ADB - Android Debugger

- location ~/Android/Sdk/platform-tools/adb, add dir to PATH
- adb shell
  - su (on emulators, not on devices)
- look for files (find)
- run sqlite
SQLiteOpenHelper -- A class to help with SQLite databases:
- onCreate(SQLiteDatabase) -- initial creation of the database
- onUpgrade(SQLiteDatabase,int oldVersion,int newVersion)
- onDowngrade(SQLiteDatabase, int oldVersion, int newVersion)
- onOpen(SQLiteDatabase)
- getReadableDatabase(), returns a SQLiteDatabase
- getWritableDatabase(), returns a SQLiteDatabase
- SQLiteOpenHelper(Context String name, SQLiteDatabase.CursorFactory, int version)
  - null for CursorFactory gets default one

public class TimeDB extends SQLiteOpenHelper {

  public TimeDB(Context context) {  super(context, "TimeDB", null, 1);  }

  @Override  public void onCreate(SQLiteDatabase db) {
    // Only one table for version 1
    db.execSQL("create table sessions (_id integer primary key, ....);" );
  }

  @Override public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {
    // Similar SQL to modify/upgrade
  }
}
Simple Usage of "TimeDB"

Access:  TimeDB db = new TimeDB(this);  // this is an activity
Use:  db.getWritableDatabase()
  .execSQL("SQL statement")
  .rawQuery(...)  -- returns a Cursor
  .insert()
  .delete()
  .query() (many ....)

Sample code:

Cursor dbCursor = db.getReadableDatabase().
 .rawQuery("select _id, ascTime, duration, sent from sessions order by _id", null);
listAdapt.swapCursor(dbCursor).close();
listAdapt.notifyDataSetChanged();

A lot more in the SQLiteDatabase Class.

Cursor needs at least one call to run.
Cursor sesList = db.getReadableDatabase().rawQuery(
    "select _id, ascTime, duration, pauses, pausetime "
    + "from sessions" + (all ? "" : " where sent==0"), null);
String body = "Data to Report \n";

sesList.moveToFirst();
while (!sesList.isAfterLast()) {
    body += sesList.getString(1) +": worked for " + sesList.getString(2) +"\n";
    int pauses = sesList.getInt(3);
    if (pauses > 0)
        body += " " + pauses + " pause" + (pauses > 1 ? "s" : "")
            + " for " + sesList.getString(4) +"\n";
    sesList.moveToNext();
}
sesList.close();
TableLayout -- similar to tables in HTML
- Layout for rows and columns on the screen
- Rows are each a TableRow
- Number of columns is the row with the most elements
- A row single cell can span multiple columns

Database
- Original project -- Use TableLayout, SQL database
  - Note when onCreate() is called in the handler
  - File is with the app

- Not shown -- cursor adaptor
  - DB as data source for a RecyclerView, ListView, or other such view
  - Database handler needs to stay open longer
  - Build views from cursor row data
Content Providers

- Application to provide data to other apps ... e.g. Contacts
- Typically implemented using a DB under it (SQLite db on Android)
- `android.content.ContentProvider` class
  - `onCreate()` -- Typical start, often has a SQLiteDB as argument
  - `query(URI, String[], String, String[], String)`
  - `insert(URI, ContentValues)`
  - `update(URI, ContentValues, String, String[])`
  - `delete(URI, String, String[])`
  - `getType()` -- MIME type of the store data
- Content URI -- the way to identify a content provider
  - Also, identify data wanted from a content provider
    - `package_name`, `package_name/data`, `package_name/data/key`
  - `UriMatcher` class -- help extract
- Content Resolver -- access to a Content Provider
  - `getContentResolver()`, has the methods `query()`, `insert()`, ...
  - `ContentValues` -- class to send the Content Providers data via methods
- Manifest file needs a `<provider>` tag
  - `android:authority` -- URI of provider, `android:name` -- name of the class
Database example -- turned into a content provider
  (Appears to me to be a poor example ...)

Other methods of interest
  applyBatch(ArrayList<ContentProviderOperation>)
  bulkInsert(URI, ContentValues[])
  shutdown()
  and several others

Contacts -- standard content provider on the platform
  Not in our book
  URI: com.android.contacts