UNIVERSITY of WASHINGTON

Android 210 - Lecture 6
Databases and Content Provider

Margaret Maynard-Reid
February 23, 2015

Topics

- Homework 3 solution
- Databases - create your own db
- Content Provider - access content
- Homework 4 requirements
- Sample Code:
  - SampleDatabase
  - SampleAcessContent

Android Stories

- Android 5.1 Review - What's New?
- Google concerned with Xiaomi’s ability to compete with apps and services
- Google and Apple Fight for the Car Dashboard

Homework 3 Solution

- Walk through Homework 3 solution

Review from last week

- What is stored on internal storage?
- What is stored on external storage?
- What are SharedPreferences?
- Can you access the files on the internal storage via DDMS/File Explorer?
- Are files secured on the external storage?

Android Storage Options

- Files
- SharedPreferences
- Databases

Android Databases

- Only one option provided by the platform
  - SQLite
- Others exist but you mileage may vary
  - CouchDB
- If you are really storing massive amounts of data, look to the cloud

SQLite

- A lightweight library used by Android to provide a relational database access via a weakly typed dynamic SQL syntax

http://www.sqlite.org/

SQLite Uses

- Local Private Database Storage
- Content Provider Backing Storage

SQLite Types

- NULL – A NULL value type
- INTEGER – A signed integer stored in 1, 2, 3, 4, 6, 8 bytes
- REAL – A floating point value stored as an 8-byte floating point number
- TEXT – A text string stored in UTF-8, UTF-16 BE or UTF-16 LE
- BLOB – A binary blob of data

Basic SQL

- SELECT – Select data from a table
- UPDATE – Update data from a table
- DELETE – Deletes data from a table
- INSERT – Inserts data into a table in a database
- CREATE {DATABASE | TABLE} – Creates a new element
- ALTER {DATABASE | TABLE} – Updates an element

*** https://www.sqlite.org/lang.html

Basic SQL Select

- SELECT {projection}
- FROM {table}
- WHERE {conditions}
- ORDERBY {argument}

Basic SQL

- SELECT * FROM books
- SELECT * FROM books WHERE _id=5
- SELECT _id, name FROM books ORDERBY name

SQLite Tools

- Sqlite3
- SQLite Browser – http://sqlitebrowser.org/
- SQLite Administrator – http://sqliteadmin.orbmu2k.de/
- Navicat
- Apps on Google Play
  - Access Database on SDCARD
  - Access via SuperUser on Rooted phone

SQLite Db Location

Where is the db located?

- Located on internal storage, associated with your app
- Open DDMS/File Explorer,
- Find db file under

  data/data/<package name>/databases

Sqlite3

- Open command line
- *adb shell*
- *cd data/data/<package name>/databases*
- ***sqlite3 dbname*** ← *invoke sqlite3 on database*
- ***.schema*** ← print the SQL CREATE statement for an existing table
- ***.tables*** ← list all the tables in db
- ***.exit*** ← exit sqlite3

Command line shell for SQLite - http://www.sqlite.org/cli.html

SQLite Browser

- Download SQLite Browser
- Get the db file from emulator

  adb pull <remote> <local>

- Open SQLite Browser and modify db
- Copy the updated db file back to emulator

  adb push <local> <remote>

Hands-on

- Run SampleDatabase
- Locate the db on Emulator
- Use sqlite3
- Use SQLite Browser

Break

Planning

- Decide if you need a database
  - Data relationships, size, filtering, speed, convenience, preference, etc.
- Build your Schema
  - Decide what fields are needed
  - Keep CRUD in Mind
    - Create
    - Read
    - Update
    - Delete
- Invest in a good SQLite book & understand basic SQL

Create Schema Class

Create static classes to maintain and consolidate your schema

– You will find yourself using constants to define for tables, queries, ids, plus much more

SQLiteOpenHelper

Helper class to implement database access

- Provides access to a SQLiteDatabase
- Helps handle transactions
- Helps handle versioning
  - Upgrades
  - Downgrades

Extend SQLiteOpenHelper class to make your own

Tips on SQLiteOpenHelper

- Thin Database Helper
  - Just deals with the administration of the database
  - Logic and queries are stored externally
  - More Reusable
- Thick Database Helper
  - All logic and administration is done from the Helper
  - Less Reusable

SQLiteOpenHelper

```
DatabaseHelper mHelper = DatabaseHelper(this);
SQLiteDatabase db = mHelper.getReadableDatabase();
Cursor c = db.query(
        TABLE_NAME,
        DATABASE.TABLE.PROJECTION, // columns
        null,  // selection
        null, // selectionArgs
        null, // groupBy
        null, // having
        null); // orderBy
```

SQLiteOpenHelper

```
DatabaseHelper mHelper = DatabaseHelper(this)

Cursor c = mHelper.getAllBooks();
```

Creating a SQLite DB

Generally two ways to create a SQLite DB

1. android.database.sqlite – SQLiteDatabase (Memory)
    - create()
    - openDatabase()
    - openOrCreateDatabase()
2. SQLiteOpenHelper (Internal Storage)
    - getReadableDatabase()
    - getWritableDatabase()

SQLiteDatabase CRUD

- SQLiteDatabase.insert() : Create
- SQLiteDatabase.query() : Read
- SQLiteDatabase.update() : Update
- SQLiteDatabase.delete() : Delete

Querying

SQLiteDatabase

- ***execSQL(String sql)***

Execute a statement that doesn't return data, or that you don't care about return data

Querying

SQLiteDatabase - query()

- ***tables*** – The tables to run the query against
- ***columns*** – Columns to include in rows returned
- ***where*** – Filter the rows returned
- ***groupBy*** – Filter that groups rows returned
- ***having*** – Filter row groups to include in the rows returned
- ***orderBy*** – Orders the rows returned
- ***limit*** – Limits the number of rows returned

Querying

SQLiteQueryBuilder

- Useful for Strict checks, Building Unions, Convenience
- buildQueryString()
  - distinct – Set if rows should be unique
  - tables – The tables to run the query against
  - columns – Columns to include in rows returned
  - where – Filter the rows returned
  - groupBy – Filter that groups rows returned
  - having – Filter row groups to include in the rows returned
  - orderBy – Orders the rows returned
  - limit – Limits the number of rows returned

Projection

What columns of data you want returned

- Just an array of Strings
- Sometimes you will see it abstracted to reduce errors but the fact remains: it is an array of column titles you want in your result set

Cursor - what is it?

- A pointer to a current row in a result set
- Starts “before” the first entry in the set
- Need to call moveToFirst() or moveToNext()

Iterating the Cursor

Long Form

```
Cursor c = SQLiteDatabase.query(...)
c.moveToFirst();
while (c.isAfterLast() == false) {
}
```

Iterating the Cursor

Short Form

```
Cursor c = SQLiteDatabase.query(...)
while (c.moveToNext()) {
}
```

Hands-on

- Walk through **SampleDatabase**
- Look at the database created via **DDMS File Explorer**

Break

Content Providers

- Part of an Android application
  - Defined in the application manifest as <provider>
  - Maps to a ContentProvider class in your project
- Provide managed and secured access to data
- They encapsulate the data with a consistent, standardized URI-based access
- Useful as a cross-process interface for data-sharing amongst running processes

System Content Providers

Android applications also provide content

- People
- Calendar
- Gallery
- Etc.

Available System Content

- Browser
  - Bookmarks
- Calendar
  - Attendees
  - Events
  - Reminders
- CallLog
- ContactsContract
  - Name
  - Phones
  - Photos etc.

- MediaStore
  - Audio
    - Albums
    - Artists
    - Playlists
  - Images
  - Video
- Settings
- SyncState
- UserDictionary
- VoicemailContract

http://developer.android.com/reference/android/provider/package-summary.html

Access a Content Provider

- Get permission to the Content Provider

```
<uses-permission android:name="android.permission.READ_CONTACTS"/>
```

- Then access it via
  - ContentResolver or
  - CursorLoader

Access a Content Provider

android.content.ContentResolver (API 1)

- uri - The provider table used in resolution
- projection - The columns that should be returned
- selection - Criteria for selecting rows
- selectionArgs - Replace ? arguments from Selection
- sortOrder - The order of the rows returned

Access a Content Provider

android.content.CursorLoader (API 11)

- context - the current context
- uri - The provider table used in resolution
- projection - The columns that should be returned
- selection - Criteria for selecting rows
- selectionArgs - Replace ? arguments from Selection
- sortOrder - The order of the rows returned

What is a CursorLoader

An Android Loader built from AsyncTaskLoader

- Targets a ContentProvider
- Loads data asynchronously
- Handles the Cursor lifecycle

Content URI Format

- General Form – **content://authority/path/id**
- Authority Examples – android.provider

ContactsContract AUTHORITY	**"com.android.contacts"**
MediaStore AUTHORITY	**"media"**
CalendarContract AUTHORITY	**"com.android.calendar"**

Content URI Examples

- content://media/external/images/media/9134
- content://com.android.contacts/data/123
- content://com.mybooks.access/books/914
- content://com.mybooks.
  access/books/authors/42

Access image from MediaStore

- Use ContentResolver

ContentResolver.query(MediaStore.Images.Media.EXTERNAL_CONTENT_URI, projection, selection, selectionArgs, sortOrder)

- Use CursorLoader (preferred)

CursorLoader(context, MediaStore.Images.Media.EXTERNAL_CONTENT_URI,

projection, selection, selectionArgs, sortOrder)

Access Content

At a minimum, you will need

- A Content URI
- A Projection
- A ContentResolver or CursorLoader
- A Cursor
- Something to do with the data you retrieved

Accessing Content

Sometimes to get what you need involves a couple of lookups. In a Contact example:

- Get their Display Name
- Get their Email Addresses
- Get their Status
- ...

Hands-on

- Walk through SampleAccessContent

Homework 4

- Go over requirements
- Due on March 9, 2015 6PM
- No late homework accepted