Embedded and Mobile Wireless Systems (EWireless) 2016/2017

Android Guide

Guide to create a database

Database - Package

- The package to import is android.database.sqlite

Define a Schema and Contract

- It is not a requirement, but a formal way to declare how the database is organized.

```
public final class FeedReaderContract {
    /* Inner class that defines the table contents */
    public static abstract class FeedEntry implements BaseColumns {
        public static final String TABLE_NAME = "entry";
        public static final String COLUMN_NAME_ENTRY_ID = "entryid"; //1st
        public static final String COLUMN_NAME_TITLE = "title";      //2nd
        public static final String COLUMN_NAME_SUBTITLE = "subtitle";//3rd
        ...
    }
}
```

Create a Database Using a SQL Helper

- The database is created using the schema we declared earlier.

```
private static final String TEXT_TYPE = " TEXT";
private static final String COMMA_SEP = ",";
private static final String SQL_CREATE_ENTRIES =
        "CREATE TABLE " + FeedEntry.TABLE_NAME +
        " (" +
        FeedEntry._ID + " INTEGER PRIMARY KEY AUTOINCREMENT," +
        FeedEntry.COLUMN_NAME_ENTRY_ID + TEXT_TYPE + COMMA_SEP +  //1st column
        FeedEntry.COLUMN_NAME_TITLE + TEXT_TYPE + COMMA_SEP +     //2nd column
        FeedEntry.COLUMN_NAME_SUBTITLE + TEXT_TYPE + COMMA_SEP +  //3rd column
        ... // Any other options for the CREATE command
        " )";

private static final String SQL_DELETE_ENTRIES =
    "DROP TABLE IF EXISTS " + FeedEntry.TABLE_NAME;
```

- SQL_CREATE_ENTRIES is written based on the following SQLite create table command:
    - *CREATE TABLE entry*
{
entryid TEXT,
title TEXT,
subtitle TEXT
}
- Here is the implementation using the command we created earlier.

```
public class FeedReaderDbHelper extends SQLiteOpenHelper {
// If you change the database schema, you must increment the database
version.
    public static final int DATABASE_VERSION = 1;
    public static final String DATABASE_NAME = "FeedReader.db";

    public FeedReaderDbHelper(Context context) {
        super(context, DATABASE_NAME, null, DATABASE_VERSION);
    }
    public void onCreate(SQLiteDatabase db) {
        db.execSQL(SQL_CREATE_ENTRIES);
    }
    public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion)
{
        db.execSQL(SQL_DELETE_ENTRIES);
        onCreate(db);
    }
}
```

- To access your database, instantiate your subclass of SQLiteOpenHelper:

```
FeedReaderDbHelper mDbHelper = new FeedReaderDbHelper(getContext());
```

- An example for the complete SQLiteOpenHelper (for the EMF only) is given at the end of this guide. You can change or modify it according to your application.

Put Information into a Database

- The values are insert into the database using ContentValues.

```
// Gets the data repository in write mode
SQLiteDatabase db = mDbHelper.getWritableDatabase();

// Create a new map of values, where column names are the keys
ContentValues values = new ContentValues();
values.put(FeedEntry.COLUMN_NAME_ENTRY_ID, id);
values.put(FeedEntry.COLUMN_NAME_TITLE, title);
values.put(FeedEntry.COLUMN_NAME_CONTENT, content);

// Insert the new row
db.insert(
         FeedEntry.TABLE_NAME,
         FeedEntry.COLUMN_NAME_NULLABLE,
         values);
```

- Make sure the SQLiteOpenHelper is already instantiated.

Read Information from a Database

- The query() method is used for reading information from the database.

```
SQLiteDatabase db = mDbHelper.getReadableDatabase();

// Define a projection that specifies which columns from the database
// you will actually use after this query.
String[] projection = {
    FeedEntry._ID,
    FeedEntry.COLUMN_NAME_TITLE,
    FeedEntry.COLUMN_NAME_UPDATED,
    ...
    };

// How you want the results sorted in the resulting Cursor
String sortOrder =
    FeedEntry.COLUMN_NAME_UPDATED + " DESC";

Cursor c = db.query(
    FeedEntry.TABLE_NAME,  // The table to query
    projection,                               // The columns to return
    selection,                                // The columns for the WHERE clause
    selectionArgs,                            // The values for the WHERE clause
    null,                                     // don't group the rows
    null,                                     // don't filter by row groups
    sortOrder                                 // The sort order
    );
```

- Before reading values, you must moves the cursor to the desired position.
- You can use the following command to move the cursor’s position.

```
cursor.moveToFirst();
cursor.moveToLast();
cursor.moveToNext();
cursor.moveToPrevious();
cursor.moveToPosition(i);
```

- You can use the following command to read the data from the cursor position, where i is the number of column to retrieve value from.

```
cursor.getString(i);
cursor.getDouble(i);
cursor.getFloat(i);
cursor.getInt(i);
cursor.getDouble(i);
cursor.getShort(i);
cursor.getLong(i);
```

- The example of the implementation is as follow:

```
cursor.moveToPosition(i);
EntryId[i]      = cursor.getString(1);
CoorX[i]        = cursor.getInt(2);
CoorY[i]        = cursor.getInt(3);
EmfX[i]         = cursor.getInt(4);
EmfY[i]         = cursor.getInt(5);
EmfZ[i]         = cursor.getInt(6);
```

Delete Information from a Database

- To delete an information from a database, you can use the following command.

```
// Define 'where' part of query.
String selection = FeedEntry.COLUMN_NAME_ENTRY_ID + " LIKE ?";
// Specify arguments in placeholder order.
String[] selectionArgs = { String.valueOf(rowId) };
// Issue SQL statement.
db.delete(table_name, selection, selectionArgs);
```

Update a Database

- To update the database, you can use the following command.

```
SQLiteDatabase db = mDbHelper.getReadableDatabase();

// New value for one column
ContentValues values = new ContentValues();
values.put(FeedEntry.COLUMN_NAME_TITLE, title);

// Which row to update, based on the ID
String selection = FeedEntry.COLUMN_NAME_ENTRY_ID + " LIKE ?";
String[] selectionArgs = { String.valueOf(rowId) };

db.update(
    FeedReaderDbHelper.FeedEntry.TABLE_NAME,
    values,
    selection,
    selectionArgs);
```

Reference:

http://developer.android.com/training/basics/data-storage/databases.html

The following is an example for a complete SQLiteOpenHelper (for the EMF only). Modify as necessary.

- This file contains a completed code for “Define a Schema and Contract” and “Create a Database Using a SQL Helper”.
- The database in this file is designed with the following information:
    - Database name: Fingerprint.db
    - Table name: EmfPosition
    - Number of column: 6 (7 including _ID)
    - Column name 0 = _ID (Autoincrement)`
    - Column name 1 = EntryId (Data type: Text)
    - Column name 2 = CoordinateX (Data type: Integer)
    - Column name 3 = CoordinateY (Data type: Integer)
    - Column name 4 = EMFx (Data type: Real)
    - Column name 5 = EMFy (Data type: Real)
    - Column name 6 = EMFz (Data type: Real)
- You can use the file as it is (Copy and paste into Android Studio).
- You can modify the file according to your designed application (adding table, adding column or etc)
- This SQLiteOpenHelper can be instantiate using:

```
FeedReaderDbHelper mDbHelper = new FeedReaderDbHelper(getContext());
```

FeedReaderDbHelper.java

```
package com.example.arief;

import android.content.Context;
import android.database.sqlite.SQLiteDatabase;
import android.database.sqlite.SQLiteOpenHelper;
import android.provider.BaseColumns;

public class FeedReaderDbHelper extends SQLiteOpenHelper {
    /* Inner class that defines the table contents */
    public static abstract class FeedEntry implements BaseColumns {
        public static final String TABLE_NAME = "EmfPosition";
        public static final String COLUMN_NAME_ENTRY_ID = "EntryId";
        public static final String COLUMN_NAME_COORDINATEX = "CoordinateX";
        public static final String COLUMN_NAME_COORDINATEY = "CoordinateY";
        public static final String COLUMN_NAME_EMFX = "EMFx";
        public static final String COLUMN_NAME_EMFY = "EMFy";
        public static final String COLUMN_NAME_EMFZ = "EMFz";
    }

    private static final String TEXT_TYPE = " TEXT";
    private static final String INTEGER_TYPE = " INTEGER";
    private static final String REAL_TYPE = " REAL";
    private static final String COMMA_SEP = ",";
    private static final String SQL_CREATE_ENTRIES =
            "CREATE TABLE " + FeedEntry.TABLE_NAME + " (" +
                    FeedEntry._ID + " INTEGER PRIMARY KEY AUTOINCREMENT," +
                    FeedEntry.COLUMN_NAME_ENTRY_ID + TEXT_TYPE + COMMA_SEP +
                    FeedEntry.COLUMN_NAME_COORDINATEX + INTEGER_TYPE +
COMMA_SEP +
                    FeedEntry.COLUMN_NAME_COORDINATEY + INTEGER_TYPE +
COMMA_SEP +
                    FeedEntry.COLUMN_NAME_EMFX + REAL_TYPE + COMMA_SEP +
                    FeedEntry.COLUMN_NAME_EMFY + REAL_TYPE + COMMA_SEP +
                    FeedEntry.COLUMN_NAME_EMFZ + REAL_TYPE +
                    " )";

    private static final String SQL_DELETE_ENTRIES =
            "DROP TABLE IF EXISTS " + FeedEntry.TABLE_NAME;

    // If you change the database schema, you must increment the database
version.
    public static final int DATABASE_VERSION = 1;
    public static final String DATABASE_NAME = "Fingerprint.db";

    public FeedReaderDbHelper(Context context) {
        super(context, DATABASE_NAME, null, DATABASE_VERSION);
    }
    public void onCreate(SQLiteDatabase db) {
        db.execSQL(SQL_CREATE_ENTRIES);
    }
    public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion)
{
    // This database is only a cache for online data, so its upgrade policy
is
    // to simply to discard the data and start over
    db.execSQL(SQL_DELETE_ENTRIES);
    onCreate(db);
    }
}
```