CS493

Regis University Senior Capstone

Week Five

Intructor: Mr. Allan Rossi

Author: Mr. Jason N Mansfield

The Object Based Study Application

August 3, 2013

Contents

August 3, 2013

1 Continued efforts with ListView and SQLite3

I am pleased to announce that my knowledge of maintaining and utilizing the SQLite3 database (*Saving Data in SQL Databases* 2013) in Android has become almost second nature to me at this time. Unfortunately, I am just now beginning to learn about the effects removing data can have on the ListView (*ListView* 2013). Before I explain my conundrum allow me to pass some details. I intended to release a beta this week in accordance with my schedule. This application is extremely buggy and while I do not believe it will harm your phone it should be deleted after examining. The current beta release is codenamed DataFace and can be cloned using Git here:

```
git clone https://github.com/trentonknight/DataFaceProject.git
```

The project apk can be downloaded using wget, curl or with a browser from here:

```
wget https://github.com/trentonknight/DataFaceProject/blob/master/DataFace/build/apk/DataFace-release-unsigned.apk
```

The codename for the project was unintentional. I simply began making headway while using this particular version. Currently, the ListView is created by a class I created called the ListViewLoader, not very original I know, this class extends a ListActivity (*ListActivity* 2013) and implements a LoaderManager (*LoaderManager* 2013) which uses Cursors (*Cursor* 2013). The ListViewLoader is using a ListView which creates the array type list which lists one specific item I have selected out of all the columns. In this particular case I have asked the Key Object name to be shown. Here is a view of the class DataBaseH which handle all the database calls:

```
public Cursor getAllColumns(){
SQLiteDatabase db = this.getReadableDatabase();
return db.query(KEYS.TABLE_NAME, new String[] {KEYS.KEY_ID,
KEYS.KEY_OBNAME, KEYS.KEY_CONTENT},null,null,null,null,null);
}
```

Using the class getAllColumns the ListViewLoader class can now access this selection with a simple statement such as:

```
final DatabaseH db = new DatabaseH(this);
db.onOpen(db.getReadableDatabase());
final Cursor data = db.getAllColumns();
```

2 Implementing Adapters and Cursors in Android

One of the best and worst features I have discovered while developing this particular application is the use of adapters (*Adapter* 2013). To being with, and in a large part due to the tutorials I was using, I started my original ListView using a ArrayAdapter (*ArrayAdapter* 2013). Additional research and study lead me to believe that the SimpleCursorAdapter (*SimpleCursorAdapter* 2013) was more suited for applications use of SQLite3. I have had some difficulty with implementing a delete feature. Deleting from the database is easy enough but getting the ListView to recognize the item removal has been a major issue for me. So far the tutorials and books I have read seem to skirt around this very important feature. Today I was able to completely re-write my Parent ListView class and implement the SimpleCursorAdapter. It is my hope that this class offers more features which allow me to properly remove items from the ListView with ease. To demonstrate the use of the database, Cursor to the database, and the adapter being passed to the ListView, here is a code snippet of these features all together:

```
String[] fromColumns = {DatabaseH.KEYS.KEY_OBNAME};
int[] toViews = {android.R.id.text1};
final DatabaseH db = new DatabaseH(this);
db.onOpen(db.getReadableDatabase());

final Cursor data = db.getAllColumns();


mAdapter = new SimpleCursorAdapter(this, android.R.layout.simple_list_item_1,
data, fromColumns, toViews, 0);
final ListView listView = getListView();
listView.setAdapter(mAdapter);
listView.setOnItemClickListener(new AdapterView.OnItemClickListener() {
@Override
public void onItemClick(AdapterView<?> adapterView, View view, int i, long l) {
passDataToTheChild(i);
db.close();
}
});
```

3 Whats your intent

One major feature I took for granted when I began this project was the powerful use of what the Android API calls Intents (*Intent* 2013). Intents are an interesting concept defined by the latest API as:

> An Intent provides a facility for performing late runtime binding between the code in different applications. Its most significant use is in the launching of activities, where it can be thought of as the glue between activities. It is basically a passive data structure holding an abstract description of an action to be performed.

The Intent feature is what allowed me to pass a Bundle (*Bundle* 2013) containing a needed variable from the Parent ListActivity to my Child Activity or more specifically from a Parent class to a Child class. It was necessary for me to pass the ListViews current position so the correct child was retrieved:

```
public void passDataToTheChild(int position){

Intent intent = new Intent(this, ListChildActivity.class);
Bundle bundle = new Bundle();
bundle.putInt("position", position);
intent.putExtras(bundle);
startActivity(intent);

}
```

After which the Child Activity a class called ListChildActivity receives the bundle in the following fashion:

```
Bundle bundle = getIntent().getExtras();
int newContent = bundle.getInt("position");
```

It may not seem worth all those calls for a single variable but it was a vital part of my child reading the proper column in the database. It is possible I will find a better way to identify the current location of the ListView within the Child Activity, but for the time being this is the best solution I'm aware of. My Goal for next week is to solidify how the ListView communicates with the adapter and Cursor; I believe once I can grasp those concepts I can then remove the proper ListView Items to match the database deletions.

Bibliography

Adapter (2013). Android Open Source Project. URL: http://developer.android.com/reference/android/widget/Adapter.html.

ArrayAdapter (2013). Android Open Source Project. URL: http://developer.android.com/reference/android/widget/ArrayAdapter.html.

Bundle (2013). Android Open Source Project. URL: http://developer.android.com/reference/android/os/Bundle.html.

Cursor (2013). Android Open Source Project. URL: http://developer.android.com/reference/android/database/Cursor.html.

Intent (2013). Android Open Source Project. URL: http://developer.android.com/reference/android/content/Intent.html.

ListActivity (2013). Android Open Source Project. URL: http://developer.android.com/reference/android/app/ListActivity.html.

ListView (2013). Android Open Source Project. URL: http://developer.android.com/reference/android/widget/ListView.html.

LoaderManager (2013). Android Open Source Project. URL: http://developer.android.com/reference/android/app/LoaderManager.html.

Saving Data in SQL Databases (2013). Android Open Source Project. URL: http://developer.android.com/training/basics/data-storage/databases.html.

SimpleCursorAdapter (2013). Android Open Source Project. URL: http://developer.android.com/reference/android/support/v4/widget/SimpleCursorAdapter.html.