Lab 5: Content Providers, Advanced DBs, Facebook, and APIs

Dr. Kanda Runapongsa Saikaew
Department of Computer Engineering
Faculty of Engineering
Khon Kaen University
http://twitter.com/krunapon

Agenda

- Content Providers
- Advanced Databases
  - Search
  - Update
- Facebook
- APIs

The Need for Content Providers

- Databases created in Android are visible only to the application that created them
- SQLite database created on Android by one application is usable only by that application, not by other applications
- If you need to share data between applications, you need to use the *content provider model* as recommended in Android

Using a Content Provider

- In Android, a content provider is a specialized type of data store that exposes standardized ways to retrieve & manipulate the stored data
- Android ships with useful content providers

Content Provider	Intended Data
Browser	Browser bookmarks, browser history, etc.
CallLog	Missed calls, call details, etc.
Contacts	Contact details
MediaStore	Media files such as audio, video and images
Settings	Device settings and preferences

Content Provider URI

- To query a content provider, you provide a query string in the form of a URI, using syntax

<standard_prefix>://<authority>/<data_path>/<id>

Examples:

- content://browser/bookmarks
- content://contacts/people

To retrieve a particular contact, you can specify the URI with a specific ID:

- content://contacts/people/3

Example: Using Call Log Provider

- Need to add in the READ_CONTACTS permission in the AndroidManifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest
xmlns:android="http://schemas.android.com/
apk/res/android" ...>
    <application android:icon="@drawable/icon"
android:label="@string/app_name"> ...
    </application>
<uses-permission
android:name="android.permission.READ_CONTACTS"
</uses-permission>
</manifest>
```

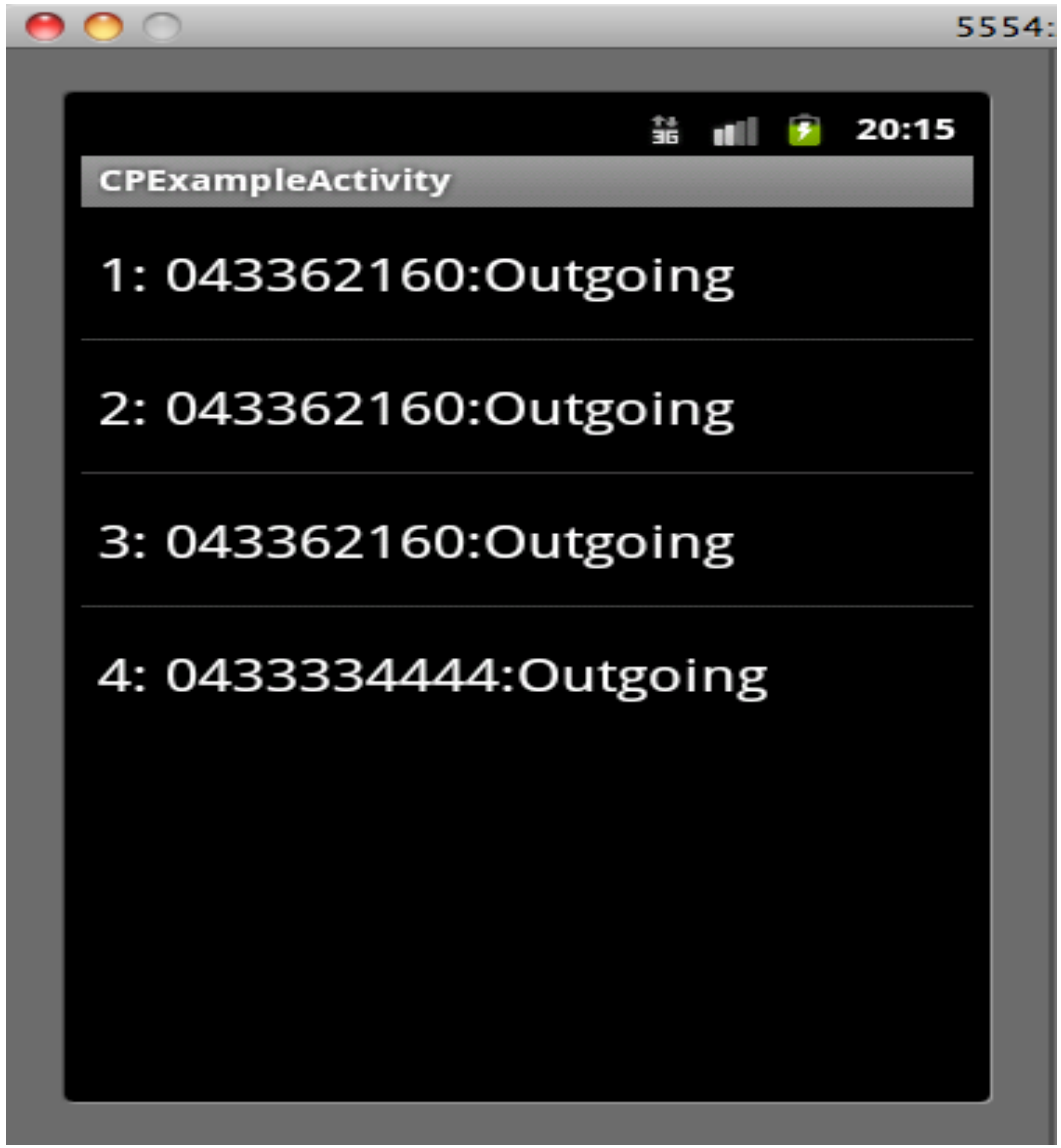
Example: CPExampleActivity (1/2)

```
public class CPExampleActivity extends Activity {
 /** Called when the activity is first created. */
 @Override public void onCreate(Bundle
savedInstanceState) { super.onCreate
(savedInstanceState); setContentView(R.layout.
main);
  Uri allCalls = Uri.parse("content:
//call_log/calls");
 Cursor c = managedQuery(allCalls, null, null,
null, null);
```

Example: CPExampleActivity (2/2)

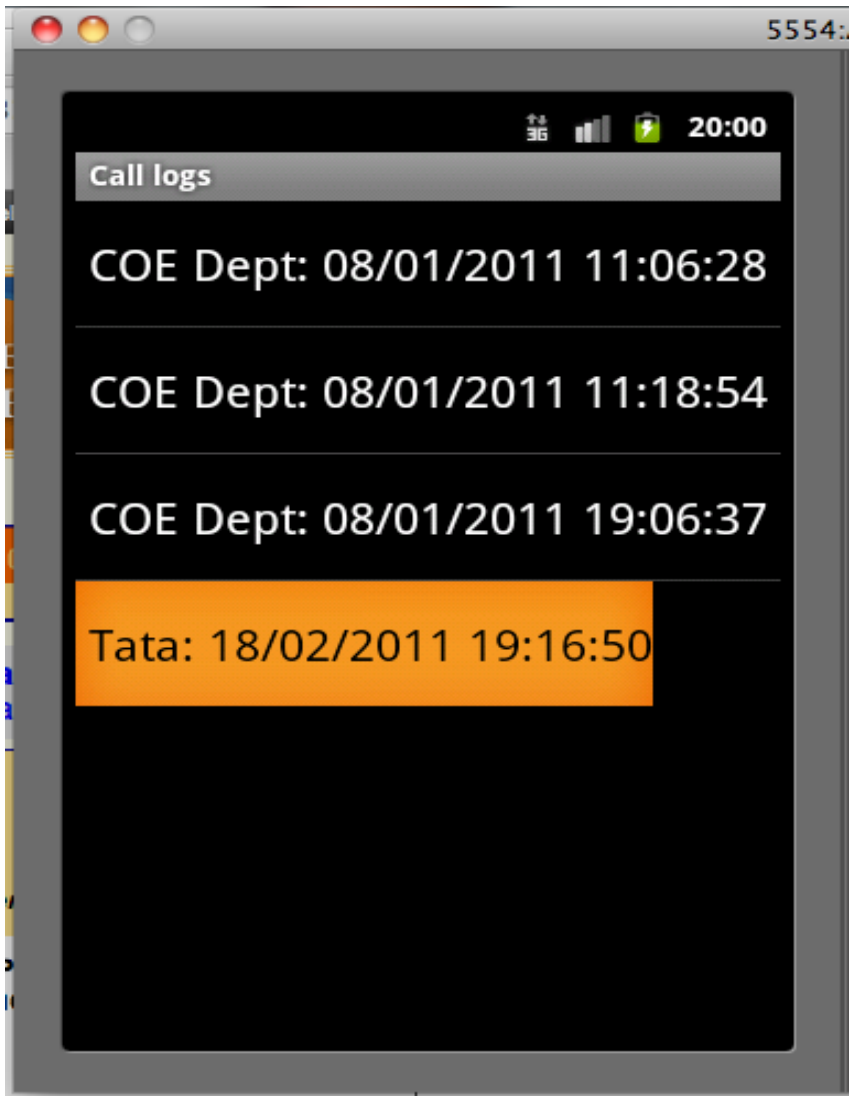
```
if (c.moveToFirst()) {
 do{
 String callType = "";
 switch (Integer.parseInt(c.getString( c.getColumnIndex(Calls.
TYPE)))) {
 case 1: callType = "Incoming";  break;
 case 2: callType = "Outgoing"; break;
 case 3: callType = "Missed"; }
 list.add(c.getString(c.getColumnIndex(Calls._ID)) + ": " +  c.getString(c.
getColumnIndex(Calls.NUMBER)) + ":" +
                    callType) ;
          } while (c.moveToNext());
        }
    // Get the list view
        ListView listView = (ListView) findViewById(R.id.listView);
        ArrayAdapter<String> aa = new ArrayAdapter<String>(this,
android.R.layout.simple_list_item_1, list);
        listView.setAdapter(aa);
```

Call Log


5554:
20:15
CPExampleActivity
1: 043362160:Outgoing
2: 043362160:Outgoing
3: 043362160:Outgoing
4: 0433334444:Outgoing

Call Log

- Access other members (name and call time) of android call logs



A Content Provider

- A content provider is a class that implements a standard set of methods to let other applications store and retrieve the type of data that is handled by that content provider
- If you want your data to be public and handle by many applications create your own content provider
- **Application can perform following operations on content provider -**

1. Querying data
2. Modifying records
3. Adding records
4. Deleting records

Standard Content Provider

- Android provide some standard content provider, which are already implemented in Android e.g. contacts, images on device
- Using these content providers the application can access contact information, images available on the device
- **Querying data:**
  - The query string in content provider is different than standard sql query
  - For any operation like select, add, delete, modify we required content provider URI
  - The URI consist of three parts, the string “content://”, segment representing kind of data to retrieve and optional ID of specific item of the specified content provider

DBAdapter Helper Class

- A good practice for dealing with databases is to create a helper class to encapsulate all the complexities of accessing the database so that it's transparent to the calling code
- So, create a helper class called DBAdapter that creates, opens, closes, and uses a SQLite database
- Within the DBAdapter class, you extend the SQLiteOpenHelper class—an Android helper class for database creation and versioning management. In particular, you override the onCreate() and onUpgrade() methods

SQLiteOpenHelper

- The onCreate() method creates a new database if the required database is not present
- The onUpgrade() method is called when the database needs to be upgraded. This is achieved by checking the value defined in the DATABASE_VERSION constant.
- For this implementation of the onUpgrade() method, you will simply drop the table and create the table again
- You can now define the various methods for opening and closing the database, as well as the methods for adding/editing/deleting rows in the table

Cursor

- Notice that Android uses the Cursor class as a return value for queries
- Think of the Cursor as a pointer to the result set from a database query
- Using Cursor allows Android to more efficiently manage rows and columns as and when needed
- You use a ContentValues object to store key/value pairs. Its put() method allows you to insert keys with values of different data types

DBAdapter (1/6)

```
public class DBAdapter {
 public static final String KEY_ROWID = "_id";
 public static final String KEY_ISBN = "isbn";
 public static final String KEY_TITLE = "title";
 public static final String KEY_PUBLISHER = "publisher";
 private static final String TAG = "DBAdapter";
 private static final String DATABASE_NAME = "books";
 private static final String DATABASE_TABLE = "titles";
 private static final int DATABASE_VERSION = 1;
private final Context context;
 private DatabaseHelper DBHelper;
 private SQLiteDatabase db;
```

DBAdapter (2/6)

```
private static final String DATABASE_CREATE =
 "create table titles (_id integer primary key  autoincrement, "
+ "isbn text  not null, title text not null, "
+ "publisher text not null);";

public DBAdapter(Context ctx) {
     this.context = ctx;
     DBHelper = new DatabaseHelper(context);
}

private static class DatabaseHelper extends
SQLiteOpenHelper {
    DatabaseHelper(Context context) {
         super(context, DATABASE_NAME, null,
 }
```

DBAdapter (3/6)

```
@Override public void onCreate(SQLiteDatabase db) {
execSQL(DATABASE_CREATE);
}
 @Override public void onUpgrade(SQLiteDatabase db,
int oldVersion, int newVersion) {
   Log.w(TAG, "Upgrading database from version "
   + oldVersion + " to " + newVersion + ",
  which will destroy all old data");
   db.execSQL("DROP TABLE IF EXISTS titles");
   onCreate(db);
}
}
```

DBAdapter (4/6)

```
//---opens the database---
 public DBAdapter open() throws SQLException {
 db = DBHelper.getWritableDatabase();
  return this;
 }
 //---closes the database---
 public void close() {
  DBHelper.close();
 }
 //---insert a title into the database---
 public long insertTitle(String isbn, String title,  String publisher) {
 ContentValues initialValues = new ContentValues();
 initialValues.put(KEY_ISBN, isbn);
 initialValues.put(KEY_TITLE, title);
 initialValues.put(KEY_PUBLISHER, publisher);
 return db.insert(DATABASE_TABLE, null, initialValues);
 }
```

DBAdapter (5/6)

```
//---deletes a particular title---
 public boolean deleteTitle(long rowId) {
 return db.delete(DATABASE_TABLE, KEY_ROWID + "=" + rowId,
null) > 0;
 }
 //---retrieves all the titles---
 public Cursor getAllTitles() {
 return db.query(DATABASE_TABLE,
new String[] {
 KEY_ROWID, KEY_ISBN, KEY_TITLE, KEY_PUBLISHER},
 null, null, null, null, null);
 }
```

DBAdapter (6/6)

```
//---retrieves a particular title---
 public Cursor getTitle(long rowId) throws SQLException {
 Cursor mCursor = db.query(true, DATABASE_TABLE,
new String[] { KEY_ROWID, KEY_ISBN, KEY_TITLE,
KEY_PUBLISHER }, KEY_ROWID + "=" + rowId, null, null, null, null,
null);
 if (mCursor != null) {
 mCursor.moveToFirst();
 }
 return mCursor; }
//---updates a title---
public boolean updateTitle(long rowId, String isbn, String title, String
publisher) {
ContentValues args = new ContentValues();
args.put(KEY_ISBN, isbn);
args.put(KEY_TITLE, title);
args.put(KEY_PUBLISHER, publisher);
return db.update(DATABASE_TABLE, args, KEY_ROWID + "=" +
rowId, null ) > 0; } }
```

BooksDB (1/4)

```
public class BooksDB extends Activity {
private ArrayList<String> list;
DBAdapter db;
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        list = new ArrayList<String>();
        db = new DBAdapter(this);
     db.open();
        long id;
        id = db.insertTitle("0470285818",
        "C# 2008 Programmer's Reference", "Wrox");
```

BooksDB (2/4)

```
    id = db.insertTitle("047017661X",
    "XML and Web Services", "KKU");
    id = db.insertTitle("0470171423",
    "Programming for Android","Se-ed");
    db.close();
  //---get a title---
    db.open();
    Cursor c = db.getAllTitles();
    if (c.moveToFirst()) {
    do {
    displayTitle(c);
    } while (c.moveToNext());
    } else {
        list.add("No title found");
    }
```

BooksDB (3/4)

```
        db.close();
        // Get the list view
        ListView listView = (ListView) findViewById(R.id.listView);
        ArrayAdapter<String> aa = new ArrayAdapter<String>
(this, android.R.layout.simple_list_item_1, list);
        listView.setAdapter(aa);
    }
    public void displayTitle(Cursor c)
    {
        list.add(
                "id: " + c.getString(0) + "\n" +
                "ISBN: " + c.getString(1) + "\n" +
                "TITLE: " + c.getString(2) + "\n" +
                "PUBLISHER:  " + c.getString(3));
    }
```

BooksDB (4/4)

```
public void updateTitle(Cursor c, String newTitle) {
    String id = c.getString(0);
    String isbn = c.getString(1);
    String title = newTitle;
    String publisher = c.getString(3);
    db.updateTitle(new Long(id), isbn,
        title, publisher);
    }
}
```

BooksDB Output

Search Query

Syntax:

public Cursor query (boolean distinct, String table, String[] columns, String selection, String[] selectionArgs, String groupBy, String having, String orderBy, String limit)

Example:

```
db.query(
true /*true if you want each row to be unique, false otherwise. */,
DATABASE_TABLE /* table name */,
new String[] { KEY_ROWID, KEY_ISBN, KEY_TITLE, KEY_PUBLISHER }
/* columns */,
KEY_ROWID + "=" + rowId /* selection */,
null /* selection arguments */,
null /* group by */,
null /* having */,
null /* order by */,
null /* limit */);
```

Update Query

Syntax:

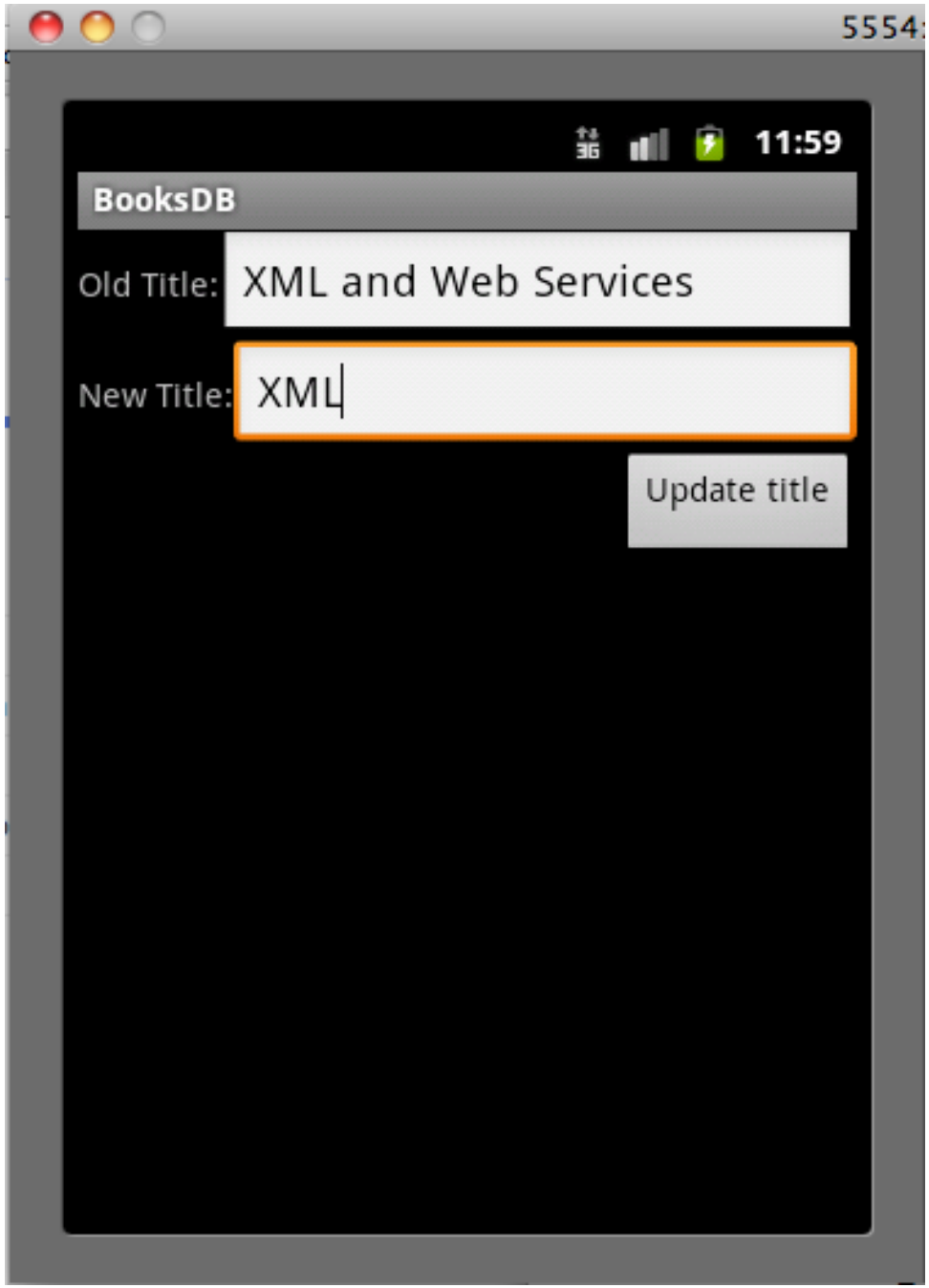
public int update (String table, ContentValues values, String whereClause, String[] whereArgs)

- This function return the number of rows affected

Example:

```
db.update(DATABASE_TABLE /* table name */,
args /* content values */,
KEY_ROWID + "=" + rowId /* where clause */,
null  /* where arguments */)
```

BooksDBUpdate Output



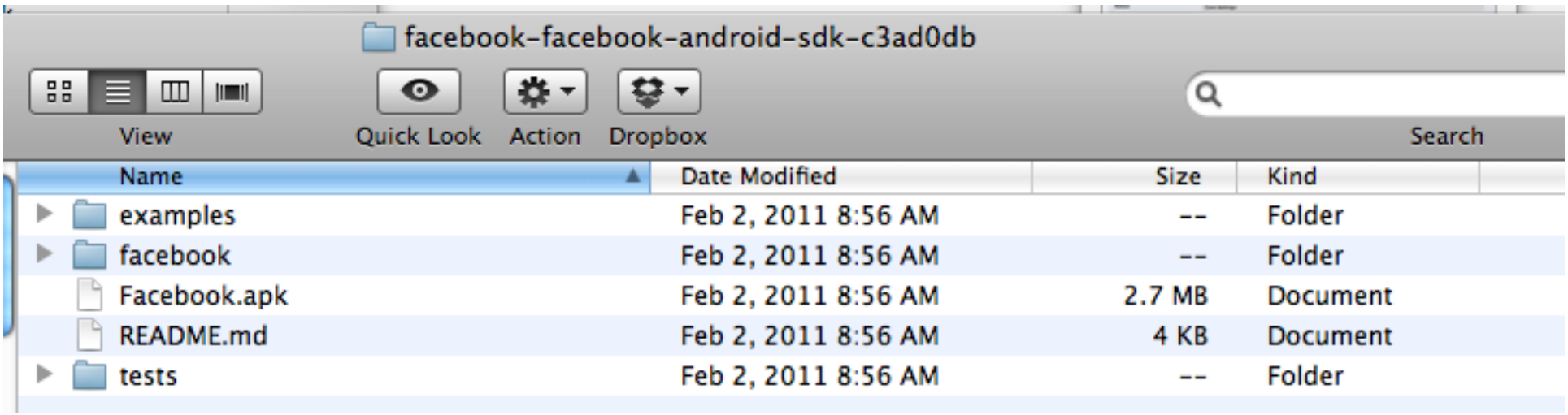

```
sqlite> select * from titles;
1|0470285818|C# 2008 Programmer's Reference|Wrox
2|047017661X|XML and Web Services|KKU
3|0470171423|Programming for Android|Se-ed
sqlite> select * from titles;
1|0470285818|C# 2008 Programmer's Reference|Wrox
2|047017661X|XML|KKU
3|0470171423|Programming for Android|Se-ed
```

How to Develop Facebook App for Android

1. Install Facebook SDK for Android
2. Generate and export signature for Android App
3. Apply for Facebook Application ID
4. Test Sample Project

Install Facebook SDK for Android

- Get the git file
  - 1. Download via web browser
    - https://github.com/facebook/facebook-android-sdk
  - 2. Use Git
    - Clone the GitHub repository: git clone git://github.com/facebook/facebook-android-sdk.git
- Extract the git file, then you should get this directory



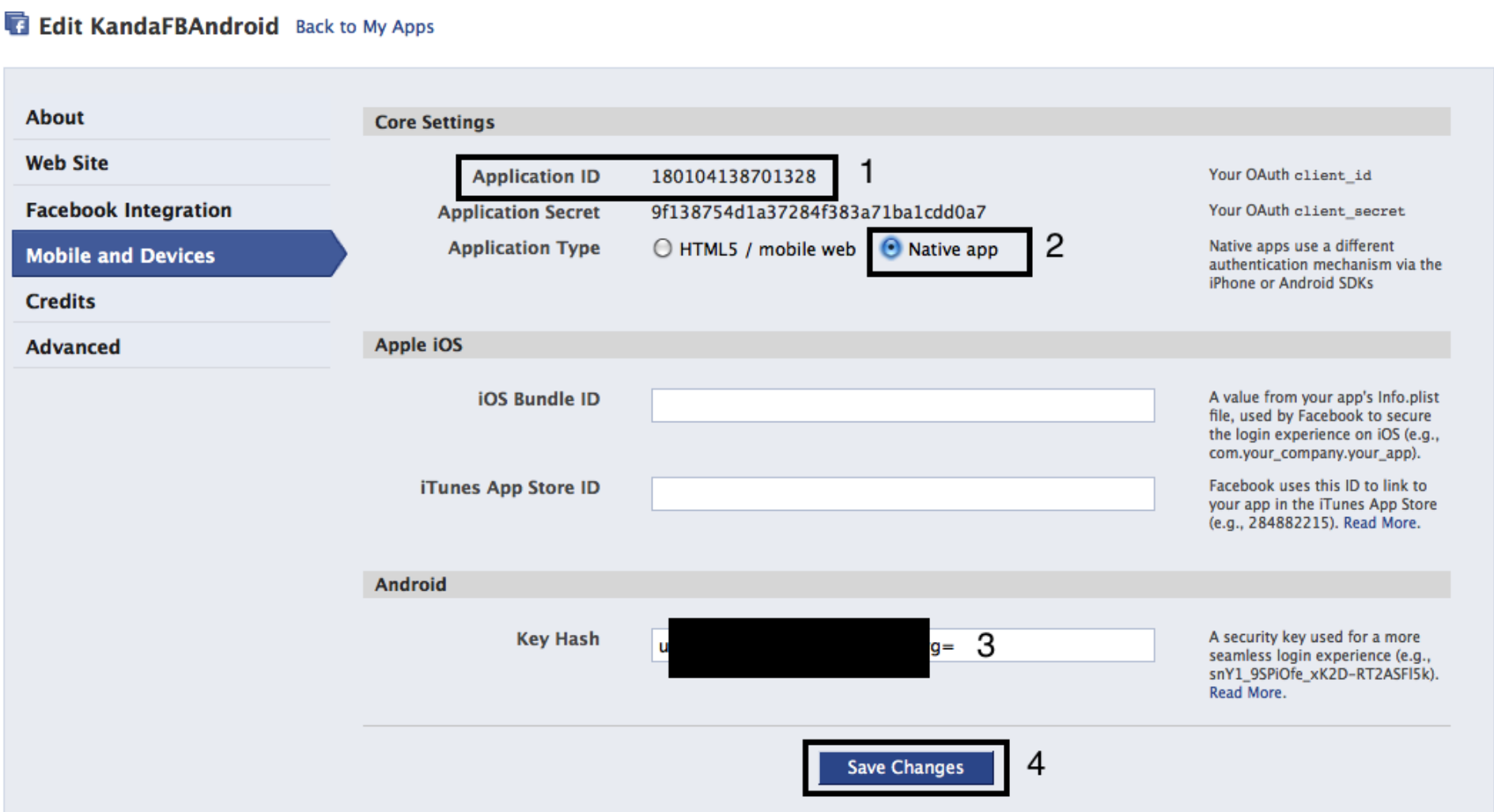
Generate and export signature for Android App

- Using command keytool and use keystore password as android
  - keytool -exportcert -alias androiddebugkey -keystore ~/.android/debug.keystore | openssl sha1 -binary | openssl base64

```
Macbooks-MacBook-Pro:facebook-android Macbook$ keytool -exportcert -alias androi
ddebugkey -keystore ~/.android/debug.keystore | openssl sha1 -binary | openssl b
ase64
Enter keystore password:  android
u[redacted]wQ=
```

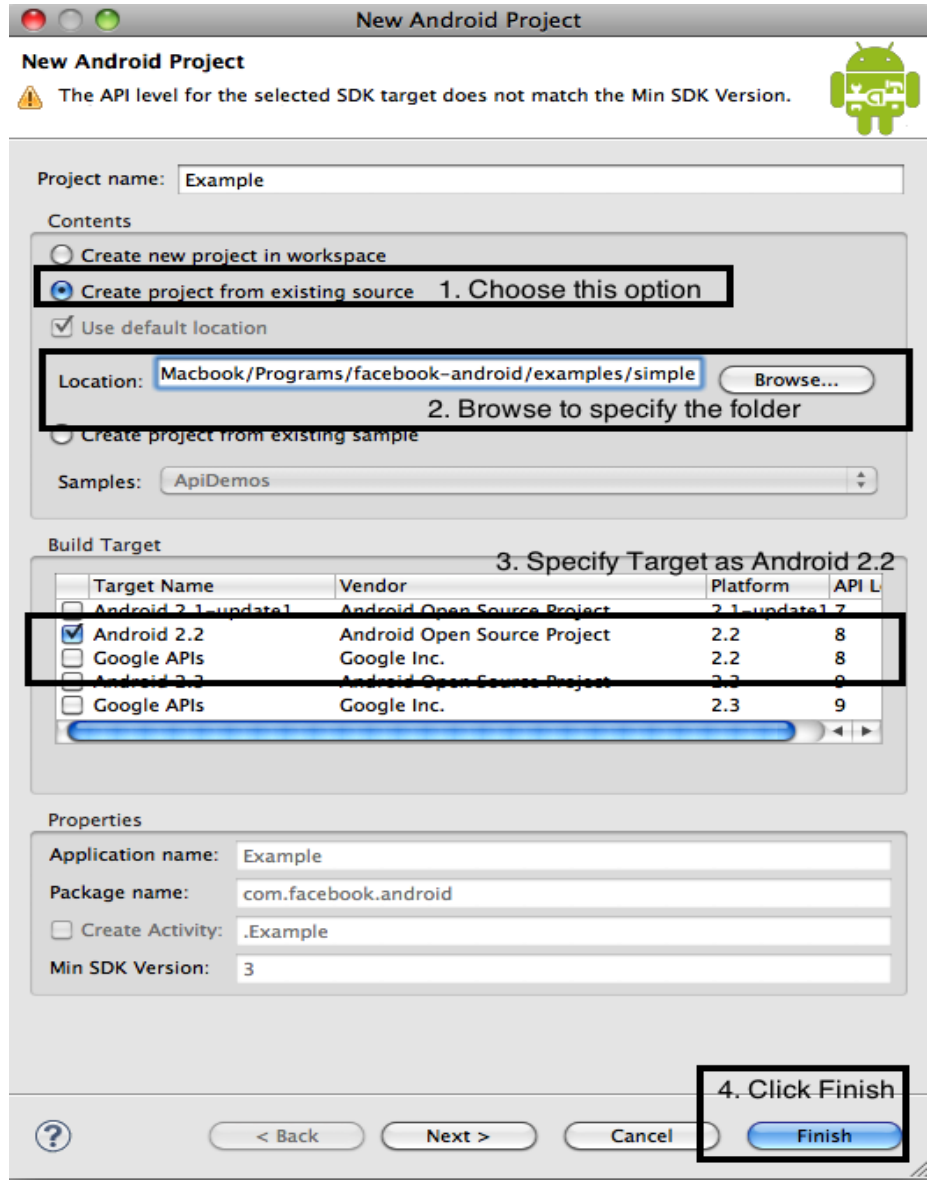
- Then copy the signature that you have which is marked with black color in the above picture

Apply for Facebook App ID

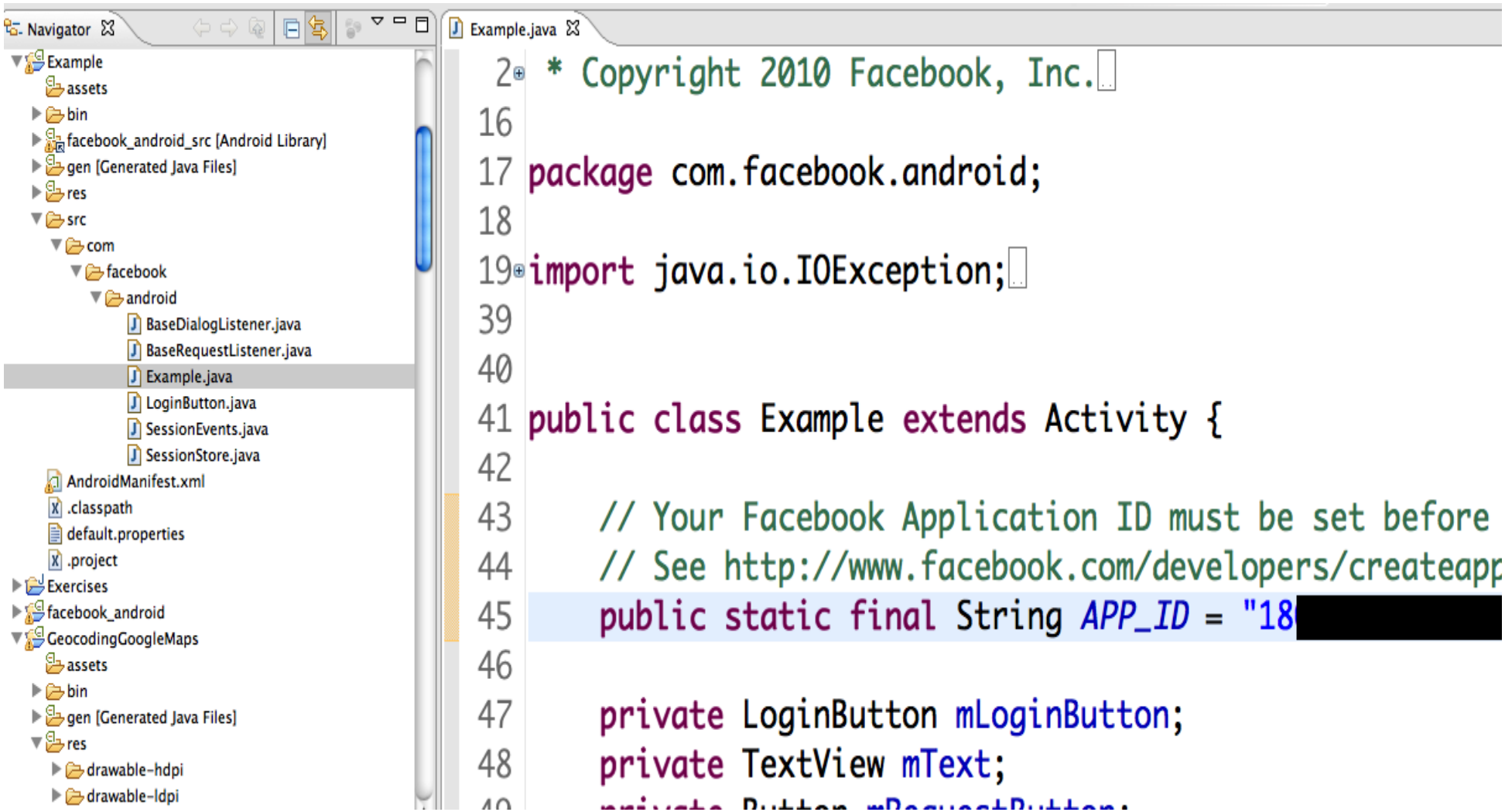


Open Facebook Android Sample Project

- Click File > New > Android Project

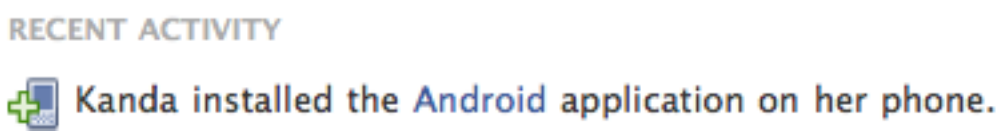


Edit the File to Insert App ID



Facebook Android App (1/5)

- After successfully running and installing the sample Android application on your Android device or your Android emulator, there will be the status posted on your Facebook wall indicating that you have installed the Android application on your phone



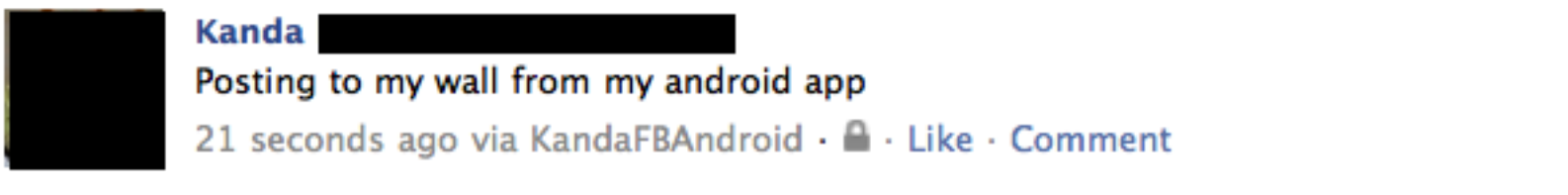
Facebook Android App (2/5)

Facebook Android App (3/5)

Facebook Android App (4/5)

Facebook Android App (5/5)

- What you type on the Android application now appears on your Facebook wall

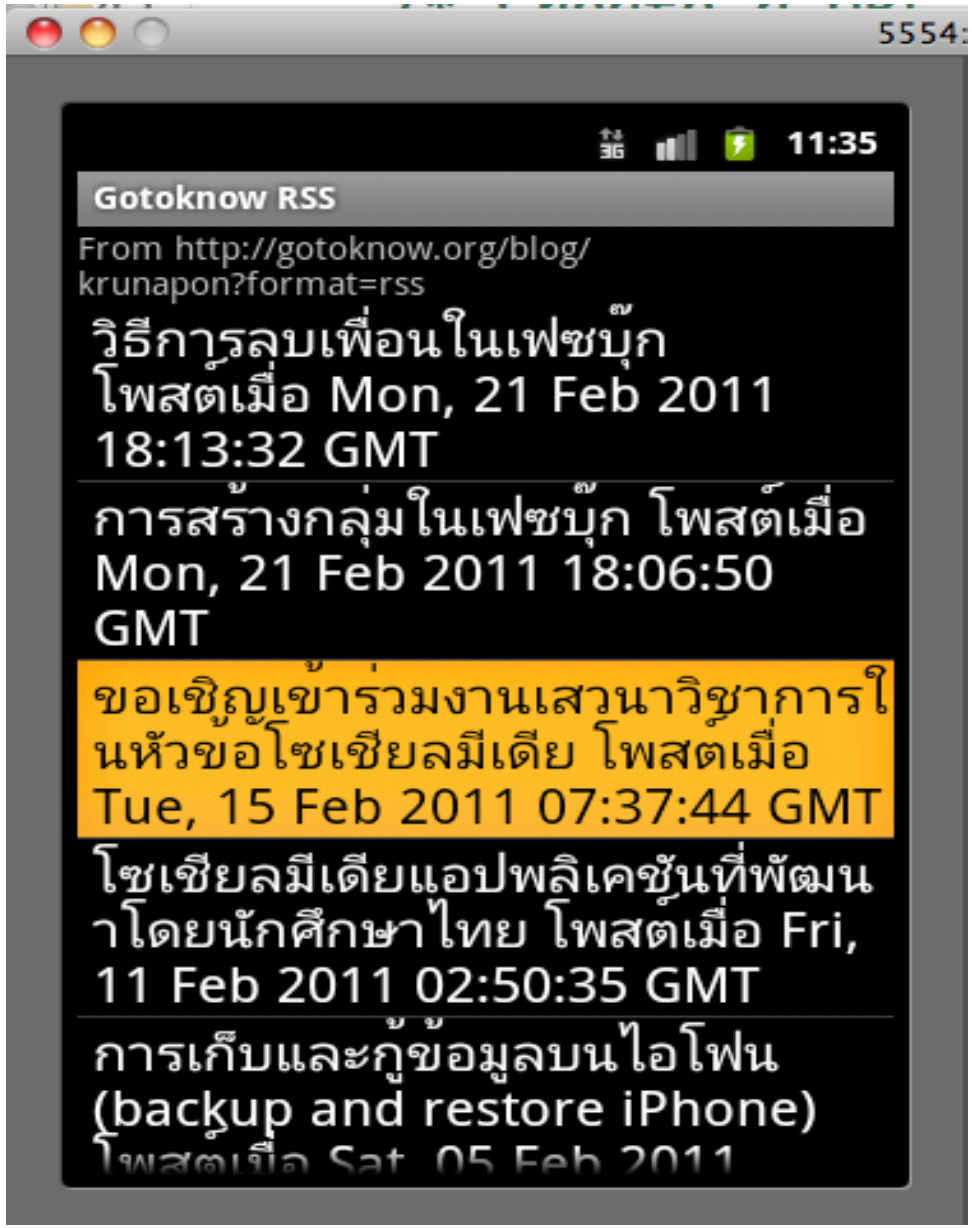


Getting RSS Feed Using URL and Java IO

```
// Here, we simply use class URL and Java IO classes to
// read and access the information from the given URL name
URL url = new URL(urlName);
InputStream in = url.openStream();
InputSource isr = new InputSource(in);
isr.setEncoding("utf-8");

// Assume that the data returned from calling the given URL
// URL is in XML format
DocumentBuilder builder = DocumentBuilderFactory.
newInstance()
.newDocumentBuilder();
Document doc = builder.parse(isr);
...
```

Result from Accessing Gotoknow RSS



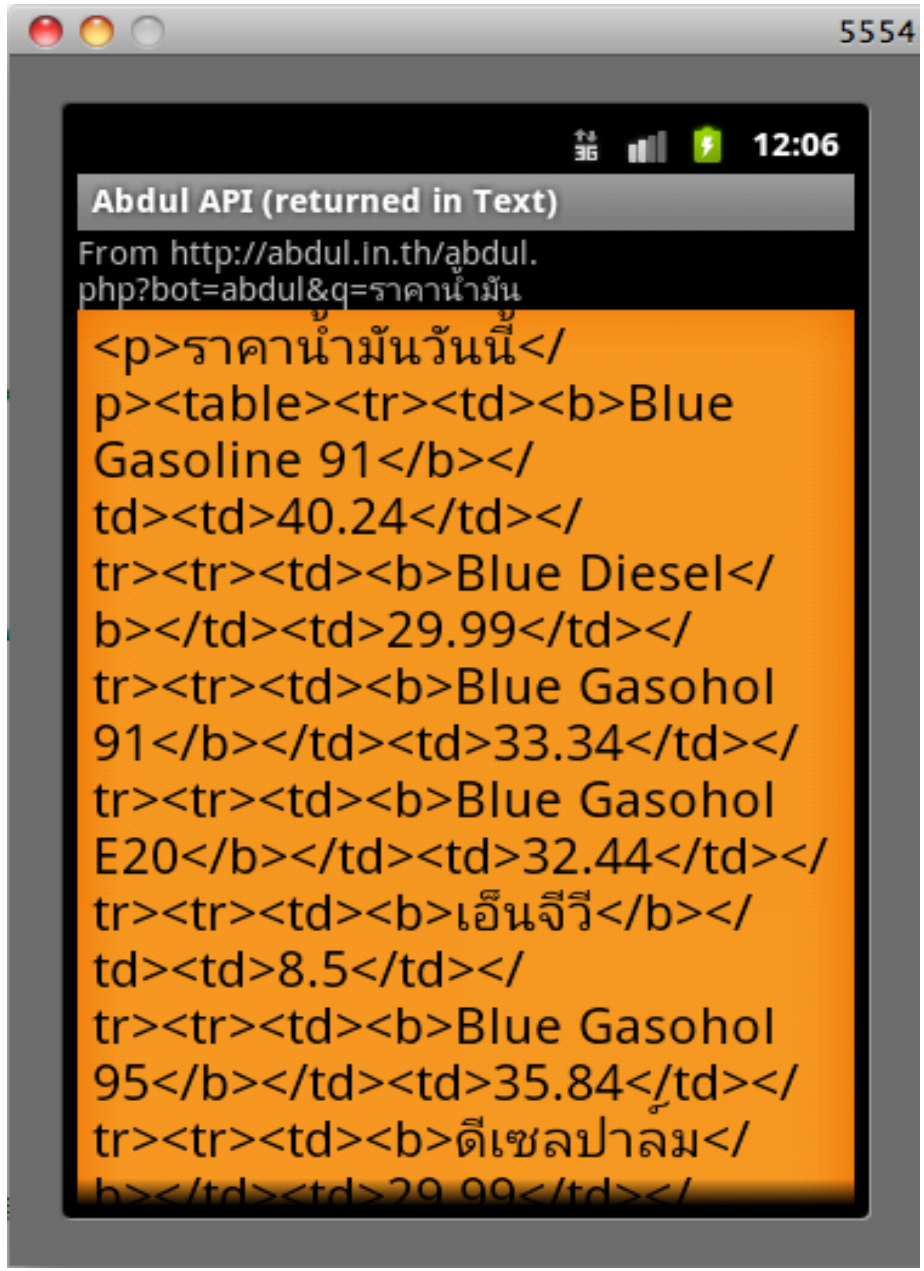
Getting Text Using URL and URLConnection

```
URL url = new URL(urlName);
URLConnection connection;
connection = url.openConnection();
HttpURLConnection httpConnection =
(HttpURLConnection) connection;
int responseCode = httpConnection.getResponseCode();
if (responseCode == HttpURLConnection.HTTP_OK) {
InputStream in = httpConnection.getInputStream();
String result = convertStreamToString(in);
items.add(result);
}
```

convertStreamToString

```
public String convertStreamToString(InputStream is)
    throws IOException {
if (is != null) {
    Writer writer = new StringWriter();
    char[] buffer = new char[1024];
    try {
        Reader reader = new BufferedReader(
                new InputStreamReader(is, "UTF-8"));
        int n;
        while ((n = reader.read(buffer)) != -1) {
            writer.write(buffer, 0, n);
        }
    } finally {
        is.close();
    }
    return writer.toString();
} else {
    return "";
}
}
```

Text Result of the Given URL



Getting Data from HttpGet and HttpClient

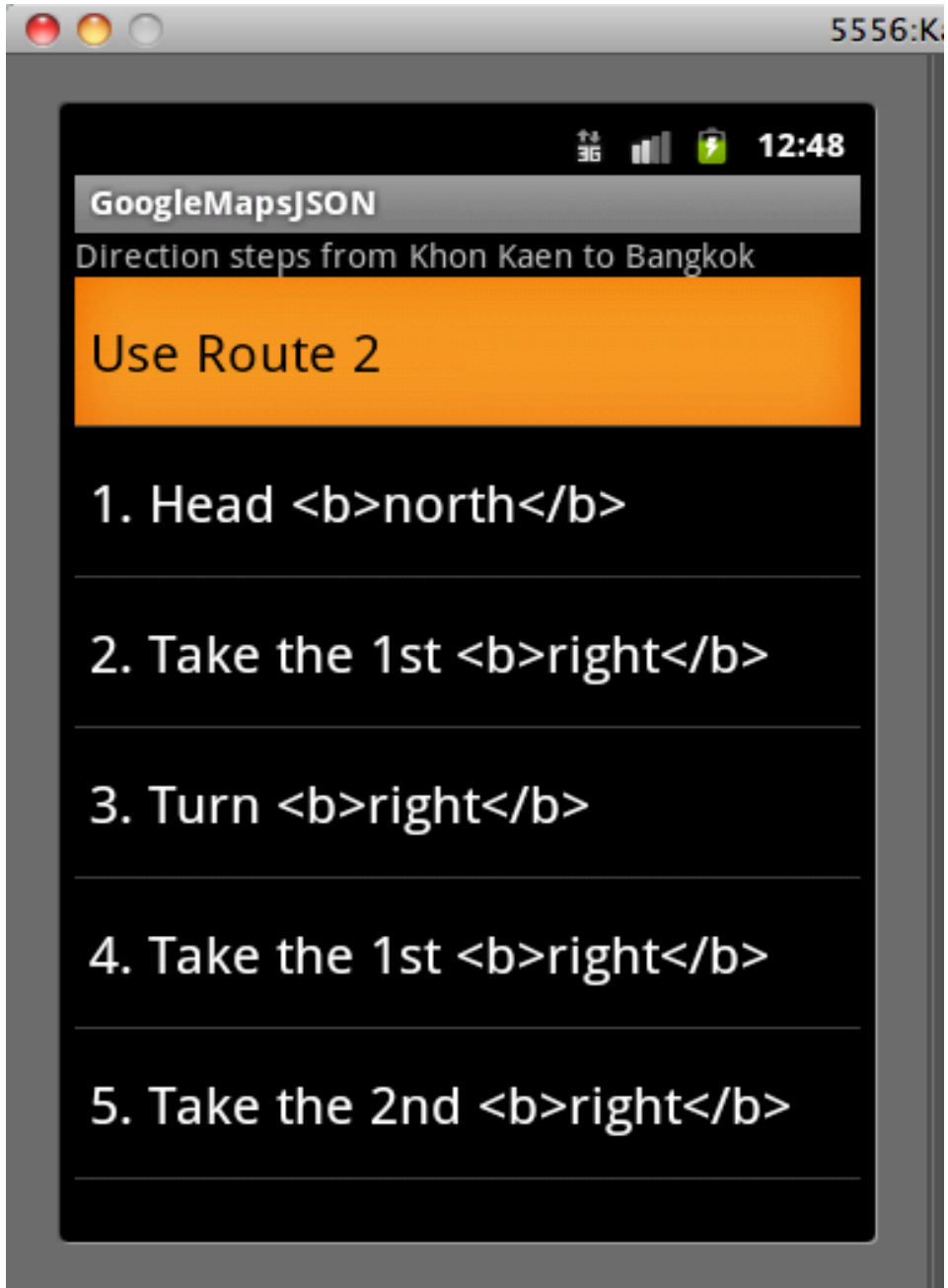
```
 String url = "http://maps.googleapis.
com/maps/api/directions/json?origin=Khon%
20Kaen&destination=Bangkok&sensor=false";
        HttpGet httpGet = new HttpGet(url);
HttpClient client = new DefaultHttpClient();
HttpResponse response;
StringBuilder stringBuilder = new StringBuilder();
try {
response = client.execute(httpGet);
HttpEntity entity = response.getEntity();
```

Converting from InputStream to JSONObject

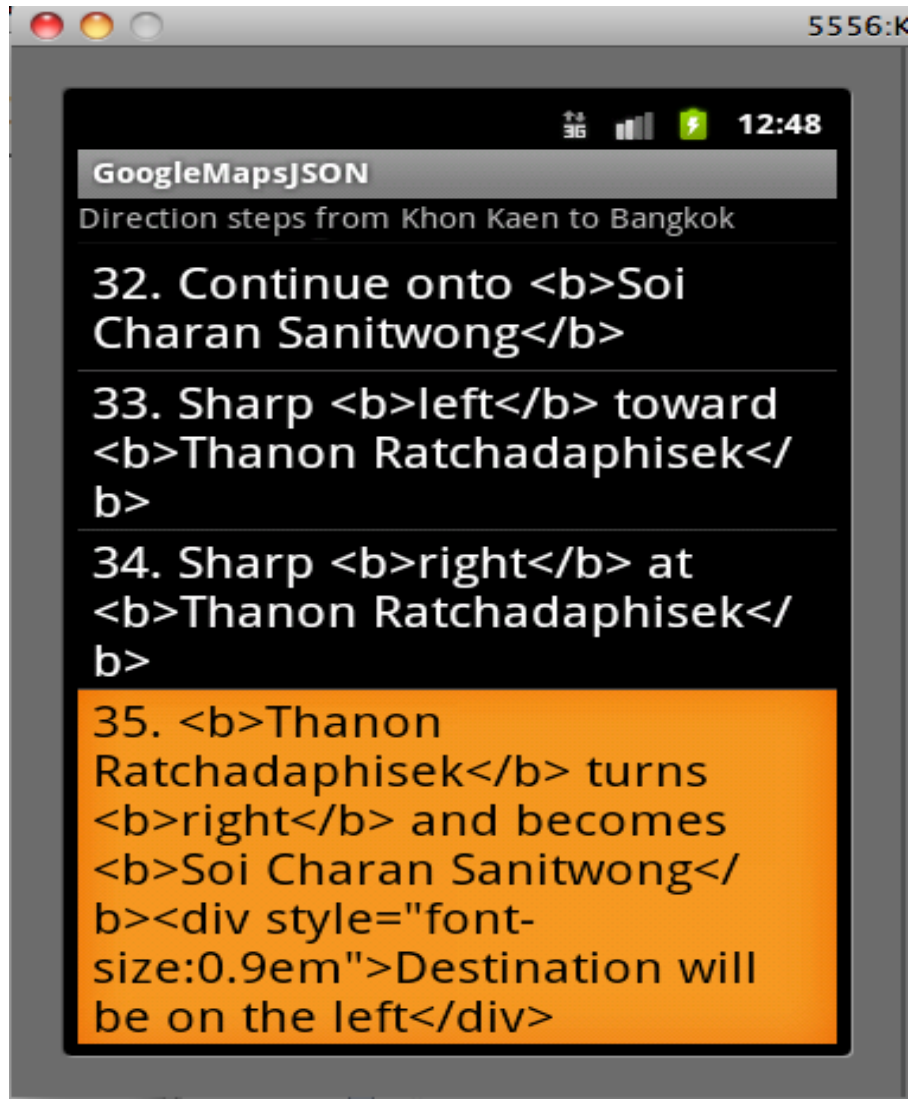
```
InputStream stream = entity.getContent();
int b;
while ((b = stream.read()) != -1) {
stringBuilder.append((char) b);
}
} catch (ClientProtocolException e) {
} catch (IOException e) {
}

JSONObject jsonObject = new JSONObject();
try {
jsonObject = new JSONObject(stringBuilder.toString());
} catch (JSONException e) {
// TODO Auto-generated catch block
e.printStackTrace();
}
```

Result from Accessing JSON Data



Result from Accessing JSON Data



References

- Wei-Meng Lee, "Using and Implementing Content Providers in Android", http://www.devx.com/wireless/Article/41133
- Class android.provider.CallLog.Calls http://developer.android.com/reference/android/provider/ CallLog.Calls.html
- Wei-Meng Lee, "Creating and Using Database in Android", http://www.devx.com/wireless/Article/40842
- https://github.com/facebook/facebook-android-sdk/
- http://developers.facebook.com/docs/guides/mobile/