The Basis of Data

Steven R. Bagley

So far…

- How to create a UI
  - View defined in XML
  - Java-based `Activity` as the Controller
- Services
  - Long running processes
- `Intent`s used to send messages between things asynchronously

Data

- Apps generally store, process and display data
- Normally stored in files
- Android doesn't have a traditional file system…
- Why not?

Security mainly
Put also enables the app to be cleanly removed
Storage at a premium on the device…

Data storage on Android

- Shared Preferences
- File-based storage
  - Internal Data Storage
  - External Data Storage
- SQLite Database
- Network

File access

- Android uses the standard Java I/O classes to access files
  - `File`, `InputStream`, `OutputStream`, etc…
- However, obtained in a different way…

Show's the power of OO abstraction same interface but different access mechanism

Internal Data Storage

- Internal Data storage is private to the app
- Other apps (and the user) cannot access it
- Removed on uninstall
- Data is stored in Files

(Access Without permission)

Accessing Internal Storage

- Two methods are used to access files on internal storage
- `Context.openFileOutput(String name, int mode)`
  Returns a `FileOutputStream`
- `Context.openFileInput(String name)`
  Returns a `FileInputStream`
- Don't forget to catch `IOExceptions`

Go give a demo of how we can access this…

Cache Files

- Android provides a standard place to store (small) cache files
- Use `getCacheDir()` to get a `File` for the directory
- Still need to manage the files yourself

External Data Storage

- Every Android device provides externally-accessible storage, e.g. SD card
- Can be mounted externally (and/or disconnected)
- Before accessing files you need to check the state of external storage
- Files deleted when app uninstalled…

External Data Storage

- Check state with
  `Environment.getExternalStorageState()`
- Returns a `String` containing the details
- Compare with the constants:
  `Environment.MEDIA_MOUNTED`
  `Environment.MEDIA_MOUNTED_READ_ONLY`
- To check the state…

Accessing External Storage

- Use `Context.getExternalFilesDir(String type)` to obtain a `File` for the directory
- If you pass a `type` (it can be `null`) then returns a sub-directory of appropriate type
  - Used to enable the Media scanner to categorize material
  - Use `File` object returned to `createNewFile()`

DIRECTORY_MUSIC
DIRECTORY_RINGTONES
etc…
Go modify demo

Structured Data

- Often the data we are storing is structured
- And we want to query it based on that structure
- Could store this in a file and write our own routines to access it
- Normally, we'd use a database to store it

E.g Address Book, music library

Android Databases

- Android comes with local database
- Complete with the ability to run SQL queries
- Each app's databases are local to it…

Android and SQLite

- Uses SQLite
- Public Domain software library
- “A software library that implements a self-contained, serverless, zero-configuration, transactional SQL database engine.”
- Most widely deployed software engine on the planet…

Android and SQLite

- Wrapped up in two main classes…
- Database represented by `SQLiteDatabase`
  - Lets us run SQL queries on the database
- Also provides `SQLiteOpenHelper` to help create the database

Creating a DB

- `SQLiteOpenHelper` manages database creation and upgrades between versions
- Create a subclass of it
- Override `onCreate` to provide the code to create the database
- Using SQL `CREATE TABLE`
- Handled automatically

See example and documentation for how to call superclass

Accessing the DB

- Create an instance of our `SQLiteOpenHelper` subclass
- Obtain reference to `SQLiteDatabase` using:
  `getReadableDatabase()`
  `getWriteableDatabase()`
- Return the same object, unless memory is low and can only open the DB readonly

Querying the DB

- `SQLiteDatabase` has many methods
- `void execSQL()`
  used to run SQL queries that don't return anything
- More useful are `query()` and `rawQuery()`
- These return a `Cursor` object that can be used to access the data

Querying the DB

- `Cursor rawQuery(String sql, String[] selectionArgs)` processes SQL query in `sql`
- SQL has to be parsed so there is also `query()` where the SQL is exploded into separate strings

```
Cursor query(String table, String[] columns,
             String selection, String[] selectionArgs,
             String groupBy, String having,
             String orderBy)
```

`Cursor` object

- Pretty self explanatory object
- Enables you to step over all the rows returned by a query
- Has a `close()` method to close the query when you are finished (don't wait for it to be garbage collected)

Methods	Description
`int getColumnCount()`	Return the total number of columns
`int getColumnIndex(String name)`	Return the index for the column named `name` (or `-1`)
`String getColumnName(int columnIndex)`	Return the name of the column
`int getCount()`	Return the number of rows

Methods	Description
`byte[] getBlob(int columnIndex)`	Return value of the column as a byte array
`float getFloat(int columnIndex)`	Return value of the column as a float
`int getInt(int index)`	Return the value of column as an int
`int getString(int index)`	Return the value of column as a String
…	And so on, you name it there is a method for every type

Returns data from the current row…

Methods	Description
`boolean isFirst()` `boolean isLast()`	Is this the first/last row in the result?
`boolean move(int offset)`	Move the cursor relative to the current position, offset places
`boolean moveToFirst()` `boolean moveToLast()`	Guess…
`boolean moveToNext()`	moves to the next row, false if past the end of the result

Returns data from the current row…

Content Providers

- Access to data is restricted to the app that owns it
- If we want other apps to access our data, or we want to access other apps data
- Then we need to provide or make use of a `ContentProvider`...