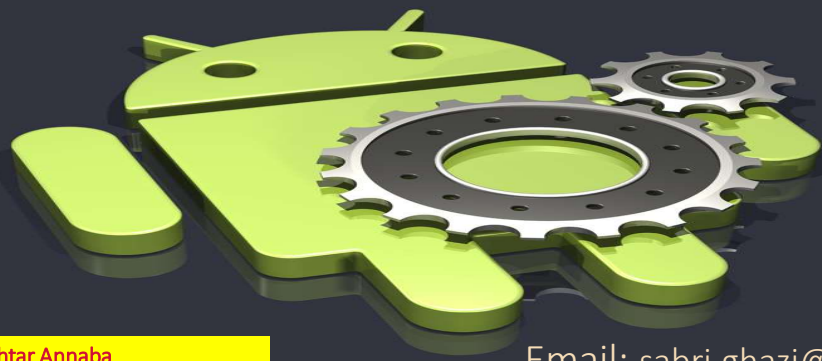


Département d'Informatique, L3, S6
Cours : Développement Mobile

Persistence



Université Badji Mokhtar Annaba

Email: sabri.ghazi@univ-annaba.dz

Plan du cours

Introduction

Préférences
Partagées

Stockage dans un
fichier local
(Interne, Externe)

Bases de Données
& Fournisseurs de
Contenu

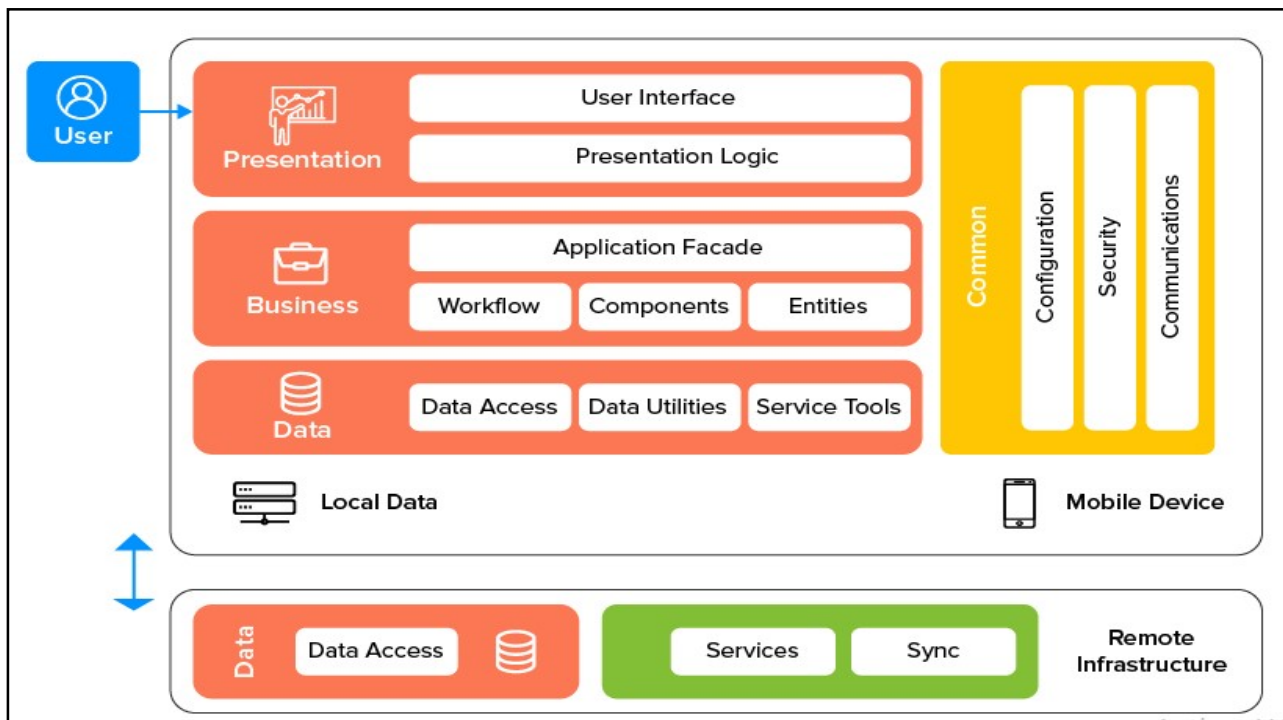
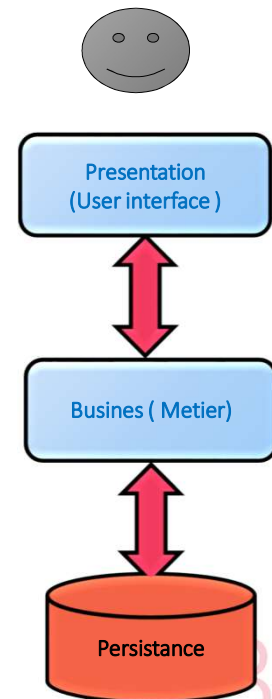
Services REST
REpresentational
State Transfer

2

La persistance c'est quoi ?

Persistence = Sauvegarder les données d'une façon permanente.

- Elle peut se faire selon plusieurs méthodes:
 - **Locales** : Fichier Local, SQLITE
 - **Distantes** : REST, cloud, FireBase etc,



Pourquoi persister ?

5

Accéder à des données métier

Avoir un référentiel local de travail

Faire vivre les données au-delà des applications

Partager les données entre applications

Méthodes de persistance

6

Paramètres locaux

Stockage Interne

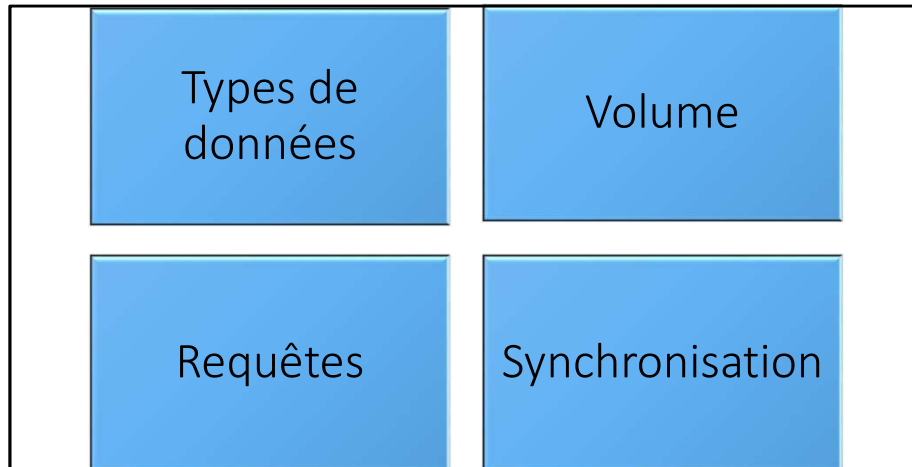
Stockage Externe

Bases de données SQLite

REST

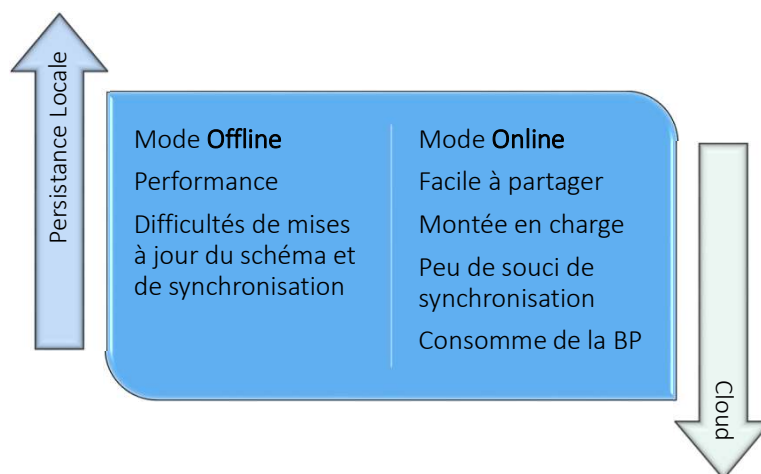
Comment choisir ?

7



Persistence Locale vs Cloud

8



Préférences Partagées

9

Mécanisme de
stockage clé/valeur

Idéal pour stocker
des données
basiques (entiers,
booléens,...)

Convient aux
paramètres
persistés

Préférences Partagées et Préférences

10

Deux types : par activité
(préférences) et
partagées

getPreferences() renvoie
les préférences de
l'activité

getSharedPreferences()
renvoie les préférences
partagées

Les méthodes getXXX
renvoient lisent les
données en fournissant
une clé

Lecture des préférences - Exemple

11

```
SharedPreferences params = activity.getSharedPreferences(PARAMS_NAME, 0);  
String text = params.getString("monparam", "défaut");
```

Modification des préférences

12

La classe
SharedPreferencesEditor
renvoie un éditeur des
préférences

Les méthodes putXXX
changent une
préférence par clé

La méthode « commit »
sauvegarde les
changements

Sauvegarde des préférences - Exemple

13

```
SharedPreferences prms = activity.getSharedPreferences(PARAMS_NAME, 0);
SharedPreferences.Editor editeur = prms.edit();
editeur.putString("monparam", " NouvelleValeur ");
editeur.commit();
```

Stockage dans des fichiers.

- Permet de stocker dans le disque dur de l'appareil.
- Fichier crée par l'application et stocker dans le répertoire de l'application.
- Lorsque l'application est désinstallée les fichiers stockés en internes sont aussi supprimés.
- Les fichiers créés ne sont pas accessibles par les autres applications.

14

SQLite

15

Android permet un accès structuré en combinant SQLite et les fournisseurs de contenu

SQLite est une BDD relationnelle légère et open source supportée nativement par Android

Les fournisseurs de contenu sont une abstraction permettant d'utiliser et de partager des données

SQLite

16

BDD relationnelle et très rapide lancée en 2000

Open Source

Supportée nativement dans Android

Les BDD sont stockées dans le répertoire /data/package

Accès exclusif à l'application qui a créé la BDD

Accès en utilisant la classe SQLiteDatabase


```

public class DataBaseHandler extends SQLiteOpenHelper {
    public static String creationQuery=
        "create table Book( id integer, title char(30), auteur char(30))";
    public static String db_name="library";

    public DataBaseHandler(@Nullable Context context,
        @Nullable String name,
        @Nullable SQLiteDatabase.CursorFactory factory,
        int version) {
        super(context, name, factory, 1);
    }

    @Override
    public void onCreate(SQLiteDatabase db) {
        db.execSQL(creationQuery);
    }
    @Override
    public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {
        db.execSQL("DROP TABLE IF EXISTS library");
        db.execSQL(creationQuery);
    }

    public void ajouterBook(String title, String auteur){
        SQLiteDatabase db = this.getWritableDatabase();
        ContentValues contentValues = new ContentValues();
        contentValues.put("title",title);
        contentValues.put("auteur",auteur);
        db.insert("Book",null,contentValues);
    }
}

```

Ajout d'une ligne dans la table Book

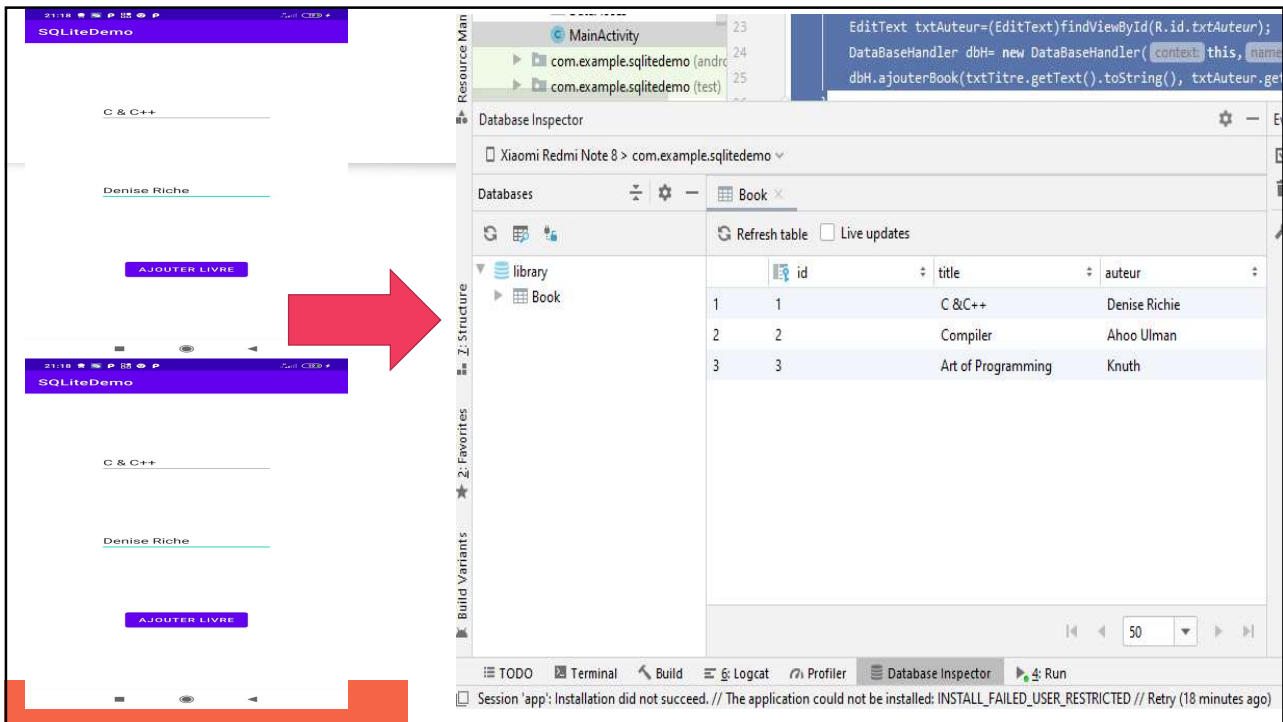
```

public void createBook(View v){
    EditText txtTitre=(EditText)findViewById(R.id.txtTitre);
    EditText txtAuteur=(EditText)findViewById(R.id.txtAuteur);

    DataBaseHandler dbH= new DataBaseHandler(this,"library",null,1);

    dbH.ajouterBook(txtTitre.getText().toString(), txtAuteur.getText().toString());
}

```



Resource Man

```
MainActivity 23
  com.example.sqlitedemo (andre) 24
  com.example.sqlitedemo (test) 25
```

```
EditText txtAuteur=(EditText)findViewById(R.id.txtAuteur);
DatabaseHandler dbH= new DataBaseHandler(this, name);
dbH.ajouterBook(txtTitre.getText().toString(), txtAuteur.get
```

Database Inspector

Xiaomi Redmi Note 8 > com.example.sqlitedemo

Databases

Book

Refresh table Live updates

id	title	auteur
1	C & C++	Denise Richie
2	Compiler	Ahoo Ulman
3	Art of Programming	Knuth

Build Variants

TODO Terminal Build Logcat Profiler Database Inspector Run

Session 'app': Installation did not succeed. // The application could not be installed: INSTALL_FAILED_USER_RESTRICTED // Retry (18 minutes ago)

Lister les lignes d'une table

```
public ArrayList<Book> listAllBooks(){
    ArrayList<Book> ret= new ArrayList<Book>();
    SQLiteDatabase db = this.getWritableDatabase();
    Cursor cur=db.rawQuery("SELECT * FROM BOOK", null);
    Book b=null;
    if( cur.moveToFirst()){
        while (cur.moveToNext()){
            b=new Book(cur.getInt(0),
                cur.getString(1),
                cur.getString(2));
            ret.add(b);
        }
    }
    return ret;
}
```

22:33

SQLiteDemo

```
Book(id=2, title='Compiler', auteur='Ahoo Ulman')
Book(id=3, title='Art of Programming', auteur='Knuth')
Book(id=4, title='Title', auteur='auteur')
Book(id=5, title='Title', auteur='auteur')
Book(id=6, title='Title', auteur='auteur')
Book(id=7, title='le petit prince', auteur='sain exebury')
Book(id=8, title='le petit prince', auteur='sain exebury')
Book(id=9, title='le petit prince', auteur='sain exebury')
Book(id=10, title='le petit prince', auteur='sain exebury')
Book(id=11, title='le petit prince', auteur='sain exebury')
Book(id=12, title='le petit prince', auteur='sain exebury')
```

Cours 4 : Persistence

Fournisseurs de contenu

21

Abstraction permettant de publier et de consommer les données en utilisant des url « content://schema »

Permettent à d'autres applications d'accéder aux données (lecture / écriture / requêtes)

Android fournit des FC natifs tels que les contacts, les fichiers multimédia et le calendrier

Valeurs de contenu

22

Un ensemble de clés/valeurs

Facilite l'obtention ou la fourniture de données

La méthode put ajout une clé/valeur

La méthode getAsXXX renvoie une valeur typée en fournissant une clé

Les curseurs

23

Les curseurs sont des pointeurs sur les résultats d'une requête

Un curseur permet de renvoyer les données de la position en cours

Les curseurs peuvent changer de position en utilisant moveToXXX

getCount renvoie le nombre de lignes

getXXX renvoie la valeur d'une colonne en donnant sa position

getColumnName renvoie le nom d'une colonne

Les requêtes

24

SQLiteDatabase supporte deux types de requêtes: structurées et natives

Les méthodes « query » et «.rawQuery »

Les deux méthodes renvoient des curseurs

Modification de données

25

SQLiteDatabase inclut les méthodes « insert », « update » et « delete »

Les valeurs de contenu sont utilisés pour correspondre avec les champs / valeurs

Ouverture / Création de BDD

26

Utiliser SqliteOpen Helper

Abstraction permettant de décider si la BDD doit être créée ou mise à jour

Accès direct

Méthode openOrCreateDatabase

Fournisseurs de contenu

27

Classes héritant de
ContentProvider

Enregistrés dans le
fichier manifest

Permet aux données
d'être utilisées par
d'autres application et
d'autres composants

Doit implémenter
« onCreate »,
« query », « update »,
« delete », « insert » et
« getType »

La méthode « query »
renvoie un curseur

Enregistrement d'un fournisseur de contenu

28

Ajouter une balise
« provider » dans le
fichier manifeste

Une déclaration indique
le type du fournisseur
et son autorité

L'autorité est l'URL
d'accès aux données

La classe fournisseur
doit définir une
constante publique
« CONTENT_URI » qui
est l'URL d'accès

Exemple

29

Balise Provider

```
<provider android:name=".MonFournisseur"  
android:authorities="dz.esi.prototype"/>
```

Constante

```
public static final Uri CONTENT_URI =  
Uri.parse("content://dz.esi.prototype/elements");
```

Matching d'urls

30

Principe

Utiliser les URL pour retrouver des données

Configuration

Utiliser la classe UriMatcher

Exemple1 : renvoyer tous les éléments

```
content://dz.esi.prototype/elements
```

Exemple 2 : renvoyer un seul élément

```
content://dz.esi.prototype/elements/25
```

Configuration de Matching : exemple

31

```
// constantes de matching
private static final int TOUS = 1;
private static final int UN_SEUL = 2;

static {
    uriMatcher = new UriMatcher(UriMatcher.NO_MATCH);
    uriMatcher.addURI("dz.esi.prototype", "elements", TOUS);
    uriMatcher.addURI("dz.esi.prototype", "elements/#", UN_SEUL);
}
```

Utilisation des fournisseurs de contenu

32

L'accès à un fournisseur de contenu se fait en utilisant la classe « ContentResolver »

La classe utilise des méthodes telles que « query » ou « insert » conjointement à une URI pour accéder au fournisseur correspondant

On obtient une instance de « ContentResolver » en appelant « getContentResolver »

Utilisation des fournisseurs de contenu natifs

33

Android fournit un ensemble de FC natifs (par exemple, pour consulter les contacts)

L'accès à ces FC se fait également en utilisant `contentResolver`

La classe « `ContactsContract` » définit un ensemble de constantes permettant d'utiliser le FC

Consultation des contacts - Exemple

34

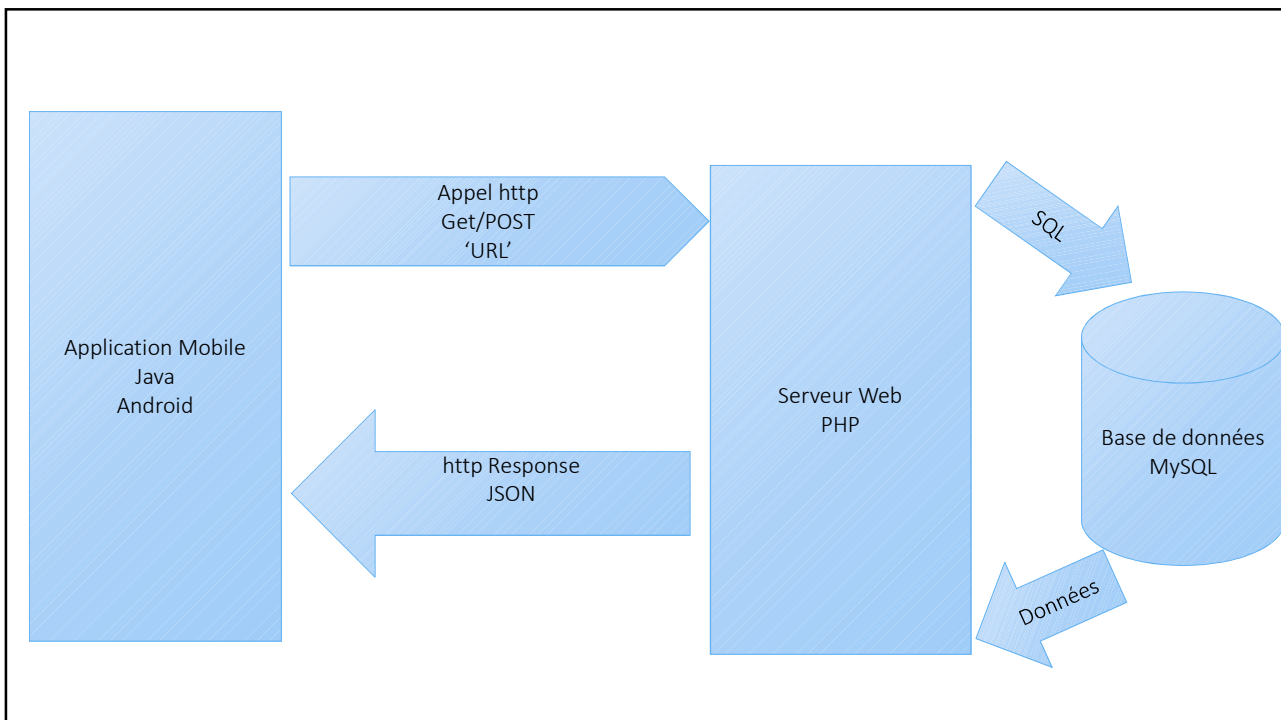
```
public List<Contact> getContacts() {
    List<Contact> contactList = new ArrayList<Contact>();
    String[] projection = new String[]{ContactsContract.Contacts._ID,
ContactsContract.Contacts.DISPLAY_NAME};
    String selection = null;
    Cursor cursor =
getContentResolver().query(ContactsContract.Contacts.CONTENT_URI,
projection, selection, null, null);

    while (cursor.moveToNext()) {
        Contact contact = getContact(cursor);
        contactList.add(contact);
    }

    cursor.close();
    return contactList;
}
```

Section 6 : Services REST

35



Services REST

37



Processus

38



Implémentation de client REST

39

Client HTTP

- DefaultHttpClient
- HttpURLConnection

Lecture JSON

- JSONObject
- JSONArray

Tâches asynchrones

- AsyncTask

Exemple code

```
public class EtudiantReader extends AsyncTask<String, Void, Etudiant> {
    private Exception error = null;
    private IEtudiantHolder holder;

    public EtudiantReader(IEtudiantHolder holder)
    {
        this.holder = holder;
    }

    @Override
    protected Etudiant doInBackground(String... params) {
        try {
            String matricule = params[0];
            DefaultHttpClient httpClient = new DefaultHttpClient();
            String url = "http://etudiantsapi.azurewebsites.net/api/etudiants/" +
            matricule;

```

40

```

        HttpGet get = new HttpGet(url);
        HttpResponse httpResponse = httpClient.execute(get);
        HttpEntity httpEntity = httpResponse.getEntity();
        String response = EntityUtils.toString(httpEntity);
        JSONObject jsonObj = new JSONObject(response);
        String nom = jsonObj.getString("Nom");
        String prenom = jsonObj.getString("Prenom");
        return new Etudiant(matricule,nom,prenom);
    }
    catch (Exception e)
    {
        this.error = e;
        return null;
    }
}

```

Exemple 2 Envoie d'une requête http avec Image

```

public class TestMultipartPost extends Activity {
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        try {
            Ion.with(this, "http://monSiteWeb/formulaire.php")
                .setMultipartParameter("Nom", "Ghazi")
                .setMultipartParameter("Prenom", "Sabr")
                .setMultipartFile("datafile",
                    new File(Environment.getExternalStorageDirectory()+"/photo.png"))
                .asString().setCallback(new FutureCallback<String>() {
                    @Override
                    public void onCompleted(Exception e, String result)
                    {
                        System.out.println(result);
                    }
                });
        } catch (Exception e) {
            // Do something about exceptions
            System.out.println("Got exception: " + e);
        }
    }
}

```

- <https://github.com/koush/ion#jars> (ion and androidasync)
- <https://code.google.com/p/google-gson/downloads/list> (gson)