

FORMATION ANDROID



QUI SUIS-JE ?

- Anthony Monteiro
- Indépendant sur Toulouse
- Vidéos de formation :
<https://www.linkedin.com/learning/instructors/anthony-monteiro>
 - Java pour les développeurs Android
 - Google Map
 - Interaction avec les appareils
 - Stockage des données
 - Publier une application
- Site internet : www.amonteiro.fr
- Contact : contact@amonteiro.fr

Présentation et tour de table.

PLAN

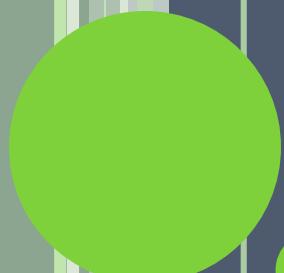
- Premiers pas
 - Présentation / Installation de l'IDE
- Rappels sur Java
- L'univers d'Android
- Architecture d'un projet
 - Les différents types de fichiers et classes
 - Les Activités
- IHM
 - Layouts, composants graphiques
 - Gestion des événements
 - RecyclerView/ListView
 - Communication entre activités

PLAN

- AlertDialog, Toast et Menu
- BroadCast
- Services
- Handler
- AsyncTask
- Les fragments
- Conseils d'architecture
- Ergonomie et User Expériences
- SQLite
 - Avec et sans GreenDAO

PLAN

- Présentation de librairies existantes
 - GraphView
 - Zbar
 - Facebook
 - Scribe
- Notification et GCM
- Web
 - Requête HTTP
 - JSON
 - WebService
- Google Map

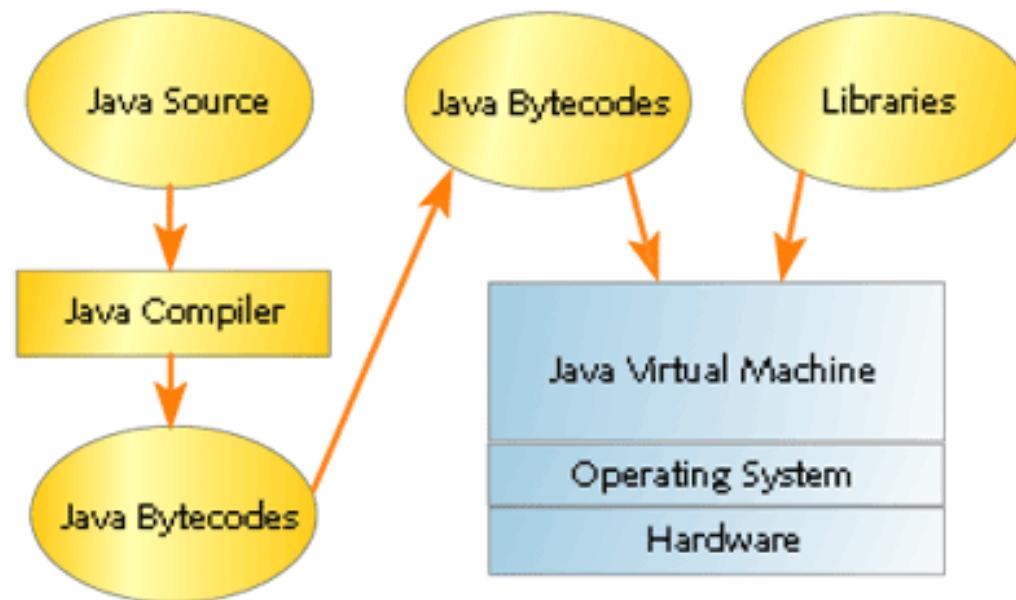


7

RAPPELS JAVA

RAPPELS JAVA

- La JVM



RAPPELS JAVA

```
//Classe
public class Eleve {
    //Attribut
    private String nom;
    //constructeur
    public Eleve(String nom) {
        this.nom = nom;
    }
    //methode
    private void doSomething(){
        ...
    }

    //getter
    public String getNom() {
        return nom;
    }
    //setter
    public void setNom(String nom) {
        this.nom = nom;
    }
}
```

RAPPELS JAVA

```
private void genocide() {  
  
    Eleve eleve = null;  
    //NullPointerException  
    eleve.getNom();  
  
    eleve = new Eleve("Bob"); //Allocation Mémoire pour créer Bob  
    eleve.setNom("John"); //Bob préfère qu'on l'appelle John  
  
    //On réassigne le pointeur, plus personne ne pointe sur l'espace mémoire de John  
    //il va être « garbage collecté ». Adieu John  
    eleve = new Eleve("Candy");  
  
    //Nous avons 2 pointeurs sur Candy  
    Eleve eleve2 = eleve;  
  
    //Nous n'avons plus qu'un pointeur sur Candy  
    eleve = null;  
  
    //Plus personne ne pointe sur Candy. Prépare toi à être recycler  
    eleve2 = null;  
}
```

RAPPELS JAVA

- Conditions

```
boolean condition = true;  
Eleve eleve = new Eleve("bob");  
  
//Condition  
if(eleve == null || condition) {  
    //...  
}  
else if(eleve.getNom().equals("bob")) {  
    //...  
}  
else {  
    //...  
}
```

RAPPELS JAVA

- Les collections

	Utilisation générale
List	ArrayList LinkedList
Set	HashSet TreeSet LinkedHashSet
Map	HashMap TreeMap LinkedHashMap

- SparseArray HashMap<Integer, ?>

RAPPELS JAVA

- **List**

```
//List
ArrayList<Eleve> eleveArrayList = new ArrayList<>();
eleveArrayList.add(eleve);

//Parcours de liste
for (Eleve e : eleveArrayList) {
    e.setNom(e.getNom() + "_eleve");
}

for(int i=0; i<eleveArrayList.size(); i++) {
    Eleve e = eleveArrayList.get(i);
    e.setNom(e.getNom() + "_eleve");
}

//While
int i = eleveArrayList.size() -1;
while(i>=0) {
    Eleve e = eleveArrayList.get(i);
    e.setNom(e.getNom() + "_eleve");
    i--;
}
```

RAPPELS JAVA

- **Tableau**

```
//Déclaration d'un tableau de primitive ou d'objet
float[] monTab;
Eleve[] mesEleves;

//Instanciation du tableau, taille fixe obligatoire
monTab = new float[10];
mesEleves = new Eleve[10];

//Ecrire dans un tableau
monTab[0] = 3;
mesEleves[1] = new Eleve();

//Lire dans un tableau
int i = monTab[0];
Eleve temp = meEleves[1];

//Taille du tableau
int size = monTab.length;
```

RAPPELS JAVA

- **HashMap**

```
//Déclaration
HashMap<Integer, Eleve> mesEleves;

//Instanciation de la Hashmap
mesEleves = new HashMap<Integer,Eleve>();

//Ajout / remplace
mesEleves.put(3, new Eleve());

//Récupération
Eleve eleve = mesEleves.get(3);

//Taille de la hashmap
int size = mesEleves.size();

//Parcours
for(Map.Entry<Integer, Eleve> unEleve : mesEleves.entrySet()) {
    String cle = unEleve.getKey();
    Eleve temp = unEleve.getValue();
}
```

HÉRITAGE : UTILISATION DE LA CLASSE MÈRE

- Pour accéder aux attributs de la classe mère ceux-ci doivent avoir été déclarés avec la visibilité **protected ou public**
- L'équivalent de **this** pour la classe mère est **super**

```
1 public class Personne {  
2     protected String nom;  
3     public Personne(String nom) {  
4         this.nom = nom;  
5     }  
6 }  
1 public class Eleve extends Personne {  
2     protected String section;  
3     public Eleve(String nom, String section) {  
4         super(nom); //Appel du constructeur de Personne  
5         this.section = section;  
6     }  
7 }  
1 public void main(String[] args) {  
2     Eleve e = new Eleve("Bob", "Math"); //ok  
3     Personne p = new Personne("Bob2"); //ok  
4     Personne p2 = new Eleve("Bob3", "Math"); //Ok Eleve est aussi de type personne  
5     // KO, la classe personne n'est pas de type Eleve.  
5     Eleve im1 = new Personne("Bob4");  
6 }
```

- Ordre d'exécution des lignes :

- 1 - ... - 4 - 3 - 4 - 3 - 4 - 5 - 5 - 6 - 6

HÉRITAGE : REDÉFINITION

- Une extension peut redéfinir une méthode de sa classe mère

- Elle doit conserver la même signature

```
1 public class Personne {  
2     protected String nom;  
3     public void print() {  
4         System.out.println("Je m'appelle " + nom);  
5     }  
6 }  
  
1 public class Eleve extends Personne {  
2     protected String section;  
3  
4     @Override  
5     public void print() {  
6         //Si je souhaite appeler la classe de la méthode mère  
7         super.print();  
8         System.out.println(" et je suis en classe de " + section);  
9     }  
10}  
  
1 public static void main(String[] args) {  
2     Personne m1 = new Eleve();  
3     m1.print();  
4     Personne m2 = new Personne();  
5     m2.print();  
6 }
```

- Ordre d'exécution des lignes :

- 1 - 2 - 3 - 5 - 7 - 3 - 4 - 5 - 7 - 9 - 4 - 5 - 3 - 4 - 5 - 6

RAPPELS JAVA

- Interface

```
//Déclaration  
  
public interface OnClickOnEleve{  
    void onClick (Eleve eleve);  
}  
  
//Implémentation  
public classe MyClass implements OnClickOnEleve{  
    public void onClick (Eleve eleve) {  
        //...  
    }  
}
```

GESTION DES EXCEPTIONS

○ Définition

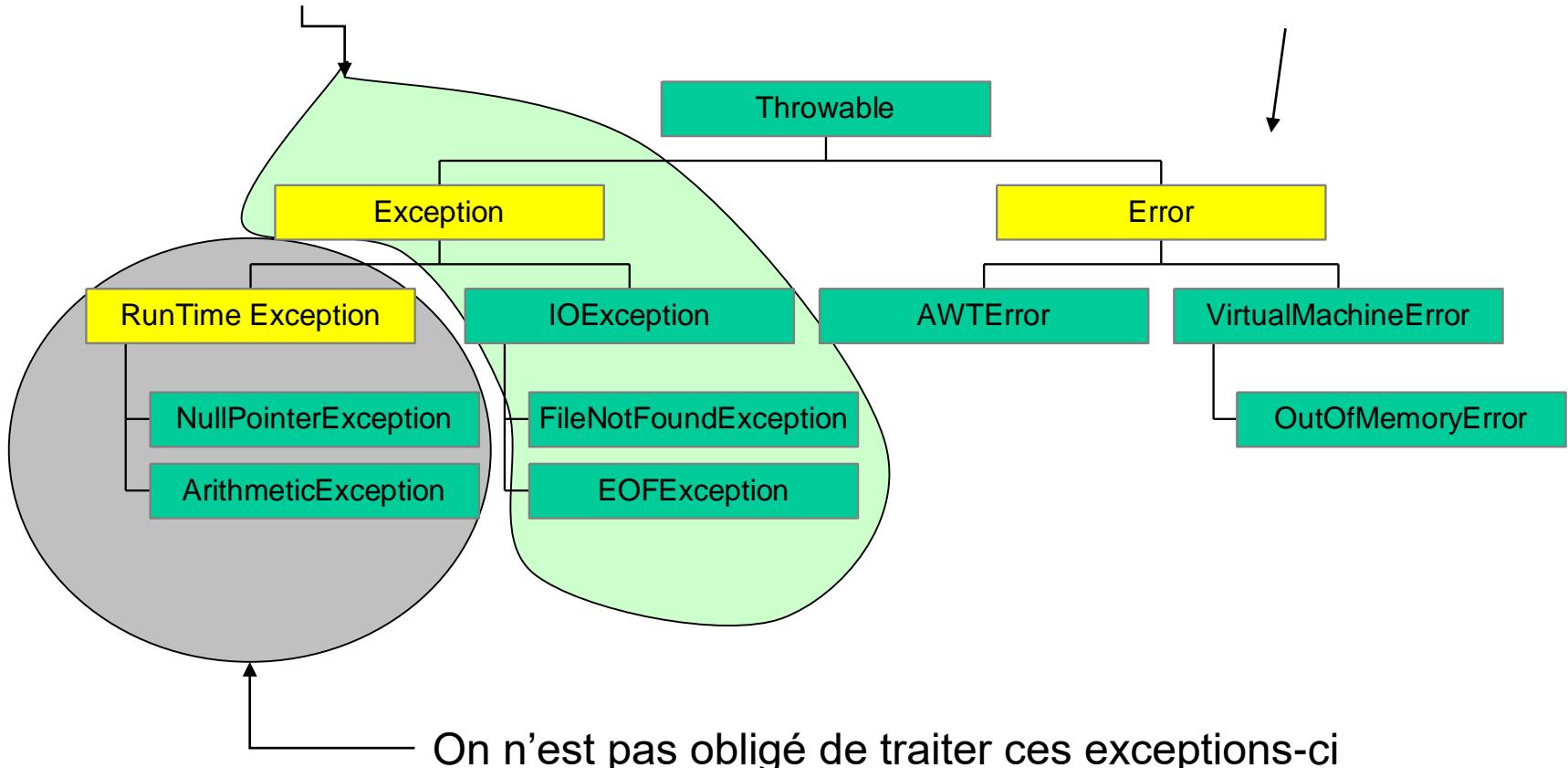
- Une exception est un signal qui indique que quelque chose d'exceptionnel (comme une erreur) s'est produit. Elle interrompt le flot d'exécution normal du programme.

○ A quoi ça sert

- Gérer les erreurs est indispensable : Ariane 5
- Mécanisme simple et lisible
 - Regroupement du code de gestion de l'erreur
 - Possibilité de transmettre l'erreur.

HIÉRARCHIE DES EXCEPTIONS

On doit traiter ces exceptions-ci



GESTION DES EXCEPTIONS

- Créer sa propre classe d'exception

```
public class MyException extends Exception {  
    public MyException(String errorMessage) {  
        super(errorMessage);  
    }  
}
```

- Déclencher une exception

```
if (number > 10) {  
    throw new MyException("Bad number : " + number);  
}
```

GESTION DES EXCEPTIONS

- Propager une exception

```
public void test(int number) throws MyException {  
    if (number > 10) {  
        throw new MyException("Bad number : " + number);  
    }  
}
```

- Traiter une exception

```
public void test2() {  
    try {  
        test(15);  
    }  
    catch (MyException e) {  
        e.printStackTrace();  
    }  
    catch (Exception e) {  
        e.printStackTrace();  
    }  
}
```

GESTION DES EXCEPTIONS

```
public void test2() throws Exception {  
    try {  
        test(5);  
    }  
    catch (MyException e) {  
        e.printStackTrace();  
    }  
    catch (Exception e) {  
        throw e;  
    }  
    finally {  
        // Passera toujours ici avec ou sans Exception  
    }  
}
```

GESTION DES EXCEPTIONS

- Créer sa propre classe d'exception

```
public class MyException extends Exception {  
    public MyException(String errorMessage) {  
        super(errorMessage);  
    }  
  
    //un 2eme constructeur pour transmettre la StackTrace  
    public MyException(String message, Throwable e){  
        super(message, e);  
    }  
}
```

EXERCICE

- Créez votre classe InvalidNumberException
- Créez une méthode « generateException » levant cette exception
- Créez une méthode « throwException » appelant generateException mais ne traitant pas l'exception.
- Créez une méthode « catchException » appelant throwEception, catchant l'InvalidNumberException et retournant une Exception.



26

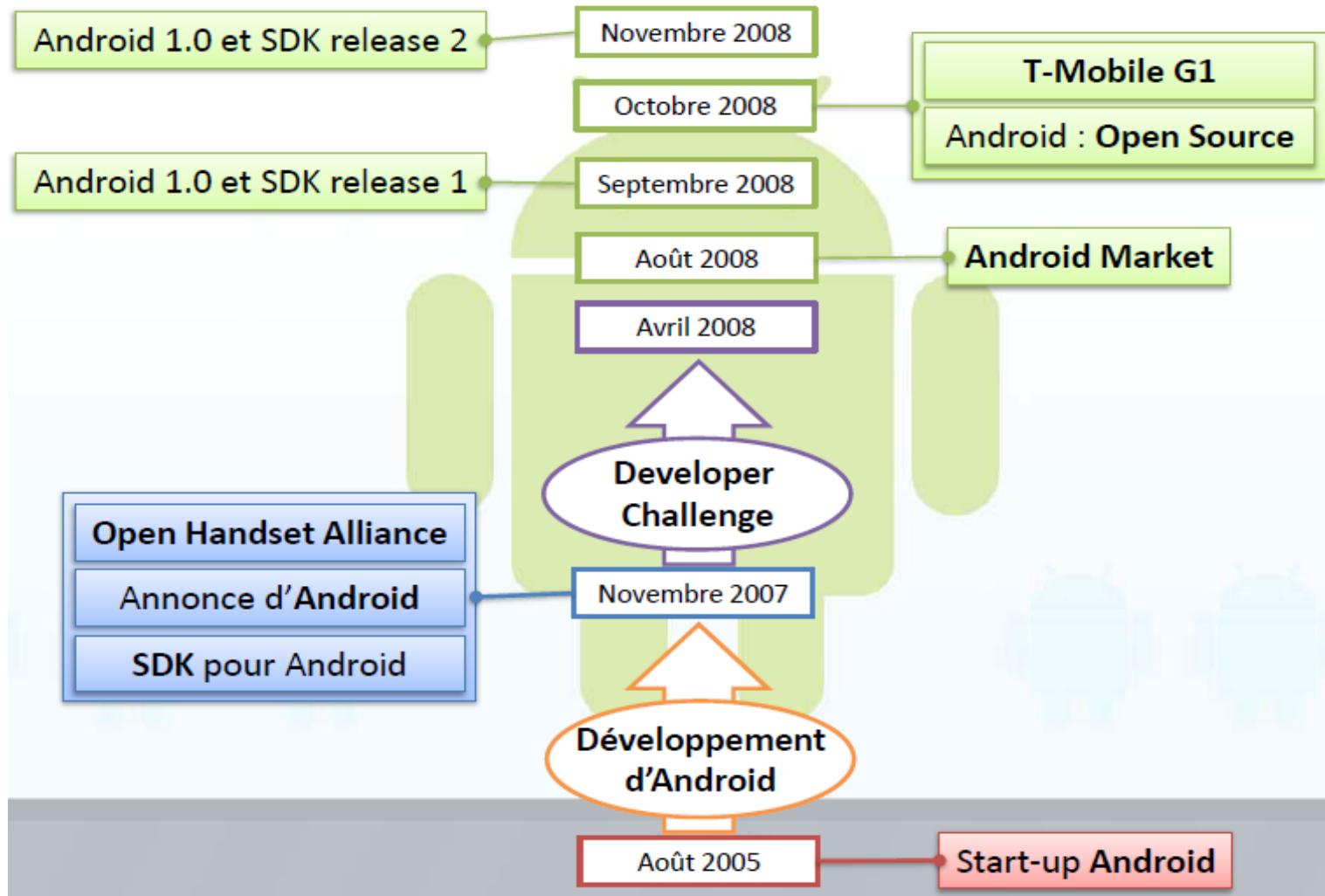
ANDROID

Présentation

INTRODUCTION

- Android est un système d'exploitation pour téléphone portable de nouvelle génération développé par Google. Celui ci met à disposition un kit de développement (SDK) basé sur le langage Java.
- OS complètement ouvert au développeur:
 - Lancer des appels, sms, emails
 - Accès au hardware (gps, appareil photo, wifi)
 - Accès à toutes les fonctionnalités du téléphone
 - => Applications plus riches

EVOLUTION



EVOLUTION

ANDROID



Cupcake



Donut



Eclair



Froyo



Gingerbread



Honeycomb

ANDROID



Ice Cream Sandwich



Jelly Bean



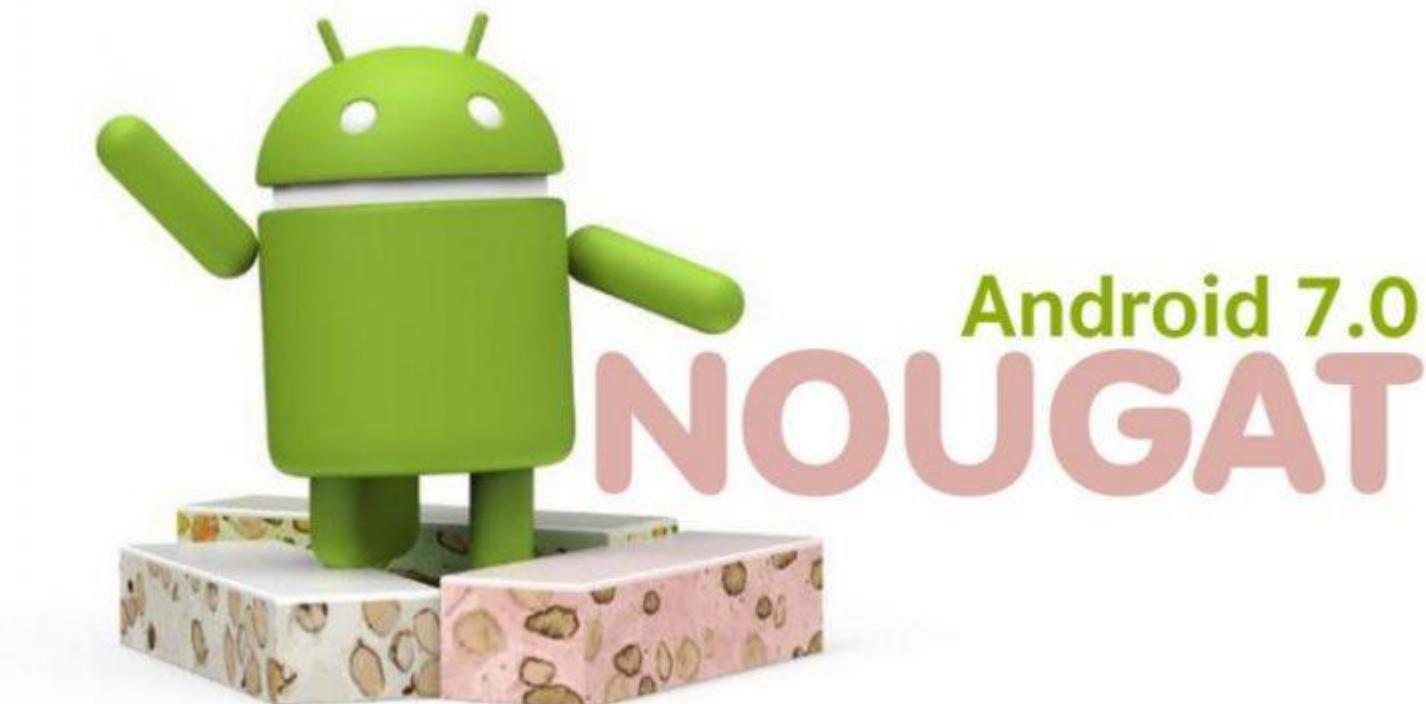
KitKat



Lollipop



Marshmallow

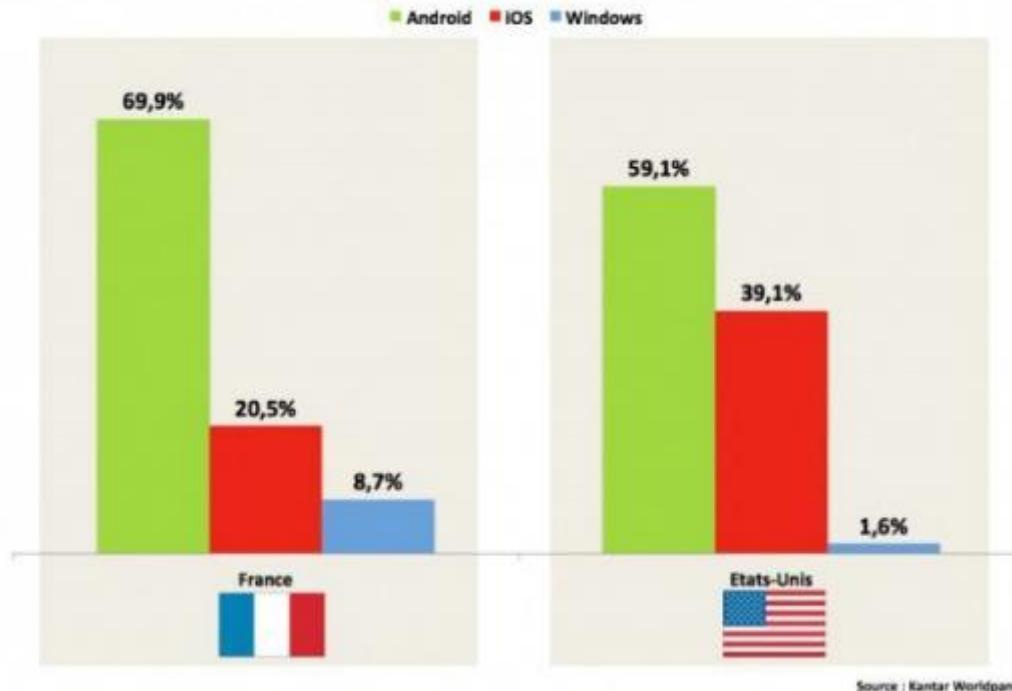


30

Anthony Monteiro

EN CHIFFRE

Part de marché des systèmes d'exploitation pour smartphones
sur la période octobre-décembre 2015



EN CHIFFRE



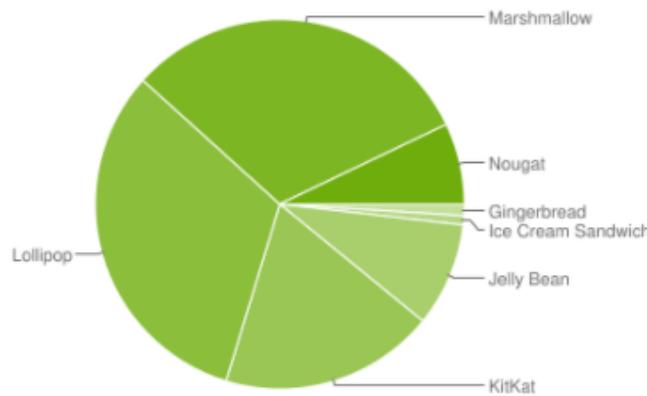
- Supercell (Clash of Clans, Hay Day et Boom Beach)
 - 515M € de bénéfice en 2015
 - 402M€ en publicités
- 2% des applications génèrent 90% des revenus

DÉVELOPPER POUR ANDROID

- Gratuit
- Application distribuable rapidement
 - Par mail
 - Par son propre «store»
 - Via Google Play (payant 25\$)
- Programmation en Java
 - Orientée Objet
 - Possibilité de réutiliser du code métier existant
- API Android / Google Service
 - Pas besoin d'écrire du code bas niveau

RÉPARTITION DES OS PAR VERSION

Version	Codename	API	Distribution
2.3.3 - 2.3.7	Gingerbread	10	1.0%
4.0.3 - 4.0.4	Ice Cream Sandwich	15	0.8%
4.1.x	Jelly Bean	16	3.2%
4.2.x		17	4.6%
4.3		18	1.3%
4.4	KitKat	19	18.8%
5.0	Lollipop	21	8.7%
5.1		22	23.3%
6.0	Marshmallow	23	31.2%
7.0	Nougat	24	6.6%
7.1		25	0.5%



Data collected during a 7-day period ending on May 2, 2017.

Any versions with less than 0.1% distribution are not shown.

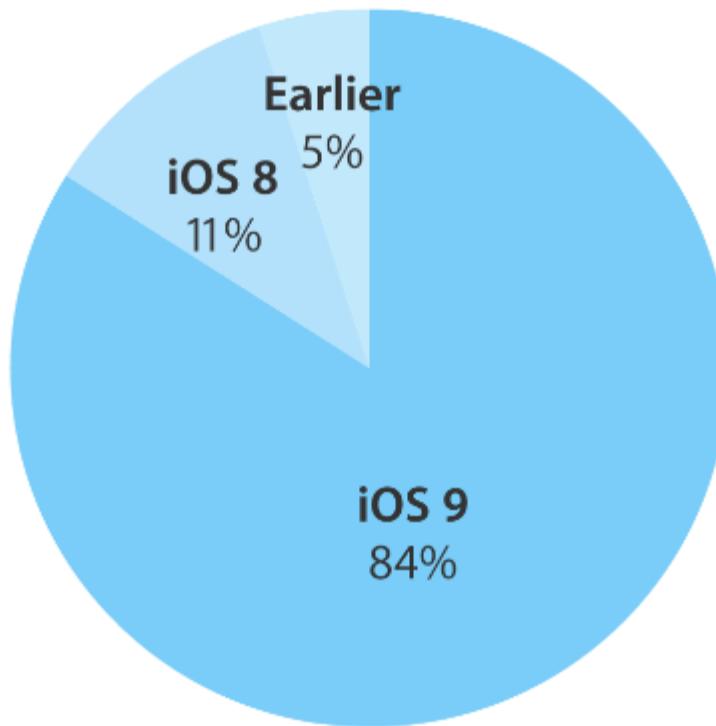
Source : <http://developer.android.com/about/dashboards/index.html>

34

Anthony Monteiro

iOS

84% of devices are using iOS 9.



As measured by the App Store on May 9, 2016.

35

Anthony Monteiro

BONNES PRATIQUES

- Respecter la charte d'Android.
- Respecter les bonnes pratiques du système.
- Android est différent d'iOS.
- Respecter l'utilisateur
 - Ses données
 - Sa confidentialité
- Respecter ses ressources
 - CPU
 - Batterie
 - Mémoire
- Prévenir l'utilisateur

ABUS DE POUVOIR

- MÉFIEZ-VOUS DES APPLICATIONS LAMPE DE POCHE
 - https://play.google.com/store/apps/details?id=goldenshotesttechnologies.brightestflashlight.free&hl=fr_FR
 - <http://www.phonandroid.com/mefiez-vous-applications-lampe-poche-vous-etes-espionnes.html>

ABUS DE POUVOIR

○ MÉFIEZ-VOUS DES APPLICATIONS LAMPE DE POCHE



Privacy Flashlight

SnoopWall

La version 1.1.0 peut accéder aux éléments suivants :

Appareil photo/Micro

- prendre des photos et filmer des vidéos

Autre

- contrôler la lampe de poche

Des fonctionnalités peuvent être automatiquement ajoutées au sein de chaque groupe en cas de mise à jour de l'application "Privacy Flashlight". [En savoir plus](#)

Fermer

envoyer un e-mail à
noopwall.llc@gmail.com

Autori
Affich

38

ABUS DE POUVOIR

○ MÉFIEZ-VOUS DES APPLICATIONS LAMPE DE POCHE



The screenshot shows a mobile application interface for "Brightest Lampe de Poche" by GoldenShores Technologies, LLC. At the top, there is a large image of a white LED lightbulb. Below it, the app's name is displayed in green, followed by the developer's name in smaller text. A scrollable list of permissions is shown, divided into sections with icons:

- afficher les connexions Wi-Fi
- Identifiant de l'appareil et informations relatives aux appels
 - voir l'état et l'identité du téléphone
- Autre
 - désactiver ou modifier la barre d'état
 - Lire les paramètres et les raccourcis de la page d'accueil
 - contrôler la lampe de poche
 - empêcher la mise en veille de l'appareil
 - afficher les connexions réseau
 - bénéficier d'un accès complet au réseau
 - Installer des raccourcis

At the bottom of the screen, a note states: "Des fonctionnalités peuvent être automatiquement ajoutées au sein de chaque groupe en cas de mise à jour de l'application "Brightest Lampe de Poche". [En savoir plus](#)". There is also a "Fermer" button.

ABUS DE POUVOIR

○ MÉFIEZ-VOUS DES APPLICATIONS LAMPE DE POCHE

La version 2.4.2 peut accéder aux éléments suivants :

Données de localisation

- position approximative (réseau)
- position précise (GPS et réseau)

Photos/Contenus multimédias/Fichiers

- Modifier ou supprimer le contenu de la mémoire de stockage USB
- Tester l'accès à la mémoire de stockage protégée

Appareil photo/Micro

- prendre des photos et filmer des vidéos

Informations relatives à la connexion Wi-Fi

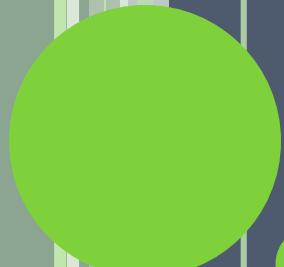
- afficher les connexions Wi-Fi

Identifiant de l'appareil et informations relatives aux appels

- voir l'état et l'identité du téléphone

Autre

- désactiver ou modifier la barre d'état
- Lire les paramètres et les raccourcis de la page d'accueil
- contrôler la lampe de poche
- empêcher la mise en veille de l'appareil
- afficher les connexions réseau
- bénéficier d'un accès complet au réseau
- Installer des raccourcis
- Désinstaller les raccourcis



41

ANDROID

Outils de développement

LE KIT DE DÉVELOPPEMENT

- De quoi avons-nous besoin?

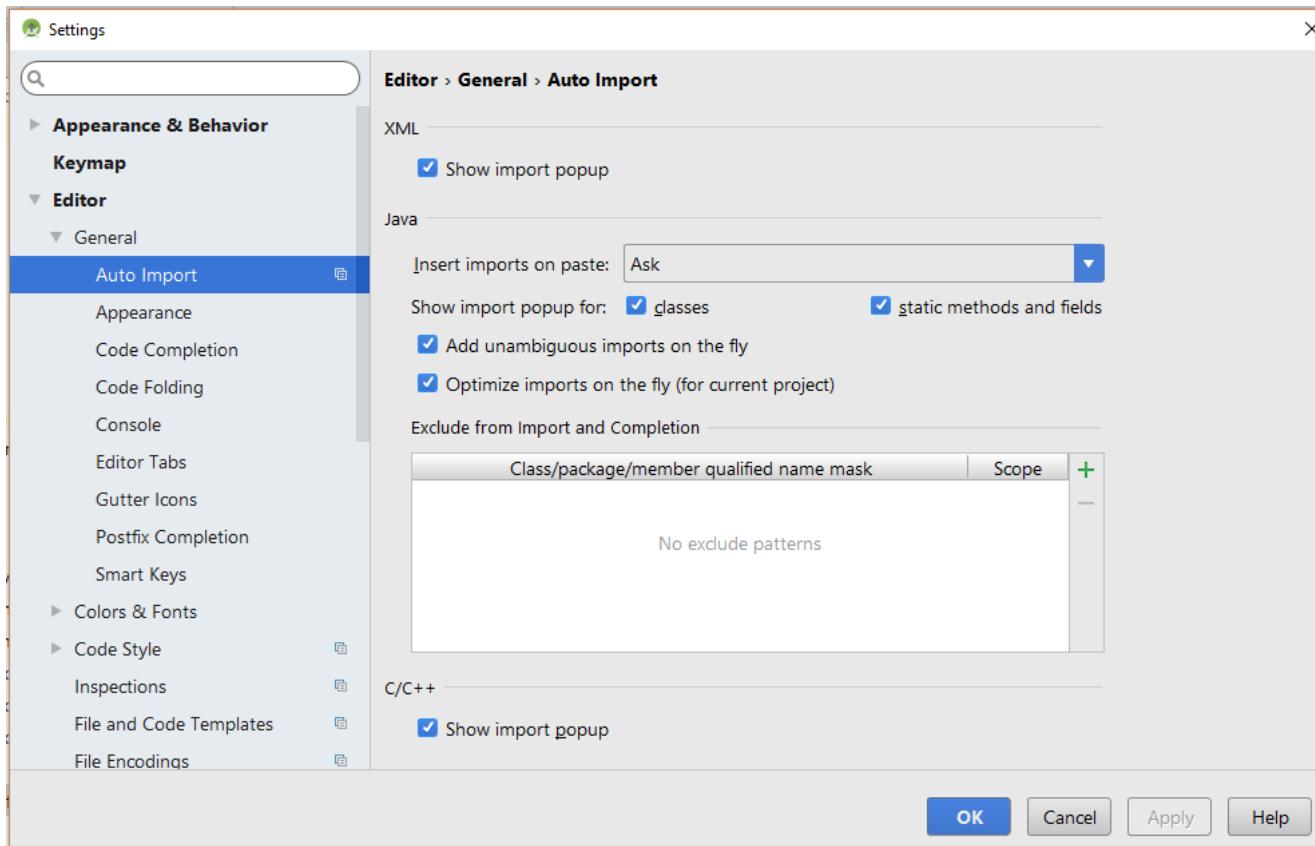
- Un Environnement de développement
 - Android Studio (basé sur IntelliJ)
 - Eclipse (avec plugin Android)
- Le framework Android
 - Téléchargeable via le SDK Manager
- Un Device
 - Réel de test (préférable)
 - Simulateur créé depuis le AVD Manager ou depuis Genymotion

ANDROID STUDIO

- IDE développé par Google (2013)
- Basé sur IntelliJ Community Edition
- Multiplateforme (MacOS, Windows, Linux)
- Installe automatiquement le SDK Manager et le AVD Manager
- **NECESSITE d'avoir le JDK installé !!!**
- <http://developer.android.com/sdk/installing/studio.html>

RÉGLAGES ANDROID STUDIO

○ Activer l'Auto-Import



44

RÉGLAGES ANDROID STUDIO

- Ajouter l'AndroidSettings
 - File -> Import Settings

TESTER SON APPLICATION

- Impossible sur chaque type de téléphone.
- Minimum :
 - Device réel
 - 3 tailles d'écran (petite, normale (nexus 4), tablette).
 - 3 densités (mdpi, hdpi, xhdpi).
 - Avec réseau faible
 - Sur un Samsung
 - Portrait / paysage

UNE FOIS SUR LE PLAYSTORE

- Librairie d'Analytics d'application
 - Google Analytics
 - Capptain
 - CrashLitycs
 - Accra
- Serveur Push
 - java-apns
 - Capptain
- Ils existent des centaines de librairies :
 - <http://android-arsenal.com/free>

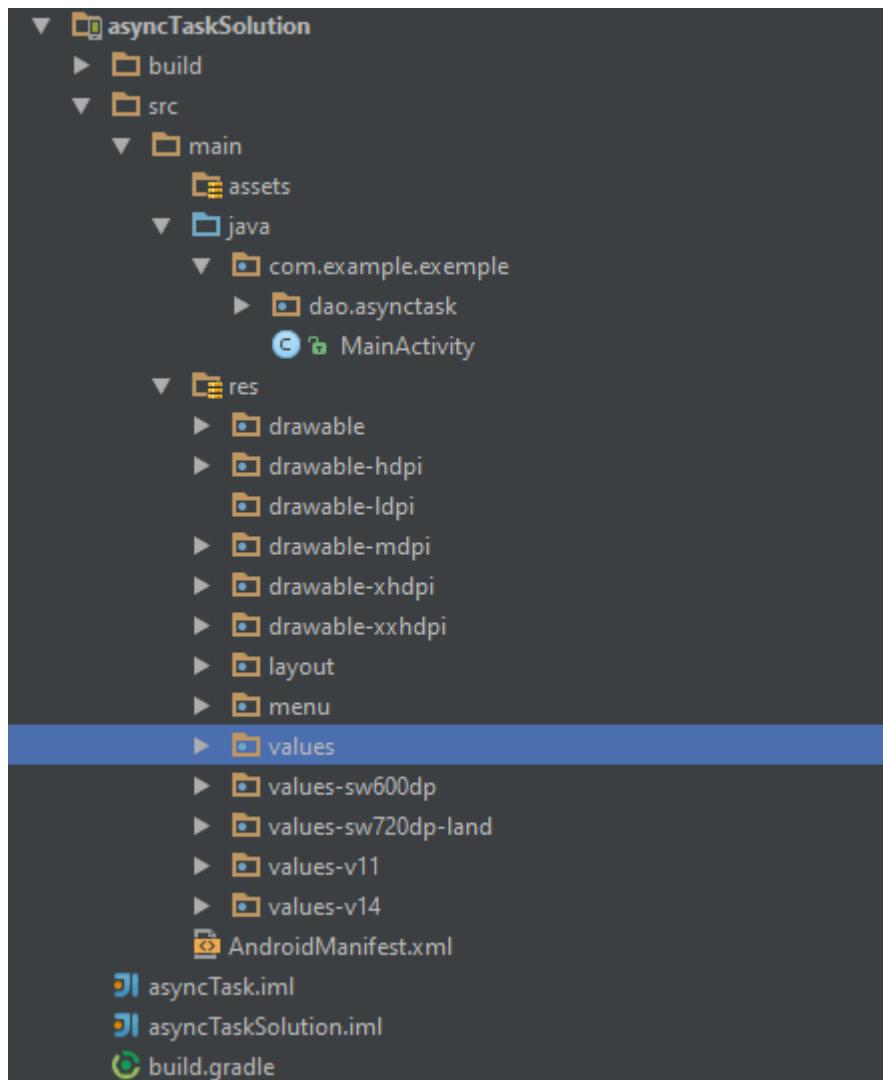


48

ANDROID

Architecture d'une application

QUE CONTIENT MON PROJET ?



LE RÉPERTOIRE GEN

- La classe R (générée)

```
public final class R{  
    public static final class drawable {  
        public static final int icon=0x7f020000;  
    }  
    public static final class layout {  
        public static final int main=0x7f030000;  
    }  
}
```

LE RÉPERTOIRE RES

```
//strings.xml
<string name="today">Aujourd'hui</string>
<string name="annonce">Le petit %1$s est attend à l'accueil par %2$s</string>

//dimens.xml
<dimen name="standard_height">48dp</dimen>

// Retrouver le texte
Resources resources = context.getResources();
String today= resources.getString(R.string.today);
String annonce = resources.getString(R.string.annonce, "Timmy", "sa maman");

//Les chiffres
int standard_height = getResources().getDimensionPixelSize(R.dimen.standard_height);
```

ANDROIDMANIFEST.XML

- Fichier permettant de configurer votre application.
- Plusieurs sortes de configuration:
 - Informations générales (version, packages)
 - Informations concernant l' application : activity, attributs de l' application
 - Permission : pour autoriser l' application à avoir accès à certaines ressources (géolocalisation, internet)
 - Instrumentation : correspond aux classes de test associées.

ANDROID MANIFEST.XML

```
<?xml version="1.0" encoding="utf-8"?>
<manifest package="com.httpexample"
    xmlns:android="http://schemas.android.com/apk/res/android">

    <uses-permission android:name="android.permission.INTERNET"/>

    <application
        android:icon="@drawable/ic_launcher"
        android:label="@string/app_name"
        android:theme="@style/AppTheme">
        <activity
            android:name=".MainActivity"
            >
            <intent-filter>
                <action android:name="android.intent.action.MAIN"/>

                <category android:name="android.intent.category.LAUNCHER"/>
            </intent-filter>
        </activity>
    </application>
```

GRADLE

- Equivalent de maven et Ant

```
    android {  
        compileSdkVersion 20  
        buildToolsVersion "20.0.0"  
  
        defaultConfig {  
            applicationId "com.facebooklogin"  
            minSdkVersion 15  
            targetSdkVersion 19  
            versionCode 1  
            versionName "1.0"  
        }  
    }  
  
dependencies {  
    compile fileTree(include: ['*.jar'], dir: 'libs')  
    compile project(':Simple Facebook')  
    compile files('libs/commons-lang3-3.2.1.jar')  
}
```

TP

- Créez un nouveau projet
- Le lancer dans un émulateur
- Mettre le HelloWorld dans strings.xml
- Changer la langue du helloworld en fonction de la langue du téléphone.



56

IHM

ANDROID

- Construire une interface graphique
- Pas de traitement lourd sur l'UIThread. (Service, AsyncTask...)
- Les modifications d'IHM uniquement sur l'UIThread

CONSTRUIRE SON INTERFACE GRAPHIQUE

- 2 possibilités
 - En code
 - En XML
- Privilégier autant que possible le XML
 - Séparation comportement et visuel (MVC)
 - Outil de rendu graphique en XML
- Utilisation du code pour des changements dynamiques

LAYOUTS

- Les layouts sont des composants permettant de positionner les différents composants graphiques.
- Ce sont des conteneurs d'éléments visuels (ils peuvent contenir d'autre vues et même d'autres layouts) représentés sous forme de fichier xml ou créés directement dans le code.
- Plusieurs types de layouts :
 - LinearLayout
 - RelatifLayout
 - TableLayout
 - ...

LES LAYOUTS ET LE XML

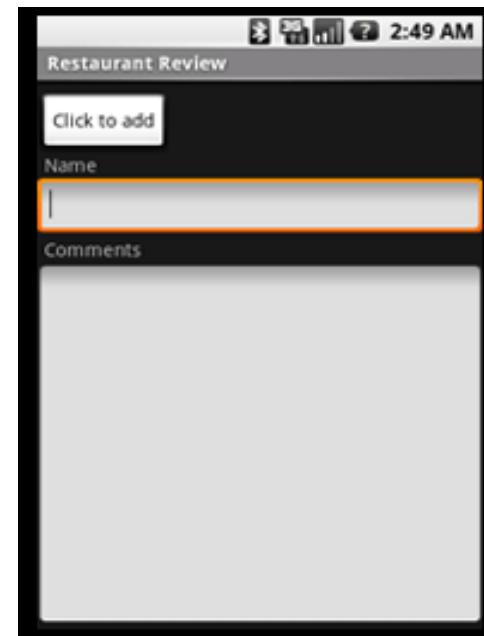
```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    // Attributs du layout
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical" >

    <!-- Contenu du layout -->

</LinearLayout>
```

LINEAR LAYOUT

- Layout le plus utilisé
- Container permettant de placer les éléments en ligne

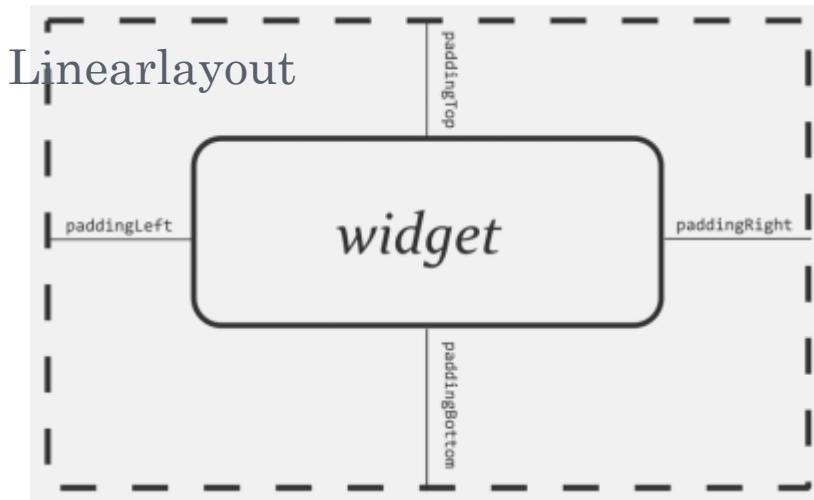


ATTRIBUTS DU LINEAR LAYOUT

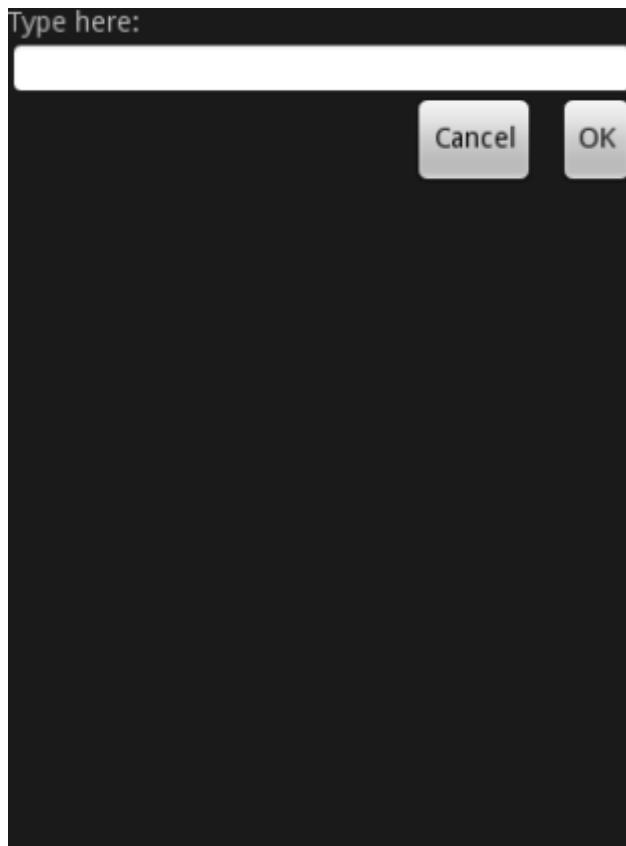
- **Orientation** : indique si le layout ajoute les composants par ligne ou par colonne
- **Width et height** :
 - wrap-content : Uniquement l'espace nécessaire
 - match_parent : Toute la place possible
 - 50p : Une taille précise
- **Gravity** : par défaut, l'alignement se fait de haut en bas
- **Weight** : définit l'espace que prendra la vue associée

LINEAR LAYOUT

- Padding : spécifie l' espace entre le composant et son wrapper



Exemple



TEXTVIEW

- Champ texte non modifiable par l' utilisateur
- En code :
 - new TextView([Context](#) context)
 - utilisation de setText() et getText() pour gérer le contenu
- Code Html accepté
 - <http://daniel-codes.blogspot.fr/2011/04/html-in-textviews.html>

```
<TextView  
    android:layout_width="fill_parent"  
    android:layout_height="wrap_content"/>
```

EDITTEXT

- Champ texte modifiable par l'utilisateur
- En code :
 - new EditText(Context context)
 - utilisation de setText() et getText() pour gérer le contenu
 - setHint() pour afficher un texte grisé quand il n'y a pas de contenu

```
<EditText  
    android:layout_width="fill_parent"  
    android:layout_height="wrap_content"/>
```

BUTTON

- Un simple bouton à appuyer.
 - `setText(var)` : ajouter un texte.

```
<Button  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content"/>
```

Checkbox

- Sélectionne des informations

New CheckBox

```
<CheckBox android:id="@+id/checkbox"
          android:layout_width="wrap_content"
          android:layout_height="wrap_content"
          android:text="check it out" />
```

RADIOBUTTON

○ Boutons à choix exclusifs

 New RadioButton

```
<RadioGroup  
    android:layout_width="fill_parent"  
    android:layout_height="wrap_content"  
    android:orientation="vertical">  
  
    <RadioButton android:id="@+id/radio_red"  
        android:layout_width="wrap_content"  
        android:layout_height="wrap_content"  
        android:text="Red" />  
  
    <RadioButton android:id="@+id/radio_blue"  
        android:layout_width="wrap_content"  
        android:layout_height="wrap_content"  
        android:text="Blue" />  
  
</RadioGroup>
```

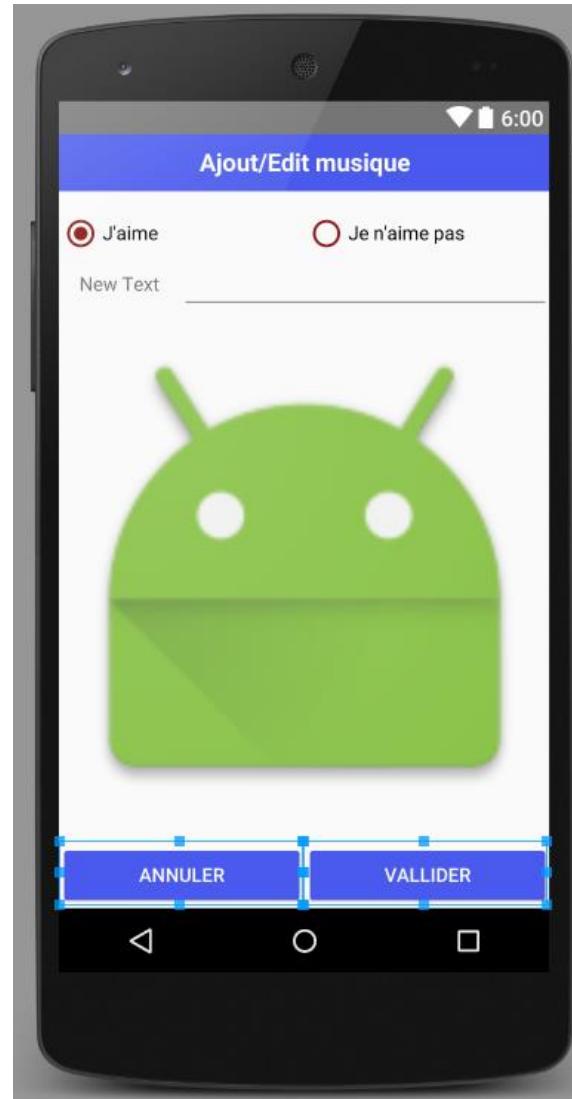
ImageView

- Affiche une image simple
 - setImageResource(int)

```
<ImageView  
    android:layout_width="wrap_content"  
    android:layout_height="match_parent"  
    android:src="@drawable/icon"/>
```

TP : construction d'une vue

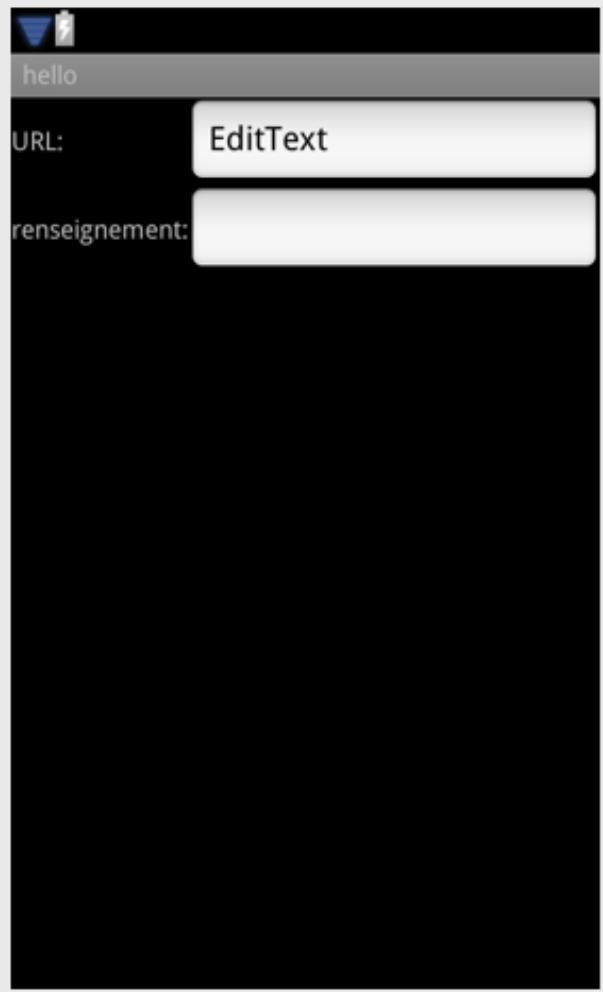
Réaliser et afficher le layout suivant



TABLELAYOUT

- Le fonctionnement du tableLayout ressemble beaucoup à celle des tables en HTML
- Les éléments enfants sont des tableRow où on ajoute nos composants
- On les utilise pour aligner les formulaires

Exemple



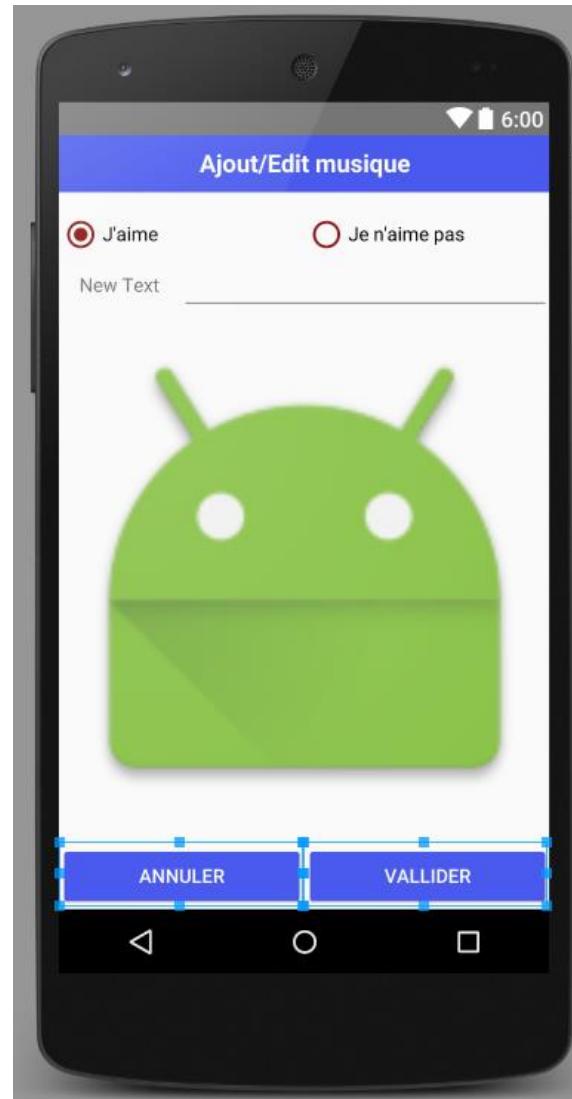
```
<TableLayout  
    xmlns:android="http://schemas.android.com/apk/res/android"  
    android:stretchColumns="1"  
    android:layout_height="fill_parent"  
    android:layout_width="fill_parent">  
    <TableRow android:id="@+id/ligne1">  
        <TextView  
            android:text="URL:"  
            android:id="@+id/textView1"></TextView>  
        <EditText android:layout_height="wrap_content"  
            android:text="EditText"  
            android:layout_width="wrap_content"  
            android:id="@+id/EditText01"></EditText>  
    </TableRow>  
    <TableRow android:id="@+id/ligne2">  
        <TextView android:text="renseignement:"/>  
        <EditText android:id="@+id/entry"  
            android:layout_height="wrap_content"  
            android:layout_width="wrap_content"/>  
    </TableRow>  
</TableLayout>
```

RELATIVE LAYOUT

- Positionner les composants de façon relative entre eux (leurs positions dépendent d'eux-mêmes)
- Position relative à un container
 - android:layout alignParentTop: le composant est aligné par rapport au début du container
 - android:layout alignParentBottom: le composant est aligné par rapport à la fin du container
 -
 - android:layout alignParentLeft: le composant est aligné par rapport à la partie gauche du container
 -
 - android:layout alignParentRight: le composant est aligné par rapport à la partie droite du container

TP : construction d'une vue

Réaliser et afficher le layout suivant à l'aide d'un RelativeLayout



AFFICHER DES MESSAGES EN CONSOLE

- Plusieurs niveaux de
 - Verbose
 - Debug
 - Information
 - Warning
 - Error
- Utilisation des méthodes statiques de la classe Log(
respectivement les fonctions v(),d(),i(),w(), et e())
- Chaque message est associée a un tag facilitant le filtre des messages dans la console

EXEMPLE

- Bonne Pratique :
 - Définir dans le fichier de constante les tags pour être sur d'avoir toujours le même dans la console.
 - Définir une variable permettant de savoir si on est en prod.

```
Log.v("MON_TAG", "activity: main_activity ");
```

LIRE UNE STACKTRACE

The screenshot shows the Android Logcat window. At the top, it displays log entries from an emulator (Emulator Nexus_5X_API_25) running Android 7.1.1, API level 25. The application ID is com.example.mac.app1 (PID 4852). A search bar at the top right includes options for 'Warn', 'Regex', and 'Show only selected application'. Below the log entries is a preview of the Android emulator screen, which shows a notification: 'App1 a cessé de fonctionner.' (App1 stopped working) and a 'Rerunner l'application' (Restart application) button. The main area contains the stack trace:

```
11-03 10:33:22.435 4852-4852/? W/art: Unexpected CPU variant for X86 using defaults: x86_64
11-03 10:33:22.495 4852-4852/? W/System: ClassLoader referenced unknown path: /data/app/com.example.mac.appl-1/lib/x86_64
11-03 10:33:22.522 4852-4852/? W/art: Before Android 4.1, method android.graphics.PorterDuffColorFilter android.support.graphics.drawable.VectorDrawableCompat$ColorFilterImpl.update PorterDuffColorFilter
11-03 10:33:22.583 4852-4852/? E/AndroidRuntime: FATAL EXCEPTION: main
    Process: com.example.mac.appl, PID: 4852
    java.lang.RuntimeException: Unable to start activity ComponentInfo{com.example.mac.appl/com.example.mac.appl.MainActivity}
        at android.app.ActivityThread.performLaunchActivity(ActivityThread.java:2665)
        at android.app.ActivityThread.handleLaunchActivity(ActivityThread.java:2726)
        at android.app.ActivityThread.-wrap12(ActivityThread.java)
        at android.app.ActivityThread$H.handleMessage(ActivityThread.java:1477)
        at android.os.Handler.dispatchMessage(Handler.java:102)
        at android.os.Looper.loop(Looper.java:154)
        at android.app.ActivityThread.main(ActivityThread.java:6119) <1 internal calls>
        at com.android.internal.os.ZygoteInit$MethodAndArgsCaller.run(ZygoteInit.java:886)
        at com.android.internal.os.ZygoteInit.main(ZygoteInit.java:776)
Caused by: java.lang.NullPointerException: Attempt to invoke a virtual method on a null object
        at com.example.mac.appl.MainActivity.onCreate(MainActivity.java:15)
        at android.app.Activity.performCreate(Activity.java:6679)
        at android.app.Instrumentation.callActivityOnCreate(Instrumentation.java:1118)
        at android.app.ActivityThread.performLaunchActivity(ActivityThread.java:2618) <6 more...
```

A red circle highlights the 'Logcat' tab at the bottom left of the window.

Provoquer une Erreur

```
public class MainActivity extends AppCompatActivity {  
  
    @Override  
    protected void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        setContentView(R.layout.activity_main);  
  
        Button bt_valider = null;  
        bt_valider.getText();  
    }  
}
```

TP

- Lire la stackTrace provoquée par l'erreur du slide précédent.
- Trouver la ligne qui provoque l'erreur

Du XML au code

○ Comment

- Mettre du texte dans le label?
- Etre informé du click sur un bouton?

○ Solution:

- Récupérer les instances qui nous intéressent
- Appeler les méthodes des composants
- Ajouter des « écouteurs d'événements »

Du XML au code

○ Dans le XML

- On ajoute un identifiant au composant

```
<Button  
    android:id="@+id/am_bt_next"  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content"/>
```

○ Dans le code de l'activity

- On récupère le composant via son identifiant

```
Button bt_next; //Déclaration en tant qu'attribut  
//Dans le onCreate  
bt_next = (Button) findViewById(R.id.am_bt_next);
```

Du XML au code

```
public class MainActivity extends AppCompatActivity {  
  
    //Déclaration d'un pointeur vers un Button  
    private Button bt_valider;  
  
  
  
    @Override  
    protected void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        //Chargement de l'interface graphique  
        setContentView(R.layout.activity_main);  
  
        //On récupère le Button qui a été créé par le fichier XML  
        bt_valider = (Button) findViewById(R.id.bt_valider);  
    }  
}
```

GESTION DES ÉVÉNEMENTS

- Un écouteur d'événements est appelé en java un « Listener ».
- Certains composants peuvent réagir à des événements
- Ces composants proposent une interface pour écouter leurs événements
 - `public void setOnXListener(OnXListener listener)`

```
//Que l'activité implémente l'interface : (Dans le onCreate)  
//Sur le this en erreur -> Alt+entrée -> implements Listener  
myButton.setOnClickListener(this);
```

QUELQUES APIs

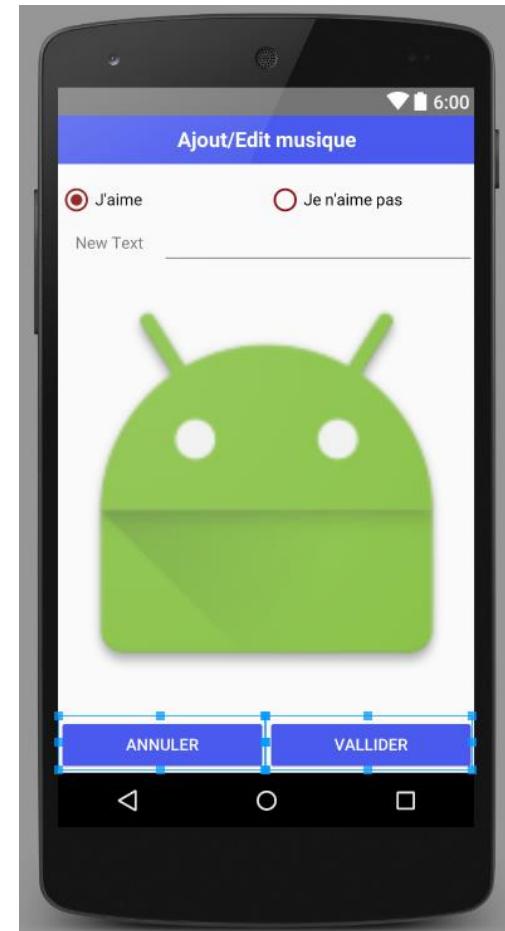
- View(clic)
 - setOnClickListener(...)
 - setOnLongClickListener(...)
- EditText (Chaque touche appuyée)
 - setOnKeyListener(...)
- CheckBox / RadioButton (Coché ou décoché)
 - setOnClickListener(...)
 - setOnCheckedChangeListener(...)

XML TO CODE

- Un site permettant de générer le code à partir du xml
 - <https://www.buzzingandroid.com/tools/android-layout-finder/>

TP : construction d'une vue

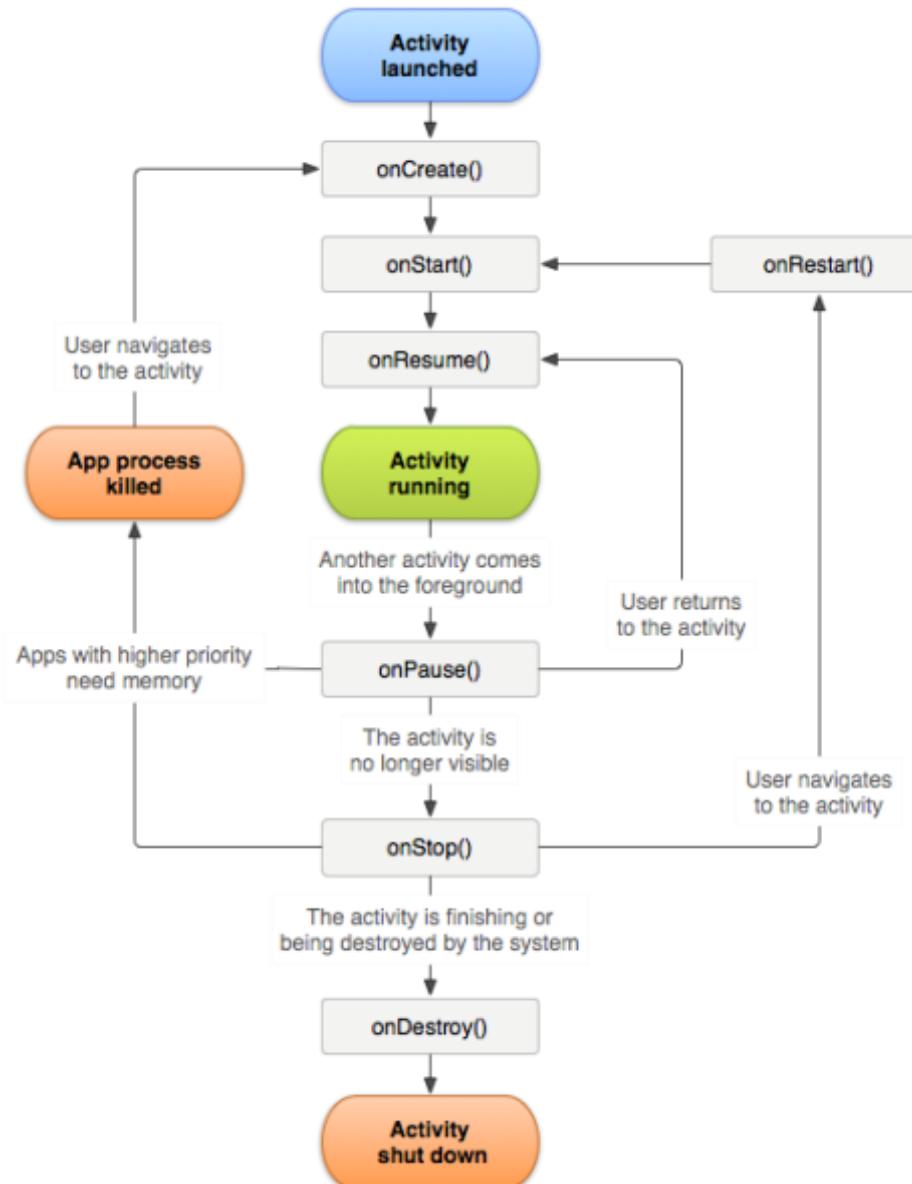
- **Etape 1 : Gérer les évènements**
 - Un clic sur **Valider**, affiche le radioButton sélectionné dans l'editText.
 - Un clic sur **Annuler**, efface l'editText et désélectionne les radioButton.
- **Etape 2 : Gestion d'images**
 - Valider : affiche l'image ‘done’
 - Annuler affiche l'image ‘delete forever’
 - Base d'icônes Material Design :
<https://design.google.com/icons/>
 - Prendre sur fond noir, taille 48dp, Png
- **Etape 3 : changer la couleur de l'image dynamiquement grâce à la méthode ColorFilter**
 - `imageView.setColorFilter(Color.CYAN);`



LES ACTIVITÉS

- Composant de base d'une application
- Représente un écran (une tâche) de l'application
- **DOIT** être déclarée dans le fichier manifest
- Programmation événementielle
 - Pas de main()
 - Répondre à des événements
 - Activité démarrée, click sur le bouton menu...
 - possède un cycle de vie géré par le système
 - reçoit des événements à différents moments de sa vie

LES ACTIVITÉS



- **Active** : Au 1^{er} plan
 - **En pause** : Visible mais elle n'a plus la main, une notification ou une autre activité est active.
 - **Stoppée** : Existe, mais n'est plus visible. L'activité ne peut interagir avec l'utilisateur que par notification.
 - **Morte** : L'activité n'est pas lancée.

CLASSE ACTIVITY

```
public class MainActivity extends Activity {
    private TextView tv;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        tv = (TextView) findViewById(R.id.tv);
    }

    @Override
    protected void onResume() {
        super.onResume();
    }

    @Override
    protected void onPause() {
        super.onPause();
    }

    @Override
    protected void onDestroy() {
        super.onDestroy();
    }
}
```

90

Anthony Monteiro

CYCLE DE VIE

- **onCreate** initialise l' activité (création d' un objet à partir du layout xml, affectation des variables, chargement des données sur les composants)
- **onDestroy** est appelée à la destruction et se charge de fermer toutes les connexions
- **onStop** aura la charge de stopper les threads, animations, et plus généralement les process qui vont agir sur l' interface
- Utilisation de **onStart** pour lancer/relancer ces process
- **onPause/Resume** doit être léger de façon à ne pas ralentir la machine lors du redémarrage (on peut s'en servir pour s'abonner à des broadcasts)

LANCER UNE ACTIVITY

```
//Généralement dans un onClick  
Intent intent = new Intent(this, SecondActivity.class);  
startActivity(intent);  
// Tuer l'activité courante  
finish();
```

- Ajouter une seconde Activity (à la main)
 - SecondActivity.java
 - activity_second.xml
 - Déclaration dans l'AndroidManifest.xml (Sans l'intent filter)
- Créer un bouton sur la 1^{er} activity qui redirige sur la seconde



ANNOTATION AVEC BUTTERKNIFE

```
protected Button bt;  
bt = (Button) findViewById(R.id.bt);
```

Devient...

```
@BindView(R.id.bt)  
protected Button bt;
```

Extraire une Liste de composant

```
@BindViews({R.id.tv1, R.id.tv2, R.id.tv3, R.id.tv4})  
protected List<TextView> tvList;
```

OnClick

```
@OnClick({R.id.bt, R.id.bt2})  
public void onClick(View view) {  
}
```

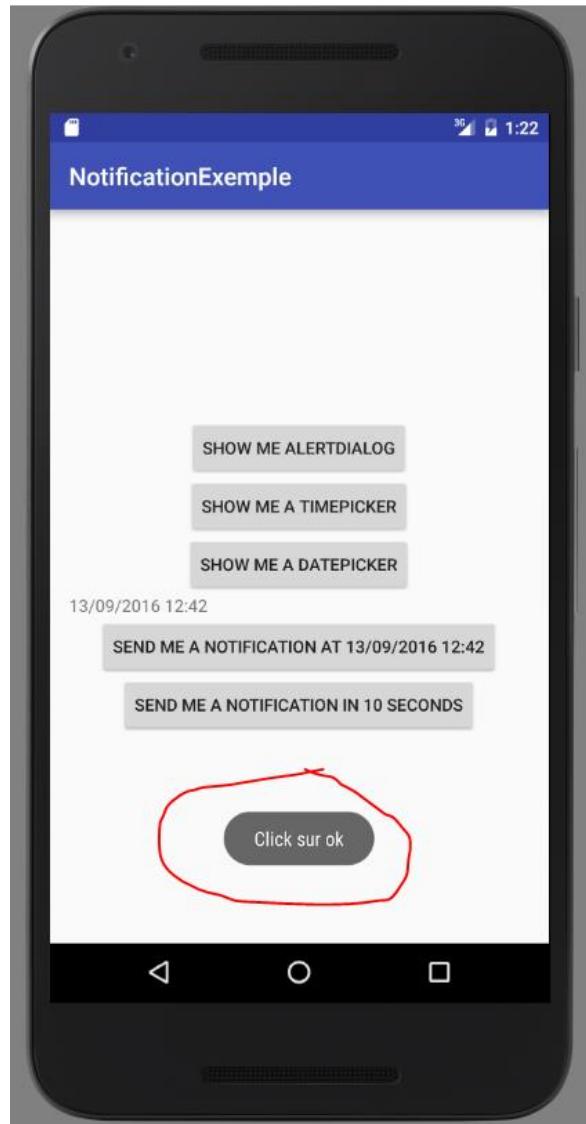
Lien :

<http://jakewharton.github.io/butterknife/>

MENUS ET BOÎTES DE DIALOGUE

95

TOAST



TOAST

- Moyen simple et rapide pour afficher une information à l'utilisateur
- Affichage limité à quelques secondes
- Disparition automatique

```
//Si on est dans une activity, on met this en guise de contexte  
Toast.makeText(context, "Ceci est un Toast", Toast.LENGTH_LONG).show();
```

LES BOÎTES DE DIALOGUES

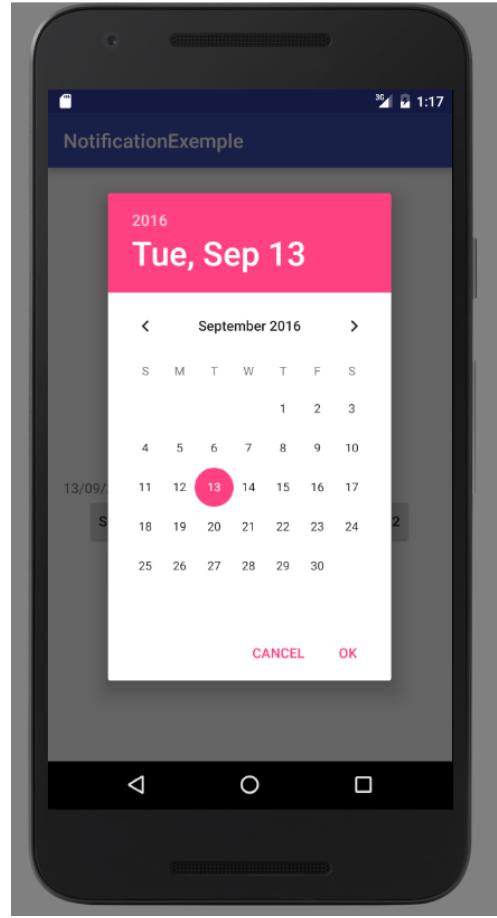
- Les boîtes de dialogue sont des éléments visuels flottants
- Il existe plusieurs sortes de dialogue :
 - alertDialog
 - DatePicker
 - TimePicker..
- Héritent de la classe “Dialog”
- Permet à l’utilisateur de faire des actions rapides comme répondre à des questions, voir les messages...

TIMEPICKER



```
//(Context, callback, heure, minute, 24h format)
//Pour le callback -> Alt+entree -> implémente méthode -> Génère la méthode onTimeSet
TimePickerDialog timePickerDialog = new TimePickerDialog(this, this, 14, 33, true);
timePickerDialog.show();
```

DATEPICKER

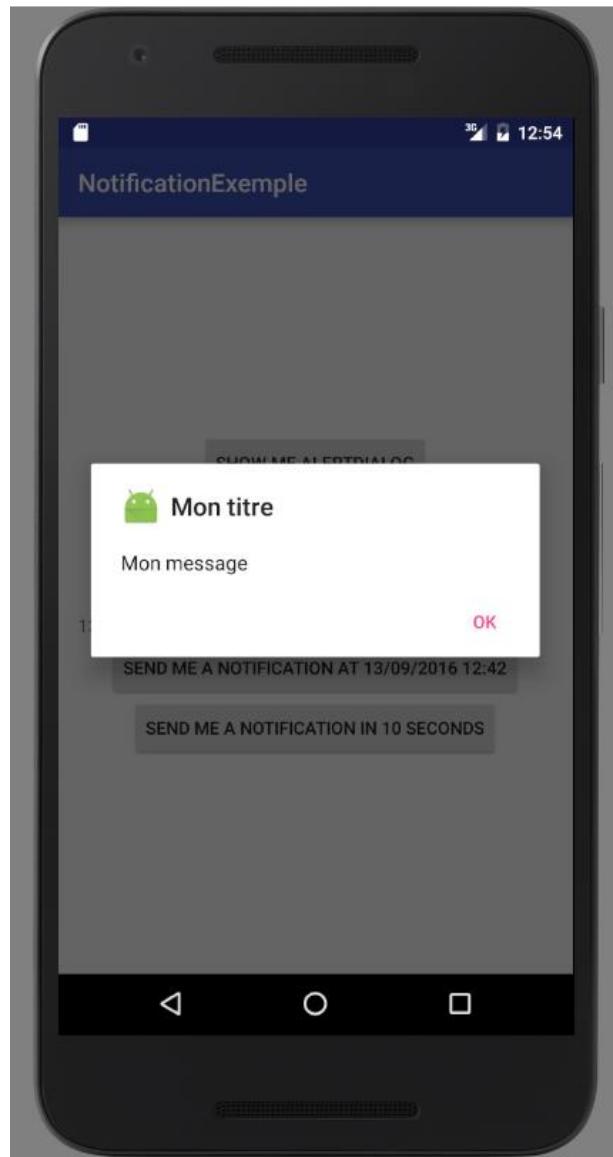


```
//Gestion de la date
Calendar calendar = Calendar.getInstance();
//Création de la fenêtre
//Pour le callback -> Alt+entree -> implémente méthode -> Génère la méthode onTimeSet
DatePickerDialog datePickerDialog = new DatePickerDialog(this, this,
        calendar.get(Calendar.YEAR), calendar.get(Calendar.MONTH),
        calendar.get(Calendar.DAY_OF_MONTH));
//Afficher la fenêtre
datePickerDialog.show();
```

100

Anthony Monteiro

ALERTDIALOG

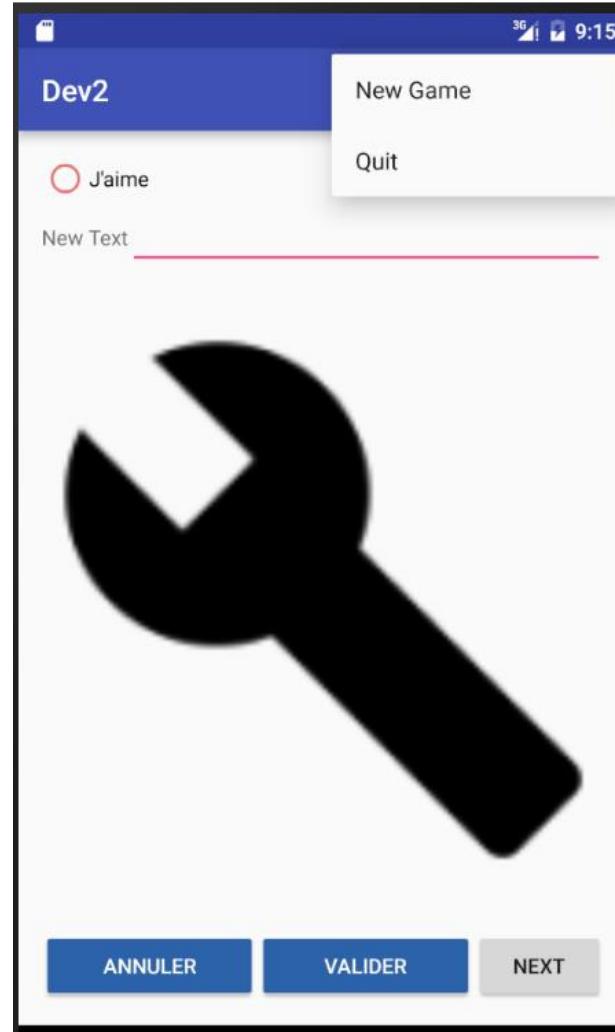


101

ALERTDIALOG

```
//Préparation de la fenêtre
AlertDialog.Builder alertDialogBuilder = new AlertDialog.Builder(this);
//Message
alertDialogBuilder.setMessage("Mon message");
//titre
alertDialogBuilder.setTitle("Mon titre");
//bouton ok
alertDialogBuilder.setPositiveButton("ok",
    new DialogInterface.OnClickListener() {
        public void onClick(DialogInterface dialog, int which) {
            //Affiche un toast apres le click sur le bouton ok
            Toast.makeText(MainActivity.this, "Click sur ok",
                Toast.LENGTH_SHORT).show();
        }
    });
//Icone
alertDialogBuilder.setIcon(R.mipmap.ic_launcher);
//Afficher la fenêtre
alertDialogBuilder.show();
```

MENU



MENU

- Pré construit par le système
- La classe Activity met à disposition 2 méthodes à surcharger

```
//Remplir le menu (Se génère en écrivant le début de « onCreate »)
```

```
@Override  
public boolean onCreateOptionsMenu(Menu menu) {...}
```

```
//Intercepter un click sur le menu
```

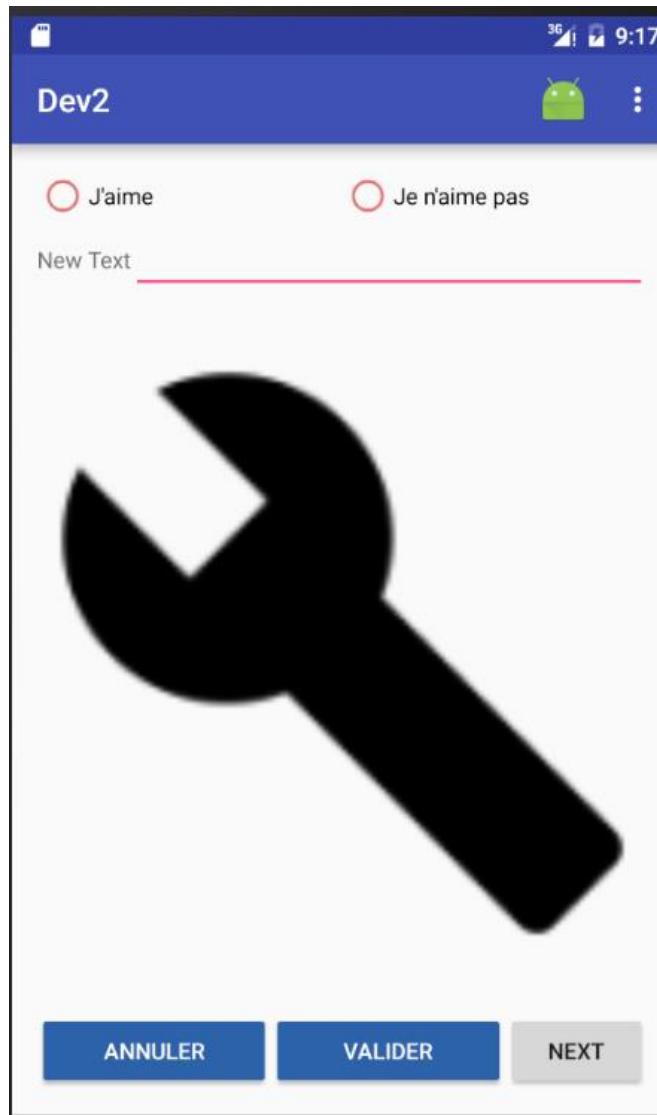
```
@Override  
public boolean onOptionsItemSelected(MenuItem item) {...}
```

Exemple

```
/* Crée le menu, 25 et 26 sont les id pour retrouver les
menuItem */
public boolean onCreateOptionsMenu(Menu menu)
{
    menu.add(0, 25, 0, "New Game");
    menu.add(0, 26, 0, "Quit");
    return super.onCreateOptionsMenu(menu);
}

/* Handles item selections */
public boolean onOptionsItemSelected(MenuItem item)
{
    switch (item.getItemId())
    {
        case 25:
            newGame();
            break;
        case 26:
            quit();
            break;
    }
    return super.onOptionsItemSelected(item);
}
```

MENU AVEC ICÔNE



106

MENUTEM AVEC ICÔNES

- Ajouter une icône à un menultem

```
menu.add(0,25, 0, "New Game").setIcon(R.mipmap.ic_launcher);
```

- L'afficher

```
menu.add(0,25, 0, "New Game").setIcon(R.mipmap.ic_launcher)  
        .setShowAsAction(MenuItem.SHOW_AS_ACTION_ALWAYS);
```

DÉTAILS

- `MenuItem.SHOW_AS_ACTION_ALWAYS` :
 - Nous permet d'afficher l'icone même s'il y a peu de place
- `MenuItem.SHOW_AS_ACTION_IF_ROOM` :
 - Affiche l'icone seulement s'il y a de la place dans la barre
- `MenuItem.SHOW_AS_ACTION_WITH_TEXT` :
 - Affiche le texte à côté de l'icone

MENUITEM AVEC ICÔNES VERSION XML

○ Chargement du menu depuis l'XML

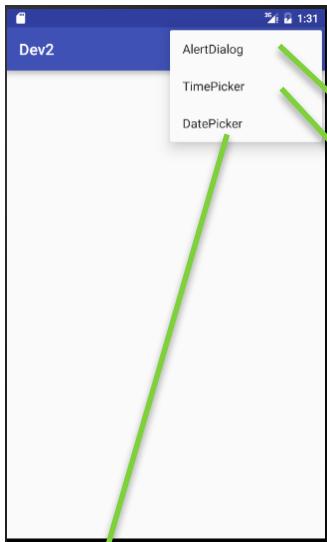
```
@Override  
public boolean onCreateOptionsMenu(Menu menu) {  
    //Version chargement du fichier menu à parti d'un XML  
    getMenuInflater().inflate(R.menu.menu_main_activity, menu);  
    return super.onCreateOptionsMenu(menu);  
}
```

○ Menu_main_activity.xml

```
<menu xmlns:android="http://schemas.android.com/apk/res/android"  
      xmlns:app="http://schemas.android.com/apk/res-auto">  
  
    <item  
        android:id="@+id/menu_add"  
        android:icon="@drawable/ic_action_add_group"  
        android:orderInCategory="100"  
        android:title="@string/st_menu_add"  
        app:showAsAction="ifRoom"/>  
  
</menu>
```

TP MENU – DIALOG - TOAST

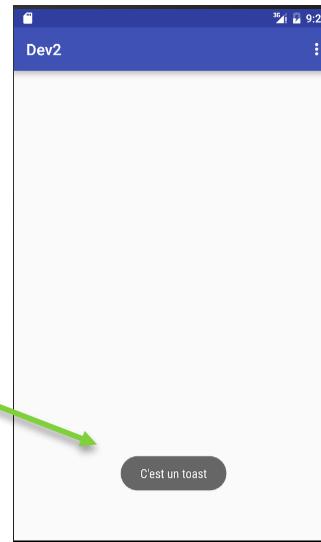
Menu



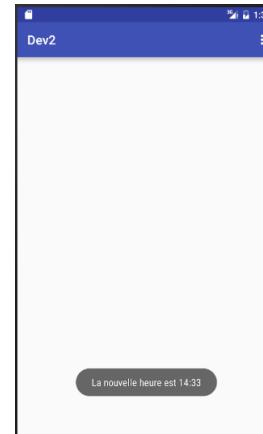
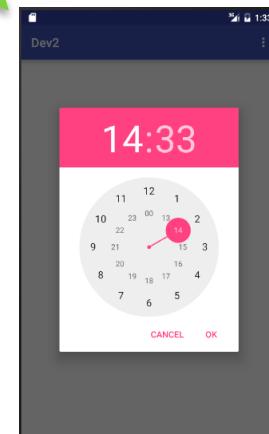
AlertDialog

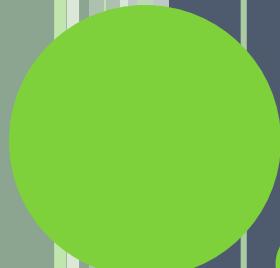


Toast



- Pour les plus rapides faire le TimePicker et le DatePicker





111

BROADCAST

ANDROID

- Répondre à des événements extérieurs

Broadcast receiver

- Le concept des broadcast receiver est simple:

- Réceptionner un **intent** qui broadcaste sur toutes les applications
- Chaque classe broadcast receiver doit être déclarée dans le manifest et associée à des broadcasts
- Une seule méthode **onReceive()**
- Un BroadcastReceiver ne vit que le temps de traiter votre **onReceive()**.

```
public class MyBroadcastReceiver extends BroadcastReceiver {  
    @Override  
    public void onReceive(Context context, Intent intent) {  
        ...  
    }  
}
```

Broadcast receiver

- Active / désactive le mode avion:

```
<manifest ... >

    <application ...>

        //Attention à ne pas mettre le receiver dans l'activity
        <receiver
            android:name="broadcast.MyBroadcastReceiver"
        >
            <intent-filter>
                <action android:name="android.intent.action.AIRPLANE_MODE" />
            </intent-filter>
        </receiver>

    </application>
</manifest>
```

Broadcast receiver

○ Lecture et réception des SMS:

```
<manifest ... >

    //Attention à ne pas mettre les permissions dans l'application
    <uses-permission android:name="android.permission.RECEIVE_SMS" />
    <uses-permission android:name="android.permission.READ_SMS" />

    <application ...>
        //Attention à ne pas mettre le receiver dans l'activity
        <receiver
            android:name="broadcast.MyBroadcastReceiver"
        >
            <intent-filter>
                //Non proposé dans l'auto-complétion.
                <action android:name="android.provider.Telephony.SMS_RECEIVED" />
            </intent-filter>
        </receiver>
    </application>
</manifest>
```

TP : broadcast

- Créer une application qui affiche un Toast quand le téléphone passe et sort du mode Avion
- (Ne fonctionne pas sur les simulateurs avec API 26 et +)
- Pour les plus rapides : Même chose quand le Wifi s'active et se désactive. Classe WifiManager

Déclencher son propre Broadcast

○ Déclencher un Broadcast

```
public void sendBroadcast() {  
    Intent intent = new Intent();  
    //intent.setAction("android.intent.action.AIRPLANE_MODE");  
    intent.setAction("com.broadcast.MY_CUSTOM_INTENT");  
    sendBroadcast(intent);  
}
```

○ S'abonner dans le manifest

```
<receiver android:name="MyReceiver">  
    <intent-filter>  
        <action android:name="android.intent.action.AIRPLANE_MODE"/>  
        <action android:name="com.broadcast.MY_CUSTOM_INTENT"/>  
    </intent-filter>  
</receiver>
```

S'abonner en Java

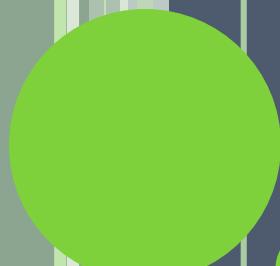
```
// Pointeur vers le Broadcast
private BroadcastReceiver receiver;

@Override
public void onCreate(Bundle savedInstanceState){

    // Création du Broadcast
    receiver = new BroadcastReceiver() {
        @Override
        public void onReceive(Context context, Intent intent) {
            //do something
        }
    };

    // S'abonner au BroadCast
    IntentFilter filter = new IntentFilter();
    filter.addAction("android.net.wifi.WIFI_STATE_CHANGED");
    registerReceiver(receiver, filter);
}

@Override
protected void onDestroy() {
    super.onDestroy();
    // Se désabonner sinon fuite mémoire
    if (receiver != null) {
        unregisterReceiver(receiver);
        receiver = null;
    }
}
```

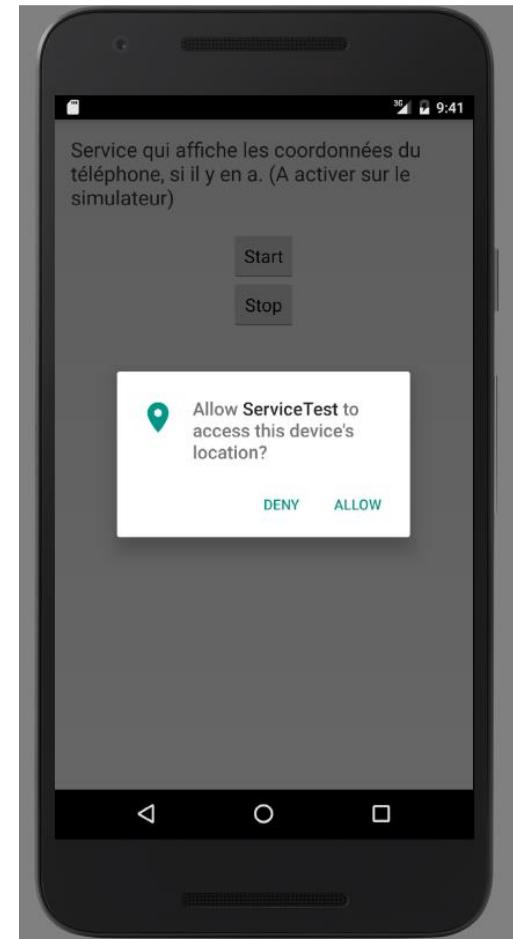


119

PERMISSION

Permission à la volée

- **Arrivée avec Marshmallow**
- **Il va falloir gérer**
 - le avant et le après Marshmallow
 - L'acceptation ou le refus
 - Le retrait de permission
- **3 types de permission**
 - Normal : A l'installation
 - Bluetooth, Internet, Vibreur...
 - Dangereuse : A la volée
 - Contact, sms, localisation...
- **Mais ce n'est pas si compliqué!!**



120

Permission à la volée

- **Déclarer la permission dans l'AndroidManifest (Au dessus d'application)**

```
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
```

- **Si c'est une permission dangereuse, gérer la permission à la volée dynamiquement (donc en Java)**
 - **Etape 1 :** Est-ce qu'on a déjà la permission ?
 - Oui : On peut utiliser la fonctionnalité
 - Non : On demande la permission
 - **Etape 2 :** On demande la permission
 - **Etape 3 :** On exploite la réponse
 - Oui : On peut utiliser la fonctionnalité
 - Non : On indique à l'utilisateur le refus

Permission à la volée

○ Vérification et demande de la permission

```
//Etape 1 : Est ce qu'on a déjà la permission ?  
//Attention prendre le Manifest d'android.util  
if (ContextCompat.checkSelfPermission(this, Manifest.permission.ACCESS_FINE_LOCATION)  
    == PackageManager.PERMISSION_GRANTED) {  
    //On a la permission  
}  
else {  
    //Etape 2 : On demande la permission  
    ActivityCompat.requestPermissions(this,  
        new String[]{Manifest.permission.ACCESS_FINE_LOCATION}, 25);  
}
```

○ Quand utiliser ce code ?

- Lancement de l'application
- Au moment où on en a besoin

Permission à la volée

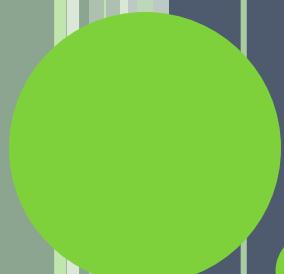
- **Traiter le résultat de la demande en surchargeant la méthode d'Activity :**

➤ onRequestPermissionsResult

```
@Override
public void onRequestPermissionsResult(int requestCode, String[] permissions, int[] gr) {
    super.onRequestPermissionsResult(requestCode, permissions, gr);
    //Est ce que c'est la permission qu'on a demandé ?
    if (requestCode == 25) {
        //On verifie la réponse
        if (ContextCompat.checkSelfPermission(this, Manifest.permission.ACCESS_FINE_LOCATION)
                == PackageManager.PERMISSION_GRANTED) {
            //ON a la permission
        }
        else {
            //On n'a pas la permission
        }
    }
}
```

TP PERMISSION

- Faites une demande d'autorisation d'utiliser la localisation à tester sur
 - Un device avant Marshmallow
 - Un device Marshmallow et +
- On utilisera la localisation plus tard !!!



125

SERVICES

Services

- Les services ont pour but de réaliser des tâches de fond pour une durée indéfinie. Exemple lecteur de musique.

- Les services doivent être déclarés dans le fichier `AndroidManifest.xml`:

```
<service android:name=".subpackagename.ServiceName"/>
```

- S'exécute sur le ThreadUI

- Intérêt par rapport aux threads: Le système interagit avec, par exemple quand il lui dit qu'il va s'arrêter (`onDestroy`).

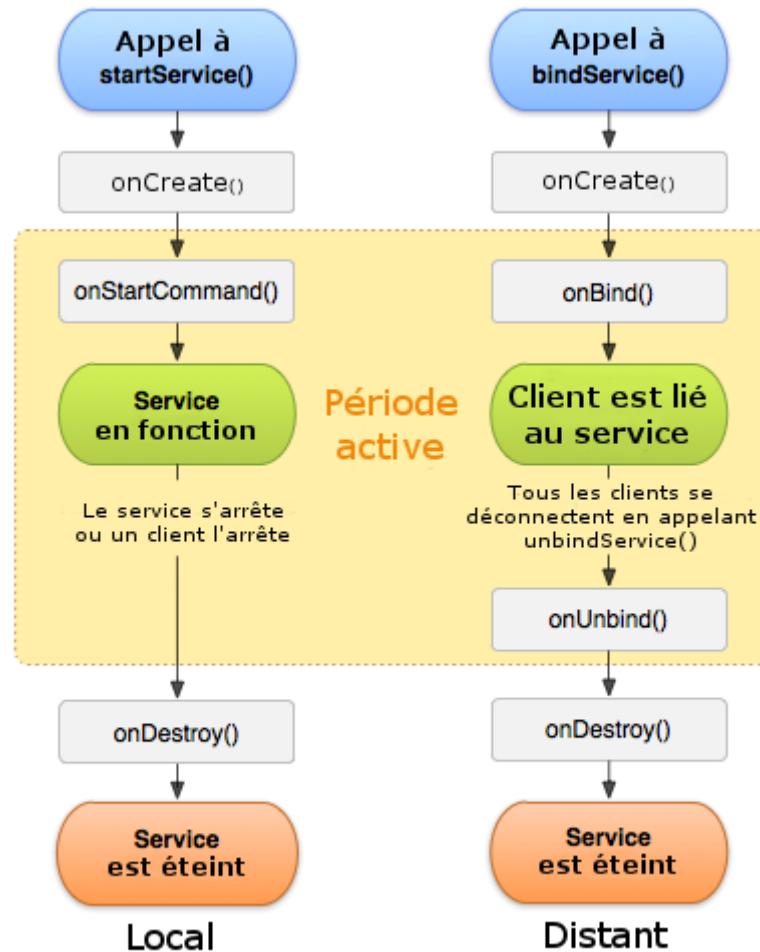
Services

- 2 types de service
 - Local : Même processus que l'application
 - Remote : En dehors du processus de l'application
- Ils doivent étendre la classe Service dont vous devrez surcharger les méthodes suivantes en fonction de vos besoins

```
void onCreate(); // initialisation des ressources  
void onStartCommand(Intent intent, int flags, int startId); // SDK>2.0 la tâche de fond démarre  
void onDestroy(); // libération des ressources
```

```
IBinder onBind(Intent intent); // connexion client distant  
boolean onUnbind(Intent intent); // déconnexion d'un client  
void onRebind(Intent intent); // Récuperer un service
```

Services



Exemple

```
public class BackgroundService extends Service {

    private Timer timer;

    @Override
    public void onCreate() {
        super.onCreate();
        timer = new Timer();
        timer.scheduleAtFixedRate(new TimerTask() {
            @Override
            public void run() {
                // Executer votre tâche
            }
        }, 0, 60000);

    }

    @Override
    public int onStartCommand(Intent intent, int flags, int startId) {
        return START_NOT_STICKY;
    }

    @Override
    public void onDestroy() {
        super.onDestroy();
        this.timer.cancel();
    }

    @Override
    public IBinder onBind(Intent intent) {
        return null;
    }
}
```

SERVICE

- Si le service existe déjà, lors d'un second appel, on ne passera pas par la méthode onCreate!!

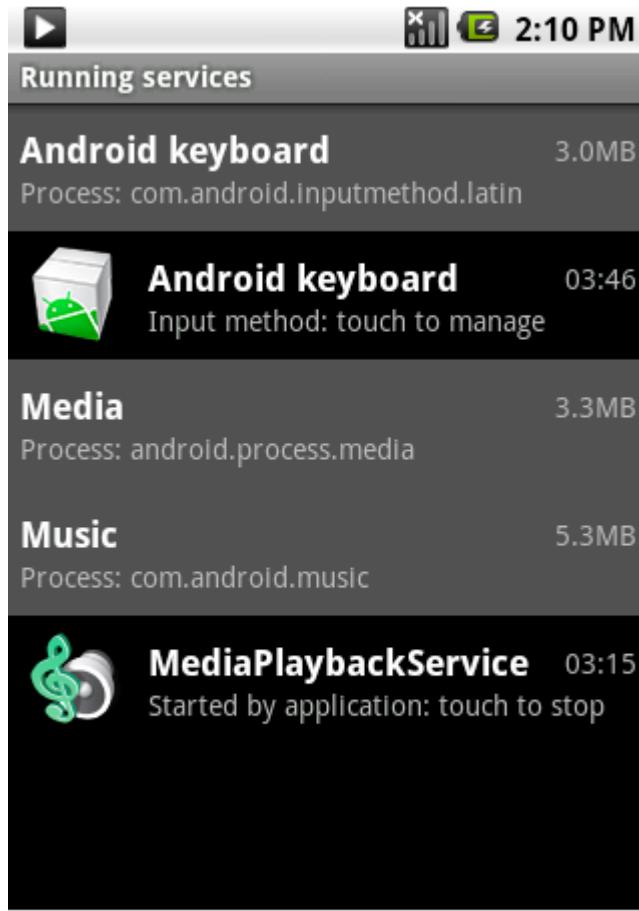
130

ONSTARTCOMMAND

OnStartCommand(Intent intent, int flags, int startId)

- startId : Au 1^{er} lancement vaut 1, au 2^{eme} vaut 2...
- Valeur de retour: elle correspond au comportement que doit adopter le service par rapport au processus qui l'a créé.
Plusieurs constantes sont disponibles dans la classe service.
 - START_STICKY: par défaut, redémarre le service et donc le processus, après un arrêt inattendu de celui ci (ex: kill du process dans le DDMS).
 - START_NOT_STICKY: le service n'est pas redémarré lors d'un arrêt inattendu de notre processus
 - START_REDELIVER_INTENT: pareil que START_STICKY mais redélivre l'intent en paramètre.

Services



- Il est possible de voir la liste des services exécutés en allant dans :
Menu > Settings > Applications > Running Services > du téléphone:

Démarrer/arrêter un service

```
//Démarre le service  
startService(new Intent(this, BackgroundService.class));
```

```
// stop le service  
stopService(new Intent(this, BackgroundService.class));
```

stopSelf() dans le service pour se stopper lui-même.

TP : services

- Créer un service qui sera démarré grâce à une interface possédant des boutons start/stop.
- Il affichera un toast avec les coordonnées GPS du téléphone.
- LE TIMER EST INNUTILE POUR CE TP

```
//Si on a pas la permission on ne s'abonne pas (onCreate)
if (ContextCompat.checkSelfPermission(this,
        Manifest.permission.ACCESS_FINE_LOCATION) != PackageManager.PERMISSION_GRANTED) {
    return;
}

//Minimum (et non égale, c'est Android qui gère) 5 secondes et 200m de difference.(onCreate)
//Le this ici représente l'interface « LocationListner » à implémenter MANUELLEMENT
locationMgr = (LocationManager) getSystemService(Context.LOCATION_SERVICE);
if (locationMgr.getAllProviders().contains(LocationManager.NETWORK_PROVIDER))
    locationMgr.requestLocationUpdates(LocationManager.NETWORK_PROVIDER, 5000, 200, this);
if (locationMgr.getAllProviders().contains(LocationManager.GPS_PROVIDER))
    locationMgr.requestLocationUpdates(LocationManager.GPS_PROVIDER, 5000, 200, this);

@Override
public void onLocationChanged(Location location) {
    Double latitude = location.getLatitude();
    Double longitude = location.getLongitude();
    //do something
}
//Désabonnement (onDestroy)
locationMgr.removeUpdates(this);
```

- Permission

```
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
```

LA CLASSE APPLICATION

```
public class MyApplication extends Application {  
  
    private static Bus bus;  
    public static Bus getBus() {  
        return bus;  
    }  
  
    @Override  
    public void onCreate() {  
        super.onCreate();  
        bus = new Bus();  
    }  
  
}
```

○ La déclarer dans l'AndroidManifest.xml

```
<application  
    android:name=".MyApplication"  
    android:allowBackup="true"  
    android:icon="@mipmap/ic_launcher"  
    android:label="CDI17"  
    android:roundIcon="@mipmap/ic_launcher_round"  
    android:supportsRtl="true"  
    android:theme="@style/AppTheme">
```

COMMUNICATION INTERTHREAD

- Librairie otto

- <http://square.github.io/otto/>

- Classe Application

```
private static Bus eventBus;  
eventBus = new Bus(); //dans le onCreate  
  
public static Bus getEventBus() {  
  
    return eventBus;  
}
```

- Classe émettrice

```
MyApplication.getEventBus().post(new EleveBean("prénom", "nom"));
```

- Classe de réception

```
MyApplication.getEventBus().register(this); //OnStart  
@Subscribe  
public void afficherEleve(EleveBean eleve) {  
    //traitements  
}  
  
MyApplication.getEventBus().unregister(this); //OnStop
```

INTENTSERVICE

- Sous classe de Service
- Permet de ne pas gérer de Thread dans le service et gère une liste d'attente d'appel.

```
public class MyIntentService extends IntentService {  
  
    public MyIntentService () {  
        super("UnNomAuHasard");  
    }  
  
    protected void onHandleIntent(Intent intent) {  
        //Gérer la requête  
    }  
}
```

- Grâce à Otto, communiquer avec le service précédent pour récupérer les coordonnées.
- Ajouter une fonctionnalité au service permettant de lui demander de se tuer

MYSERVICEBINDER

- Permet de récupérer l'instance d'un service

```
public class MyServiceBinder extends Binder {

    private BackgroundService backgroundService;

    //on recoit l'instance du service
    public MyServiceBinder(BackgroundService backgroundService) {
        super();
        this.backgroundService = backgroundService;
    }

    /** @return l'instance du service */
    public BackgroundService getBackgroundService() {
        return backgroundService;
    }
}
```

Service avec Binder

```
public class BackgroundService extends Service {

    private Timer timer;
    private IBinder iBinder = null; //l'instance du binder correspondant à notre service

    @Override
    public void onCreate() {
        super.onCreate();
        iBinder = new MyServiceBinder(this);
        timer = new Timer();
    }

    @Override
    public int onStartCommand(Intent intent, int flags, int startId) {
        timer.scheduleAtFixedRate(new TimerTask() {
            @Override
            public void run() {
                // Executer votre tâche
            }
        }, 0, 60000);
        return START_NOT_STICKY;
    }

    @Override
    public void onDestroy() {
        super.onDestroy();
        this.timer.cancel();
    }

    @Override
    public IBinder onBind(Intent intent) {
        return iBinder;
    }
}
```

Et dans l'activité...

```
public class MyActivity extends Activity {  
  
    // ServiceConnection permet de gérer l'état du lien entre l'activité et le service.  
    private ServiceConnection serviceConnection;  
    //L'instance du service  
    private BackgroundService myService;  
  
    //Lance ou récupère l'instance du service  
    private void bindToService() {  
        if (serviceConnection == null) {  
            serviceConnection = new ServiceConnection() {  
  
                //le service s'est déconnecté  
                public void onServiceDisconnected(ComponentName name) {  
                    myService = null;  
                    //Do something  
                }  
  
                //le service se connecte  
                public void onServiceConnected(ComponentName arg0, IBinder binder) {  
                    //on récupère l'instance du service dans l'activité  
                    myService = ((MyServiceBinder) binder).getBackgroundService();  
                }  
            };  
        }  
        Intent intent = new Intent(this, BackgroundService.class);  
  
        //démarrer le service si il n'est pas démarré  
        startService(intent);  
        //lance le binding du service  
        bindService(intent, serviceConnection, Context.BIND_AUTO_CREATE);  
    }  
    ...  
}
```

Et dans l'activité...

```
public class MyActivity extends Activity {

    private void stopService() {
        //on détruit le service
        if (myService != null) {
            myService.stopSelf();
            myService = null;
        }

        if (serviceConnection != null) {
            unbindService(serviceConnection);
            serviceConnection = null;
        }
    }

    @Override
    protected void onPause() {
        super.onPause();
        //Sinon impossible de tuer l'activité
        if (serviceConnection != null) {
            unbindService(serviceConnection);
        }
        updateDataService = null;
        serviceConnection = null;
    }
}
```

TP : services

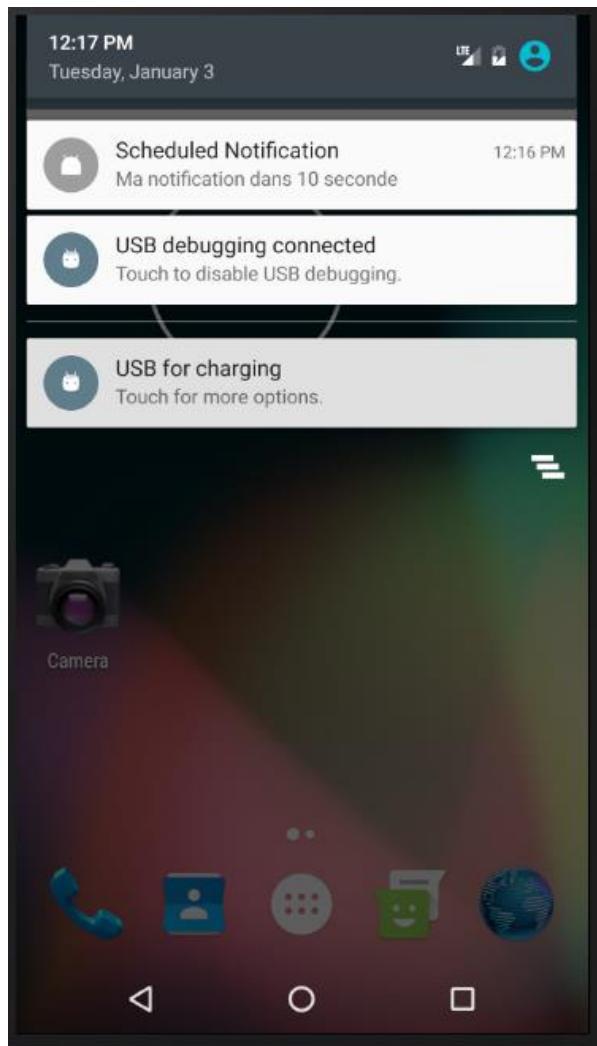
- Comprendre le service avec Bind
 - Projet : ServiceBinding



144

NOTIFICATION

Notifications



Notifications

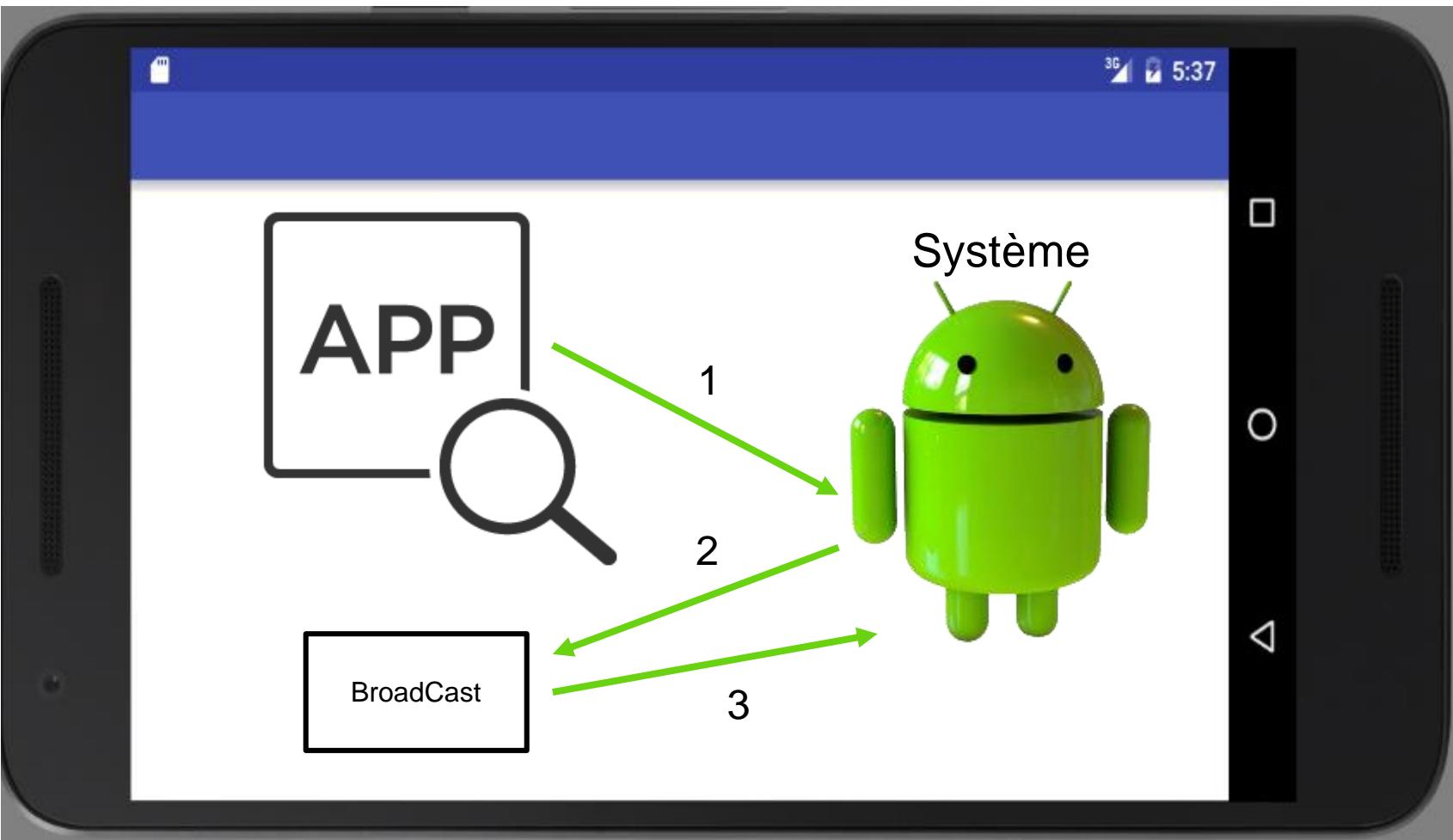
- Lorsque l'utilisateur visualisera la notification, il doit pouvoir revenir sur l'activité émettrice de la notification.
- Pour cela, vous devez utiliser un type spécifique d'Intent, le PendingIntent.

```
PendingIntent pendingIntent = PendingIntent.getActivity(this, 28,  
new Intent(this, MainActivite.class), PendingIntent.FLAG_ONE_SHOT);
```

Notifications

```
//Envoyer une notification immediate  
public static void createInstantNotification(Context context, String message) {  
  
    //Ce qui se passera quand on cliquera sur la notif  
    Intent intent = new Intent(context, MainActivity.class);  
    PendingIntent pendingIntent = PendingIntent.getActivity(context, 28, intent,  
        PendingIntent.FLAG_ONE_SHOT);  
  
    //La notification  
    Notification notification = new NotificationCompat.Builder(context, "toto")  
        .setSmallIcon(R.mipmap.ic_launcher)  
        .setContentTitle("Le titre")  
        .setContentText(message)  
        .setContentIntent(pendingIntent).build();  
  
    //Envoyer la notification  
    NotificationManager notificationManager = (NotificationManager)  
        context.getSystemService(Context.NOTIFICATION_SERVICE);  
  
    //ENVOIE  
    notificationManager.notify(29, notification);  
}
```

Notifications à retardement



148

Notifications à retardement

```
private void scheduleNotification(String message, long delay) {  
  
    //La notification  
    NotificationCompat.Builder builder = new NotificationCompat.Builder(this, "Toto");  
    builder.setContentTitle("Scheduled Notification");  
    builder.setContentText(message);  
    builder.setSmallIcon(R.mipmap.ic_launcher);  
  
    //Redirection vers le broadcast  
    Intent notificationIntent = new Intent(this, NotificationPublisherBR.class);  
    notificationIntent.putExtra("MaCle", builder.build());  
    PendingIntent pendingIntent = PendingIntent.getBroadcast(this, 0, notificationIntent,  
PendingIntent.FLAG_UPDATE_CURRENT);  
  
    //La dans le futur  
    long futureInMillis = SystemClock.elapsedRealtime() + delay;  
    AlarmManager alarmManager = (AlarmManager) getSystemService(Context.ALARM_SERVICE);  
    alarmManager.set(AlarmManager.ELAPSED_REALTIME_WAKEUP, futureInMillis, pendingIntent);  
}
```

Notifications à retardement

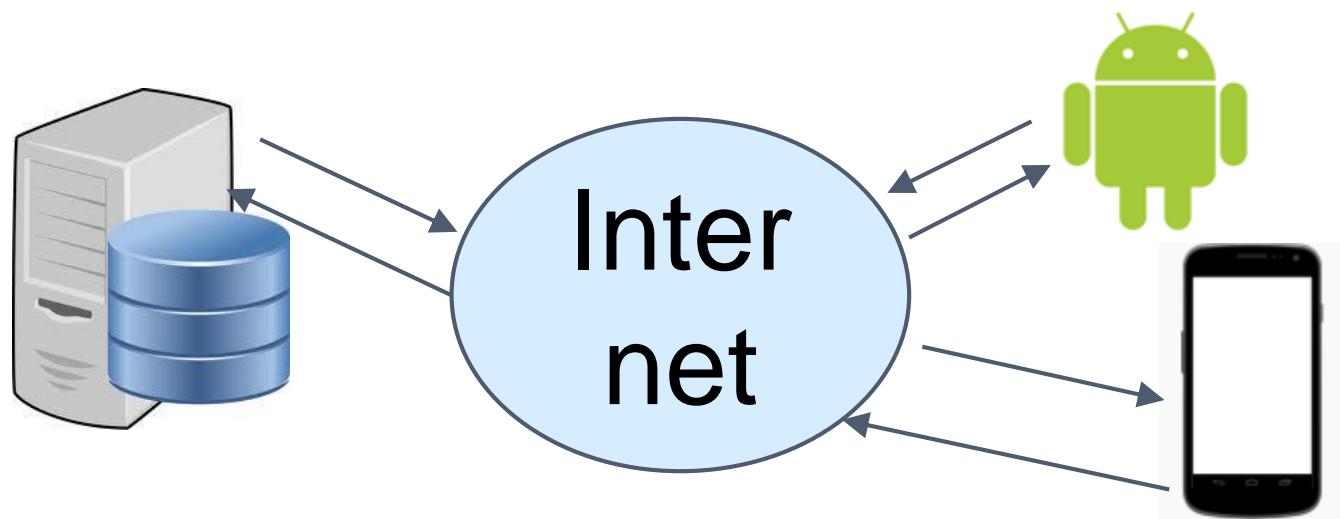
```
public class NotificationPublisherBR extends BroadcastReceiver {  
  
    public void onReceive(Context context, Intent intent) {  
        //On récupère la notification reçue  
        Notification notification = intent.getParcelableExtra("MaCle");  
  
        //on l'affiche  
        NotificationManager notificationManager = (NotificationManager)  
            context.getSystemService(Context.NOTIFICATION_SERVICE);  
        notificationManager.notify(1, notification);  
    }  
}
```

TP : notifications

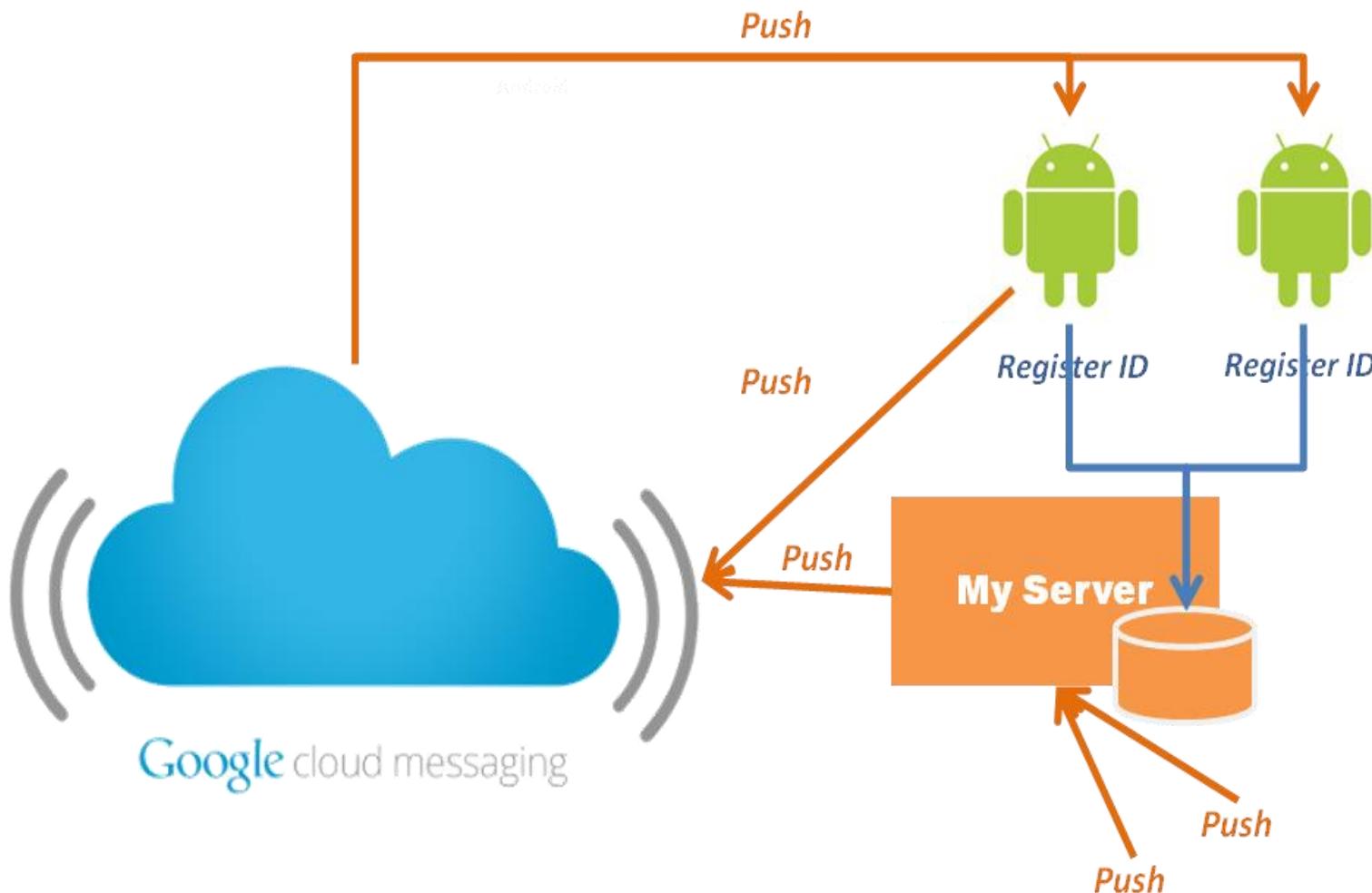
- Créer un activité avec 2 boutons

- Le 1^{er} qui affiche une notification immédiate
- Le 2eme qui affiche une notification dans 30 secondes même si l'application est tuée.

Google cloud messaging



Google cloud messaging





FireBase

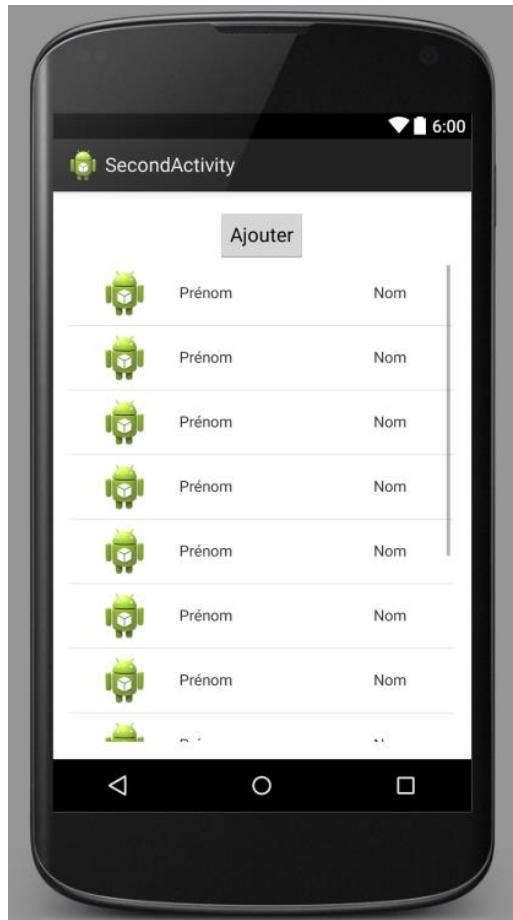
- Suite d'outils de Google pour les applications mobiles.
 - Envoie de notification
 - Gestion d'analytics
 - Publicité
 - Multiplateforme (Android et iOS)
 - Gratuit jusqu'à une certaine quantité.
 - Intégré à la console developer
- <https://firebase.google.com/>
- Présentation de son utilisation lors d'une session du TAUG
 - <https://www.youtube.com/watch?v=gHDWuhDZTUl>

Pour continuer

- Créer un compte et son projet sur la console de développement de Google.
 - <https://developers.google.com/>
- Activer Google Cloud Messaging
- Et suivre le tuto Google du moment...

ANDROID

- Afficher les données sous forme de liste



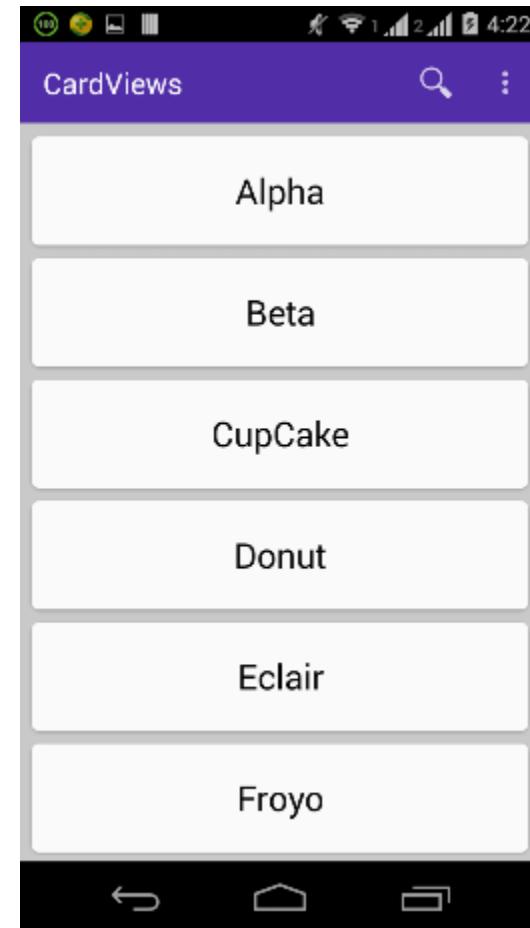
NOUVEAU COMPOSANT DE LOLLIPOP

○ RecycleView

- La nouvelle ListView de Lollipop
- Permet des animations à l'utilisation

○ CardView

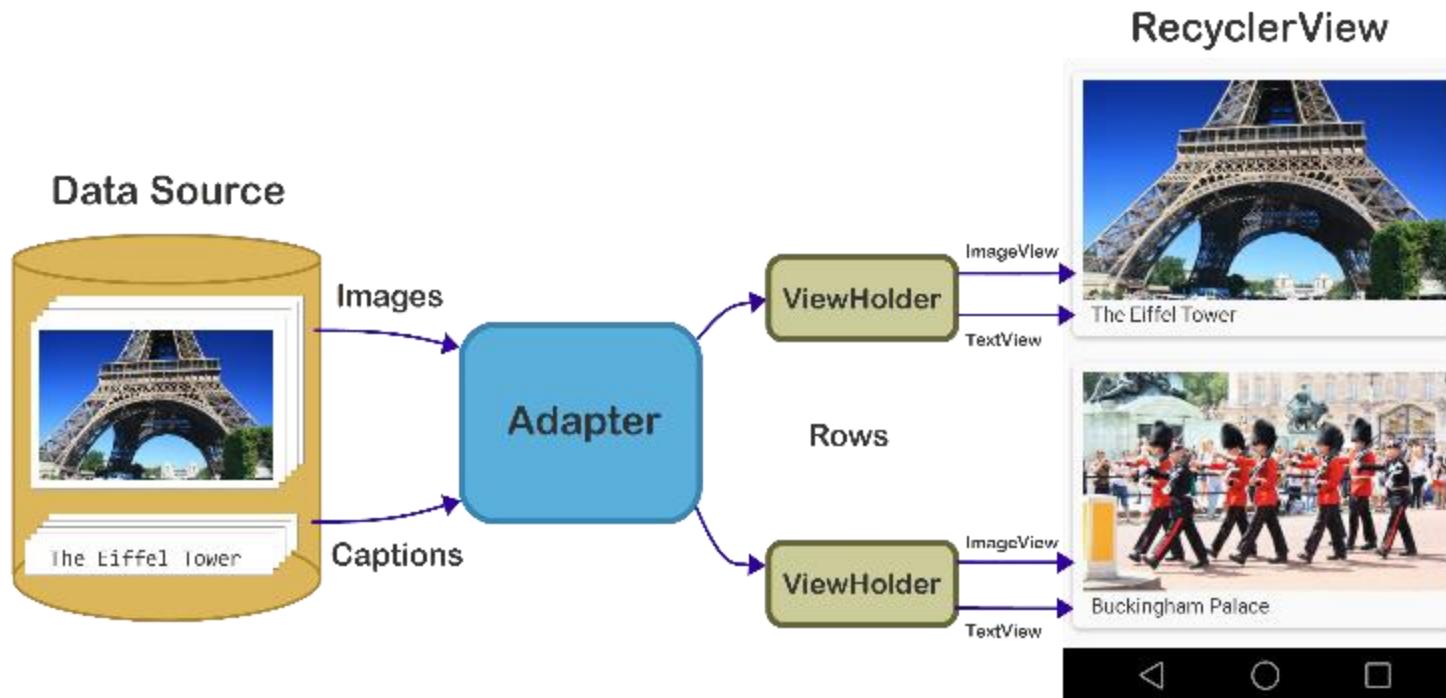
- Composant XML
- Créer un bloc avec une ombre.



LISTVIEW / RECYCLEVIEW

- Recycler les cellules, pour le bien de votre device.
- Créer un fichier XML représentant le layout **d'une ligne**
- Créer un adapter indiquant comment afficher chaque ligne
- Utiliser un composant d'affichage d'adapter:
 - RecyclerView
 - ListView

RECYCLEVIEW



RECYCLEVIEW : LE COMPOSANT

- Ajouter la librairie dans dependencies dans build.gradle

- *//Recyclerview dans le build.gradle*
compile 'com.android.support:recyclerview-v7:24.+'

- Utiliser le composant dans les layouts :

```
<android.support.v7.widget.RecyclerView  
    android:id="@+id/rv"  
    android:layout_width="match_parent"  
    android:layout_height="match_parent"/>
```

- Récupérer le composant dans l'activité :

```
RecyclerView rv;  
rv = (RecyclerView) findViewById(R.id.rv); //onCreate
```

RECYCLEVIEWADAPTER

```
public class RVAdapter //1
    extends RecyclerView.Adapter<RVAdapter.ViewHolder> {//3

    //Classe qui stocke les composants graphiques d'1 ligne
    protected static class ViewHolder extends RecyclerView.ViewHolder { //2

        public TextView ec_tv_nom, ec_tv_prenom;
        public ImageView ec_iv;

        public ViewHolder(View itemView) {
            super(itemView);
            ec_tv_nom = (TextView) itemView.findViewById(R.id.ec_tv_nom);
            ec_tv_prenom = (TextView) itemView.findViewById(R.id.ec_tv_prenom);
            ec_iv = (ImageView) itemView.findViewById(R.id.ec_iv);
        }
    }

    public RVAdapter.ViewHolder onCreateViewHolder(ViewGroup vg, int viewType) {...} //4
    public void onBindViewHolder(RVAdapter.ViewHolder holder, int position) {...} //4
    public int getItemCount() {...} //4
}
```

RECYCLEVIEWADAPTER

```
public class RVAdapter extends RecyclerView.Adapter<RVAdapter.ViewHolder> {

    protected static class ViewHolder extends RecyclerView.ViewHolder {...}

    //Détermine quel fichier XML on utilise pour représenter une cellule
    @Override
    public RVAdapter.ViewHolder onCreateViewHolder(ViewGroup vg, int viewType) {
        View v=LayoutInflater.from(vg.getContext()).inflate(R.layout.eleve_cellule, vg, false);
        return new RVAdapter.ViewHolder(v);
    }

    public void onBindViewHolder(RVAdapter.ViewHolder holder, int position) {...}
    public int getItemCount() {...}
}
```

RECYCLEVIEWADAPTER

```
public class RVAdapter extends RecyclerView.Adapter<RVAdapter.ViewHolder> {

    private ArrayList<Eleve> eleveBeanList;
    //Constructeur
    public RVAdapter(ArrayList<Eleve> eleveBeanList) {
        this.eleveBeanList = eleveBeanList;
    }

    protected static class ViewHolder extends RecyclerView.ViewHolder {...}
    public RVAdapter.ViewHolder onCreateViewHolder(ViewGroup vg, int viewType) {...}

    //Remplir les composants graphique de chaque cellule
    @Override
    public void onBindViewHolder(RVAdapter.ViewHolder holder, int position) {
        //L'élève correspondant à la ligne
        Eleve eleve = eleveBeanList.get(position);
        holder.ec_tv_nom.setText(eleve.getNom());
        holder.ec_tv_prenom.setText(eleve.getPrenom());
    }

    public int getItemCount() {...}
}
```

RECYCLEVIEWADAPTER

```
public class RVAdapter extends RecyclerView.Adapter<RVAdapter.ViewHolder> {

    private ArrayList<Eleve> eleveBeanList;
    public RVAdapter(ArrayList<Eleve> eleveBeanList) {...}
    protected static class ViewHolder extends RecyclerView.ViewHolder {...}
    public RVAdapter.ViewHolder onCreateViewHolder(ViewGroup vg, int viewType) {...}
    public void onBindViewHolder(RVAdapter.ViewHolder holder, int position) {...}

    //Combien de cellule on affiche
    @Override
    public int getItemCount() {
        return eleveBeanList.size();
    }
}
```

RecyclerView (Côté Activity)

```
//Données  
  
private ArrayList<Eleve> eleveList;  
private RVAdapter rVAdapter;  
//Composant graphique afficheur de RecyclerView.Adapter  
private RecyclerView recycleView;  
  
//onCreate  
//Création de la liste  
eleveList = new ArrayList<Eleve>();  
//Instanciation d'un RVAdapter  
rVAdapter = new RVAdapter(eleveList);  
recycleView = (RecyclerView) findViewById(R.id.recycleView);  
// L'adapter que l'on souhaite afficher  
recycleView.setAdapter(rVAdapter);  
//Réglage : Est ce qu'on affiche ligne par ligne ou  
recycleView.setLayoutManager(new LinearLayoutManager(this));  
//new GridLayoutManager(this, 2) //Sous forme de tableau à 2 colonnes  
// Réglage : type d'animation qu'on utilise  
recycleView.setItemAnimator(new DefaultItemAnimator());
```

RecyclerView Actualisation et animation

```
//Indiquer que ma source de donnée à changé et qu'il faut rafraichir  
rVAdapter.notifyDataSetChanged();  
  
//Si on sait ce qui a changé, on l'indique pour une animation d'insertion  
rVAdapter.notifyItemInserted(0);  
//Animation de suppression  
rVAdapter.notifyItemRemoved(0);  
//Animation de déplacement  
rVAdapter.notifyItemMoved(0, 3);
```

1 parmi les 4

CARDVIEW: LE COMPOSANT

- Ajouter la librairie dans dependencies dans build.gradle

```
//CardView  
compile 'com.android.support:cardview-v7:23.+'
```

- Utiliser le composant dans les layouts (Il ne peut contenir qu'un seul enfant):

```
<android.support.v7.widget.CardView  
    xmlns:android="http://schemas.android.com/apk/res/android"  
    xmlns:card_view="http://schemas.android.com/apk/res-auto"  
    android:layout_width="match_parent"  
    android:layout_height="wrap_content"  
    android:layout_margin="6dp"  
    card_view:cardCornerRadius="4dp"    >
```

TP – RECYCLEVIEW

- **Etape 1 : Créez une RecyclerView d'élève**
 - Un bouton « ajouter » ajoute un élève à la liste
 - Pour le nom et le prénom de l'élève : eleve0, eleve1, eleve2...
 - Pas d'image pour le moment
- **Etape 2 : Ajouter une animation d'insertion.**
- **Etape 3 : Ajouter un CardView à votre layout de cellule**
- **Etape 4 (pour les plus avancés) : Gestion d'images en ligne**
 - Utiliser la librairie Glide pour gérer le chargement des images depuis une URL
 - <http://www.tutos-android.com/tag/glide-tuto>
 - Attention, pour aller chercher les images, il faut la permission d'aller sur internet!!
 - Quelques url d'images d'élèves :
 - https://pixabay.com/static/uploads/photo/2014/04/02/17/02/girl-307747_960_720.png
 - <http://www.coloriage.tv/dessincolo/ecolier.png>
 - <http://clamart-lafontaine-blog.e-monsite.com/medias/images/eleve.png>
 - <http://ekladata.com/2NvmX2GdczA71ZMewxFwyR9CesE@350x586.png>

RECYCLEVIEW: CLIC SUR UNE CELLULE

- Ajout d'un id au layout de la cellule :

```
    android:id="@+id/root"
```

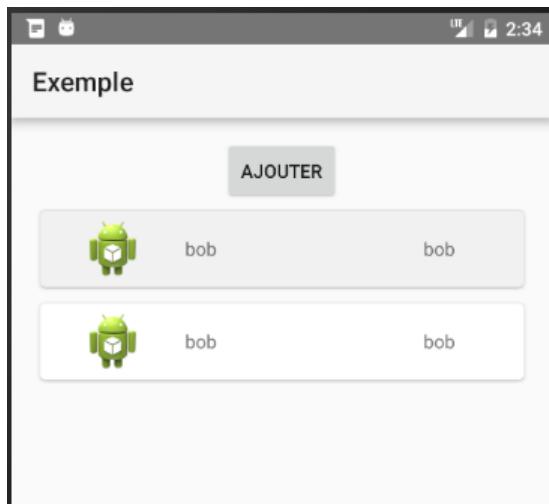
- On complète notre **ViewHolder** pour qu'il stocke ce pointeur
- Et dans le **onBindViewHolder()** on intercepte le clic :

```
holder.root.setOnClickListener(new View.OnClickListener() {  
    @Override  
    public void onClick(View v) {  
        //un clic  
    }  
});
```

RECYCLEVIEW: ANIMATION DU CLIC

- Pour ajouter une animation lors du clic, on l'ajoute au background de la cellule dans le layout du XML

```
<LinearLayout  
    android:id="@+id/root"  
    android:layout_width="match_parent"  
    android:layout_height="match_parent"  
    android:background="?android:selectableItemBackground"  
    android:orientation="horizontal">  
  
</LinearLayout>
```



CRÉER UN CALLBACK : CLASSE ÉMÉTRICE

- Créer un Callback à l'aide d'une interface :

```
public interface OnEleveClickListener {  
    void onEleveClic(Eleve eleve);  
}
```

- Transmettre ce CallBack grâce à un setter

```
private OnEleveClickListener eleveClickListener;  
  
public setOnEleveClickListener (OnEleveClickListener eleveClickListener) {  
    this.eleveClickListener = eleveClickListener;  
}
```

- L'utiliser quand on souhaite prévenir la classe réceptrice

```
if (eleveClickListener != null) {  
    eleveClickListener.onEleveClic(eleve);  
}
```

CRÉER UN CALLBACK : CLASSE RÉCÉPTRICE

- S'abonner à la classe émettrice comme le onClick d'un bouton
- Alt+ entrée -> Génèrera la méthode onEleveClick

```
rvAdapter.setOnEleveClickListener(this);
```

TP – RECYCLEVIEW CLIC

- **Etape 1:** Un clic long sur un élève le supprime de la liste
- **Etape 2 :** Un clic sur un élève le positionne en 1^{er} position



174

ASYNCTASK



AsyncTask

- AsyncTask ou asynchrone task, comme un Thread.
- Avantage
 - Le thread se crée automatiquement
 - Et la communication entre les différents thread est simplifiée.
- A utiliser pour les traitements lourds
 - Gros calculs
 - Requête WS
- Une tache ne peut être exécutée qu'une seule fois.

Requête longue

```
public class WebServiceUtils { //Classe appartenant au model. Gérera les futurs requêtes

    public static Eleve loadEleveFromWeb() {
        SystemClock.sleep(5000); //Attente de 5 secondes

        return new Eleve("Toto", "Tata");
    }

}
```

AsyncTask (Simple)

```
public class ChargementEleveAT extends AsyncTask {  
  
    private Eleve resultat;  
  
    // Garantie en dehors de l'UIThread  
    protected Object doInBackground(Object... params) {  
        // TRAITEMNT LONG MAIS INTERDIT DE TOUCHER AUX COMPOSANT GRAPHIQUE  
        resultat = WebServiceUtils.loadEleveFromWeb();  
        return null;  
    }  
  
    // Dans UIThread  
    protected void onPostExecute(Object object) {  
        // Méthode appelée quand le doInBackground est terminé  
        //On peut modifier les composants graphiques  
        monTv.setText("Nom : " + resultat.getText());  
    }  
}
```

AsyncTask (Simple)

○ Utilisation

```
ChargementEleveAT chargementEleveAT = new ChargementEleveAT();  
chargementEleveAT.execute();
```

- Une tache ne peut être exécutée qu'une seule fois.
- Etat de l'AsyncTask : `chargementEleveAT.getStatus()`
 - Status.*PENDING* (*Prête*)
 - Status.*RUNNING* (*En cours d'exécution*)
 - Status.*FINISHED* (*Terminée*)

TP AsyncTask

- Reprendre le TP de l'affichage des élèves sur un RecycleView
- **Etape 1 :** Créer une classe MonAsyncTask en classe interne (dans le même fichier) de l'Activity et qui appellera :
`WebServiceUtils.loadEleveFromWeb()`
- **Etape 2 :** Ajouter un bouton « Charger » qui lance cette AsyncTask
- **Etape 3 (pour les plus rapides)** : Ajouter une ProgressDialog pendant le chargement
- **Etape 4 (pour les plus rapides)** : Déplacer l'AsyncTask dans son propre fichier et faire en sorte que cela fonctionne

AsyncTask (Avancé)

```
public class ChargementEleveAT extends AsyncTask<Params, Progress, Result>
public class ChargementEleveAT extends AsyncTask<Void, Pair<Integer, Integer>, ArrayList<Eleve>> {
    public ChargementEleveAT() {
    }

    @Override
    protected void onPreExecute() {
        super.onPreExecute();
    }
    // Garantie en dehors de l'UIThread
    protected ArrayList<Eleve> doInBackground(Void... params) {
        // traitement
        publishProgress(new Pair<Integer, Integer>(3,100));
        return new ArrayList<Eleve>();
    }
    //UIThread
    protected void onProgressUpdate(Pair<Integer, Integer>... values) {
        // mise à jour progress bar ou UI
    }
    // UIThread
    protected void onPostExecute(ArrayList<Eleve> eleveList) {
    }
}
```

AsyncTask (Avancé)

o Utilisation

```
private ChargementEleveAT chargementEleveAT = null;

if (chargementEleveAT == null || chargementEleveAT.getStatus() == Status.FINISHED) {
    chargementEleveAT = new ChargementEleveAT();
    chargementEleveAT.execute();
}
```

AsyncTask

- Arrêter une AsyncTask (Par exemple dans le onStop ou le onDestroy de l'activity)

```
chargementEleveAT.cancel(true);
```



- Si le « doInBackground » a commencé il finira son exécution
- Faire stopper le doInBackground dans celui-ci

```
if(isCancelled()) {  
    return null;  
}
```

TP AsyncTask

- **Etape 1 :** Déplacer la classe AsyncTask dans son propre fichier
- **Etape 2 :** L'AsyncTask devra rendre son résultat à l'aide d'un callBack.
Créer à l'aide de l'interface:

```
public interface CallBack {  
    void eleveLoaded(List<Eleve> eleve);  
}
```



WEB

184

WebView, WebService, HttpURLConnection

CONNEXION HTTP

- 2 classes
 - HttpClient à partir de Eclair (à ne plus utiliser)
 - HttpURLConnection (à partir de GingerBread)
 - Compression, cache
- On peut aussi
 - Utiliser des cookies
 - Utiliser GET, POST, WebSocket
 - Proxies
 - Cache des réponses HTTP

CONNEXION HTTP

Tester la connectivité

- Réalisable depuis le thread principale.

```
public static boolean isInternetConnexion(Context context) {  
    ConnectivityManager cm = (ConnectivityManager) context.getSystemService(Context.CONNECTIVITY_SERVICE);  
    return cm != null && cm.getActiveNetworkInfo() != null && cm.getActiveNetworkInfo().isConnected();  
}  
  
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE"/>
```

FAIRE UNE REQUÊTE GET

○ OkHttp

- <http://square.github.io/okhttp/>
- ```
public static String sendGetOkHttpRequest(String url) throws Exception {
 Log.w("tag", "url : " + url);
 OkHttpClient client = new OkHttpClient();
 //Création de la requête
 Request request = new Request.Builder().url(url).build();
 //Execution de la requête
 Response response = client.newCall(request).execute();
 //Analyse du code retour
 if (response.code() != 200) {
 throw new Exception("Réponse du serveur incorrect : " + response.code());
 }
 else {
 //Résultat de la requête.
 //ATTENTION .string() ne peut être appelée qu'une seule fois.
 return response.body().string();
 }
}
```

# FAIRE UNE REQUÊTE POST

```
public static String sendPostOkHttpRequest(String url, String paramJson)
throws Exception {
Log.w("tag", "url : " + url);
OkHttpClient client = new OkHttpClient();
MediaType JSON = MediaType.parse("application/json; charset=utf-8");
//Corps de la requête
RequestBody body = RequestBody.create(JSON, paramJson);

//Création de la requête
Request request = new Request.Builder().url(url).post(body).build();
//Execution de la requête
Response response = client.newCall(request).execute();
//Analyse du code retour
if (response.code() != 200) {
 throw new Exception("Réponse du serveur incorrect : " + response.code());
}
else {
 //Résultat de la requête.
 return response.body().string();
}
```

# WEBVIEW

## ○ Un composant comme les autres

```
<WebView
 android:layout_width="fill_parent"
 android:layout_height="fill_parent" />
```

```
WebView myWebView = (WebView) findViewById(R.id.webview);
myWebView.loadUrl("http://www.amonteiro.fr");
//charger un site en local
myWebView.loadUrl("file:///android_asset/index.html");
//charger une page
myWebView.loadData("<html><body>Bonjour !</body></html>" , "text/html", "UTF-8");
```

## ○ Réglages

```
//Sinon cela lance le navigateur du téléphone
myWebView.setWebViewClient(new WebViewClient());
//Activr le Javascript (Attention aux performances)
WebSettings webviewSettings = myWebView.getSettings();
webviewSettings.setJavaScriptEnabled(true);
```

# WEBVIEWCLIENT

- Interagir avec les pages grâce au WebClientView
  - Et que les futures pages ne se lancent pas dans le navigateur.

```
webView.setWebViewClient(new MyWebViewClient());

public class MyWebViewClient extends WebViewClient {
 ...
 // Pour afficher des sites en https
 public void onReceivedSslError(WebView view, SslErrorHandler handler, SslError error) {
 handler.proceed(); // Ignore SSL certificate errors
 }
 ...
}
```

- Les méthodes

```
public void onPageStarted(WebView view, String url, Bitmap favicon)
public void onPageFinished(WebView view, String url)
```

190

- Créer un projet chargeant une page avec une webView et avec OkHttp

```
<uses-permission android:name="android.permission.INTERNET"/>
```

# WEBSERVICE

## ○ SOAP

- Protocole non compatible nativement avec Android.
- Plus difficile à développer
- Librairie kSOAP 2

## ○ REST

- Privilégié par google
- Lié au modèle de transport HTTP

# WEBSERVICE REST

## ○ JSON (*JavaScript Object Notation*)

- Dérivé de la notation des objets du langage JavaScript
- 2 types d'élément
  - Clé / valeur
  - Liste ordonnée

## ○ Avantages

- Peu verbeux, ce qui le rend lisible aussi bien par un humain que par une machine
- Facile à apprendre, car sa syntaxe est réduite et non extensible
- Ses types de données sont connus et simples à décrire.

# JSON

## ○ A quoi cela ressemble ?

//Succès

```
{ "results":
 [
 {
 "ville": "Saint-Ouen",
 "cp": 93400
 },
 {
 "ville": "La Plaine-Saint-Denis",
 "cp": 93210
 },
 {
 "ville": "Levallois-Perret",
 "cp": 92300
 }
],
 "nbr": 3
}
```

//Echec

```
{
 "errors": {
 "message": "Aucun terme trouve",
 "code": "2"
 }

```

# JSON OUTILS

- Notepad++
  - JSON Viewer
- Chrome
  - JSON Formatter
- Librairie GSON pour gagner du temps.
  - Permet de sérialiser/désérialiser du JSON
  - <https://sites.google.com/site/gson/gson-user-guide>
  - compile 'com.google.code.gson:gson:2.3'
- Générer beans Java à partir du JSON
  - <http://pojo.sodhanalibrary.com>

# GSON FONCTIONNEMENT

- Créer les beans de réception

- Parser le flux

```
//Création de l'objet

private Gson gson = new Gson();

//parsing du flux ou du String contenant du JSON
ResultBean result = gson.fromJson(monStringJson , ResultBean.class);

//Parser une ArrayList typée
ArrayList<Message> list = gson.fromJson(monStringJson,
 new TypeToken<ArrayList<ResultBean>>() {}.getType());
```

- Traiter le résultat

# JSON BEAN

- Parsing par introspection
  - A part vérifier, rien à faire!!

```
public class ResultBean {

 private ArrayList<CityBean> results;
 private int nbr;
 private ErrorBean errors;

}
```

# JSON

## ○ A quoi cela ressemble ?

//Succès

```
{ "results":
 [
 {
 "ville": "Saint-Ouen",
 "cp": 93400
 },
 {
 "ville": "La Plaine-Saint-Denis",
 "cp": 93210
 },
 {
 "ville": "Levallois-Perret",
 "cp": 92300
 }
],
 "nbr": 3
}
```

//Echec

```
{
 "errors": {
 "message": "Aucun terme trouve",
 "code": "2"
 }

```

# CLIENT SERVEUR : BONNES PRATIQUES

- Créer une librairie de Bean commune sur un repository Maven
  - Rien à faire côté client en cas de changement sur les beans
  - 1 seul test de parsing à faire pour les 2.
  - Ajouter dans la librairie les méthodes d'appel des WS

- Créer un projet permettant à partir d'un code postal de récupérer la ville correspondante.
  - Récupérer une clé sur le site  
<http://www.citysearch-api.com/>
  - Information sur le Webservice  
<http://www.citysearch-api.com/fr/tutorial/3/widget-recherche-ville-code-postal.html#liste>
  - Exercice et Solution projets : webServiceGSON
    - `private final static String API_LOGIN = "login=webserviceexemple";  
private final static String API_KEY = "apikey=s0f940dd26cf107eabf8bf6827f87c3ca8e8d82546";`

# Retrofit

- Appel WebService par injection

- <http://square.github.io/retrofit/>

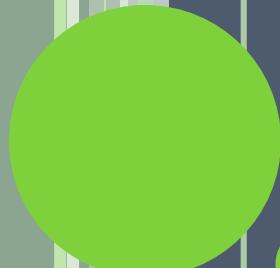
- Créer une interface

```
public interface CPService {
 @GET("/cp={cp}")
 ResultBean getCp(@Path("cp") String cp);

 @GET("/")
 ResultBean getCpV2(@Query("cp") String cp);
}
```

- Utilisation

```
CPService service = new Retrofit.Builder().baseUrl("www.aaaa.fr/... ")
 .build().create(CPService.class);
ResultBean result = service.getCp(postalCode);
```



202

## PERSISTANCE

SharedPreferences, SQLite, GreenDao

# PRÉFÉRENCES PARTAGÉES

- Stockage des informations sous forme clef / valeurs
- Mise en place rapide
- Idéales pour de petites quantités d'informations
  - nom de l'utilisateur
  - email
  - préférence de tri d'une liste...
- Classe de base : SharedPreferences

# PRÉFÉRENCES PARTAGÉES

- Il est possible de récupérer cet objet de plusieurs façons avec des droits différents :
  - De manière générale:
  - `getSharedPreferences(String nom, int droit)`
- Informations disponibles en lecture seule par des méthodes associées :
  - `getString()`, `getInt()`...

# PRÉFÉRENCES PARTAGÉES

- Permissions
- Il existe 3 type de paramètres:
  - MODE\_PRIVATE: par défaut, créé et accessible uniquement dans l' application courante.
  - MODE\_WORLD\_PRIVATE : les autres applications y ont accès en lecture seule.
  - MODE\_WORLD\_WRITABLE : lecture/écriture partout.

# PRÉFÉRENCES PARTAGÉES

- L'enregistrement de préférences se fait avec un objet de type Editor, récupéré par la fonction SharedPreferences.edit();

```
SharedPreferences mPrefs = context.getSharedPreferences("MonFichier",
MODE_PRIVATE);
```

- Ecrire

```
Editor editor = mPrefs.edit();
editor.putBoolean("first_start", false);
editor.apply(); //commit()
```

- Lire

```
return mPrefs.getBoolean("first_start", true);
```

# INTRODUCTION AUX BASES DE DONNÉES

- Sqlite est beaucoup utilisé dans les systèmes embarqués car il allie simplicité et mémoire légère.
- Pour Android, la base de données Sqlite est native et directement connectée à la machine virtuelle. De ce fait, toutes les applications peuvent l'utiliser.
- Cependant son API n'est pas jdbc mais une API plus légère

# Exemple

```
db.execSQL("create table produits (_id integer primary key
autoincrement,
+ "codebarre text not null, titre text not null,
+ "description text not null"
+ ");");
```

# CRÉER UNE BDD

- Aucune base de données n'est fournie automatiquement par Android.
- Il faudra la créer et la remplir.
- Pour cela, il faut utiliser une redéfinition de la classe SQLiteOpenHelper.

# Exemple : Creation de la table

```
public class MaBaseSQLite extends SQLiteOpenHelper {

 private static final String NOM_BDD = "mabase.db";
 private static final int VERSION_BDD = 1;

 public MaBaseSQLite(Context context) {
 super(context, NOM_BDD, null, VERSION_BDD);
 }

 @Override
 public void onCreate(SQLiteDatabase sqLiteDatabase) {
 }

 @Override
 public void onUpgrade(SQLiteDatabase sqLiteDatabase, int oldVersion, int newVersion) {
 }
}
```

# CRÉER UNE BDD

- Trois méthodes à ré-implémenter :
  - Le constructeur : qui a besoin du context, d'un nom et d'un numéro de version.
  - **Oncreate()**: qui nous donnera un objet SQLiteDatabase à peupler
  - **OnUpgrade()**: comportement à adopter si la version de la base change. Il possède comme argument la nouvelle version, l'actuel ainsi qu'un objet SQLiteDatabase. Pour faire la mise à jour de la base.

# Exemple : Creation de la table

```
public class MaBaseSQLite extends SQLiteOpenHelper {

 private static final String NOM_BDD = "mabase.db";
 private static final int VERSION_BDD = 1;

 public MaBaseSQLite(Context context) {
 super(context, NOM_BDD, null, VERSION_BDD);
 }

 @Override
 public void onCreate(SQLiteDatabase sqLiteDatabase) {
 //on cre la table a partir de la requete ecrite dans la variable CREATE_ELEVE_TABLE
 sqLiteDatabase.execSQL(EleveBDDManager.CREATE_ELEVE_TABLE);
 }

 @Override
 public void onUpgrade(SQLiteDatabase sqLiteDatabase, int oldVersion, int newVersion) {

 if (oldVersion < 47) {
 sqLiteDatabase.execSQL("DROP TABLE " + EleveBDD.TABLE_ELEVE + ";");
 onCreate(sqLiteDatabase);
 }
 if (oldVersion < 54) {
 db.execSQL(AnchorsTable.QUERY.Alter_54);
 }
 if (oldVersion < 59) {
 db.execSQL(AnchorsTable.QUERY.Alter_58);
 }
 }
}
```

# OUVRIR UNE CONNEXION

- Créer l'instance dans le `onCreate` de la classe Application

```
maBaseSQLite = new MaBaseSQLite(this);
```

- Selon le contexte, appel des 2 méthodes
  - `getReadableDatabase()`: ouvre la base en lecture seule
  - `getWritableDatabase()` : ouvre une connexion et accepte l'écriture.

# MYAPPLICATION

```
public class MyApplication extends Application {

 private static MaBaseSQLite maBaseSQLite;

 public static MaBaseSQLite getMaBaseSQLite() {
 return maBaseSQLite;
 }

 @Override
 public void onCreate() {
 super.onCreate();
 maBaseSQLite = new MaBaseSQLite(this);
 }
}
```

# Exemple : Insertion

```
public class EleveBDDManager {

 public static final String TABLE_ELEVE = "Eleve";
 private static final String COL_ID = "ID";
 private static final String COL_PRENOM = "Prenom";
 private static final String COL_NOM = "Nom";

 public static final String CREATE_ELEVE_TABLE = "CREATE TABLE " + TABLE_ELEVE + " (" +
 COL_ID + " INTEGER PRIMARY KEY AUTOINCREMENT, "
 + COL_PRENOM + " TEXT NOT NULL, " + COL_NOM + " TEXT NOT NULL);";

 public static void insertEleve(Eleve eleve) {

 //Ouvrir la base en écriture
 SQLiteDatabase bdd = MyApplication.getMaBaseSQLite().getWritableDatabase();

 //Création d'un ContentValues (fonctionne comme une HashMap)
 ContentValues values = new ContentValues();
 //on lui ajoute une valeur associée à une clé (qui est le nom de la colonne dans
 //laquelle on veut mettre la valeur)
 values.put(COL_PRENOM, eleve.getPrenom());
 values.put(COL_NOM, eleve.getNom());
 //on insère l'objet dans la BDD via le ContentValues
 eleve.setId(bdd.insert(TABLE_ELEVE, null, values));
 if (eleve.getId() == -1) {
 //gestion erreur
 }
 bdd.close();
 }
}
```

# Exemple : Update - Remove

```
Public static int updateEleve(Eleve eleve) {
 //Ouvrir la base en écriture
 SQLiteDatabase bdd = MyApplication.getMaBaseSQLite().getWritableDatabase();

 //La mise à jour d'un élève dans la BDD fonctionne plus ou moins comme une
insertion
 //il faut simplement préciser quel élève on doit mettre à jour grâce à l'ID
 ContentValues values = new ContentValues();
 values.put(COL_PRENOM, eleve.getPrenom());
 values.put(COL_NOM, eleve.getNom());
 int result = bdd.update(TABLE_ELEVE, values, COL_ID + " = " + eleve.getId(),
null);
 bdd.close();
 return result;
}

public static int removeEleveWithID(int id) {
 SQLiteDatabase bdd = MyApplication.getMaBaseSQLite().getWritableDatabase();
 //Suppression d'un élève de la BDD grâce à l'ID
 int result = bdd.delete(TABLE_ELEVE, COL_ID + " = " + id, null);
 bdd.close();
 return result;
}
```

# UTILISATION D' UN CURSOR

- Retourner le nombre d'enregistrements : getCount()
- Itération du cursor avec les méthodes “moveToFirst()”, “moveToNext()” et “isAfterLast()”
- Retourner les noms des colonnes avec getColumnNames(), pour les afficher
- getColumnIndex(), nous renvoie l'index du nom de la colonne donné en paramètre
- Récupération des informations de la colonne avec getString(),getInt(), etc.
- Libérer le curseur avec la méthode close()

# Exemple : Get

```
public static List<Eleve> getAllEleves() {
 SQLiteDatabase bdd = MyApplication.getMaBaseSQLite().getReadableDatabase();
 Cursor c = bdd.query(TABLE_ELEVE,
 new String[] { COL_ID, COL_PRENOM, COL_NOM },
 null, null, null, null, null);
 List<Eleve> result = cursorToEleves(c);
 bdd.close();
 return result;
}

//Cette méthode permet de convertir un cursor en list d'Eleve
private static List<Eleve> cursorToEleves(Cursor c) {
 ArrayList<Eleve> eleveListe = new ArrayList<Eleve>();

 if (c != null) {
 //On se place sur le premier élément
 if (c.moveToFirst()) {
 do {
 Eleve eleveBean = new Eleve(c.getString(c.getColumnIndex(COL_NOM)),
 c.getString(c.getColumnIndex(COL_PRENOM)), c.getLong(c.getColumnIndex(COL_ID)));
 eleveListe.add(eleveBean);
 } while (c.moveToNext());
 }
 }

 //On ferme le cursor
 c.close();

 //On retourne la liste
 return eleveListe;
}
```

# VISUALISER SA BASE DE DONNÉES



## ○ SQLiteStudio

- <http://sqlitestudio.pl/>
- Permet de travailler sur **la copie** de sa base

Il suffit d'importer le fichier .sqlite pour voir sa base de données, ses tables, leurs contenus et effectuer des requêtes dessus.



## ○ Stetho

- Base de donnée en temps réel sur Chrome

# ACCÉDER À SA BASE DE DONNÉES

- Android Device Monitor : File Explorer
  - data/data/<app\_package>/databases/nom.sqlite
- Restriction
  - Impossible sans un téléphone « root » ou Genymotion
- Contournement
  - Copier le fichier dans un répertoire accessible sans root.
  - Attention depuis Marshmallow la permission est à la volée.

Manifest.permission.***WRITE\_EXTERNAL\_STORAGE***

# Copier la base dans Download

```
/** Copier la base de donnée de l'application dans le répertoire download */
public static void CopySQLiteBaseToDownload(Context context, String dbName) {
 //OU se trouve la base de donnée
 File database = new File("data/data/" + context.getPackageName() + "/databases/" + dbName);
 //Ou on la copie
 File downloadDirectory = new
File(Environment.getExternalStoragePublicDirectory(Environment.DIRECTORY_DOWNLOADS), "/" + dbName);

 try {
 if (database.exists()) {
 if (!downloadDirectory.exists()) {
 downloadDirectory.createNewFile();
 }
 InputStream in = new FileInputStream(database);
 OutputStream out = new FileOutputStream(downloadDirectory);

 // Copy the bits from instream to outstream
 byte[] buf = new byte[1024];
 int len;

 while ((len = in.read(buf)) > 0) {
 out.write(buf, 0, len);
 }
 in.close();
 out.close();

 Toast.makeText(context, "Le fichier a été copié", Toast.LENGTH_LONG).show();
 } else {
 Toast.makeText(context, "Erreur lors de la copie", Toast.LENGTH_LONG).show();
 }
 } catch (IOException e) {
 e.printStackTrace();
 Toast.makeText(context, "Erreur lors de la copie", Toast.LENGTH_LONG).show();
 }
}

<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
```



# SQLITESTUDIO

The screenshot shows the SQLiteStudio interface with several windows open:

- Databases Window:** Shows the structure of the "ULM-db (SQLite 3)" database, including tables like MATCH\_BEAN, PLAYER\_BEAN, and TEAM\_PLAYER.
- SQL editor 1:** Contains the following SQL query:

```
1 select *
2 FROM PLAYER_BEAN PB
3 LEFT JOIN (
4 Select *
5 FROM TEAM_PLAYER TP
6 WHERE TP.TEAM_ID = 1
7) TP ON TP.PLAYER_ID = PB._id
8 WHERE TP.PLAYER_ID IS NULL
9
10
```
- SQL editor 2:** Contains the following SQL query:

```
1 Select *
2 FROM TEAM_PLAYER TP
3 WHERE TP.TEAM_ID = 1
4
```
- SQL editor 3:** Contains the following SQL query:

```
1 Select *
2 FROM PLAYER_BEAN PB
3 INNER JOIN TEAM_PLAYER TP ON TP.PLAYER_ID = PB._id
4 INNER JOIN MATCH_BEAN MB ON MB.TEAM_ID = TP.TEAM_ID
5 WHERE MB._id = 1
6 ORDER BY PB.NAME
7
```

222

Anthony Monteiro



## STETHO



- Permet de voir sa base en temps réél
  - <http://facebook.github.io/stetho/>
  - compile 'com.facebook.stetho:stetho:1.4.2'

# TP : SQLITE

- Rendre persistant la liste d'élève de la listView
- Doc SQLite :  
<http://developer.android.com/reference/android/database/sqlite/SQLiteDatabase.html>
- Solution : Module DAOSQLISolution

# CONTENTPROVIDER

- Concept permettant à une application de partager ses ressources avec d'autres applications (clients)
- Etapes
  - Créer une classe *héritant* de ContentProvider
  - Implémenter les méthodes qui font le lien avec la BDD
  - Référencer le provider dans l'AndroidManifest

# CONTENT PROVIDER

```
public class EleveContentProvider extends ContentProvider {

 @Override
 public boolean onCreate() {...}

 @Override
 public String getType(Uri uri) {...}

 @Override
 public Cursor query(Uri uri, String[] projection, String selection, String[] selectionArgs,
String sortOrder) {...}

 @Override
 public Uri insert(Uri uri, ContentValues values) {...}

 @Override
 public int delete(Uri uri, String selection, String[] selectionArgs) {...}

 @Override
 public int update(Uri uri, ContentValues values, String selection, String[] selectionArgs)
}
```

# CONTENTPROVIDER : RÉGLAGES

```
public class EleveContentProvider extends ContentProvider {
 private MaBaseSQLite maBaseSQLite;

 @Override
 public boolean onCreate() {
 maBaseSQLite = new MaBaseSQLite(getApplicationContext());
 return true;
 }

 public String getType(Uri uri) {
 return "vnd.android.cursor.item/vnd.content.provider.eleve"
 }
 public Cursor query(Uri uri, String[] projection, String selection, String[] selectionArgs,
String sortOrder) {...}
 public Uri insert(Uri uri, ContentValues values) {...}
 public int delete(Uri uri, String selection, String[] selectionArgs) {...}
 public int update(Uri uri, ContentValues values, String selection, String[] selectionArgs) {...}
}
```

# CONTENTPROVIDER : GETID

```
public class EleveContentProvider extends ContentProvider {
 private MaBaseSQLite maBaseSQLite;
 public boolean onCreate() {...}
 public String getType(Uri uri) {...}

 private long getId(Uri uri) {
 String lastPathSegment = uri.getLastPathSegment();
 if (lastPathSegment != null) {
 try {
 return Long.parseLong(lastPathSegment);
 }
 catch (NumberFormatException e) {
 Log.e("TAG", "Number Format Exception : " + e);
 }
 }
 return -1;
 }
 ...
}
```

# CONTENTPROVIDER : GET

```
public class EleveContentProvider extends ContentProvider {
 private MaBaseSQLite maBaseSQLite;
 public boolean onCreate() {...}
 public String getType(Uri uri) {...}

 @Override
 public Cursor query(Uri uri, String[] projection, String selection, String[] selectionArgs,
String sortOrder) {
 //Effectuer une requete
 long id = getId(uri);
 SQLiteDatabase db = maBaseSQLite.getReadableDatabase();
 if (id < 0) {
 return db.query(EleveBDDManager.TABLE_ELEVE, projection, selection, selectionArgs,
 null, null, sortOrder);
 }
 else {
 return db.query(EleveBDDManager.TABLE_ELEVE, projection,
 EleveBDDManager.COL_ID + "=" + id, null, null, null, null);
 }
 }
 ...
}
```

# CONTENTPROVIDER : INSERT

```
public class EleveContentProvider extends ContentProvider {
 private MaBaseSQLite maBaseSQLite;

 ...
 @Override
 public Uri insert(Uri uri, ContentValues values) {
 SQLiteDatabase db = maBaseSQLite.getWritableDatabase();
 try {
 long id = db.insertOrThrow(EleveBDDManager.TABLE_ELEVE, null, values);
 if (id == -1) {
 throw new RuntimeException(String.format(
 "Failed to insert [%s] for unknown reasons.", values));
 }
 else {
 return ContentUris.withAppendedId(uri, id);
 }
 }
 finally {
 db.close();
 }
 }
 ...
}
```

# CONTENTPROVIDER : DELETE

```
public class EleveContentProvider extends ContentProvider {
 private MaBaseSQLite maBaseSQLite;

 ...
 @Override
 public int delete(Uri uri, String selection, String[] selectionArgs) {
 long id = getId(uri);
 SQLiteDatabase db = maBaseSQLite.getWritableDatabase();
 try {
 if (id < 0) {
 return db.delete(EleveBDDManager.TABLE_ELEVE, selection, selectionArgs);
 }
 else {
 return db.delete(EleveBDDManager.TABLE_ELEVE,
 EleveBDDManager.COL_ID + " = " + id, selectionArgs);
 }
 }
 finally {
 db.close();
 }
 }
 ...
}
```

# CONTENTPROVIDER : UPDATE

```
public class EleveContentProvider extends ContentProvider {
 private MaBaseSQLite maBaseSQLite;

 ...

 @Override
 public int update(Uri uri, ContentValues values, String selection, String[] selectionArgs) {
 long id = getId(uri);
 SQLiteDatabase db = maBaseSQLite.getWritableDatabase();

 try {
 if (id < 0) {
 return db.update(EleveBDDManager.TABLE_ELEVE, values, selection, selectionArgs);
 }
 else {
 return db.update(EleveBDDManager.TABLE_ELEVE, values,
 EleveBDDManager.COL_ID + " = " + id, null);
 }
 }
 finally {
 db.close();
 }
 }
 ...
}
```

# CONTENTPROVIDER : ANDROIDMANIFEST

```
<application ...>
 <!-- Déclaration du provider, sur la classe
 Authorities = Url d'appel du provider pour le client -->
 <provider
 android:name=".dao.EleveContentProvider"
 android:authorities="com.example.exemple.dao.elevecontentorovider"
 android:exported="true"
 />
</application>
```

# CONTENTPROVIDER : CLIENT

- Application client qui va appeler le Provider de notre 1<sup>er</sup> application.
- Prérequis
  - Connaitre l'URL d'appel
  - Connaitre les colonnes de la table.
- Rien à faire dans l'AndroidManifest

# CONTENTPROVIDER : CLIENT

```
public class EleveBDDManager {

 // URI utilisé pour accéder au ContentProvider, attribue Authorities de la déclaration du
 // ContentProvider dans l'AndroidManifest.
 public static final Uri DAOURI = Uri
 .parse("content://com.example.exemple.dao.elevecontentorovider");

 public static final String COL_ID = "ID";
 public static final String COL_PRENOM = "Prenom";
 public static final String COL_NOM = "Nom";

 // Ajoute un élève à la base
 public static void addEleve(Context context) {
 ContentValues contact = new ContentValues();
 contact.put(COL_PRENOM, "From");
 contact.put(COL_NOM, "Client");
 context.getContentResolver().insert(DAOURI, contact);
 }
 public static ArrayList<Eleve> getEleves(Context context) {...}
}
```

# CONTENTPROVIDER : CLIENT

```
public class EleveBDDManager {

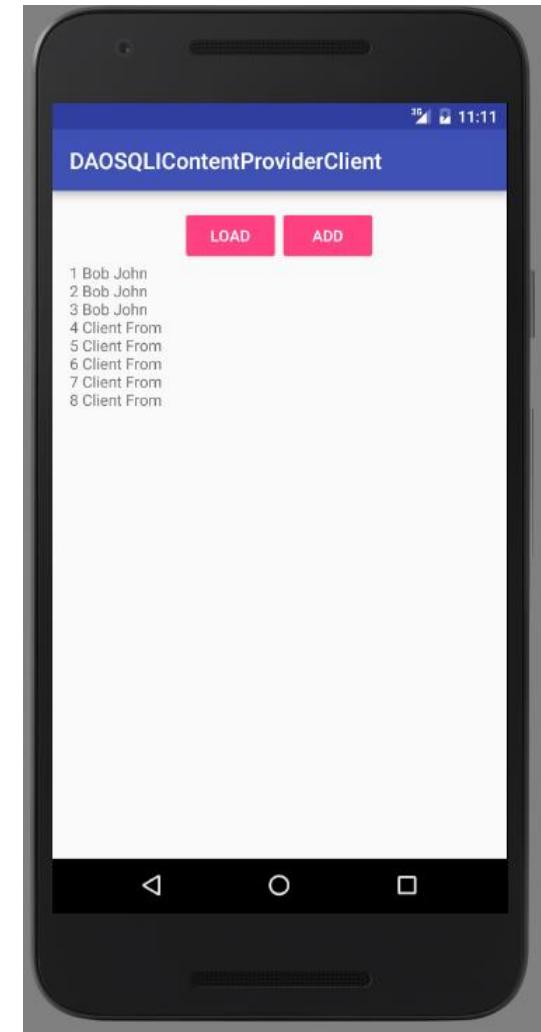
 public static final Uri DAOURI = Uri
 .parse("content://com.example.exemple.dao.elevecontentorovider");
 public static final String COL_ID = "ID";
 public static final String COL_PRENOM = "Prenom";
 public static final String COL_NOM = "Nom";

 public static void addEleve(Context context) {...}

 public static ArrayList<Eleve> getEleves(Context context) {
 String columns[] = new String[]{EleveBDDManager.COL_ID, EleveBDDManager.COL_NOM,
EleveBDDManager.COL_PRENOM};
 Cursor cur = context.getContentResolver().query(DAOURI, columns, null, null, null);
 return cursorToEleves(cur);
 }
}
```

# TP : CONTENTPROVIDER

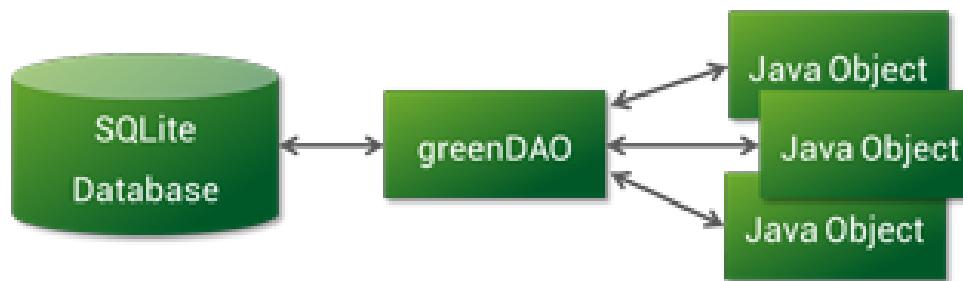
- Rendre votre Base de donnée accessible depuis l'extérieur
- Créer un client qui ajoute un élève à la base et qui charge la liste des élèves



237

# GREENDAO

- Un projet open source s'occupant du mapping (ORM).
- <http://greenrobot.org/greendao/documentation/how-to-get-started/>



# GREENDAO DAOGENERATOR

- Ajouter dans le build.gradle du module
- Puis build -> Make Project

```
buildscript {
 repositories {
 mavenCentral()
 }
 dependencies {
 classpath 'org.greenrobot:greendao-gradle-plugin:3.2.2'
 }
}
apply plugin: 'org.greenrobot.greendao'

android {
 ...
}

//Version de la base
greendao {
 schemaVersion 2
}

dependencies {
 compile 'org.greenrobot:greendao:3.2.+'
}
```

# GREENDAO DAOGENERATOR

- Transformer ses beans en table
- Ajouter l'annotation `@Entity(...)`

```
@Entity(active = true,
 //nameInDb = "ELEVE", nom de la table si different
 // Define indexes spanning multiple columns here.
 // indexes = {
 // @Index(value = "name DESC", unique = true)
 // },
 generateConstructors = true,
 generateGettersSetters = true)

public class Eleve {

 @Id(autoincrement = true)
 private Long id;
 private String nom;
 private String prenom;
}
```

# GREENDAO SETUP

```
public class MyApplication extends Application {

 private static DaoSession daoSession;

 @Override
 public void onCreate() {
 setupDatabase();
 }

 private void setupDatabase() {
 DaoMaster.DevOpenHelper helper = new DaoMaster.DevOpenHelper(this, "mytable-db");
 Database db = helper.getWritableDatabase();
 daoSession = new DaoMaster(db).newSession();
 }

 public static DaoSession getDaoSession() {
 return daoSession;
 }
}
```

# GREENDAO UTILISATION

```
public class EleveBDDManager {

 public static void insertOrUpdate(Eleve eleve) {
 getEleveDao().insertOrReplace(eleve);
 }
 public static void clearEleve() {
 getEleveDao().deleteAll();
 }
 public static void deleteEleveWithId(long id) {
 getEleveDao().deleteByKey(id);
 }
 public static Eleve getEleveForId(long id) {
 return getEleveDao().load(id);
 }
 public static ArrayList<Eleve> getAllEleve() {
 return (ArrayList<Eleve>) getEleveDao().loadAll();
 }
 private static EleveDao getEleveDao() {
 return MyApplication.getDaoSession().getEleveDao();
 }
}
```

# GREENDAO UTILISATION

## ○ Le constructeur de requête.

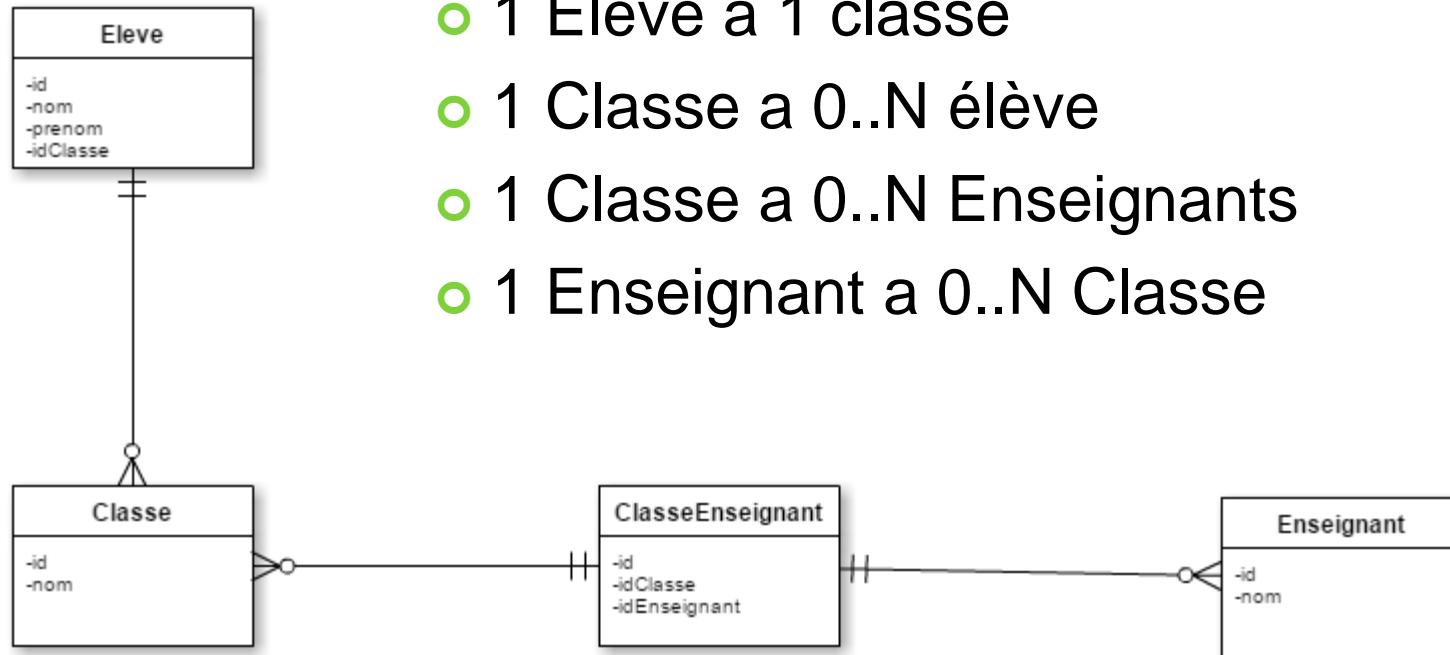
```
/**
 * Retourne une liste d'élève en fonction du prénom
 * @param context
 * @return
 */

public static List<Eleve> getEleveByPrenom(String prenom) {
 return getEleveDao().queryBuilder().where(EleveDao.Properties.Prenom.eq(prenom)).list();
}
```

# TP : GREENDAO

- Rendre persistant la liste d'élève de la listView avec GreenDAO

# GREENDAO : RELATIONNELLE



- 1 Elève a 1 classe
- 1 Classe a 0..N élève
- 1 Classe a 0..N Enseignants
- 1 Enseignant a 0..N Classe

# GREENDAO : RELATIONNELLE

- 1 Elève possède 1 classe ( 0..1)

```
@Entity(...)
public class Eleve {
 ...
 //Jointure avec la Classe : 1 élève à 1 classe
 @ToOne(joinProperty = "classeId")
 private Classe classe;
 private long classeId;
}
```

- Utilisation

```
Eleve eleve = EleveBddManager.getEleve(id);
Classe classe = eleve.getClasse();
```

# GREENDAO : RELATIONNELLE

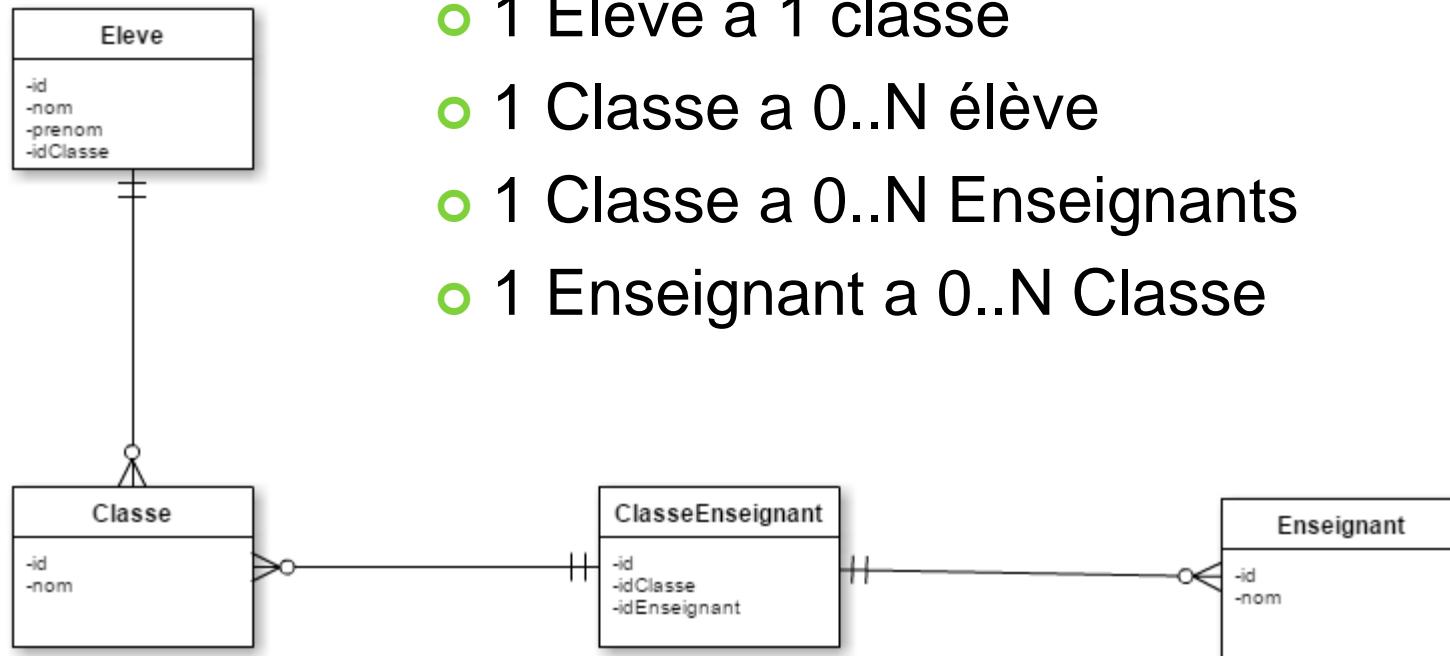
- 1 classe possède un ensemble d'élèves (0..N)

```
@Entity(...)
public class Classe {
 ...
 //Jointure Inverse Avec l'élève
 @ToMany(referencedJoinProperty = "classeId")
 @OrderBy("nom ASC")
 private List<Eleve> eleves;
}
```

- Utilisation

```
Classe classe = ClassesBddManager.getClasse(id);
List<Eleve> eleves = classe.getEleves();
```

# GREENDAO : RELATIONNELLE



# GREENDAO : RELATIONNELLE

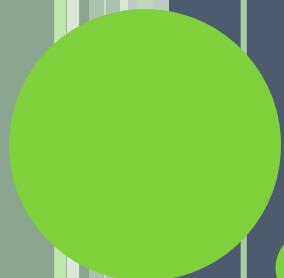
## ○ 1 classe possède un ensemble d'enseignant(0..N)

```
@ToMany
@JoinEntity()
 //Table intermediaire
 entity = ClasseEnseignant.class,
 //Id representant cette table dans la table intermediaire
 sourceProperty = "classeId",
 //Id representant la table voulu dans la table intermediraire
 targetProperty = "enseignantId"
)

private List<Enseignant> enseignantList;
```

## ○ Utilisation

```
Classe classe = ClassesBddManager.getClasse(id);
List<Enseignant> enseignants = classe.getEnseignantList();
```



GOOGLE MAPS

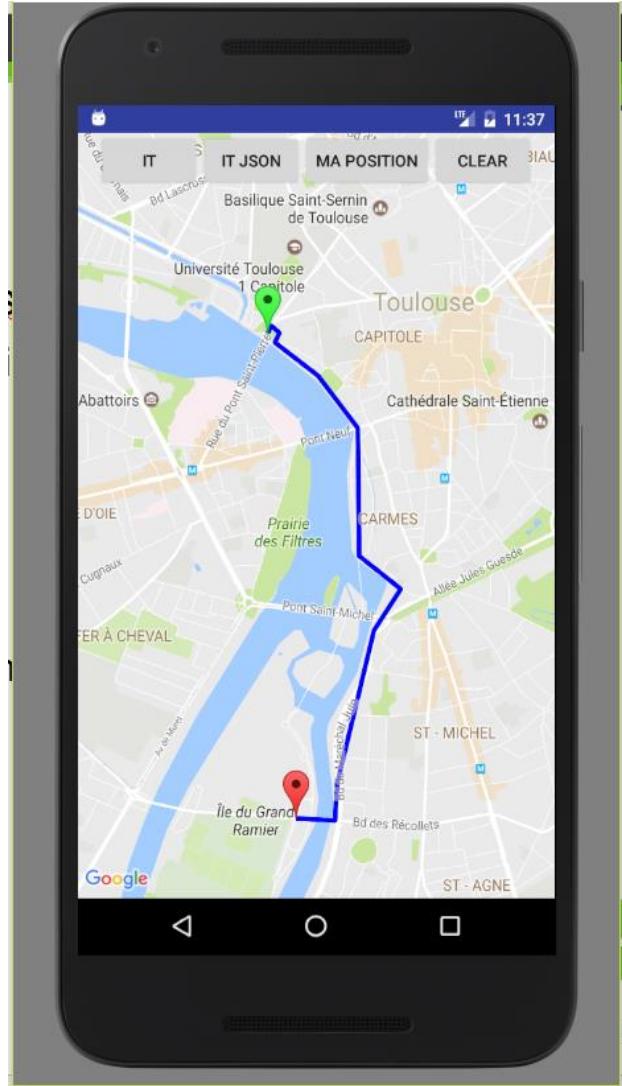
250



# GOOGLE MAPS

- Qu'est ce que Google Maps ?
  - Service de cartographie en ligne
  - Lancé en 2004 au USA
- Utilités?
  - Basées sur la localisation
  - Construire des cartes dynamiques pour les applications

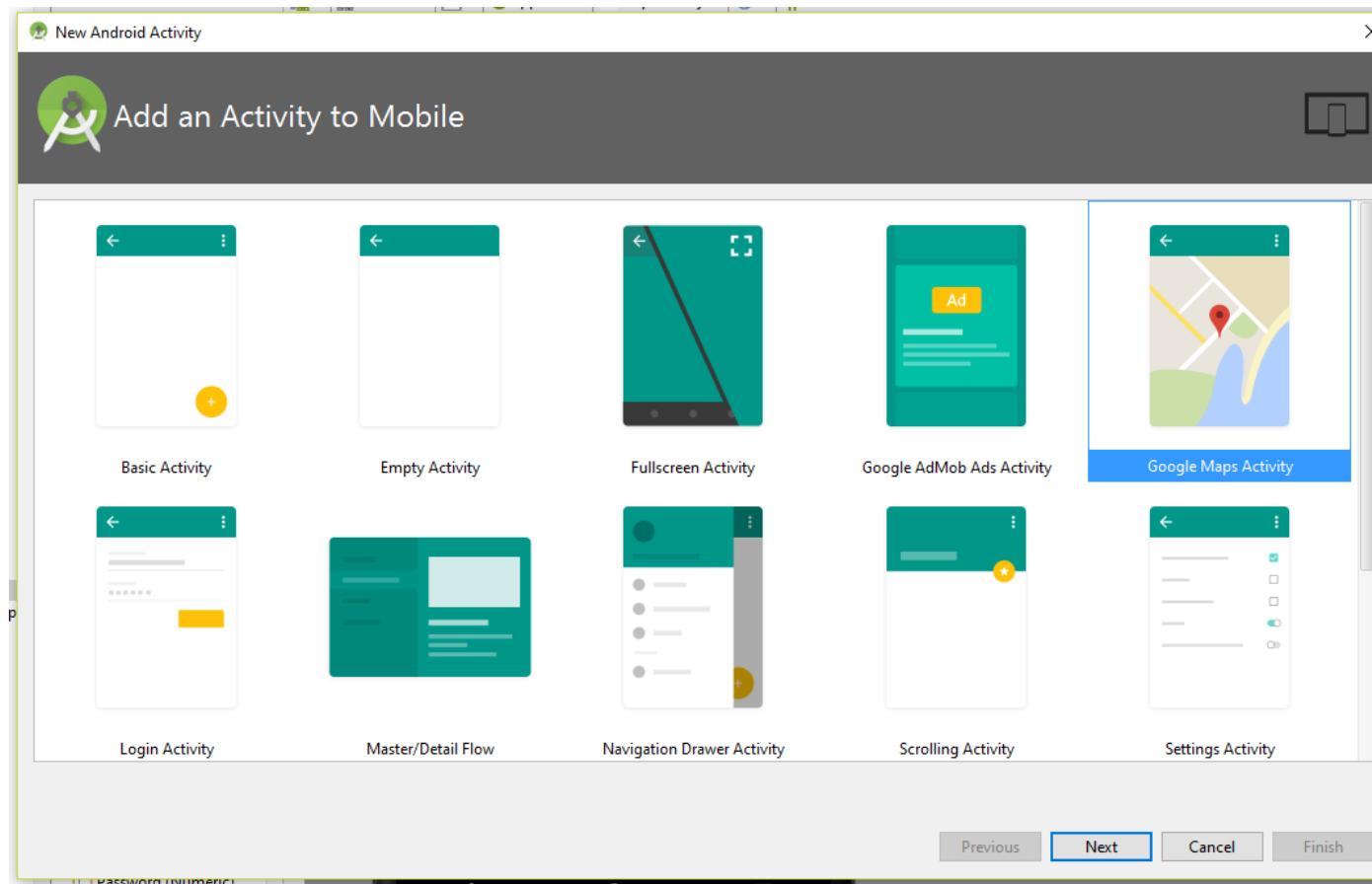
# GOOGLE MAPS : EXEMPLE



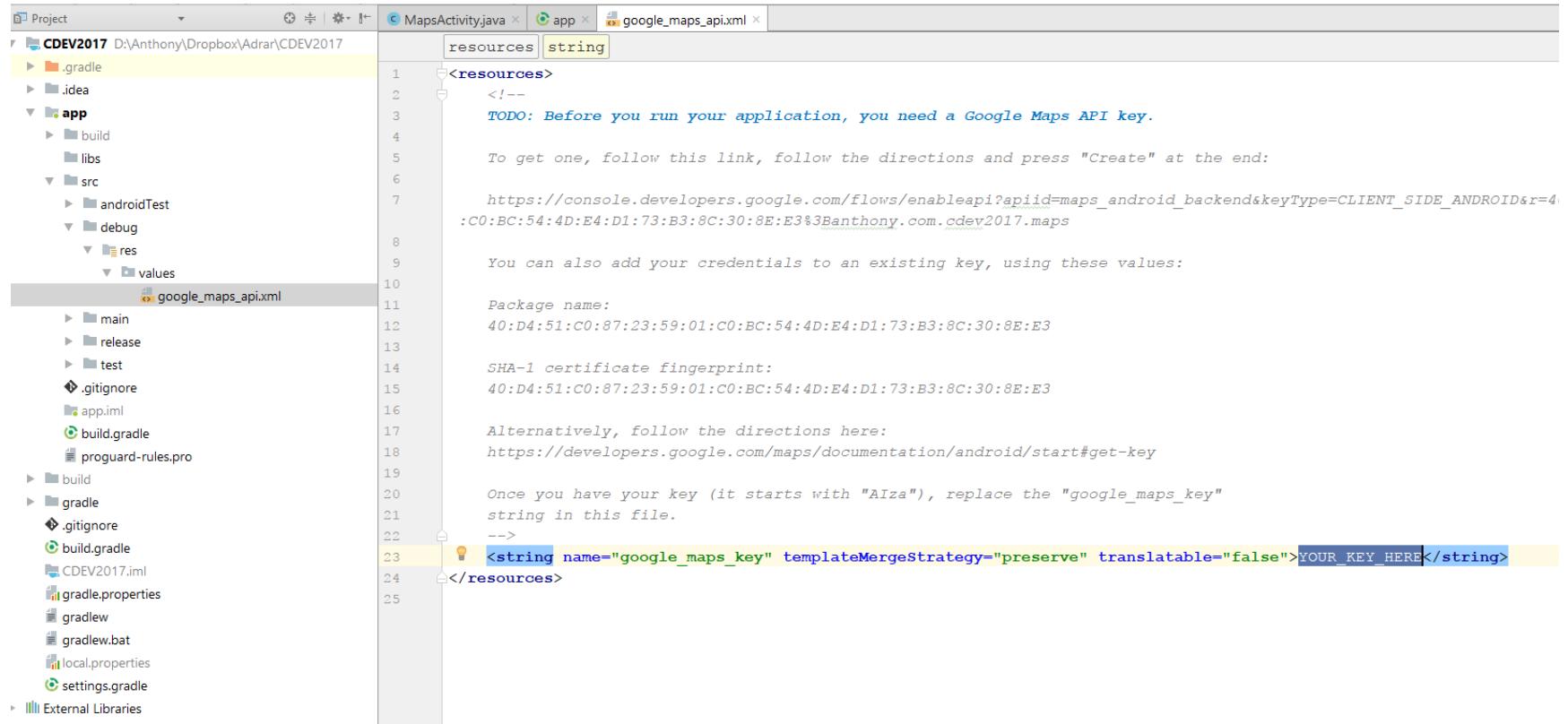
252

# GOOGLE MAPS : CRÉATION

Attention créer l'Activity dans le package de départ



# GOOGLE MAPS : LA CLÉ



```
resources string
<resources>
 <!--
 TODO: Before you run your application, you need a Google Maps API key.

 To get one, follow this link, follow the directions and press "Create" at the end:
 https://console.developers.google.com/flows/enableapi?apiId=maps_android_backend&keyType=CLIENT_SIDE_ANDROID&r=4
 :C0:BC:54:4D:E4:D1:73:B3:8C:30:8E:E3%3Banthony.com.cdev2017.maps

 You can also add your credentials to an existing key, using these values:

 Package name:
 40:D4:51:C0:87:23:59:01:C0:BC:54:4D:E4:D1:73:B3:8C:30:8E:E3

 SHA-1 certificate fingerprint:
 40:D4:51:C0:87:23:59:01:C0:BC:54:4D:E4:D1:73:B3:8C:30:8E:E3

 Alternatively, follow the directions here:
 https://developers.google.com/maps/documentation/android/start#get-key

 Once you have your key (it starts with "AIza"), replace the "google_maps_key"
 string in this file.
 -->
 <string name="google_maps_key" templateMergeStrategy="preserve" translatable="false">YOUR_KEY_HERE</string>
</resources>
```

# GOOGLE MAPS : PROBLÈMES



- Correction :

```
compile 'com.google.android.gms:play-services-maps:9.+'
```

255

# GOOGLE MAPS : ETAPES

- Générer la clé de votre projet, lien et explication :
  - src/debug/res/values/google\_maps\_api.xml

- Dans le code

```
private GoogleMap mMap;

@Override
protected void onCreate(Bundle savedInstanceState) {
 //...
 //Récupère la Map et l'active
 SupportMapFragment mapFragment = (SupportMapFragment) getSupportFragmentManager()
 .findFragmentById(R.id.map);
 mapFragment.getMapAsync(this);
 //...
}

@Override
public void onMapReady(GoogleMap googleMap) {
 //Appelé quand la map est chargée
 mMap = googleMap;
}
```

# GOOGLE MAPS : UTILISATION

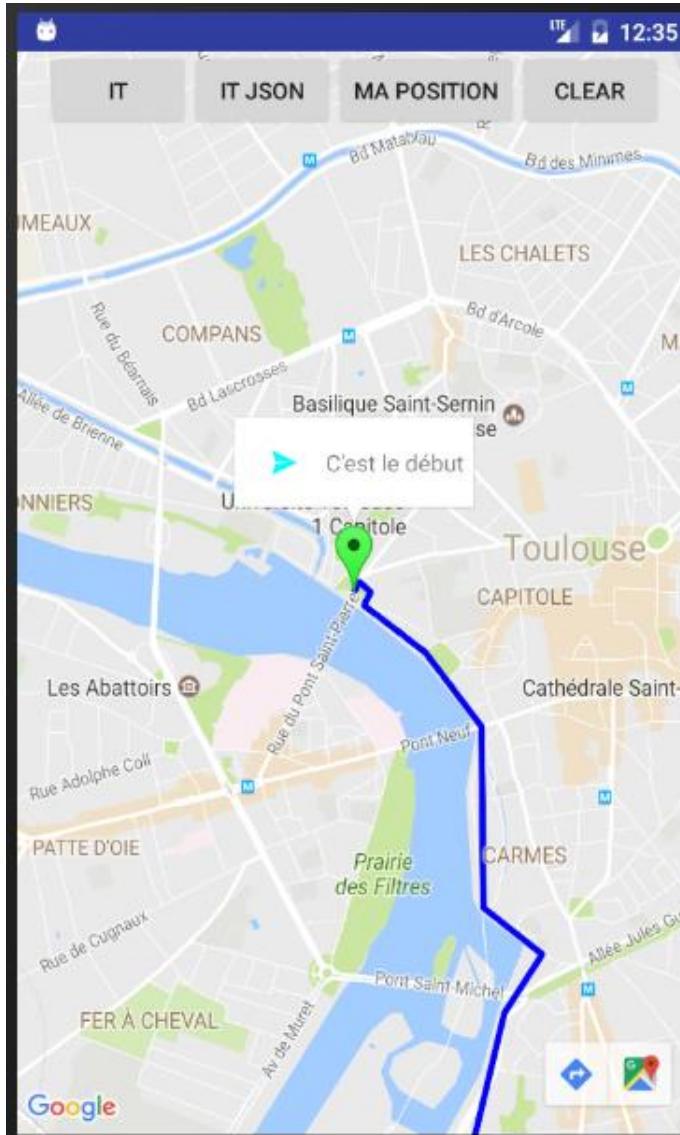
```
<RelativeLayout
 xmlns:android="http://schemas.android.com/apk/res/android"
 android:layout_width="match_parent"
 android:layout_height="match_parent"
 >

<fragment
 android:id="@+id/map"
 android:name="com.google.android.gms.maps.SupportMapFragment"
 android:layout_width="match_parent"
 android:layout_height="match_parent"
 />

<Button
 android:id="@+id/bt_go"
 android:layout_centerHorizontal="true"
 android:layout_width="wrap_content"
 android:layout_height="wrap_content"
 android:text="Go"/>

</RelativeLayout>
```

# GOOGLE MAPS : MARKER



258

# GOOGLE MAPS : MARKER

- L'objet LatLng

```
LatLng position = new LatLng(43.59999, 1.43333);
```

- L'ajouter à un bean Ville

```
public class MaVille {
 private String nom;
 private LatLng position;

 public MaVille(String nom, LatLng position) {
 this.nom = nom;
 this.position = position;
 }
 // ...
}
```

# GOOGLE MAPS : MARKER

## ○ Afficher un marker

```
MaVille toulouse = new MaVille("Toulouse", new LatLng(43.59999, 1.43333));

MarkerOptions markerToulouse = new MarkerOptions();
markerToulouse.position(toulouse.getPosition());
markerToulouse.title(toulouse.getNom());
markerToulouse.icon(BitmapDescriptorFactory.defaultMarker(BitmapDescriptorFactory.HUE_GREEN));

mMap.addMarker(markerToulouse)
 .setTag(toulouse);
```

# GOOGLE MAPS : MARKER

- Afficher sa position en temps réel sur la carte
  - Penser à gérer la permission de localisation

```
if (mMap != null) {
 mMap.setMyLocationEnabled(true);
}
```

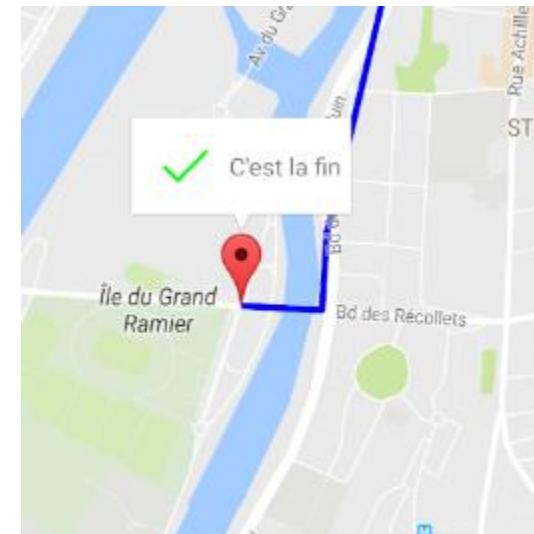
# GOOGLE MAPS : POPUP DES MARKERS

- Apres un click sur un marker, il affiche une popup
  - Par défaut n'affichera que le titre
  - Customisable

```
public void onMapReady(GoogleMap googleMap) {
 mMap = googleMap;
 //On indique qu'on souhaite customiser
 //les popups des markers
 //Génère getInfoWindow et getInfoContents
 mMap.setInfoWindowAdapter(this);
}

//Affichera la vue retournée, si null appellera getInfoContents
@Override
public View getInfoWindow(Marker marker) {
 return null;
}

//Affichera dans la fenêtre la view retournée
@Override
public View getInfoContents(Marker marker) {
 return null;
}
```



# GOOGLE MAPS : CUSTOMISER LA POPUP

```
//Affichera dans la fenêtre la view retournée
@Override
public View getInfoContents(Marker marker) {
 //Layout de la fenêtre
 View view = LayoutInflater.from(this).inflate(R.layout.marker_cellule, null);

 TextView tv = (TextView) view.findViewById(R.id.tv);
 ImageView iv = (ImageView) view.findViewById(R.id.iv);

 MaVille maVille = (MaVille) marker.getTag();
 tv.setText(maVille.getNom());
 iv.setImageResource(maVille.getLogo());
 iv.setColorFilter(Color.CYAN);

 return view;
}
```

Attention dans le XML à utiliser src et pas srcCompat sinon l'image n'apparaîtra pas

263

# GOOGLE MAPS : CLICK SUR LA POPUP

## ○ S'abonner au click sur la popup d'un marker

```
@Override
public void onMapReady(GoogleMap googleMap) {
 mMap = googleMap;

 //S'abonner aux clicks sur les markers
 //Génère onInfoWindowClick
 mMap.setOnInfoWindowClickListener(this);
}

//Le callback généré
@Override
public void onInfoWindowClick(Marker marker) {
 MaVille ville = (MaVille) marker.getTag();
 //Ferme la fenêtre
 marker.hideInfoWindow();
}
```

# GOOGLE MAPS : DÉPLACER

## ○ Update Google Services

```
//Centrer la carte sur un point (Sans animation)
mMap.moveCamera(CameraUpdateFactory.newLatLng(maVille.getPosition()));

//Centrer la carte sur un point (Avec une animation)
mMap.animateCamera(CameraUpdateFactory.newLatLng(maVille.getPosition()));
```

## ○ Effacer les markers de la carte

```
mMap.clear();
```

## ○ Changement le niveau de zoom de la carte

```
mMap.animateCamera(CameraUpdateFactory.zoomBy(12));
```

## ○ La suite sur la doc officiel de google

- <https://developers.google.com/maps/documentation/android-api/intro>

265

# CENTRER SUR UN GROUPE

## ○ Centre la caméra sur un groupe d'icônes

```
LatLngBounds.Builder latLngBounds = new LatLngBounds.Builder();
latLngBounds.include(new LatLng(1.1,1.2));
latLngBounds.include(new LatLng(1.1,1.3));
latLngBounds.include(new LatLng(1.1,1.4));
map.animateCamera(CameraUpdateFactory.newLatLngBounds(latLngBounds.build(), 100));
```

Attention au moins 2 éléments sinon ça crash !!

# CLUSTERS

- [https://developers.google.com/maps/documentation  
/android-api/utility/marker-clustering?hl=fr](https://developers.google.com/maps/documentation/android-api/utility/marker-clustering?hl=fr)



267

# CLUSTERS

- Configuration :

```
dependencies {
 compile 'com.google.maps.android:android-maps-utils:0.4+'
}
```

- Faire en sorte que notre Bean/Pojo devienne un Cluster en implementant ClusterItem

Examinons les étapes plus en détail : Pour créer notre regroupement simple de dix marqueurs, créez tout d'abord une classe `MyItem` qui implémente `ClusterItem`.

```
public class MyItem implements ClusterItem {
 private final LatLng mPosition;

 public MyItem(double lat, double lng) {
 mPosition = new LatLng(lat, lng);
 }

 @Override
 public LatLng getPosition() {
 return mPosition;
 }
}
```

# CLUSTERS MANAGER

```
//Le clusterManager est l'objet qui manipule les clusters.
private ClusterManager<MyItem> mClusterManager;

public void onMapReady(Bundle savedInstanceState) {

 ...

 // Initialize the manager with the context and the map.
 mClusterManager = new ClusterManager<MyItem>(this, mMap);

 // Point the map's listeners at the listeners implemented by the cluster
 // manager.
 mMap.setOnCameraIdleListener(mClusterManager);
 mMap.setOnMarkerClickListener(mClusterManager);

}
```

# CLUSTERS MANAGER

```
//Ajouter des clusters au ClusterManager.
mClusterManager.addItem(myItem);
```

```
//Les retirer
mClusterManager.clearItems();
```

# GOOGLE MAPS

## ○ DirectionAPI

- <https://developers.google.com/maps/documentation/directions/start?hl=fr>

271



272

## COMMUNICATION ENTRE 2 ACTIVITY

# UTILISATION DES INTENTS

- Nous l'utilisons pour lancer nos activités :

```
Intent intent = new Intent(this, SecondActivity.class);
```

- Une fois l'intent créé, nous le lançons dans notre activité

- 2 méthodes existent :

- startActivity(Intent i)
- startActivityForResult(Intent i, int req)

# UTILISATION DES INTENTS

- Possibilité d'ajouter une information de l'activité parent aux sous-activités
  - `intent.putExtra("clé", valeur);`
- Il est donc possible de transmettre tout type primitif, même des objets “serializable”.
- L'activité enfant le récupère grâce aux méthodes fournies par son intent.
  - `this.getIntent().getExtras().getString("clé");`

# Intent utilisation

## Activité 2

```
public void onCreate(Bundle savedInstanceState) {
 super.onCreate(savedInstanceState);
 setContentView(R.layout.main);
 if(getIntent().getExtras()!=null) {
 int value = getIntent().getExtras().getInt("musicUpdate");
 }
}
```

# UTILISATION DES INTENTS

- Nous pouvons aussi récupérer des informations de l'enfant pour son parent.
- Utilisation de la méthode  `setResult(int res, Intent i)` pour envoyer un résultat de l'enfant au parent.
  - 1er paramètre est un code de retour
  - 2eme paramètre est un intent qui contient des informations
- Sur le parent, implémentation de la fonction  
`onActivityResult(int requestCode, int resultCode, Intent data);`

# Exemple d'un picker

## Activité Mère

```
//Lancement de l'activite fille avec un paramètre
Intent i = new Intent(this, DatePickerActivity.class);
i.putExtra(DATE_EXTRA, dateDebut);
startActivityForResult(i, DATE_DEBUT_REQ_CODE)
```

## Activité Fille

```
//Récupération du paramètre
String date= getIntent().getExtras().getString(DATE_EXTRA);
...
//On renvoie la nouvelle valeur à l'activité mère
Intent result = new Intent();
result.putExtra(DATE_EXTRA, date);
setResult(Activity.RESULT_OK, result);
//On termine l'activité fille
finish();
```

```
//De retour sur l'activité mère on récupere la valeur
protected void onActivityResult(int requestCode, int resultCode, Intent data) {
 super.onActivityResult(requestCode, resultCode, data);
 //Si on vient bien du picker pour date de début
 if(requestCode == DATE_DEBUT_REQ_CODE) {
 if(resultCode == Activity.RESULT_OK) {
 dateDebut = data.getExtras().getString(DATE_EXTRA);
 }
 else {
 //l'utilisateur a annulé
 }
 }
}
```

# Serializable et Parcelable

- En implementant Serializable ou Parcelable, l'objet peut être transmit par intent.
- Parcelable est plus optimisé que Serializable mais nécessite de le remplir à la main.
  - Heureusement <http://www.parcelabler.com/> le fait pour vous
- Parcelable permet aussi de transmettre une liste d'objet récupérable
  - List<Eleve> eleves = getIntent().getExtras().getParcelableArrayListExtra(cle);

# Serializable et Parcelable

```
public class Eleve implements Parcelable {
 ...
 protected Eleve(Parcel in) {
 nom = in.readString();
 prenom = in.readString();
 sexe = in.readByte() != 0x00;
 id = in.readByte() == 0x00 ? null : in.readLong();
 }
 @Override
 public int describeContents() {
 return 0;
 }

 @Override
 public void writeToParcel(Parcel dest, int flags) {
 dest.writeString(nom);
 dest.writeString(prenom);
 dest.writeByte((byte) (sexe ? 0x01 : 0x00));
 if (id == null) {
 dest.writeByte((byte) (0x00));
 } else {
 dest.writeByte((byte) (0x01));
 dest.writeLong(id);
 }
 }
}
```

# ROTATION DE L'ÉCRAN

- Recréation de l'activité

- Donc repassage dans le onCreate()

- Comment gérer ?

- onSaveInstanceState / onRestoreInstanceState
  - Désactiver la recréation (Application ou Activity)  
`android:configChanges="orientation|screenSize|keyboardHidden"`

280

# TP – EXTRA

- Reprendre le TP sur la ListView, et faire en sorte de sauvegarder la liste lors de la rotation.

- Rendre les élèves Parcelable
  - <http://www.parcelabler.com/>
- Sauvegarder les éléments de l'activité

```
protected void onSaveInstanceState(Bundle outState) {
 super.onSaveInstanceState(outState);
 outState.putParcelableArrayList(EXTRA_LIST_ELEVE, eleveList);
}
```

- Charger les éléments en mémoire.

```
protected void onRestoreInstanceState(Bundle savedInstanceState) {
 if (savedInstanceState != null) {
 eleveList = savedInstanceState.getParcelableArrayList(EXTRA_LIST_ELEVE);
 //On passe par une variable intermediaire, sinon le cast ne passe pas.
 ArrayList<Eleve> temp=savedInstanceState.getParcelableArrayList(EXTRA_LIST_ELEVE);
 eleveList.clear();
 if (temp != null) {
 eleveList.addAll(temp);
 }
 }
}
```

- Solution : Module listViewRecyclage

# LES INTENTS IMPLICITES

- Nous appelons une action au lieu du nom de la classe dans l'intent
- En utilisant un Intent implicite, nous laissons Android décider de la meilleure application à lancer.
- L'action est tout simplement une chaîne de caractères.
- Cette action est souvent associée à une URI pour apporter plus de précision sur le type d'application à lancer.

# Tableau des principaux intents

Action	Uri	Description
Intent.ACTION_VIEW	geo:latitude,longitude	ouvre google maps en cherchant les lat/long
Intent.ACTION_VIEW	geo:0,0?q=street+address	ouvre google maps en cherchant l'adresse
Intent.ACTION_CALL	tel:phone_number	ouvre l'appli tel pour faire un appel
Intent.ACTION_DIAL	tel:phone_number	ouvre l'appli tel directement avec le no composé
Intent.ACTION_VIEW	http://web_address	ouvre un browser
Intent.ACTION_WEB_SEARCH	un texte	ouvre un browser avec une recherche google

# Exemple d'utilisation de l'intent implicite

```
// Call
String callUrl = "tel:" + friend.getPhoneNumber();
startActivity(new Intent(Intent.ACTION_DIAL, Uri.parse(callUrl)));

// Sms
String smsUrl = "sms:" + friend.getPhoneNumber();
startActivity(new Intent(Intent.ACTION_VIEW, Uri.parse(smsUrl)));
```

# Proposer son application pour lire un intent implicite

- Une URI commençant par geo

- Exemple : geo:47.6,-122.3
- La suite dans la doc
  - <https://developer.android.com/guide/components/intents-common.html>

- Dans le manifest

```
<activity ...>
 <intent-filter>
 <action android:name="android.intent.action.VIEW" />
 <data android:scheme="geo" />
 <category android:name="android.intent.category.DEFAULT" />
 </intent-filter>
</activity>
```

# Créer sa propre URI

- Dans le manifest

```
<activity android:name=<< MainActivity >> >
 <intent-filter>
 <action android:name="android.intent.action.VIEW" />
 <category android:name="android.intent.category.DEFAULT" />
 <category android:name="android.intent.category.BROWSABLE" />

 <data android:scheme="formation" />
 </intent-filter>
</activity>
```

- On appelle notre application
  - formation:url\_a\_parser
- Dans l'activité

```
//On récupère les données
String url_a_parser = getIntent().getData().toString();

//On affiche
Toast.makeText(this, url_a_parser , Toast.LENGTH_LONG).show();
```



287

## AFFICHAGE DYNAMIQUE

S'adapter à l'orientation et à la plateforme

# PROBLÉMATIQUE

- Différentes tailles d'écran
  - Tablette vs téléphone
- Différentes orientations
  - Landscape vs paysage
- Comment adapter l'interface graphique de son application tout en unifiant notre code?

# EXEMPLE SUR L'APPLI MUSIC

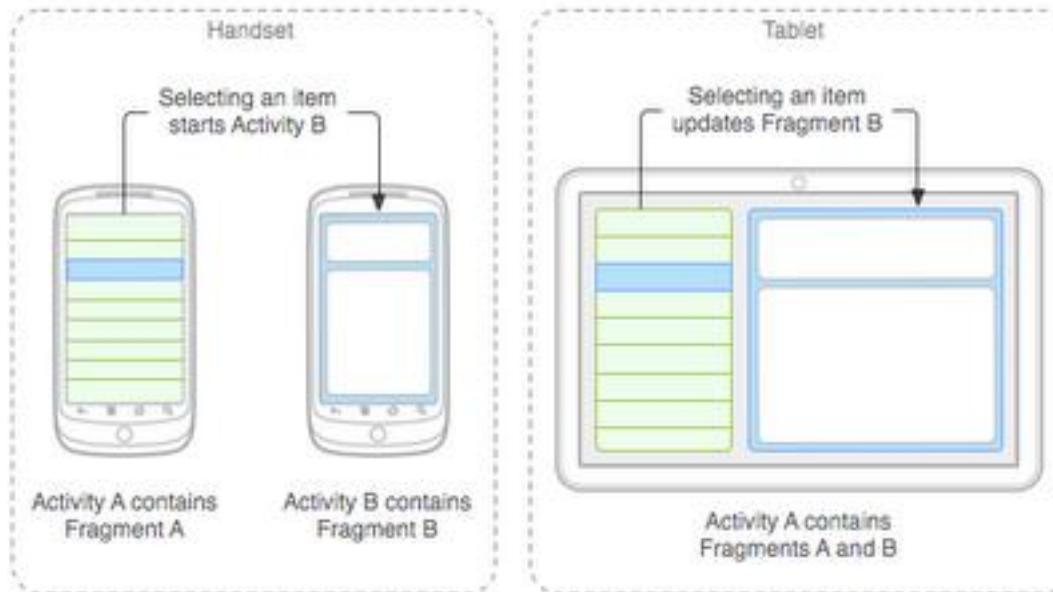
- Sur téléphone

- Un écran présentant une liste d'élève
- Sur sélection d'un élève, affichage d'un autre écran présentant le détail de celui-ci

- Sur tablette

- Un écran présentant la liste d'un coté et le détail de de l'autre

# EXAMPLE



# PROBLÈME

- La réflexion avec des activités nous obligerait à créer 3 activités (listeActivity, detailActivity et combinedActivity) et à dupliquer beaucoup de code
- Impossible de charger une Activity spécifique en fonction du device!!!!

# SOLUTION

- Dissocier la notion d'écran (activité) de la notion de fonctionnalité.
  - L'application contient 1 écran (1 activity)
  - L'application à 2 fonctionnalité (2 fragments)
    - 1 fragment List
    - 1 fragment Détail

# FRAGMENT

- Un fragment est un composant indépendant ayant son propre cycle de vie.
- Un fragment a sa propre vue et son propre model (MVC)
  - Un fragment encapsule une partie de l'interface de l'application le comportement qui lui est associé
- Un fragment doit etre gérer par une activité
- Une activité peut gerer plusieurs fragments
  - L'activity gère la navigation / communication

# CRÉATION D'UN FRAGMENT

```
public class DetailFragment extends Fragment {

 private Eleve eleve = null;
 private TextView tv;

 @Override
CE N'EST PAS UN CONSTRUCTEUR !!! C'EST ANDROID QUI APPELLE CETTE METHODE QUAND CA L'ARRANGE
 public View onCreateView(LayoutInflater inflater, ViewGroup container, Bundle savedInstanceState) {
 View rootView = inflater.inflate(R.layout.detail_fragment, container, false);
 tv = (TextView) rootView.findViewById(R.id.tv);
 return rootView;
 }

 @Override
 public void onResume() {
 super.onResume();
 refreshScreen();
 }
}
```

# LAYOUT/CONTENT\_FRAME

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
 android:layout_width="match_parent"
 android:layout_height="match_parent"
 android:orientation="horizontal" >

 <fragment
 android:id="@+id/fragment_list"
 android:name="com.example.testfragment.ListFragment"
 android:layout_width="0dp"
 android:layout_height="match_parent"
 android:layout_weight="1" />

 <FrameLayout
 android:id="@+id/f1_2"
 android:layout_width="0dp"
 android:layout_height="match_parent"
 android:layout_weight="3" />

</LinearLayout>
```

# VERS UNE SINGLE ACTIVITY APP

- La plupart des applications ne contiennent qu'une seule activité.
  - Sur téléphone :
    - Switch dynamique d'un fragment à l'autre
  - Sur tablette :
    - Affichage simultanée de plusieurs fragments
- La classe FragmentManager offre les fonctionnalités pour la gestion d'affichage des fragments

# SINGLE ACTIVITY APPLICATION

- L'activité décide dynamiquement du/des fragments à afficher en fonction du context
- Les fragments :
  - déclare leur propre API pour les mettre à jour
    - Getter/setter, reloadData() ...
  - Utilise des interfaces type onXListener pour remonter les informations à l'activity

# CONSEIL D'ARCHITECTURE

- Utiliser les activités juste pour la communication entre plusieurs fragments
  - Les Fragments n'ont pas d'IntentFilter
  - Laisser l'activité répondre aux callback des fragment et aux intents
  - L'activité sait si elle doit mettre à jour un Fragment qu'elle contient ou si elle doit appeler une autre activité

- Créer un nouveau projet avec
  - en mode portrait une liste d'élève (prénom et nom) cliquable menant sur une fiche détail reprenant le nom et prénom.
  - en mode paysage la liste sur la gauche et le detail sur la droite.
- Solution : Module FragmentFromScratch

# EXEMPLE: FRAGMENTS

- A mettre dans values et values-land

```
<resources>
 <bool name="twoPane">false</bool>
</resources>
```

- Créer les fichiers XML

- activity\_main.xml
- fragment\_list.xml
- fragment\_detail.xml

- Créer les classes Java correspondantes

- MainActivity.java
- ListFragment.java
- DetailFragment.java

300

# EXEMPLE: MAINACTIVITY

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
 android:layout_width="match_parent"
 android:layout_height="match_parent"
 android:orientation="horizontal">

 <FrameLayout
 android:id="@+id/fl_fragment1"
 android:layout_width="0dp"
 android:layout_height="match_parent"
 android:layout_weight="1" >
 </FrameLayout>

 <FrameLayout
 android:id="@+id/fl_fragment2"
 android:layout_width="0dp"
 android:layout_height="match_parent"
 android:layout_weight="3" >
 </FrameLayout>
</LinearLayout>
```

# EXAMPLE: LISTFRAGMENT

```
@Override
public View onCreateView(LayoutInflater inflater, ViewGroup container, Bundle savedInstanceState) {
 View rootView = inflater.inflate(R.layout.list_fragment, container, false);

 rv = (RecycleView) rootView.findViewById(R.id.rv);
 if (eleveList == null) {
 eleveList = new ArrayList<Eleve>();
 }
 eleveAdapter = new EleveAdapter(eleveList);
 rv.setAdapter(eleveAdapter);

 rv.setLayoutManager(layoutManager = new LinearLayoutManager(getActivity()));
 rv.setItemAnimator(new DefaultItemAnimator());

 return rootView;
}
```

# EXEMPLE: DETAILFRAGMENT

```
public View onCreateView(LayoutInflater inflater, ViewGroup container, Bundle savedInstanceState) {
 View rootView = inflater.inflate(R.layout.detail_fragment, container, false);
 tv = (TextView) rootView.findViewById(R.id.tv);
 return rootView;
}

@Override
public void onResume() {
 super.onResume();
 refreshScreen();
}

public void setEleve(Eleve eleve) {
 this.eleve = eleve;
}

public void refreshScreen() {
 if (eleve == null) {
 tv.setText("Aucun élève");
 }
 else {
 tv.setText(eleve.getPrenom() + " " + eleve.getNom());
 }
}
```

# EXEMPLE: MAINACTIVITY

```
public class MainActivity extends Activity implements OnClickListListener {

 private FrameLayout fl_fragment2;
 private ListFragment listFragment;
 private DetailFragment detailFragment = null;

 @Override
 protected void onCreate(Bundle savedInstanceState) {
 super.onCreate(savedInstanceState);
 setContentView(R.layout.activity_two_pane);

 //On définit si on utilise 1 ou 2 layout en fonction de l'appareil.
 //le fragment 2
 fl_fragment2 = (FrameLayout) findViewById(R.id.fl_fragment2);

 //Si on souhaite afficher 2 fragments en même temps
 if (MyApplication.getInstance().isTwoPane()) {
 fl_fragment2.setVisibility(View.VISIBLE);
 }
 else {
 fl_fragment2.setVisibility(View.GONE);
 }
 }
}
```

# EXEMPLE: MAINACTIVITY

```
@Override
protected void onStart() {
 super.onStart();
 //On verifie si les fragments n'existent pas déjà.
 //Ceux ci peuvent avoir été recréés par Android lors d'une rotation d'écran par exemple. On les récupère
 //grâce à leur tag.
 listFragment = (ListFragment) getFragmentManager().findFragmentByTag(ListFragment.class.toString());
 detailFragment = (DetailFragment) getFragmentManager().findFragmentByTag(DetailFragment.class.toString());

 //Si la liste n'existe pas on la crée.
 if (listFragment == null) {
 listFragment = new ListFragment();
 }
 if (MyApplication.getInstance().isTwoPane()) {
 detailFragment = new DetailFragment();
 //on le positionne sur le 2eme emplacement
 getFragmentManager().beginTransaction().replace(R.id.fl_fragment2, detailFragment,
 DetailFragment.class.toString()).commit();
 }
 //on positionne le fragment sur l'emplacement fragment1
 getFragmentManager().beginTransaction().replace(R.id.fl_fragment1, listFragment,
 ListFragment.class.toString()).commit();
}
```

- Implementer la communication entre les fragments
  - Callback vers l'activité du click sur la liste.
- Solution FragmentFromScratch

# EXEMPLE: LISTFRAGMENT

- Ajouter une interface dans ListFragment

```
public interface CallBAck{
 void onClickOnEleve (Eleve eleve);
}

private CallBack callBack = null;
public void setCallBack(CallBack cb) {
 callBack = cb;
}

@Override
public void onClick(AdapterView<?> parent, View view, int position, long id) {

 Eleve eleve = eleveList.get(position);

 if (callBack != null) {
 callBack.onClickOnEleve(eleve);
 }
}
```

# EXEMPLE: MAINACTIVITY

- S'inscrire au callBack

```
listFragment = new ListFragment();
listFragment.setCallBack(this);
```

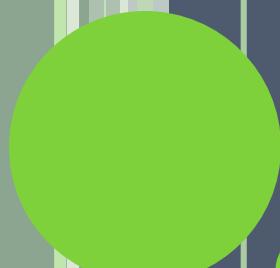
- Gestion du callBack

```
@Override
public void onClickOnEleve (Eleve eleve) {
 if (MyApplication.getInstance().isTwoPane()) {
 detailFragment.setEleve(eleve);
 detailFragment.refreshScreen();
 }
 else {
 //on demande à l'OS de remplacer le fragment
 FragmentTransaction ft = getFragmentManager().beginTransaction();
 detailFragment = new DetailFragment();
 detailFragment.setEleve(eleve);
```

ATTENTION A NE PAS MODIFIER L'INTERFACE GRAPHIQUE DU FRAGMENT QUI N'EXISTE PAS ENCORE

IL NE SERA CRÉER QUE PLUS TARD QUAND ANDROID LE SOUHAITERA

```
ft.replace(R.id.fl_fragment1, detailFragment, DetailFragment.class.toString());
ft.addToBackStack(null); // permet de revenir à l'écran d'avant avec un back bouton
ft.commit();
}
```



309

COMMON ACTIVITY

# COMMON ACTIVITY

- Une activity pour les surveiller toutes.
- Méthode applicable aux fragments.
- Un xml avec un frameLayout pour laisser la place aux activités enfants.

```
<FrameLayout
 android:id="@+id/container"
 android:layout_width="match_parent"
 android:layout_height="match_parent"/>

@Override
protected void onCreate(Bundle savedInstanceState) {
 super.onCreate(savedInstanceState);
 super.setContentView(R.layout.activity_common_layout);
}
```

# COMMON ACTIVITY

- Les Activités filles pourront définir leur XML grâce à cette méthode qu'on surcharge

```
@Override
public void setContentView(int layoutResID) {
 View v = getLayoutInflater().inflate(layoutResID, null);
 FrameLayout container = (FrameLayout) findViewById(R.id.container);
 container.removeAllViews();
 container.addView(v);

}
}
```

# COMMON ACTIVITY

## ○ Exemple d'Activity Fille

```
public class MainActivity extends CommonActivity {

 private Button bt;

 @Override
 protected void onCreate(Bundle savedInstanceState) {

 super.onCreate(savedInstanceState);
 super.setContentView(R.layout.activity_main);

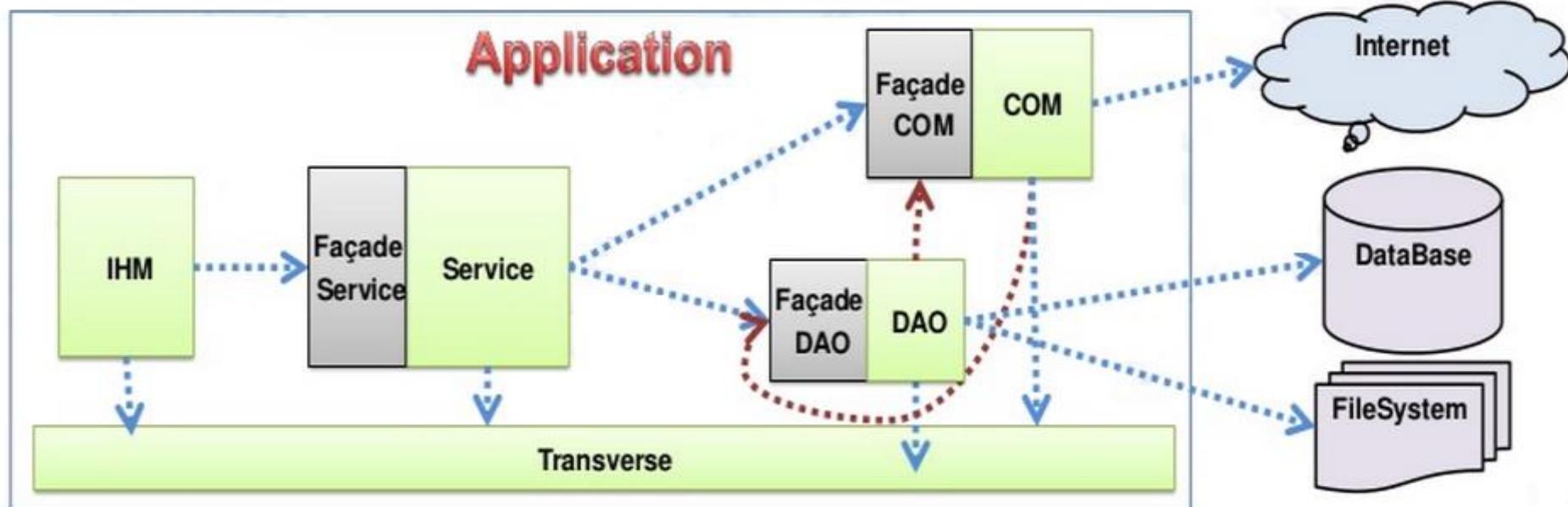
 bt = (Button) findViewById(R.id.bt);
 }
}
```

# COMMON ACTIVITY

- Permet de centraliser les informations pour chaque activité
  - Mettre un log sur l'activité courante
  - Mettre à dispo une fenêtre d'attente
  - Mettre en commun une base Graphique
  - Un ensemble de méthode

# ARCHITECTURE APPLICATION

- Suggestion d'architecture



- Couche transverse

- Beans, ExceptionManager, StringUtils, LogUtils,

314

- Créer une classe CommonActivity et son layout permettant de mettre un titre commun à toutes les activités.
- Faire en sorte que mainActivity étende de CommonActivity et puisse changer son titre.
- Masquer l'ActionBar

```
<!-- Application theme. -->

<style name="AppTheme" parent="AppBaseTheme">
 <!-- All customizations that are NOT specific to a particular API-level can go here. --
->
 <item name="windowActionBar">false</item>
 <item name="windowNoTitle">true</item>
</style>
```

# GESTION DES EXCEPTIONS

## ○ Intérêt

- Déetecter et corriger rapidement une erreur
- Prévenir l'utilisateur
- Eviter le crash de l'application
- Reporter la faute sur l'utilisateur.
- Accélérer les décisions et le travail en équipe
- Gérer plus facilement tous les cas
- Maintenances et évolutions plus faciles

# GESTION DES EXCEPTIONS

## ○ ExceptionA

- String messageUtilisateur
    - Affichage à l'écran
  - String messageTechnique
    - Envoie par mail, dans les log, remonté sur un serveur, Crashlitycs...
  - Int codeErreur
    - Pour permettre de mieux identifier des exceptions similaires.
- 
- public ExceptionA(String messageUtilisateur, String messageTechnique)
  - public ExceptionA(String messageUtilisateur, String messageTechnique, Throwable throwable)
- 
- Toujours transmettre l'exception précédente pour avoir la stackTrace complète !!!

```
catch(Exception e) {
 throw new TechnicalException("Erreur de parsing", e);
}
```

# GESTION DES EXCEPTIONS

## ○ LogicException extends ExceptionA

- C'est la faute de l'utilisateur
  - Pas de connexion internet
  - Mauvais mot de passe
  - Champs non remplis
- Pas forcément de message technique

## ○ TechnicalException extends ExceptionA

- C'est notre faute, ne doit pas se produire
  - Serveur down
  - Erreur de programmation
- Message générique pour l'utilisateur
- Message technique et remonté de l'erreur à l'équipe.

## ○ Exemple dans formationUtils

318

# CRASHLYTICS ET FLURRY

## ○ CrashLytics

- <https://try.crashlytics.com/>
- Gestion de bug
  - Imprévus : ThrowException
  - Choisis : Crashlytics.logException(e);

## ○ Flurry (Yahoo)

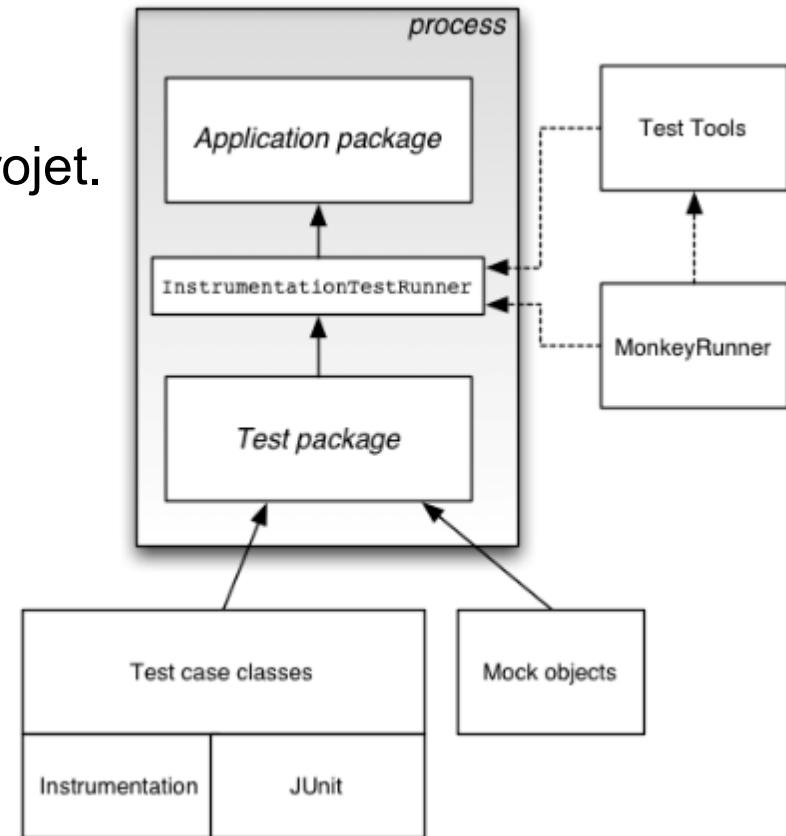
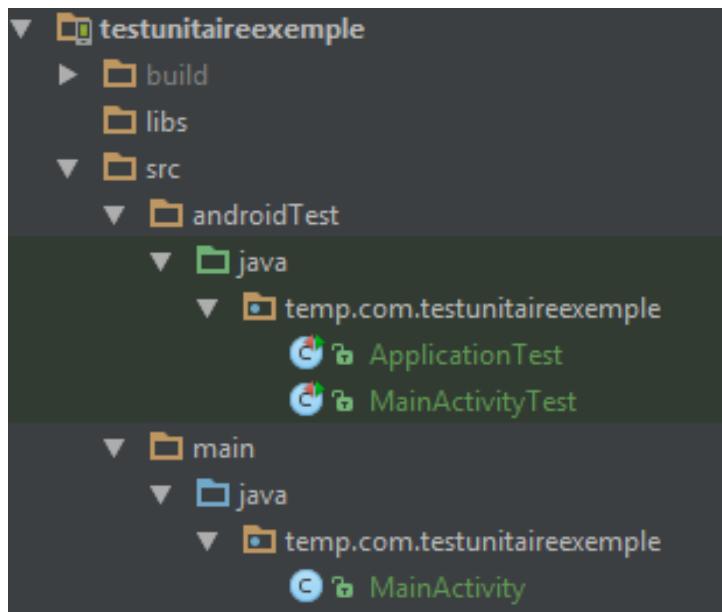
- <https://dev.flurry.com/secure/login.do>
- Gestion d'événements

## ○ Avantages

- Gratuites
- Faciles à mettre en place
- Optimisées

# TESTS UNITAIRES : ANDROID TEST FRAMEWORK

- Basé sur JUnit
- Intégré à l'IDE
- Généré par l'IDE à la création du projet.



- <https://developer.android.com/training/activity-testing/index.html>

320

# JUNIT :

## ○ Fonctionnement

- Hérite de *ActivityInstrumentationTestCase2*
- Une méthode `setUp` d'initialisation.
- Une méthode `testPreconditions()` lancée avant la séquence de test.
- Lance dans l'ordre toutes les méthodes commençant par « `test` »
- Dans les méthodes de test on utilise **les methodes** `AssertNotNull`, `assertEquals`, `Assert*`... pour vérifier les valeurs attendues.
- On peut travailler avec l'activité transmise dans le constructeur de la classe de test.

# JUNIT : QUELQUES OUTILS

- Simuler les touches du clavier
  - sendkeys(« bob »)
- Toucher une vue ou un bouton
  - TouchUtils.clickView(this, View)
- Toucher au composant graphique
  - getInstrumentation().runOnMainSync(...)
- Attendre l'UIThread
  - getInstrumentation().waitForIdleSync();

# TESTS UNITAIRES : JUNIT

```
public class MainActivityTest extends ActivityInstrumentationTestCase2<MainActivity> {

 private MainActivity mainActivity;
 private TextView tv_hello_world;
 private String helloWorldValue;

 public MainActivityTest() {
 super(MainActivity.class);
 }

 @Override
 protected void setUp() throws Exception {
 super.setUp();
 mainActivity = getActivity();
 tv_hello_world = (TextView) mainActivity.findViewById(R.id.tv_hello_world);
 helloWorldValue = getActivity().getString(R.string.hello_world);
 getInstrumentation().runOnMainSync(new Runnable() {
 @Override
 public void run() {
 //valeur par defaut
 tv_hello_world.setText(helloWorldValue);
 }
 });
 getInstrumentation().waitForIdleSync();
 }

 /** Verifie qu'on a bien la valeur attendue. */
 public void testCorrectValue() {
 assertEquals(tv_hello_world.getText(), helloWorldValue);
 }
}
```

# TESTS UNITAIRES : JUNIT

```
/**
 * Tue et relance l'activité (ne passe pas dans onSave et dans le onRestore)
 * Pour par exemple vérifier que les données sont bien enregistrées en base.
 */
public void testRestartActivitySaveValue() {
 String saveValue = "saveValue";

 getInstrumentation().runOnMainSync(new Runnable() {
 @Override
 public void run() {
 //Sur l'UIThread
 tv_hello_world.setText(saveValue);
 }
 });

 getInstrumentation().waitForIdleSync();

 // Close the activity
 mainActivity.finish();
 // Required to force creation of a new activity
 setActivity(null);

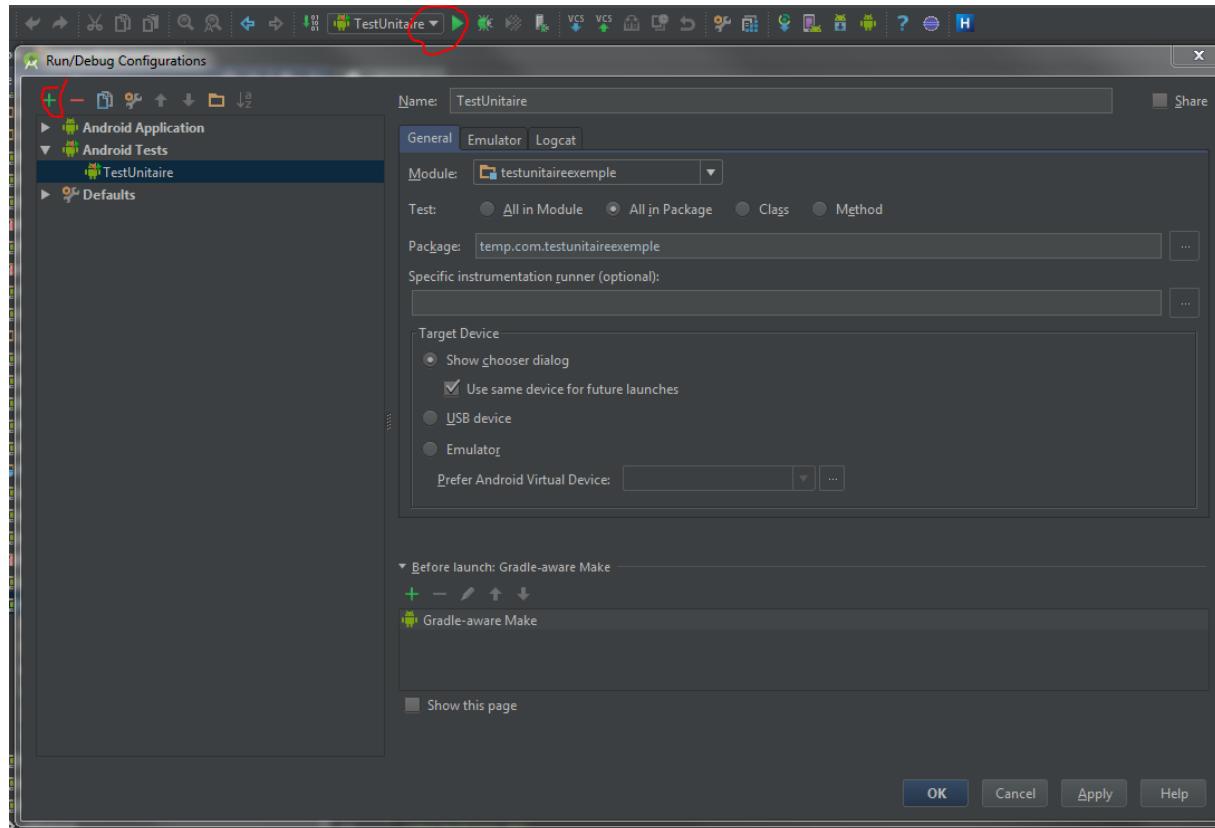
 mainActivity = getActivity();

 tv_hello_world = (TextView) mainActivity.findViewById(R.id.tv_hello_world);

 assertEquals(saveValue, tv_hello_world.getText().toString());
}
```

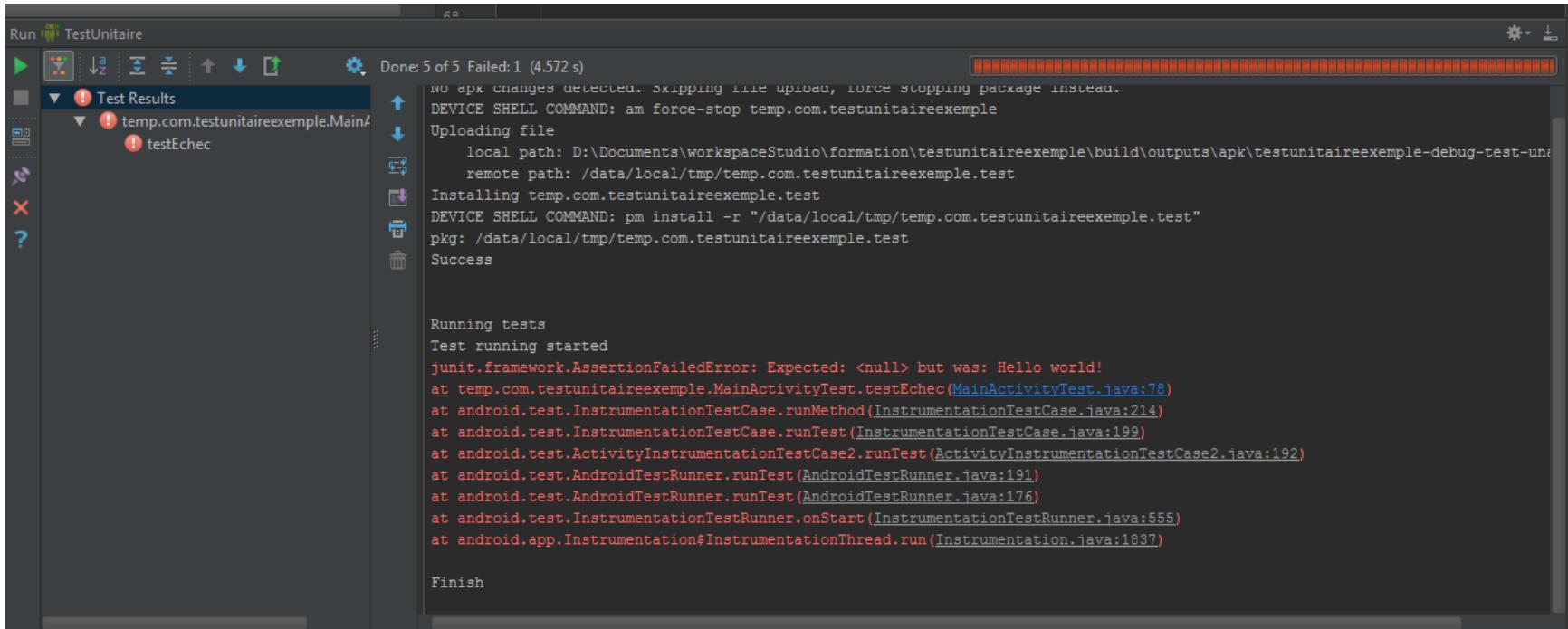
# JUNIT : LANCER LES TESTS

- Réglages d'Android Studio
  - Edit Configuration
- Fonctionne sur simulateur ou sur device réél.



325

# JUNIT : RÉSULTATS



The screenshot shows the Android Studio interface during a JUnit test run. The title bar says "Run TestUnitaire". The left sidebar has a "Test Results" section with one failed test: "temp.com.testunitaireexemple.MainActivity testEchec". The main pane displays the terminal output of the test run.

```
Done: 5 of 5 Failed:1 (4.572 s)

NO APK changes detected. Skipping file upload, force stopping package instead.
DEVICE SHELL COMMAND: am force-stop temp.com.testunitaireexemple
Uploading file
 local path: D:\Documents\workspaceStudio\formation\testunitaireexemple\build\outputs\apk\testunitaireexemple-debug-test-unit
 remote path: /data/local/tmp/temp.com.testunitaireexemple.test
Installing temp.com.testunitaireexemple.test
DEVICE SHELL COMMAND: pm install -r "/data/local/tmp/temp.com.testunitaireexemple.test"
pkg: /data/local/tmp/temp.com.testunitaireexemple.test
Success

Running tests
Test running started
junit.framework.AssertionFailedError: Expected: <null> but was: Hello world!
at temp.com.testunitaireexemple.MainActivityTest.testEchec (MainActivityTest.java:78)
at android.test.InstrumentationTestCase.runMethod (InstrumentationTestCase.java:214)
at android.test.InstrumentationTestCase.runTest (InstrumentationTestCase.java:199)
at android.test.ActivityInstrumentationTestCase2.runTest (ActivityInstrumentationTestCase2.java:192)
at android.test.AndroidTestRunner.runTest (AndroidTestRunner.java:191)
at android.test.AndroidTestRunner.runTest (AndroidTestRunner.java:176)
at android.test.InstrumentationTestRunner.onStart (InstrumentationTestRunner.java:555)
at android.app.Instrumentation$InstrumentationThread.run (Instrumentation.java:1837)

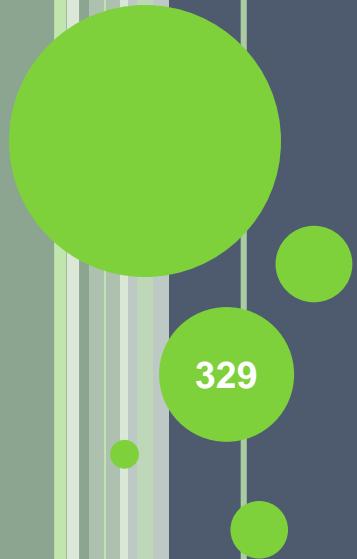
Finish
```

# JUNIT : LIMITATION

- Pas de multiple Activité
- Implémentations compliquées
- Tests lents

# TEST IHM: ROBOTIUM

- <https://code.google.com/p/robotium/>
- Extension de l'Android Test Framework
- Optimisé pour les tests fonctionnelles.
  
- Démo
- Analyse du code



# USER EXPERIENCE

# User Experience

- Qu'est ce que l'User Experience?
- Official Design Guidelines
  - <http://developer.android.com/design/index.html>
- Un peu d'aide
  - <https://android-arsenal.com/>

# Quelques exemples

- Donner un retour à l'utilisateur de son action.
  - Le bouton Bootstrap



# Quelques exemples

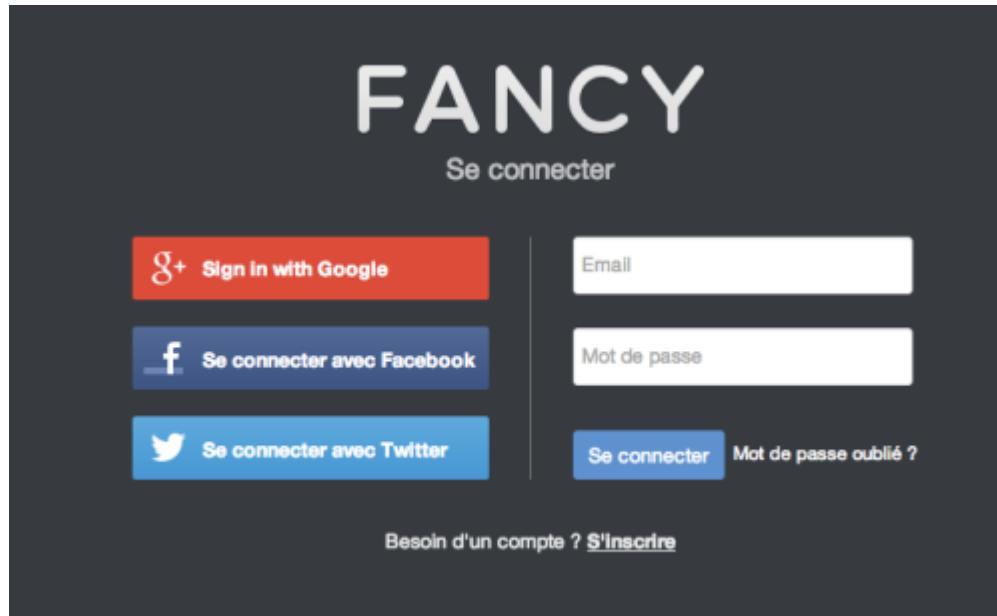
- 48 dp la taille d'un doigt
- 8dp l'espace minimum entre 2 UI élément
- Reprendre ce que l'utilisateur connaît.
  - (Facebook, Twiter, youtube...)

# Quelques exemples

- Le faire patienter (Si possible lui indiquer combien de temps)
  - ProgressBar, animation...
- Gestion d'un cache mémoire
- Gestion d'un cache de requête
- Affichage de données obsolètes avec chargement en arrière plan.
- Donner de la vie avec les animations.

# Inscription

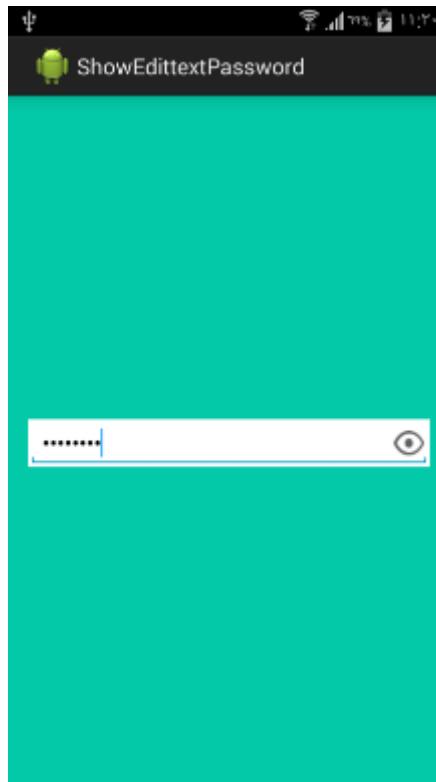
- Inscription le plus tard possible
- Utiliser la connexion par les réseau sociaux, facebook ou google+



334

# Connexion

- Librairie **ShowEdittextPassword** pour le mot de passe.



335

# Choix des couleurs

- <https://developer.android.com/design/style/color.html>

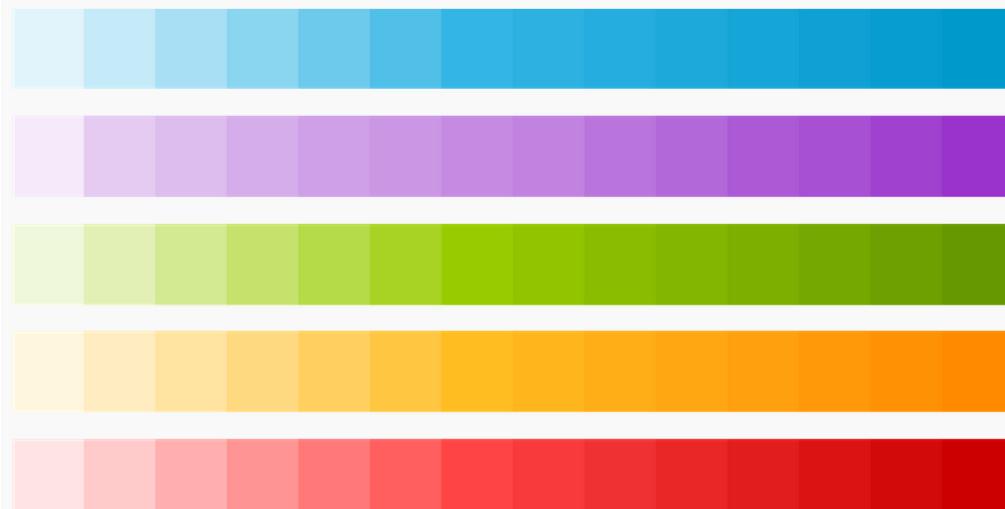
Use color primarily for emphasis. Choose colors that fit with your brand and provide good contrast between visual elements. Red and green may be indistinguishable to color-blind users.



## Palette

Blue is the standard accent color in Android's color palette. Each color has a corresponding darker shade that can be used as a complement when needed.

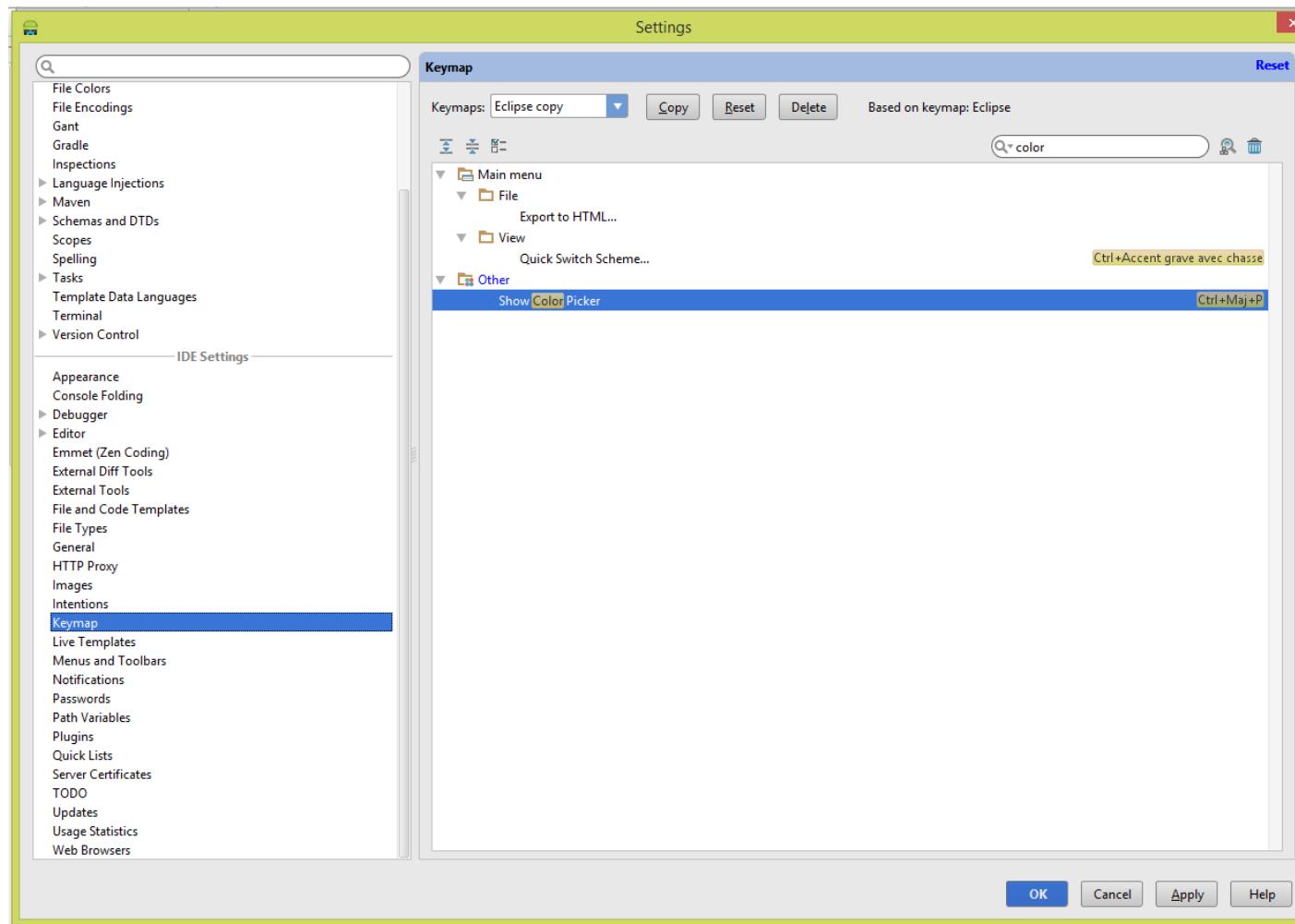
[Download the swatches](#)



336

# COLORPICKER

## Intégré à AndroidStudio



337

y Monteiro

# Material Design

- <http://www.google.com/design/spec/material-design/introduction.html>
- De nombreuses librairies dans AndroidArsenal

# Création d'un composant : BoutonWithIcon



MyBootStrapButtonWithIcon

- Objectif : Un bouton avec une image en paramètre XML

```
<com.boutonexemple.button.MyBootStrapButtonWithIcon
 android:layout_width="wrap_content"
 android:layout_height="wrap_content"
 android:src="@drawable/ic_launcher"
 app:backGroundColor="@color/main" />
```

- Départ : le fichier XML

- LinearLayout horizontale
  - ImageView
  - TextView

# Création d'un composant : BoutonWithIcon

- Dans res/values/attrs.xml on déclare les attributs du composant que l'on souhaite mettre dans le XML

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
 <declare-styleable name="stylistableButton">
 <attr name="backGroundColor" format="color"/>
 <attr name="android:src"/>
 </declare-styleable>
</resources>
```

- On crée le layout de notre composant comme un layout normal.
  - xmlns:app="http://schemas.android.com/apk/res-auto"

```
<com.boutonexemple.button.MyBootStrapButtonWithIcon
 android:layout_width="wrap_content"
 android:layout_height="wrap_content"
 android:src="@drawable/ic_launcher"
 app:backGroundColor="@color/main" />
```

# Création d'un composant : La classe

## ○ On crée notre classe composant

```
public class MyBootStrapButtonWithIcon extends LinearLayout {

 private TextView tv;
 private ImageView iv;

 private int backGroundColor;

 public MyBootStrapButtonWithIcon(Context context, AttributeSet attrs, int defStyle) {
 super(context, attrs, defStyle);
 init(context, attrs);
 }

 // ce constructeur ne devrait jamais être appelé, car il n'a pas d'AttributeSet en paramètre.
 public MyBootStrapButtonWithIcon(Context context) {
 super(context);
 }

 public MyBootStrapButtonWithIcon(Context context, AttributeSet attrs) {
 super(context, attrs);
 init(context, attrs);
 }
```

# Création d'un composant : La classe

- On crée notre classe composant

```
private void init(Context ctx, AttributeSet attrs) {

 // inflation du modèle "customtitle", et initialisation des composants Button et ImageView
 // on cherche le service Android pour instancier des vues
 LayoutInflator li = (LayoutInflator) ctx.getSystemService(Context.LAYOUT_INFLATER_SERVICE);
 View v = li.inflate(R.layout.bootstrap_button_with_icon, null);
 root = v.findViewById(R.id.root);
 tv = (TextView) v.findViewById(R.id.tv);
 iv = (ImageView) v.findViewById(R.id.iv);
 addView(v);

 // Le modèle est chargé, on a plus qu'à l'initialiser avec les attributs qu'on a reçus en
 // paramètre
 TypedArray a = ctx.obtainStyledAttributes(attrs, R.styleable.styleableButton);
```

# Création d'un composant : La classe

## On crée notre classe composant

```
TypedArray a = ctx.obtainStyledAttributes(attrs, R.styleable.styleableButton);
// on obtient un TypedArray, une classe qui a plein de méthodes getString(int index),
// getInteger(int index) (...) pour obtenir la valeur String, Integer (...)

if (a.getDrawable(R.styleable.styleableButton_android_src) != null) {
 iv.setImageDrawable(a.getDrawable(R.styleable.styleableButton_android_src));
}

//couleurs

if (a.getInt(R.styleable.styleableButton_backGroundColor, 0) != 0) {
 backGroundColor = a.getInt(R.styleable.styleableButton_backGroundColor, 0);
}
else {
 backGroundColor = Color.BLACK;
}

// On change la couleur de fond
root.getBackground().setColorFilter(backGroundColor, PorterDuff.Mode.MULTIPLY);

// on recycle, c'est pour sauver mère nature
a.recycle();
}
```

# Création d'un composant : Evènement

- Bien détecter le onPress dans le composant
  - `setOnTouchListener(this);`

```
public boolean onTouch(View view, MotionEvent motionEvent) {
 if (motionEvent.getAction() == MotionEvent.ACTION_DOWN) {
 root.getBackground().setColorFilter(backGroundColorHighlight,
PorterDuff.Mode.MULTIPLY);
 iv.setColorFilter(textColorHighlight, PorterDuff.Mode.MULTIPLY);
 tv.setTextColor(textColorHighlight);
 }
 else if (motionEvent.getAction() == MotionEvent.ACTION_UP || motionEvent.getAction() ==
MotionEvent.ACTION_CANCEL) {
 root.getBackground().setColorFilter(backGroundColor, PorterDuff.Mode.MULTIPLY);
 iv.setColorFilter(textColor, PorterDuff.Mode.MULTIPLY);
 tv.setTextColor(textColor);
 }
 return false;
}
```

## ○ Réaliser un bouton bootstrap

- Le fond arrondi

```
<?xml version="1.0" encoding="utf-8"?>
<shape xmlns:android="http://schemas.android.com/apk/res/android"
 android:padding="10dp" android:shape="rectangle" >
 <solid android:color="@color/main" />
 <corners
 android:bottomLeftRadius="5dp"
 android:bottomRightRadius="5dp"
 android:topLeftRadius="5dp"
 android:topRightRadius="5dp" />
</shape>
```

- Le bouton

```
<?xml version="1.0" encoding="utf-8"?>
<selector xmlns:android="http://schemas.android.com/apk/res/android">
 <item android:drawable="@drawable/background_button_main_on" android:state_pressed="true"/>
 <item android:drawable="@drawable/background_button_main"/>
</selector>
```

## ○ Solution : Module BoutonExemple

- Réaliser un composant dont le xml prend en paramètre
  - backgroundColor
  - backgroundColorHighlighted
  - textColor
  - TextColorHighlighted
- Et gérer le onPress qui change la couleur
- Solution : Module BoutonExemple

# Style

- On définit un style dans res/values/styles.xml

```
<style name="boutonBootstrapWithIcon">
 <item name="android:layout_width">fill_parent</item>
 <item name="android:layout_height">48dp</item>
 <item name="android:textSize">14sp</item>
 <item name="android:textStyle">bold</item>
 <item name="android:ellipsize">end</item>
 <item name="android:textColor">@android:color/white</item>
 <item name="android:textColorHighlight">@color/main</item>
</style>
```

- On l'utilise dans notre composant

```
<com.boutonexemple.button.MyBootStrapButtonWithIcon
 android:id="@+id/button3"
 style="@style/boutonBootstrapWithIcon"
 android:src="@drawable/ic_launcher"
 android:text="MyBootStrapButtonWithIcon"

 app:backGroundColor="@color/main"
 app:backGroundColorHighlight="@color/main_light"
 />
```

# Travailler avec un graphiste

- Conf TAUG

348

- Réaliser un bouton 9Patch

- Pixel gauche et haut pour la duplication
- Pixel droite et bas, pour l'espace du texte.



- Solution et image : Module BoutonExemple

349

# INKSCAPE

- Logiciel gratuit d'extraction d'éléments graphiques
  - <https://inkscape.org/fr/>
  - Format SVG

350

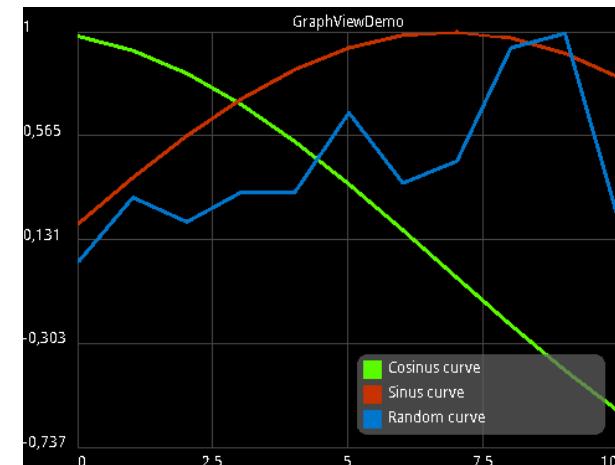
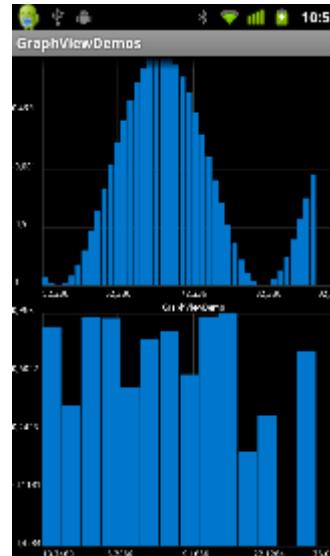
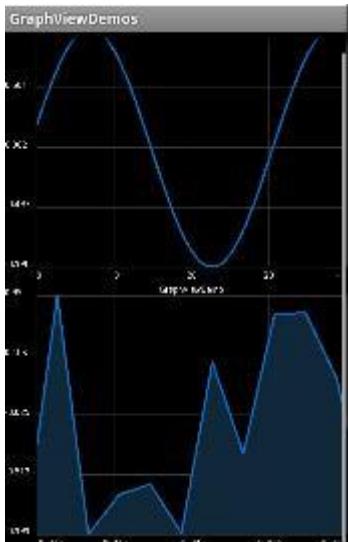
# LIBRAIRIES ANDROID

- Otto
- GreenDAO
- Picasso
- CrashLytics
- Flurry
- MaterialDialog
- Robotium
- **GraphView**
- Zbar
- **Simple-Facebook**
- **Scribe**
- **okHttp**
- **JSON**

351

# GRAPHVIEW

- Réalisation de graphiques, courbes, diagrammes...
  - <http://android-graphview.org/>
- Utilisation avec Gradle
  - compile 'com.jjoe64:graphview:3.1.3'



Anthony Monteiro

# GRAPHVIEW

## ○ Utilisation : Création d'un objet de donnée

```
public class GraphViewData implements GraphViewDataInterface {

 private double x, y;

 public GraphViewData(double x, double y) {
 this.x = x;
 this.y = y;
 }

 @Override
 public double getX() {
 return x;
 }

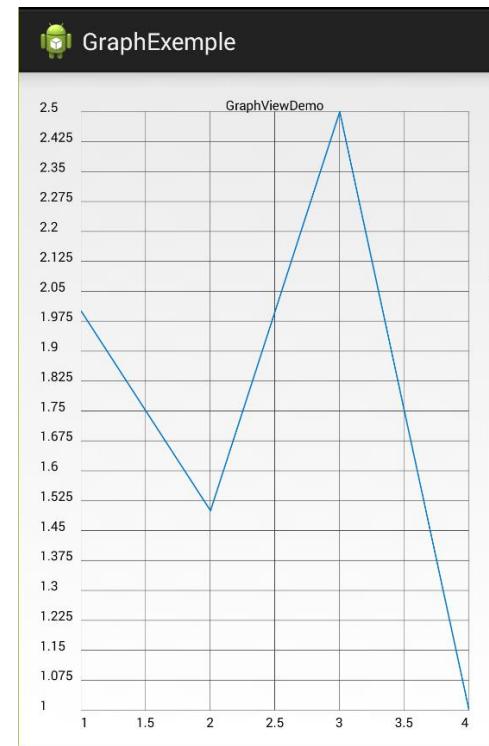
 @Override
 public double getY() {
 return y;
 }
}
```

# GRAPHVIEW

## Utilisation : Création du graph

```
private void createGraph1() {
 // init example series data
 GraphViewSeries exampleSeries = new GraphViewSeries(
 new GraphViewData[] {
 new GraphViewData(1, 2.0d), new GraphViewData(2, 1.5d),
 new GraphViewData(3, 2.5d), new GraphViewData(4, 1.0d) });

 GraphView graphView = new LineGraphView(this, "GraphViewDemo");
 graphView.addSeries(exampleSeries); // data
 ((LinearLayout) findViewById(R.id.ll_graph1)).addView(graphView);
}
```



# TP - GRAPHVIEW

- Créer un nouveau projet et afficher un 1<sup>er</sup> graph.
- Se servir de la doc sur le site pour créer d'autres versions
- Solution : Module graphView

# ZBAR OR ZXING

- <http://stackoverflow.com/questions/13268250/android-zxing-vs-zbar>
- ZXing is NOT an open-source library, or at the very best it is only "semi"-open source. You are meant to implement ZXing in tandem with its partner app, which you have to download from the PlayStore separately. This app is the one that actually does the QR Reading; all ZXing does is trigger this particular app. This means that if you are trying to integrate a QR Reader INTO your app and not call a separate one, ZXing is not what you want.
- ZXing is hackable to some extent, however as the versions have gone by the developers have purposely made it harder and harder to hack, because they don't want you to use it as a stand-alone application and bypass theirs. I tinkered with v2.1 for a day, gave up and switched to ZBar, which got me what I wanted in 10 minutes. I could have kicked myself in frustration.
- ZBar is also incredibly fast and accurate, and the tutorial is very extensive; the demo app provided even provides a "Scan Again" button if the scan turns out wrong (which I have yet to see happen). ZBar is highly customizable and doesn't have the tons of red-tape that you have to hack your way through in ZXing; it shows very simply how to get what you want out of your QR/bar code, and sending the result somewhere else is simply just a call to a different Activity.
- The final word: ZBar.

# ZBAR

- Lecteur de code barre.
  - <http://zbar.sourceforge.net/>
  - <https://github.com/dm77/barcodescanner>
- Utilisation avec Gradle
  - compile 'me.dm7.barcodescanner:zbar:1.5'

# ZBAR - DÉCLARATION

```
public class MainActivity extends Activity implements ZBarScannerView.ResultHandler {

 private FrameLayout cameraPreview;
 private ZBarScannerView mScannerView;

 @Override
 protected void onCreate(Bundle savedInstanceState) {
 super.onCreate(savedInstanceState);
 setContentView(R.layout.activity_main);
 cameraPreview = (FrameLayout) findViewById(R.id.cameraPreview);
 mScannerView = new ZBarScannerView(this);
 cameraPreview.addView(mScannerView);
 }

 @Override
 protected void onResume() {
 super.onResume();
 mScannerView.setResultHandler(this); // Register ourselves as a handler for scan results.
 mScannerView.startCamera(); // Start camera on resume
 }
 @Override
 public void onPause() {
 super.onPause();
 mScannerView.setResultHandler(null);
 mScannerView.stopCamera(); // Stop camera on pause
 }
}
```

# ZBAR - UTILISATION

```
@Override
public void handleResult(Result result) {
 // Do something with the result here
 tv_resultat.setText("Scan : " + result.getContents() + "\n" +
 "Scan format : " + result.getBarcodeFormat());
}
```

# TP - ZBAR

- Créer un lecteur de code barre.
- Solution : Module zbar\_reader

# SIMPLE FACEBOOK

- Version complète
  - <https://developers.facebook.com/docs/android>
- Version allégée du SDK de Facebook
  - <https://github.com/sromku/android-simple-facebook>
- Version très allégée (juste le login)
  - <https://github.com/greenhalolabs/facebooklogin>

# INSTALLATION

- Importer en module le projet « facebook » du sdk facebook
- Importer en module le projet « simple facebook » de la lib simple-facebook

```
dependencies {
 compile project(':facebook')
}
```

- Dans notre projet
  - compile project('Simple Facebook')

# UTILISATION

- Générer un facebook\_app\_id sur le site du sdk
- Le mettre dans le manifest

```
<meta-data
 android:name="com.facebook.sdk.ApplicationId"
 android:value="@string/facebook_app_id"/>
```

```
private SimpleFacebook mSimpleFacebook;

//OnCreate
mSimpleFacebook = SimpleFacebook.getInstance(this);

//OnResume
mSimpleFacebook = SimpleFacebook.getInstance(this);
mSimpleFacebook.eventAppLaunched();

//OnActivityResult
mSimpleFacebook.onActivityResult(this, requestCode, resultCode, data);
```

# UTILISATION

## ○ LogIn

```
if (mSimpleFacebook.isLogin()) {
 mSimpleFacebook.logout(null);
}
mSimpleFacebook.login(this);
```

## ○ LogOut

```
private void logOut() {

 mSimpleFacebook.logout(new OnLogoutListener() {
 public void onFail(String reason) {}
 public void onException(Throwable throwable) {}
 public void onThinking() {}
 public void onLogout() {}
 });
}
```

# UTILISATION

## ○ PostOnWall

```
Feed feed = new
Feed.Builder().setDescription(message).setName(getString(R.string.app_name))
.setPicture(« http://url_icone »).setLink(getString(R.string.app_url)).build();

mSimpleFacebook.publish(feed, true, new OnPublishListener() {

 @Override
 public void onFail(String reason) {}

 @Override
 public void onException(Throwable throwable) {}

 @Override
 public void onThinking() {}

 @Override
 public void onComplete(String postId) {}

});
```

# TP - FACEBOOK

- Se loguer avec Facebook
- Poster un message sur son mur.
- Solution : FacebookLogin



GIT - GITHUB

367



# GIT ?

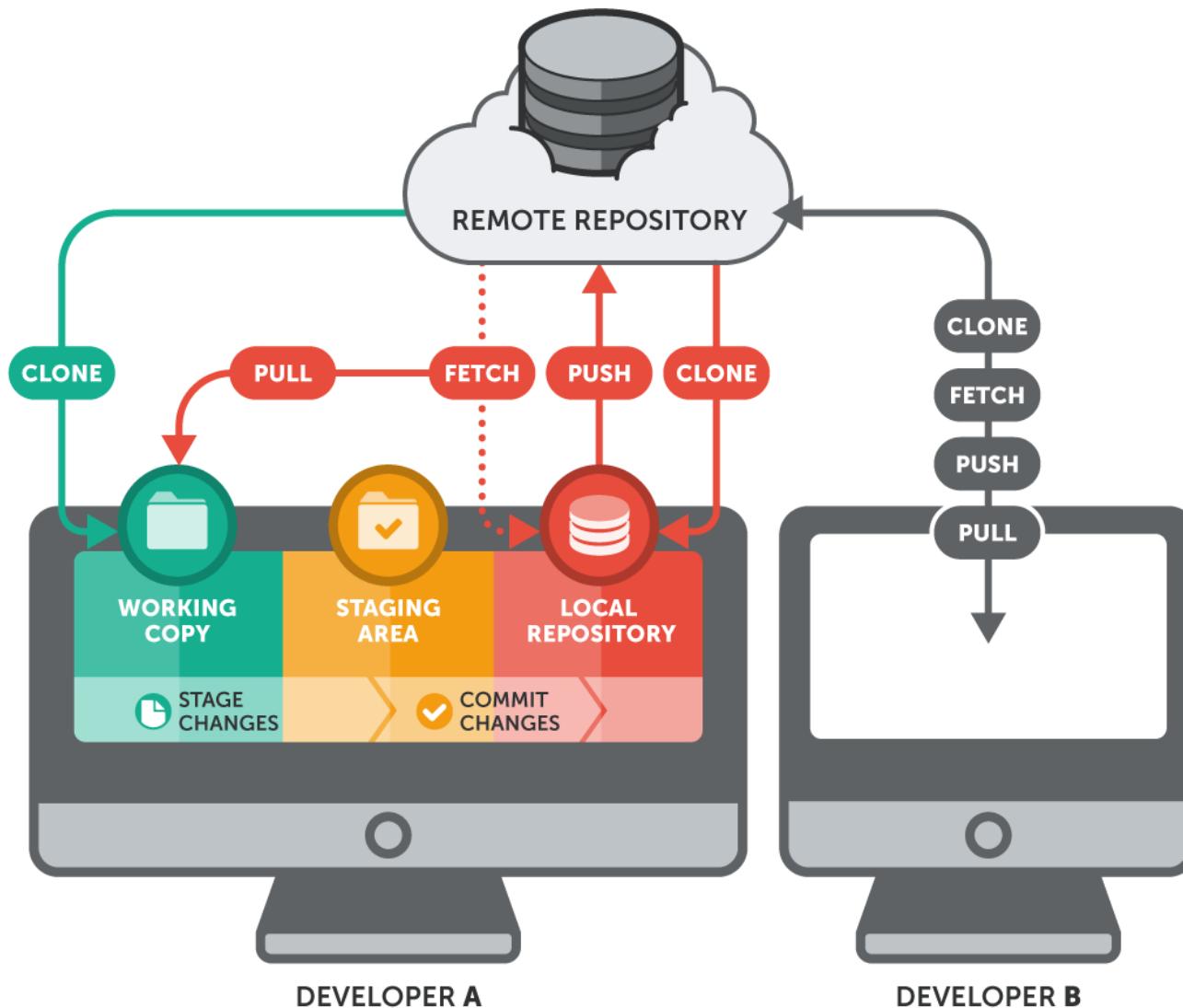
- Git permet
  - De sauvegarder votre projet sous forme de version
  - De partager un projet entre plusieurs utilisateurs
  - <https://git-scm.com/>
  - Un tuto très complet
    - <https://openclassrooms.com/courses/gerez-vos-codes-source-avec-git>

# GIT ?

- Git permet

- De sauvegarder votre projet sous forme de version
- De partager un projet entre plusieurs utilisateurs
- <https://git-scm.com/>
- Un tuto très complet
  - <https://openclassrooms.com/courses/gerez-vos-codes-source-avec-git>

# GIT : FONCTIONNEMENT



# GIT

## ○ Les mots clés

- **Clone** : Récupérer un projet du serveur en local
- **Pull** : Récupérer les nouveautés depuis le serveur
- **Add** : ajouter un fichier à Git
- **Commit** : Sauvegarder en local ses modifications
- **Push** : Envoyez ses modifications au serveur

# GIT : FONCTIONNEMENT

- Récupérer les modifications des autres avec un **Pull**
- Modifier son code et effectuer des '**commit**' réguliers => Sauvegarde en local
- Quand cela fonctionne => Effectuer un **push** pour envoyer au serveur