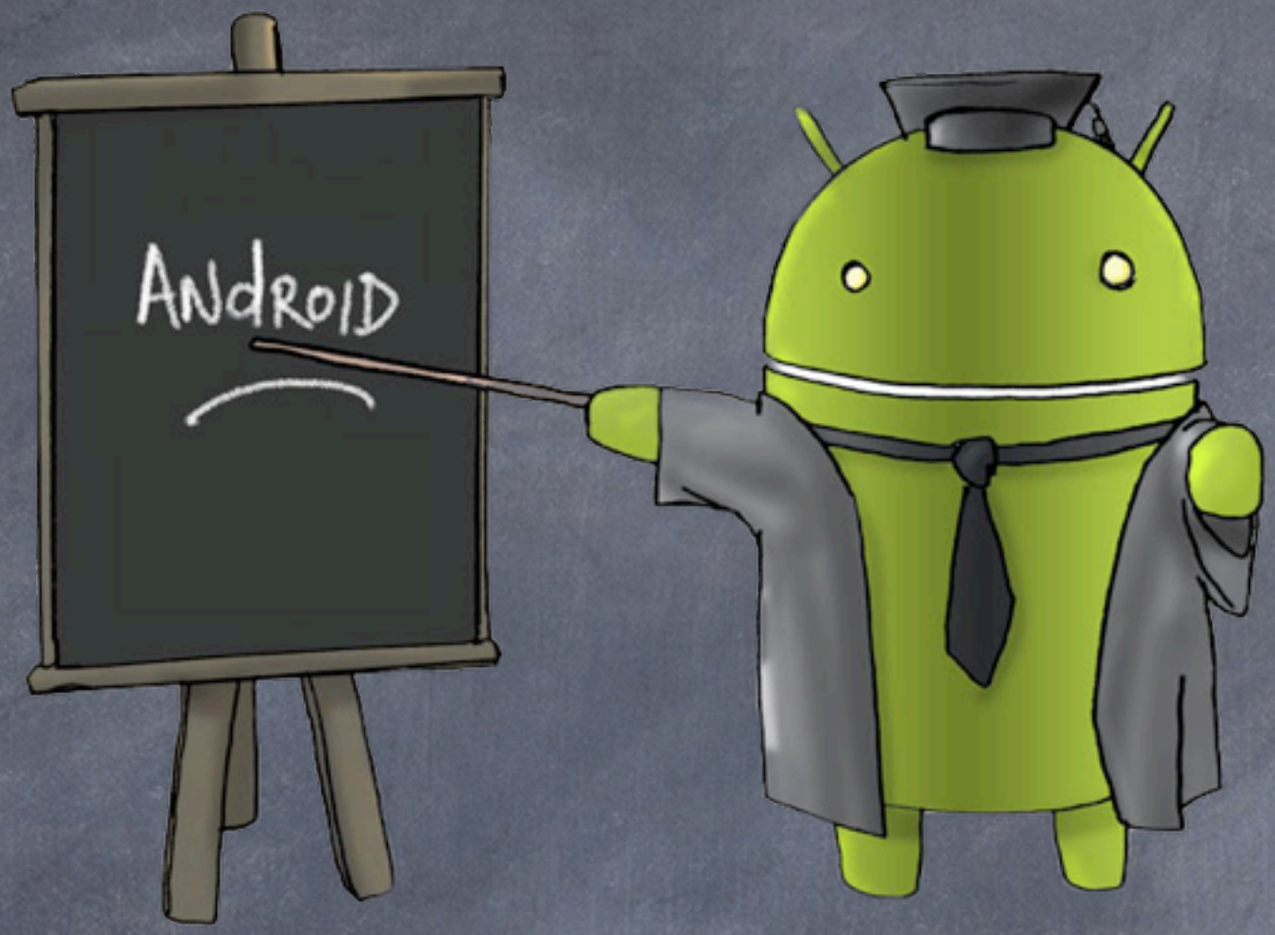


Using SQLite database in Android Apps

Presented @ Cape Town Android Meetup

Joseph Kandi
Peruzal (PTY) LTD
Email: joseph@peruzal.co.za
Web: http://www.peruzal.co.za
Facebook: http://www.facebook.com/peruzal

July 13 2013

Peruzal

Take Aways

- Add persistence to an Android App
- Use custom helper classes
- Perform CRUD(Create, Read, Update, Delete)
- Use ADB shell with sqlite3
- Use SimpleCursorAdapter
- Bind Data to a ListView

Adding A Database

- Create a Class that extends SQLiteOpenHelper
- Override the Constructor
- Override the onCreate and onUpgrade

Create the Custom Class

```java
package za.peruzal.contentprovider;

import android.database.sqlite.SQLiteDatabase;
import android.database.sqlite.SQLiteOpenHelper;

public class DatabaseHelper extends SQLiteOpenHelper {

    @Override
    public void onCreate(SQLiteDatabase arg0) {
        // TODO Auto-generated method stub

    }

    @Override
    public void onUpgrade(SQLiteDatabase arg0, int arg1, int arg2) {
        // TODO Auto-generated method stub

    }

}
```

Custom class, notice there is an error, we need a constructor

You need to override the onCreate and onUpgrade

Add A Constructor

```java
package za.peruzal.contentprovider;

import android.content.Context;
import android.database.sqlite.SQLiteDatabase;
import android.database.sqlite.SQLiteDatabase.CursorFactory;
import android.database.sqlite.SQLiteOpenHelper;

public class DatabaseHelper extends SQLiteOpenHelper {

    public DatabaseHelper(Context context, String name, CursorFactory factory,
            int version) {
        super(context, name, factory, version);
        // TODO Auto-generated constructor stub
    }

    @Override
    public void onCreate(SQLiteDatabase arg0) {
        // TODO Auto-generated method stub

    }

    @Override
    public void onUpgrade(SQLiteDatabase arg0, int arg1, int arg2) {
        // TODO Auto-generated method stub

    }

}
```

Added the constructor with Eclipse's help

Define Fields

```
package za.peruzal.contentprovider;

import android.content.Context;
import android.database.sqlite.SQLiteDatabase;
import android.database.sqlite.SQLiteDatabase.CursorFactory;
import android.database.sqlite.SQLiteOpenHelper;

public class DatabaseHelper extends SQLiteOpenHelper {

    public static final String DATABASE_NAME = "ContactsDB";
    public static final String TABLE_NAME = "contacts";
    public static final String COLUMN_KEY_ROWID = "_id";
    public static final String COLUMN_NAME = "name";
    public static final String COLUMN_EMAIL = "email";
    static final int DATABASE_VERSION = 1;
    static final String DATABASE_CREATE_SQL =
            String.format("CREATE TABLE %s(%s integer primary key autoincrement, " +
            "%s text, " +
            "%s text)", TABLE_NAME, COLUMN_KEY_ROWID, COLUMN_NAME, COLUMN_EMAIL);


    public DatabaseHelper(Context context, String name, CursorFactory factory,
            int version) {
        super(context, name, factory, version);

    }

    @Override
    public void onCreate(SQLiteDatabase arg0) {
        // TODO Auto-generated method stub

    }

    @Override
    public void onUpgrade(SQLiteDatabase arg0, int arg1, int arg2) {
        // TODO Auto-generated method stub

    }

}
```

Database name, and column name for the table

Database version, very important, used when upgrading

Statement to create the database

Modify Constructor

```java
package za.peruzal.contentprovider;

import android.content.Context;
import android.database.sqlite.SQLiteDatabase;
import android.database.sqlite.SQLiteDatabase.CursorFactory;
import android.database.sqlite.SQLiteOpenHelper;

public class DatabaseHelper extends SQLiteOpenHelper {

    public static final String DATABASE_NAME = "ContactsDB";
    public static final String TABLE_NAME = "contacts";
    public static final String COLUMN_KEY_ROWID = "_id";
    public static final String COLUMN_NAME = "name";
    public static final String COLUMN_EMAIL = "email";
    static final int DATABASE_VERSION = 1;
    static final String DATABASE_CREATE_SQL =
            String.format("CREATE TABLE %s(%s integer primary key autoincrement, " +
            "%s text, " +
            "%s text)", TABLE_NAME, COLUMN_KEY_ROWID, COLUMN_NAME, COLUMN_EMAIL);


    public DatabaseHelper(Context context) {
        super(context, DATABASE_NAME, null, DATABASE_VERSION);

    }

    @Override
    public void onCreate(SQLiteDatabase arg0) {
        // TODO Auto-generated method stub

    }

    @Override
    public void onUpgrade(SQLiteDatabase arg0, int arg1, int arg2) {
        // TODO Auto-generated method stub

    }

}
```

Use database name , context and version for the constructor

Code onCreate

```
import android.content.Context;
import android.database.sqlite.SQLiteDatabase;
import android.database.sqlite.SQLiteException;
import android.database.sqlite.SQLiteDatabase.CursorFactory;
import android.database.sqlite.SQLiteOpenHelper;
import android.util.Log;

public class DatabaseHelper extends SQLiteOpenHelper {
    static final String TAG = "DBHelper";
    public static final String DATABASE_NAME = "ContactsDB";
    public static final String TABLE_NAME = "contacts";
    public static final String COLUMN_KEY_ROWID = "_id";
    public static final String COLUMN_NAME = "name";
    public static final String COLUMN_EMAIL = "email";
    static final int DATABASE_VERSION = 1;
    static final String DATABASE_CREATE_SQL =
            String.format("CREATE TABLE %s(%s integer primary key autoincrement, " +
            "%s text, " +
            "%s text)", TABLE_NAME, COLUMN_KEY_ROWID, COLUMN_NAME, COLUMN_EMAIL);


    public DatabaseHelper(Context context) {
        super(context, DATABASE_NAME, null, DATABASE_VERSION);

    }

    @Override
    public void onCreate(SQLiteDatabase db) {
        Log.d(TAG, "Oncreate " + DATABASE_CREATE_SQL);
        try {
            db.execSQL(DATABASE_CREATE_SQL);
        } catch (SQLiteException e) {
            e.printStackTrace();
        }
    }

    @Override
    public void onUpgrade(SQLiteDatabase arg0, int arg1, int arg2) {
        // TODO Auto-generated method stub

    }
```

Create the table, this runs once

Code onUpgrade

```
static final String TAG = "DBHelper";
public static final String DATABASE_NAME = "ContactsDB";
public static final String TABLE_NAME = "contacts";
public static final String COLUMN_KEY_ROWID = "_id";
public static final String COLUMN_NAME = "name";
public static final String COLUMN_EMAIL = "email";
static final int DATABASE_VERSION = 1;
static final String DATABASE_CREATE_SQL =
        String.format("CREATE TABLE %s(%s integer primary key autoincrement, " +
        "%s text, " +
        "%s text)", TABLE_NAME, COLUMN_KEY_ROWID, COLUMN_NAME, COLUMN_EMAIL);


public DatabaseHelper(Context context) {
    super(context, DATABASE_NAME, null, DATABASE_VERSION);

}

@Override
public void onCreate(SQLiteDatabase db) {
    Log.d(TAG, "Oncreate " + DATABASE_CREATE_SQL);
    try {
        db.execSQL(DATABASE_CREATE_SQL);
    } catch (SQLiteException e) {
        e.printStackTrace();
    }
}


@Override
public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {
    //Drop the table and recreate it again
    Log.w(TAG, "Upgrading from " + oldVersion + " to version " + newVersion);
    String sql = String.format("DROP TABLEIF EXISTS %s", TABLE_NAME);
    try {
        db.execSQL(sql);
    } catch (SQLiteException e) {
        e.printStackTrace();
    }
    onCreate(db);
}
```

NB, you will lose data, need better implementation to save user data

That's it, we are done with that part

CRUD

CRUD

Implementing CRUD

- Instantiate the Database Helper Class
- Get a Writable / Readable Database
- Perform CRUD operations

Instantiate Database Helper Class

```
import java.util.Locale;

public class MainActivity extends ListActivity {
    SQLiteDatabase db;

    @Override
    protected void onPause() {
        super.onPause();
        db.close();
    }

    static final String TAG = "MainActivity";
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        DBHelper dbHelper = new DBHelper(this);

         db =  dbHelper.getWritableDatabase();
```

Declare an instance of the SQLiteDatabase

Instantiate the helper class and get a writable database

Insert into the Database

- Use the ContentValues class
- Use the various put* methods
- Use the db.insert statement

Perform Insert Operation

```
static final String TAG = "MainActivity";
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);

    DBHelper dbHelper = new DBHelper(this);

     db =  dbHelper.getWritableDatabase();

ContentValues values = new ContentValues();
values.put(DBHelper.COLUMN_NAME, "Joseph");
values.put(DBHelper.COLUMN_EMAIL, "joseph@peruzal.co.za");

//Insert into the database
long id = db.insert(DBHelper.TABLE_NAME, null, values);

if (id != 1) {
    Log.w(TAG, "Unable to insert into the database");
}
```

Instantiate an object type ContentValues, and add the values to insert into the database

Perform the insert operation. Returns 1 if successful otherwise an arbitrary number

Let's Read

Run query against database and return a Cursor

Columns we are interested in, food idea to limit to only what's needed, also define sort ordering

```
//Read from the database
String[] columns = new String[] { DBHelper.COLUMN_KEY_ROWID, DBHelper.COLUMN_NAME, DBHelper.COLUMN_EMAIL};
String sortOrder = String.format("%s ASC", DBHelper.COLUMN_NAME);
Cursor cursor =  db.query(DBHelper.TABLE_NAME, columns, null, null, null, null, sortOrder);

//Move the cursor to the first position
if (cursor.moveToFirst()) {
    do {
        Log.d(TAG,String.format("Name: %s\tEmail: %s ", cursor.getString(cursor.getColumnIndex(DBHelper.COLUMN_NAME)),
                cursor.getString(cursor.getColumnIndex(DBHelper.COLUMN_EMAIL))));
    } while (cursor.moveToNext());
}
```

NB move to first row before stating to read

Read until the end

Get column by names instead of column number

Let's whack a record

Delete a row

Define a constraint first, equivalent of the WHERE clause in SQL

```
//Delete from the database
String whereClause = String.format("%s = %d", DBHelper.COLUMN_KEY_ROWID, 1);
id = db.delete(DBHelper.TABLE_NAME, whereClause, null);

if (id > 0) {
    Log.d(TAG, "Row deleted");
}
```

Perform the delete, and note the return. Returns 0 if no rows are deleted, otherwise returns the number of rows deleted

Update

Define a constraint first, equivalent of the WHERE clause in SQL

```
//Update the table
whereClause = String.format("%s = '%s'", DBHelper.COLUMN_EMAIL, "joseph@peruzal.co.za");
values = new ContentValues();
values.put(DBHelper.COLUMN_EMAIL, "joseph@apps4u.co.za");

int rows = db.update(DBHelper.TABLE_NAME, values, whereClause, null);
if (rows > 0) {
    Log.d(TAG, String.format("Updated %d rows", rows));
}
```

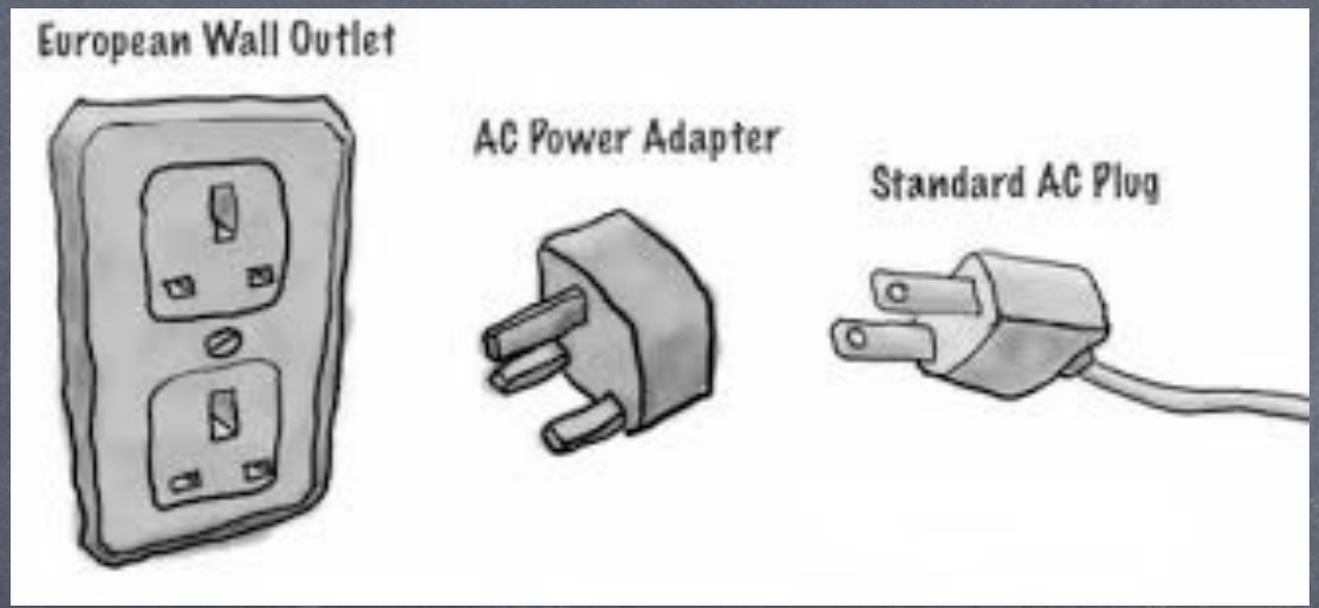
Use ContentValues class to fill in new details

Perform insert and check for the return type, greater than 0 means success

Bind Results to ListView

- Define a SimpleCursorAdapter
- Extend ListActivity for the class
- setListAdapter

SimpleCursorAdapter



Instantiate an Adapter

Define columns to glue from the database to the ListView

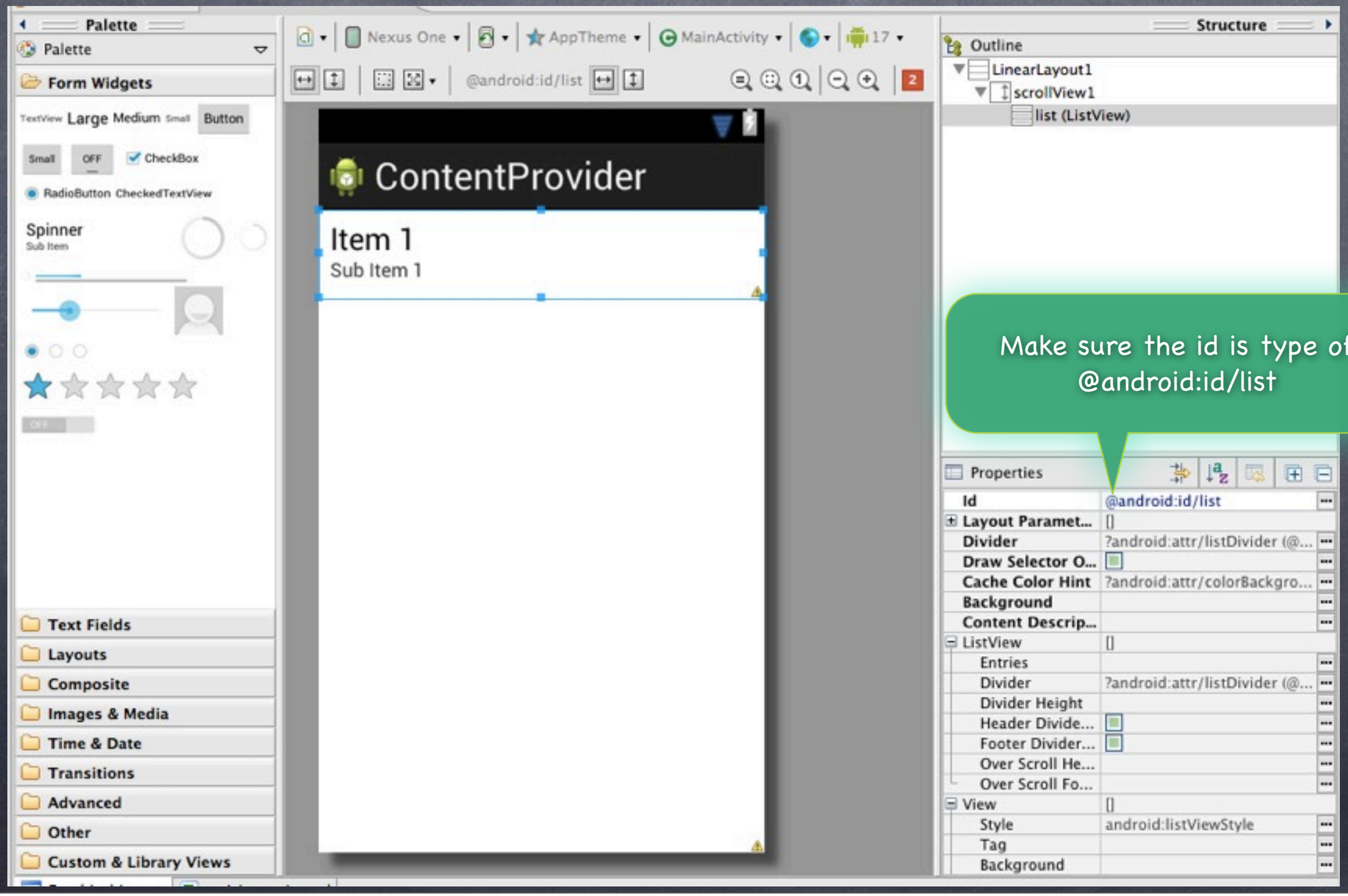
Define the UI id's to glue to the database columns

```
//Using a cursor
String[] from = new String[] {DBHelper.COLUMN_NAME, DBHelper.COLUMN_EMAIL};
int[] to = new int[] { android.R.id.text1, android.R.id.text2   };
SimpleCursorAdapter adapter = new SimpleCursorAdapter(this, android.R.layout.simple_list_item_2, cursor, from, to);

//Close the database


setListAdapter(adapter);
db.close();
```

Use the current content, the android built in layout to glue the database columns to the ListView UI

Make the link here, and don't forget to close the database

UI


Palette
Palette
Form Widgets
TextView Large Medium Small Button
Small OFF CheckBox
RadioButton CheckedTextView
Spinner
Sub Item
Text Fields
Layouts
Composite
Images & Media
Time & Date
Transitions
Advanced
Other
Custom & Library Views
Nexus One
AppTheme
MainActivity
17
@android:id/list
2
ContentProvider
Item 1
Sub Item 1
Structure
Outline
LinearLayout1
scrollView1
list (ListView)
Make sure the id is type of @android:id/list
Properties
Id
@android:id/list
Layout Paramet...
[]
Divider
?android:attr/listDivider (@...
Draw Selector O...
Cache Color Hint
?android:attr/colorBackgro...
Background
Content Descrip...
ListView
[]
Entries
Divider
?android:attr/listDivider (@...
Divider Height
Header Divide...
Footer Divider...
Over Scroll He...
Over Scroll Fo...
View
[]
Style
android:listViewStyle
Tag
Background

Thanks for your time

By the way we an have Android Development Essentials Course on special running from the 29th - 31st of July - "Shameless marketing"

http://www.facebook.com/peruzal

http://www.peruzal.co.za

Thanks for your time

By the way we an have Android Development Essentials Course on special running from the 29th - 31st of July - "Shameless marketing"

http://www.peruzal.co.za