to the

- Other apps (and the user) cannot access it: Kernel enforced user permissions
- Removed on uninstall: so sensitive data cannot be obtained
- Data is stored in files:
  - /data/data/com.example.martinstorage/files/
  - openRawResource: Can be used to read our own packaged resources

Android provides a standard place to store (small) cache files

- FingerPainterView
- /data/data/com.example.martinstorage/cache
- getCacheDir() to get a File for the directory
- We should still manage the files ourselves
  - May be deleted when internal storage becomes full / contested
  - Will be deleted when the application is uninstalled
  - A well behaved application will delete them when no longer in use. Recommended to use less than 1MB

2.1 Shared Preferences

- In internal Storage , stored on a per-application basis
- I.e. all components in an application may access the same Shared Preferences
- But should not be used for data transfer (instead of Intents, Binder etc)
- Primitive data in key-value pairs. Primitives: strings, integers etc. Not Bundles
- Can have multiple preference files per application

3 External storage

Every Android device provides externally-accessible storage, e.g. SD card

- Even those phones without an SD card. They have a logical representation of external storage
- Single storage device partitioned into internal / external. This is because all Android devices must conform to the Android API in order to be Android devices
- Private application files. Internal Storage on the External partition (Else permissions cant be overriden with other device)
- Public general files
  - World readable
  - Other applications can read and modify these files
  - Each user has their own virtual SD card

Can be mounted externally (and/or disconnected). Before accessing files need to check the state of external storage. It may not be there, or mounted by something else

3.1 Using external storage

- Should check state with `Environment.getExternalStorageState()`. It is a separate file system, potentially removable. `Environment.MEDIA_MOUNTED`, `Environment.MEDIA_MOUNTED_READ_ONLY`.
- Use `getExternalStoragePublicDirectory(String type)` to obtain a File for the directory
  - Pass a type to obtain a sub-directory for that type
  - Used to enable the Media scanner to categorize material
  - Use File object returned to createNewFile()
  - Scoped Directory access
  - With each new release, developers have been provided with new and updated APIs to work with
- `getExternalFilesDir()`
- Provides private external storage
- /sdcard/Android/data/com.example.pszmdf.fingerpainter/

4 Android Databases

- Often the data we are storing is logically structured
  - Need to query it based on that structure
  - Could store this in a file and write our own routines to access it
  - Obb virtual file system
  - StorageManager (wrapper for MountService system service)
  - Opaque Binary Blobs
  - Normally, wed use a database to store it
  - E.g. An address book, music library
- Android provides local database support
  - Complete with the ability to run full SQL queries. Each apps databases are local to it
  - Database.db stored in Internal Storage (SQLite)

4.1 Android and SQLite

- Wrapped up in two main classes
  - Database represented by SQLiteDatabase
  - Lets us run SQL queries on the database
  - SQLiteOpenHelper: Supports the Application lifecycle. `onCreate()`. Create the database the first time the application is opened. `onUpgrade(int oldVersion, int newVersion`. Changing the version number affords drop and recreation of the database
- Create an instance of our SQLiteOpenHelper subclass
- Obtain reference to SQLiteDatabase using: `getReadableDatabase(), getWriteableDatabase()` Both return the same object, unless memory is low and can only open the DB readonly

4.2 Cursors

- Provides random access to results of a query
- Enable us to step over all the rows returned by a query `moveToFirst(), moveToNext()`
- `getString(columnIndex), getInt(columnIndex)`. Where column index is index of projection result
- Has a `close()` method to close the query when finished. Shouldnt wait for it to be garbage collected
- IPC implications. Can we pass a cursor to another process? (component number 3)

4.3 Data Driven Views

Connect a cursor to a CursorAdapter and ListView

- SimpleCursorAdapter(...Cursor...)
- Map the projection to a View layout for a single item, populate a list of views
- Link resource IDs to projection columns. Requires each row to have an `_id` field
- Automatically generates a View for each row of the cursor

RecyclerView

- Optimised, flexible version of the above
- Only creates Views for visible data
- As the user scrolls, or more data is added
- Create new Views as appropriate. Re-bind old Views to new data
- Programmatically describe how to create View from data

4.4 CursorLoader

A query may last some time. Database may be large, may be in a different process. We Dont want to block the main UI thread

CursorLoader: Populates views asynchronously, Auto Updating, Monitors for notification that content has changed

4.5 Database Abstraction

Good software architecture. Separation of data model from presentation / views. Abstraction of database architecture:

- Easier to update storage code
- Expose column indices as static class variables: `c.getInt(0) -> c.getInt(DBHelper.NAME)`
- Helper methods keep database internals from leaking into other classes. Return a Collection of results rather than a Cursor. Use Cursor internally in DBHelper class
- SQL injection: Sanitise user input
- Important when thinking about the logical next step exposing data to other applications via a Component. Appropriate database schema

5 Sharing data

5.1 Content Provider

Access to data is restricted to the app that owns it

- Database is usually located in internal app-specific storage (Inaccessible by other applications)
- If we want other apps to access our data, or we want to access other apps data, or we want to be notified when data has changed

Provide or make use of a Content Provider

- Application component number 3
- Exposes data / content to other applications in a structured manner
- Fundamentally IPC via Binder (again) + ashmem with a well defined (database-like) interface

Reference section

placeholder