TACT factory
mobile agency

Fournisseur **souple** et **réactif** d'applications mobiles innovantes avec une **méthodologie** projet **rigoureuse**.

Présentation

Mickael Gaillard (Architecte logiciel)
mickael.gaillard@tactfactory.com

Yoan Pintas (Lead Developper)
yoan.pintas@tactfactory.com

Planning session 3

Phase 1 :

~~Software Lexical~~
~~Software Concept~~
~~Android OS/Spec~~
~~Java (Level 2)~~
~~Mon Activity~~
~~Doc & Lien~~

Phase 2 :

~~L'interface Utilisateur~~
~~Les Intentions~~
~~Les Interfaces Avancées~~
~~Les Intentions Avancées~~

Phase 3 :

La Persistance
Thread
L'accès au réseau
Les Frameworks tiers
Question ?

La Persistance (Preference)

Shared object : la persistance objet de votre application (petit quantité)

```
SharedPreferences settings = context.getSharedPreferences("name",
    Context.MODE_WORLD_READABLE);       // Ou Context.MODE_PRIVATE
```

Getter :

```
String value = settings.getString("key", "default value");
```

Setter :

```
SharedPreferences.Editor editor = settings.edit();
editor.putString("key", "value");
editor.commit();
```

La Persistance (SQLite - create)

SQLite engine : www.sqlite.org

- Flat file database (no TCP/IP) !!!

Le schéma de votre base de données : SQLiteOpenHelper

```
public class ApplicationSQLiteOpenHelper extends SQLiteOpenHelper {
    @Override public void onCreate(SQLiteDatabase db) {  // TODO... }
    @Override public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) { // TODO... }
}
```

- **OnCreate :** Création du schéma de la base de données (CREATE TABLE...)
- **OnUpgrade :** Evolution du schéma de la base de donnée (ALTER TABLE...)

La Persistance (SQLite - usage)

Initialisation :

```
ApplicattionSQLiteOpenHelper helper = new ApplicattionSQLiteOpenHelper(
                                                    context, "database", null, version);
SQLiteDatabase db = helper.getWritableDatabase() ;
```

Transaction :

```
db.beginTransaction();                  // BEGIN
try {
    // Save data
    db.setTransactionSuccessful();      // COMMIT
} catch  {
    // Error in between database transaction
} finally {   db.endTransaction();  }   // ROLLBACK
```

La Persistance (SQLite - CRUD)

Create (INSERT INTO):

```
ContentValues item = new ContentValues();
item.put("col", "value")
...
long id = db.insert("table_name", null, item);
```

Read (SELECT) :

```
String[] COL = {"col", "col2", ...} ;                    // Colonnes du SELECT
String[] VALUES = { " %value% " } ;                      // Valeurs de la clause WHERE
Cursor c = db.query("table_name", COL, "col2 LIKE ?" ,VALUES, null,null,null) ;
c.moveToFirst()
do {
    String value = c.getString( col_pos ) ;
} while ( c.moveToNext() );
```

La Persistance (SQLite - CRUD)

Update :

```
ContentValues item = new ContentValues();
item.put("col", "value")
...
String whereClause = "col =? ";                    // Clause WHERE
String[] whereArgs = { 1 };                        // Valeurs de la clause WHERE
int row_count = db.update("table_name", item, whereClause, whereArgs);
```

Delete :

```
String whereClause = "col =? ";
String[] whereArgs = { 1 };
int row_count = db.delete("table_name", whereClause, whereArgs);
```

La Persistance (File System)

Internal storage : (limité & supprimer a l'App)

FileOutputStream fos = **openFileOutput**("file_name.ext", Context.MODE_PRIVATE);

External storage :(partagé)

FileOutputStream fos = new FileOutputStream(new File(**getExternalFilesDir()**, "file_name.ext"));

Cache storage : (supprimer a l'App)

FileOutputStream fos = new FileOutputStream(new File(**getExternalCacheDir()**, "file_name.ext")) ;

Sous Unix (Linux, Mac OS...) le FS est sensible a la "case"
et la racine/séparateur est le '/'.

Vérifier vos permissions !!!

Thread/Runnable

Les Thread java et leurs joies :

Semaphore, Mutex, ...

La solution Android les AsyncTask ! (pour les ListView : AsyncTaskLoader)

```
private class XxxxxxTask extends AsyncTask<Params, Progress, Return> {
    protected void onPreExecute()    { }                                    // run on UI thread
    protected Return doInBackground(Params... params) { // return a T }     // run on a new thread
    protected void onProgressUpdate(Progress... progress)     { }          // run on UI thread
    protected void onPostExecute(Return ret) {  // Make your UI update }    // run on UI thread
}
new XxxxxxTask().execute(param1, param2, params3);
```

http://developer.android.com/guide/components/processes-and-threads.html

AsyncTaskLoader - AsyncTask

L'accès au réseau

D'après la doc :

java.net.*
android.net.*

Depuis 4.0, **il est obligatoire d'exécuter dans un thread différent de l'UI thread !!!**

Bas niveau :

```
Socket soc = new Socket().connect(new InetSocketAddress("192.168.1.1", 80), 700)
```

Haut niveau :

HttpURLConnection & DefaultHttpClient

Vérifier vos permissions !!!

L'accès au réseau (WS)

Appel de Web Service : (REST, SOAP, XML-RPC...)

```
HttpURLConnection urlConn = (HttpURLConnection) url.openConnection();
Try {
        urlConn.setDoOutput(true);
        urlConn.setChunkedStreamingMode(0);

        OutputStream out = new BufferedOutputStream( urlConn.getOutputStream() );
        writeStream(out);

        InputStream in = new BufferedInputStream( urlConn.getInputStream() );
        readStream(in) ;
} finally { urlConn.disconnect(); }
```

- Sample case...

Les Frameworks tiers

Analytics
Google analytics, Flurry...

Géolocalisation
Google Map SDK, OpenStreetMap.

Réseau sociaux
Google+, Facebook, Twitter, Oath...

Payement
Google Billing, PayPal, bitcoin...

Push
Google Cloud Messaging...

Distribution
Google Distribution...

AdMob
Google AdMob Ads SDK, ...

Questions ?

Change log

- Mickael Gaillard 2012 : Initial document