



# **FORMATION ANDROID**

# QUI SUIS-JE ?

- Anthony Monteiro
- Indépendant sur Toulouse
- Spécialisations
  - Architecture (Graphique, technique, projet).
  - User Experience
- Application sur le store :
  - UrbanPulse
  - Acadomia (iOS & Android)
- Site internet : [www.amonteiro.fr](http://www.amonteiro.fr)
- Contact : [contact@amonteiro.fr](mailto:contact@amonteiro.fr)

# PROJET

- Lyra Network
  - Payzen Mobile
  - SDK de paiement en ligne
- Fédération de Flying Disc France (Frisbee)
  - Ultimate Line Manager

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
  - 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



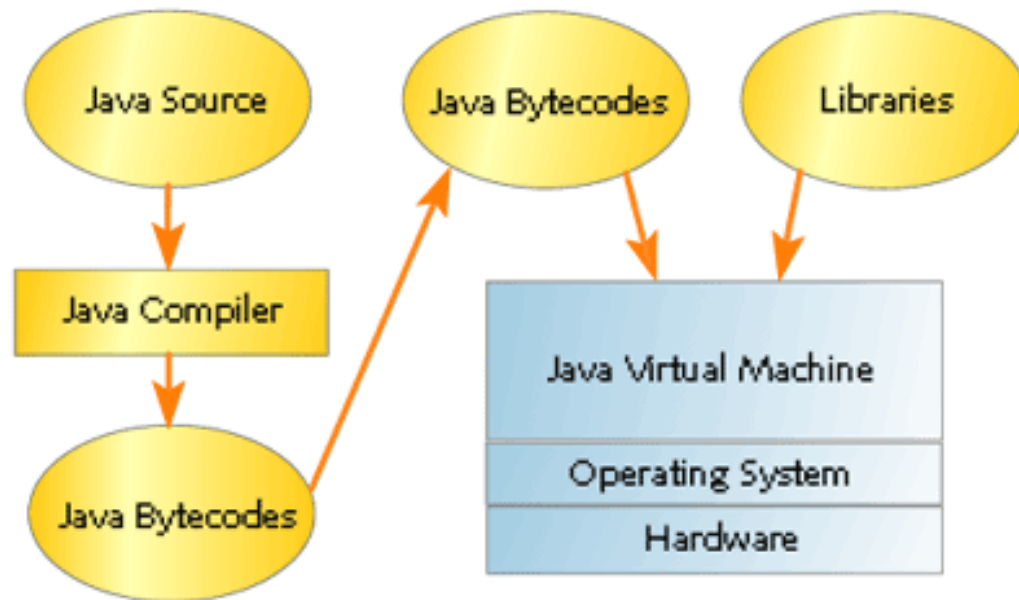
# RAPPELS JAVA

8



# RAPPELS JAVA

- La JVM



# RAPPELS JAVA

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

    //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

- Classe abstraite

```
//Déclaration
public abstract class Personne{
    private String prenom;
    protected abstract int getIdentifiant();
    protected String getPrenom() { return prenom; }
}
```

```
//Utilisation
public class Eleve extends Personne {
    public Eleve(){
        super();
    }
    @Override
    protected int getIdentifiant() { return 2; }
}
```

# RAPPELS JAVA

- Interface

//Déclaration

```
public interface CallBack{  
    void onClick (Eleve eleve);  
}
```

//Assignment

```
public void setCallBack(final CallBack cb) {  
    this.cb= cb;  
}
```

//Utilisation

```
if (cb!= null) {  
    cb.onClick(eleve);  
}
```



# RAPPELS JAVA

- Java.util
  - Arrays, Calendar, Date, Random, Timer...
- Transition List <-> Tableau
  - `Eleve[] elevTab = eleveList.toArray(new Eleve[0]);`
  - `Arrays.asList(elevTab);`
- Commons.lang (StringUtils, NumberUtils...)
  - `If(StringUtils.isNotBlank(var)) {`  
    ...  
    }



# ANDROID

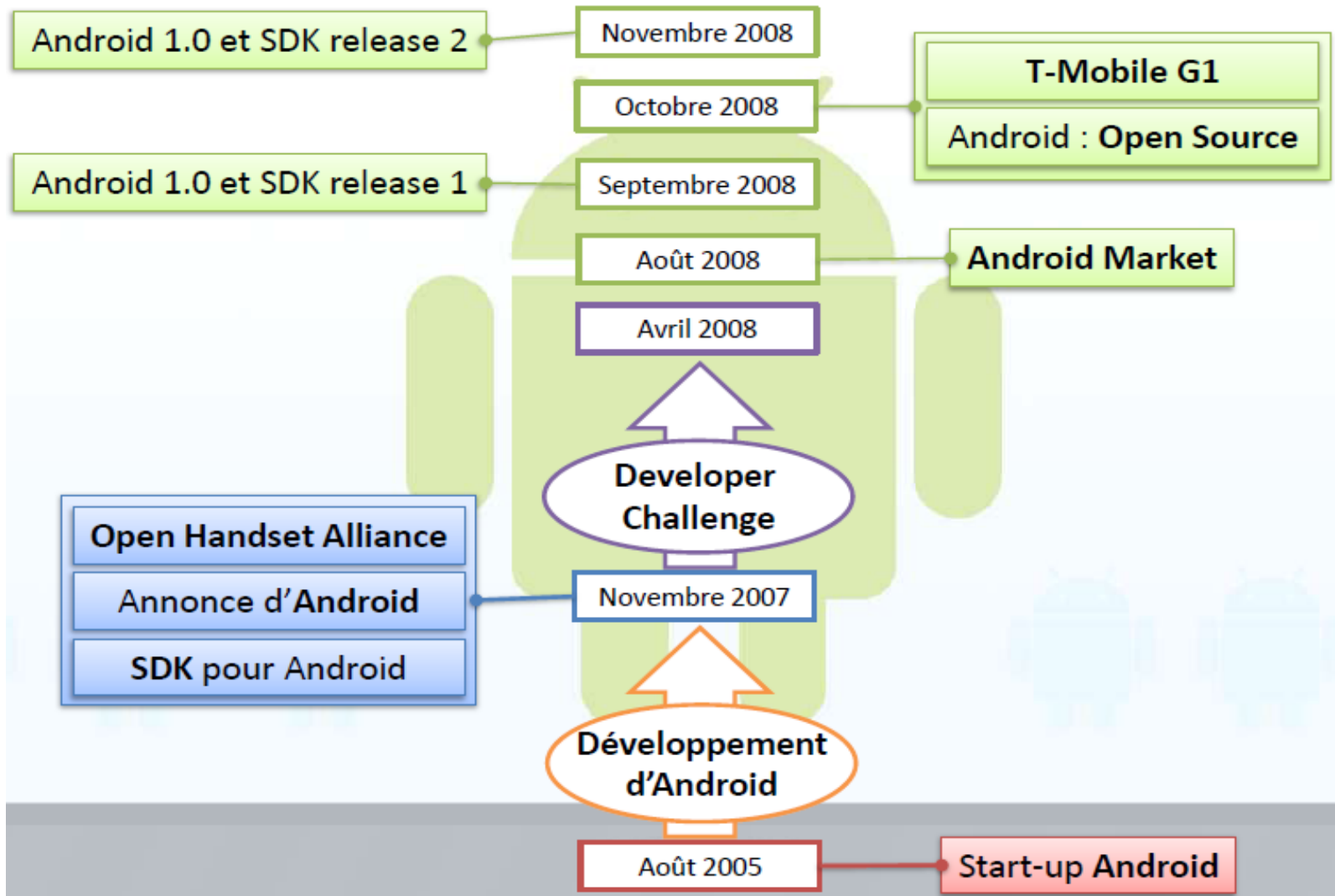
## Présentation

18

# 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



# LOLLIPOP



Android 5.0, Lollipop

# EN CHIFFRE



- 486M de smartphone vendu en 2011
- AngryBird 1M\$ / mois
- 2% des applications génèrent 90% des revenus

# EN CHIFFRE

## Ventes de smartphone en France (janvier 2015)

■ Android ■ iOS ■ Windows ■ RIM ■ Autres

Android; 65,30%



iOS; 20,20%

**Microsoft**  
Windows; 13,00%



RIM; 1,10%

Autres; 0,30%

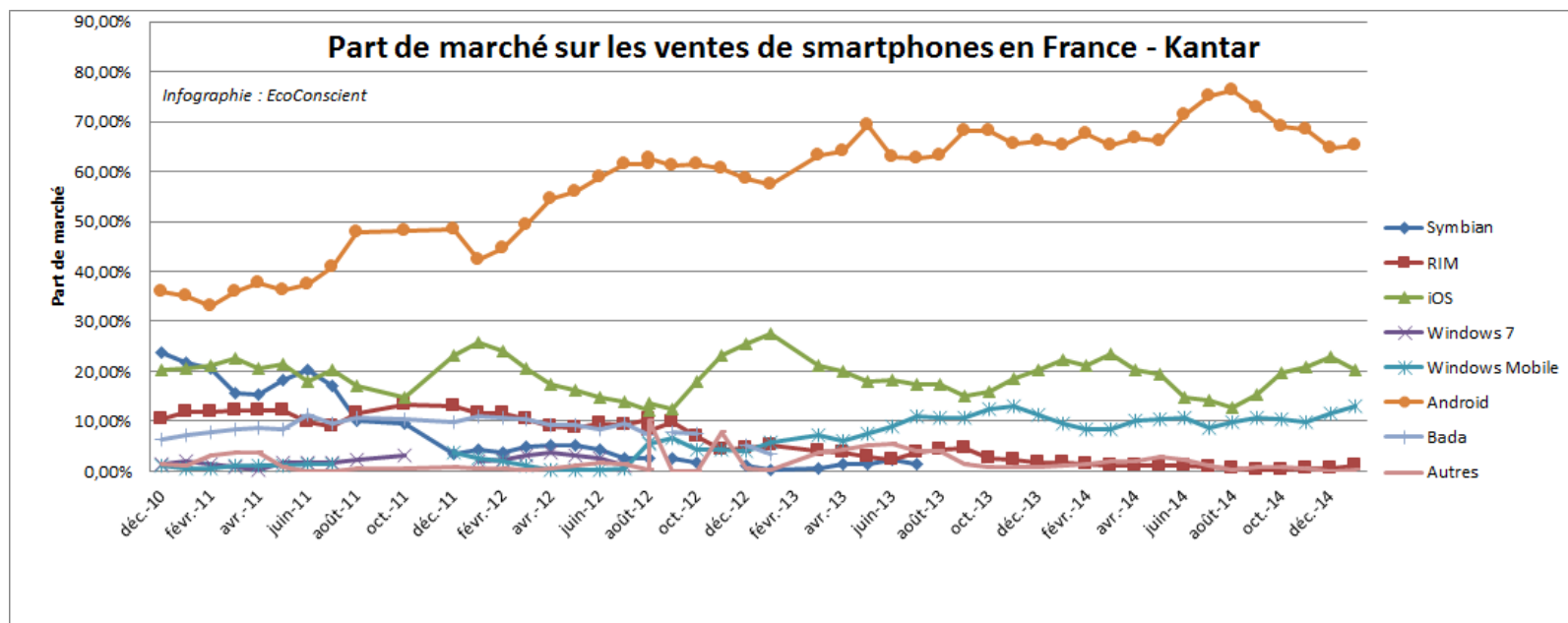
janv-15

*A propos des statistiques Kantar WorldPanel : Part de marché sur les ventes de smartphone en France. (Panel de consommateur)*

*Infographie : EcoConscient*



# EN CHIFFRE



# ANDROID C'EST

- un noyau Linux qui lui confère notamment des caractéristiques multitâches
- des bibliothèques graphiques, multimédias
- une machine virtuelle Java adaptée : la Dalvik Virtual Machine
  - ART depuis KitKat (installation--, utilisation++)
- un framework applicatif proposant des fonctionnalités de gestion de fenêtres, de téléphonie, de gestion de contenu...
- des applications dont un navigateur web, une gestion des contacts, un calendrier...

# EN IMAGE...



# DÉVELOPPER POUR ANDROID

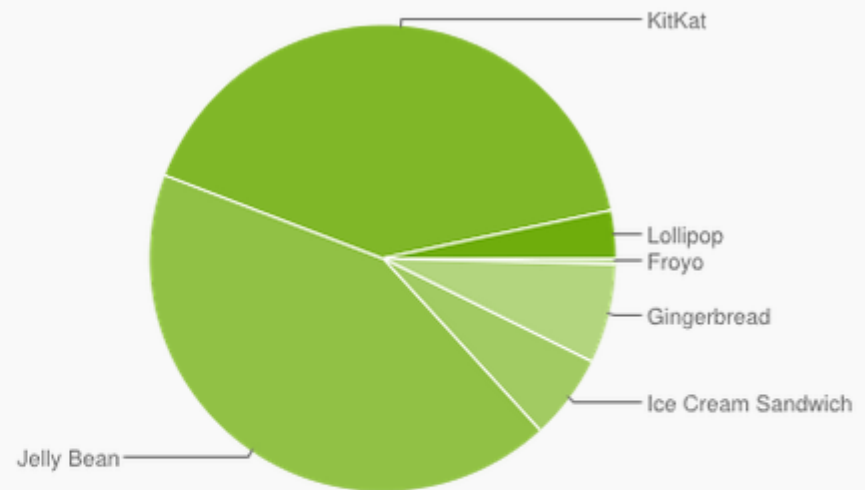
- Gratuit
- Application distribuable rapidement
  - Par mail
  - Par son propre «store»
  - Via Google Play (payant 25\$)
- Programmation en Java
  - Orientée Objet
  - Possibilité de réutiliser du code métier existant
- API Android / Google Service
  - Pas besoin d'écrire du code bas niveau

# VERSION DE L'API ANDROID

- Les API (classes et méthodes) android évoluent avec chaque nouvelle version de l'OS.
- Novembre 2014:
  - Les devices avec une version 4.0 ou supérieure représentent 89,6% du marché.

# RÉPARTITION DES OS PAR VERSION

Version	Codename	API	Distribution
2.2	Froyo	8	0.4%
2.3.3 - 2.3.7	Gingerbread	10	6.9%
4.0.3 - 4.0.4	Ice Cream Sandwich	15	5.9%
4.1.x	Jelly Bean	16	17.3%
4.2.x		17	19.4%
4.3		18	5.9%
4.4	KitKat	19	40.9%
5.0	Lollipop	21	3.3%



Data collected during a 7-day period ending on March 2, 2015.  
Any versions with less than 0.1% distribution are not shown.

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

## LIEN DIVERS

- [www.openhandsetalliance.com](http://www.openhandsetalliance.com) : alliance contribuant au développement de Android
- [source.android.com](http://source.android.com) : source pour le développement Android
- [www.android.com](http://www.android.com) : site officiel Android
- <http://developer.android.com> : documentation pour le développeur
- <http://android-france.fr> : news d'Android sur un site français
- [www.androlib.com](http://www.androlib.com) : les applications du Market sur un site en version française.



# ANDROID

Outils de développement

32



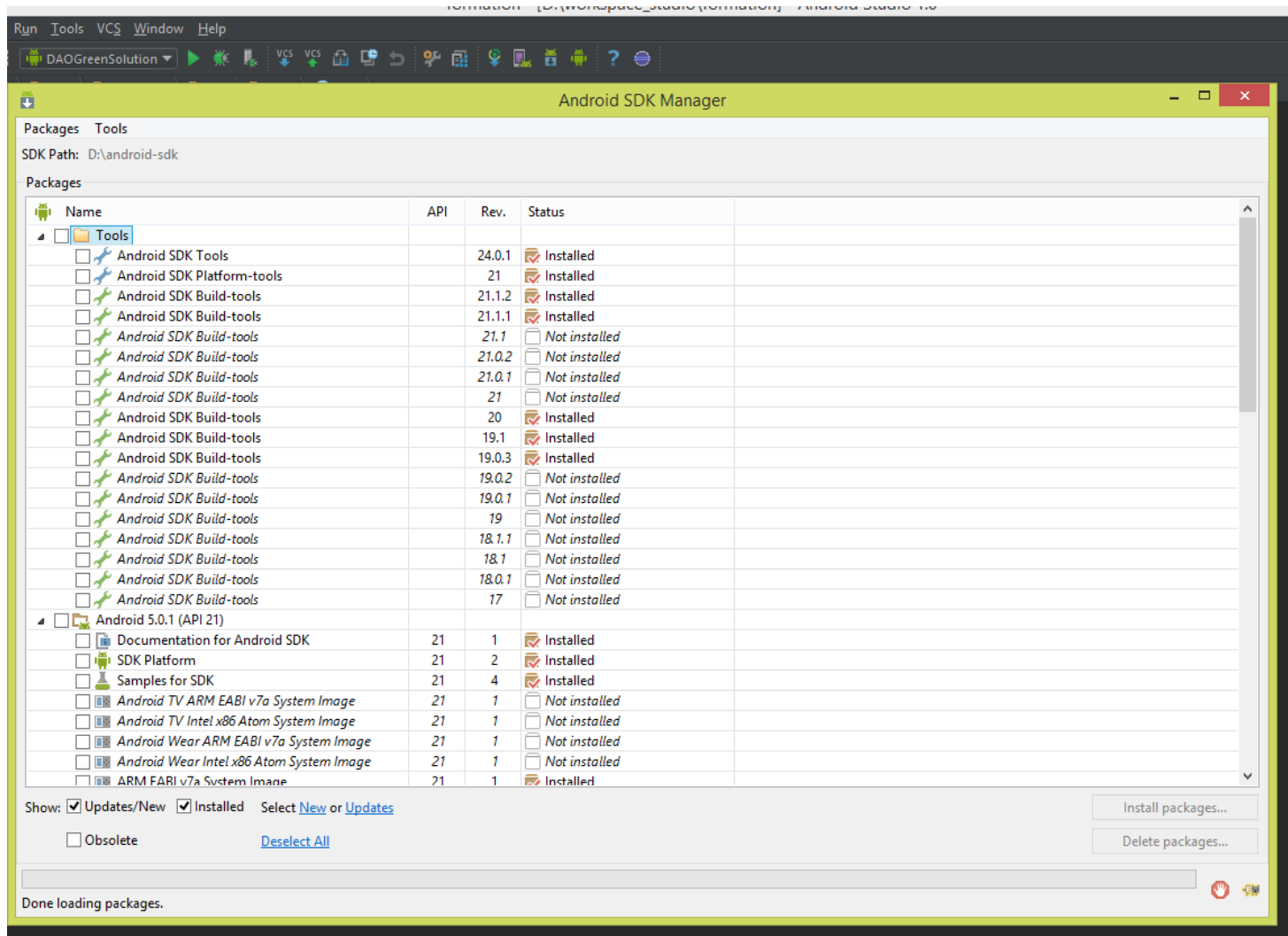
# LE KIT DE DÉVELOPPEMENT

- 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>

# SDK ANDROID



# CHARGER LES PROJETS

- Créer un compte GitHub
  - `git config --global http.proxy 'PROXY_URL'`
- Chargement du projet de formation
  - `https://github.com/Anth06ny/formationExo/tree/master`

# RÉGLAGES ANDROID STUDIO

## ○ Importer mes Settings

- File -> import settings -> cours/settingAndroidStudio.jar

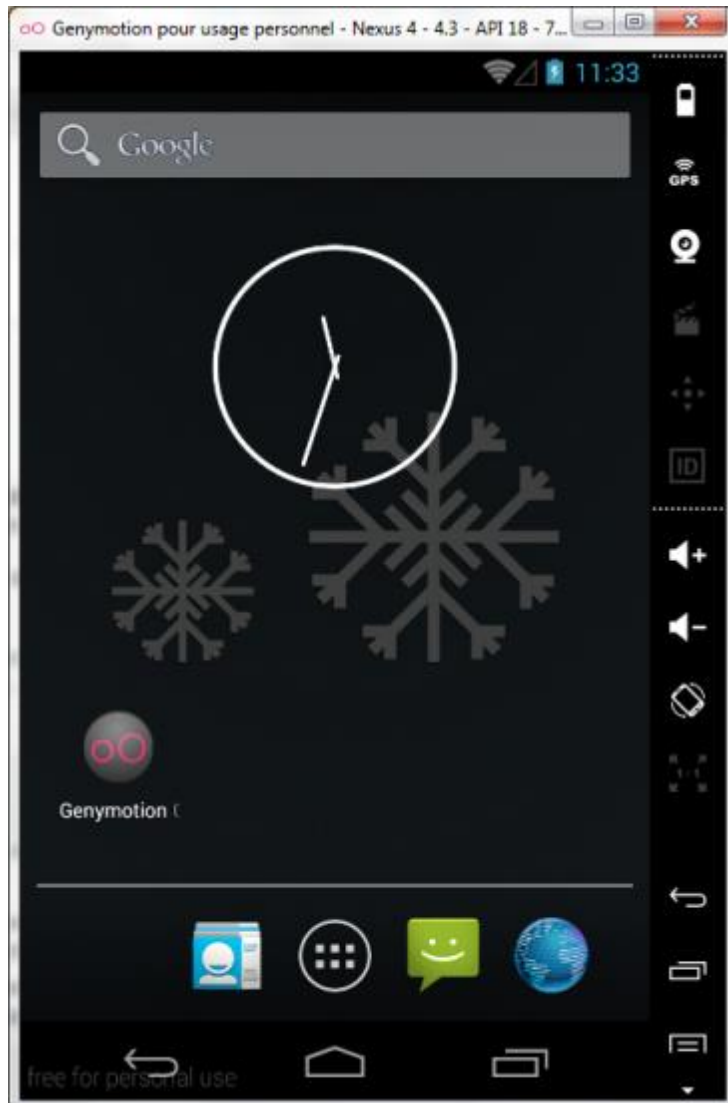
## ○ Contenu

- Raccourci clavier Eclipse
- Save Action (Import Automatique, Indent code).

# RÉGLAGES ANDROID STUDIO

- Quelques raccourcis clavier
  - Ctrl + espace : proposition de code
  - Alt + entrée : Proposition de solution
  - Ctrl + d : Supprimer ligne
  - Ctrl + shift + d : Dupliquer ligne
  - Alt + a : Revenir en arrière
  - Ctrl + g : Rechercher dans tous le projet
  - Ctrl + shift + o : Rechercher fichier dans le projet

# GENYMOTION



- L'émulateur plus rapide que le device !!!
- Les Google services ne sont plus installés.
  - [http://wiki.cyanogenmod.org/w/Google\\_Apps#Downloads](http://wiki.cyanogenmod.org/w/Google_Apps#Downloads)
- La compatibilité ARM non plus.
  - <http://filetrip.net/dl?4SUOrdCMRv>

# DALVIK DEBUG MONITOR SERVER (DDMS)

- Fonctionnalité de l'IDE que l'on ouvre via une perspective
- Liste les devices avec la possibilité d'avoir des informations sur les processus créés (thread, mémoire), fichiers systèmes
- Possibilité d'interagir avec l'émulateur, en simulant des appels ou un envoi de sms, changer de position de GPS
- Contient aussi la journalisation de toutes les activités de l'émulateur : le logCat.



# 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>



# FORMATION

Projet de travail

43

# LE PROJET

- Exercices et solution de l'ensemble des TP du cours
- Une compilation commune
  - Build.gradle + Gradle.properties
- Compilation et exécution individuelles
  - Build.gradle de chaque projet utilisant le gradle.properties commun.

# FORMATIONUTILS

- Une librairie commune
- Pour l'utiliser

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

# FORMATIONUTILS

## ○ Utils

- StringUtils

- `isNotBlank(String)` // true si non null, non blanc, non vide

- ToastUtils

- `.showToastOnUiThread(Context context, String message, int length);`

- PopupManager

- `.createProgressPopup(Activity activity, String bodyText)` //Fenêtre d'attente
- `.showPopup(Activity activity, String message, OnClickListener oclButtonOK)` //afficher une popup OK

- NotificationHelper

- `.createNotification(final Context context, Class<?> activityToLaunchOnClick)` // Afficher une notification

- Logutils

- `logTemp(message)` //Afficher un log retrouvable avec le TAG « SET\_DEBUG\_LOGTAG »
- `logException(String TAG, Throwable e);` //Afficher une exception avec stacktrace dans la console

- HttpUtils

- `isInternetConnexion(Context context);` //True si le device est connecté à internet
- `pingGoogle();` // True si google répond

# FORMATIONUTILS

## ○ Utils

- BDDUtils

- CopySQLiteBaseToDownload(String) // copier le fichier de la bdd dans download

- DateUtils

- stringToDate(String date, String format) //transformer un string en Date
- dateToString(Date date, String dateFormat) // Transformer une date en String

# FORMATIONUTILS

## ○ Bean

- Eleve

## ○ Adapter

- EleveAdapter //affiche une liste d'Eleve

## ○ Exception

- ExceptionA
- LogicException
- TechnicalException

## ○ Composant

- ButtonHighlight // Un bouton bootstrap



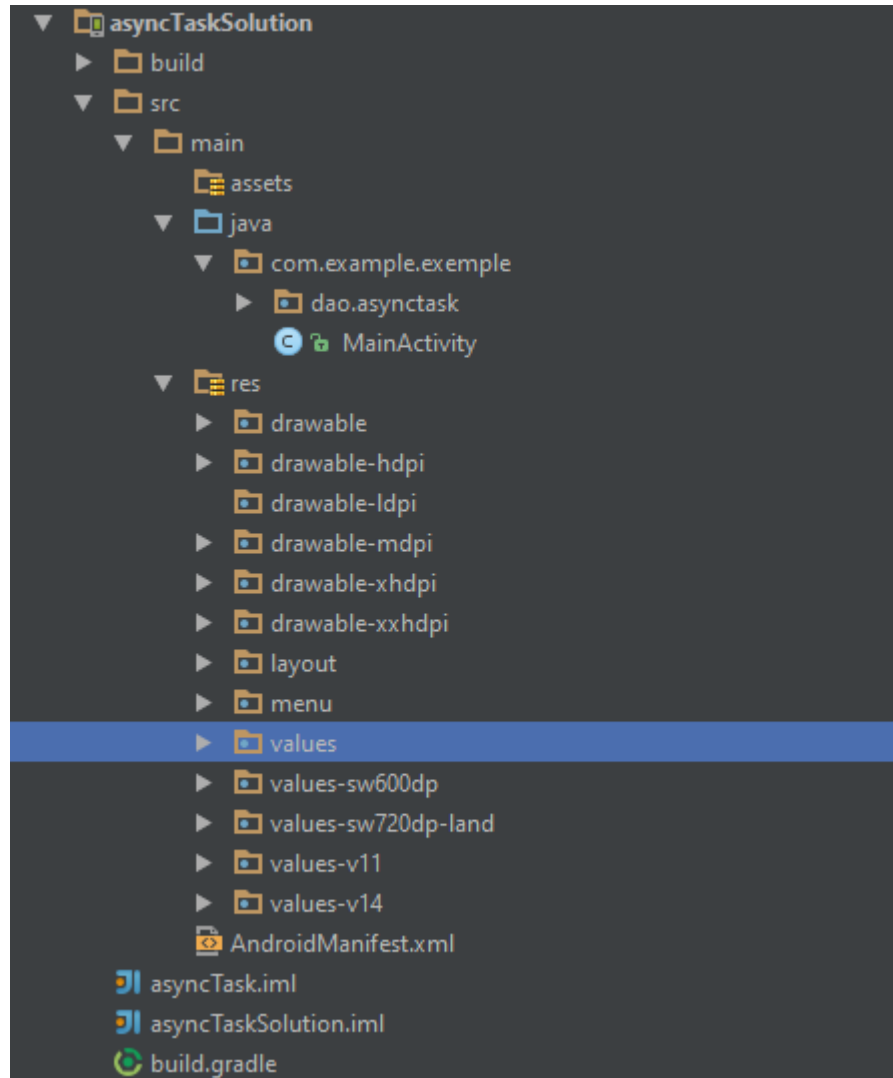


# ANDROID

## Architecture d'une application

49

# 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 JAVA

- Contient votre code source
  - Classes java
  - Différents packages

# LE RÉPERTOIRE RES

- Contient les ressources externes de l'application
  - **values** : Valeurs simples (chaines, couleurs, dimensions)
  - **drawable** : ressources images (conseil : utiliser le png)
  - **layout** : fenêtre correspondant à l'interface graphique; entièrement paramétrable en XML
  - **anim** : associées aux composants graphiques (translation, rotation)
- Possibilité de fournir des ressources pour différentes langues et tailles d'écran

# LE RÉPERTOIRE RES





















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

//dimen.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");

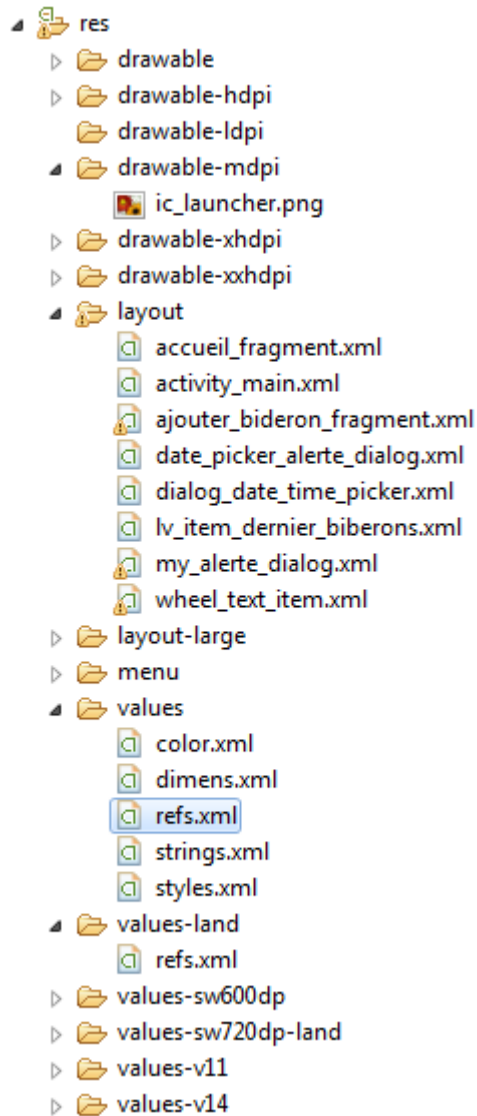
//Les chiffres
int standard_height = getResources().getDimensionPixelSize(R.dimen.standard_height);
```

# LE RÉPERTOIRE RES

Available Qualifiers	
 Country Code	
 Network Code	
 Language	
 Region	
Bidi Layout Direction	
 Smallest Screen Width	
 Screen Width	
 Screen Height	
 Size	
 Ratio	
 Orientation	
 UI Mode	
 Night Mode	
 Density	
 Touch Screen	
 Keyboard	
 Text Input	
 Navigation State	
 Navigation Method	
 Dimension	
 Version	

- On ne peut pas tout prendre tous en charge.
- Recommandation :
  - La langue
  - ~~3 tailles d'écrans (medium, large, xLarge) sw600dp~~
  - Orientation
  - Density (mdpi, ldpi, hdpi, xhdpi, )
  - Version Android

# LE RÉPERTOIRE RES



- Création de sous répertoire impossible.



# LE RÉPERTOIRE RES

- Les valeurs de bases sont dans les répertoires /res/values et peuvent contenir.
  - String : Vous pouvez utiliser les HTML tags <b>, <i> and <u>.
    - Tags compatibles : <http://daniel-codes.blogspot.fr/2011/04/html-in-textviews.html>
  - Colors : #RGB, #ARGB, #RRGGBB and #AARRGGBB
  - Dimensions: In pixels (px), inches (in), millimeters (mm), points (pt) , density-independent pixel (dp) or scale-independent pixel (sp)

# 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.

# ANDROIDMANIFEST.XML

```
<?xml version="1.0" encoding="utf-8"?>
<manifest package="com.httpexemple"
    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')  
}
```

# LA CLASSE APPLICATION

- Créée à l'initialisation de l'application, elle est accessible partout et est détruite avec l'application.
- Idéal pour initialiser les variables communes à toutes l'application.
- Initialisation de nombreuses librairies. (Capptain, Google analytics...)
- Parfait pour stocker la liste des services en cours ou appel WebService.

# LA CLASSE APPLICATION

```
public class MyApplication extends Application {

    private static MyApplication instance;
    public static MyApplication getInstance() {
        return instance;
    }

    @Override
    public void onCreate() {
        super.onCreate();
        instance = this;
    }

    /**
     * Detect si on est en simple ou double affichage
     * @return
     */
    public boolean isTwoPane() {
        return getResources().getBoolean(R.bool.twoPane);
    }
}
```

# TP

- Créez un nouveau projet
- Le lancer dans Genymotion
- Changer la langue du helloworld en fonction de la langue du téléphone.

# ANDROID - LES COMPOSANTS

- Principaux composants d'une application Android:
  - Les activités (activities)
  - Les services (services)
  - Les broadcast receivers



# LES COMPOSANTS - LES SERVICES

- Effectuent une tâche en arrière-plan
  - Téléchargement de données, jouer de la musique
- Ne présentent pas d'interface graphique
  - Se contrôlent via le code
- Emettent une / des notification(s) quand elles ont des informations à communiquer
  - «Téléchargement terminé»
- Tournent sur le processus de l'application

# LES COMPOSANTS – BROADCAST RECEIVER

- «Ecoutent» une notification donnée
  - Notification Système
    - batterie faible, sms reçu
  - Notification Interne
    - notification émise par un service
  - Notification Google
    - émise par le Google Cloud Messaging
- Exécute un code à la réception de cette notification
  - Affichage d'une notification à l'écran, code métier, arrêt d'un service...
- Possibilité d'avoir plusieurs BR par application

# 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=goldenshorestechologies.brightestflashlight.free&hl=fr\\_FR](https://play.google.com/store/apps/details?id=goldenshorestechologies.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

# ABUS DE POUVOIR

## ○ MÉFIEZ-VOUS DES APPLICATIONS LAMPE DE POCHE



**Brightest Lampe de Poche**  
GoldenShores Technologies, LLC

- 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

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](#)

Fermer

# 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

# BONNES PRATIQUES

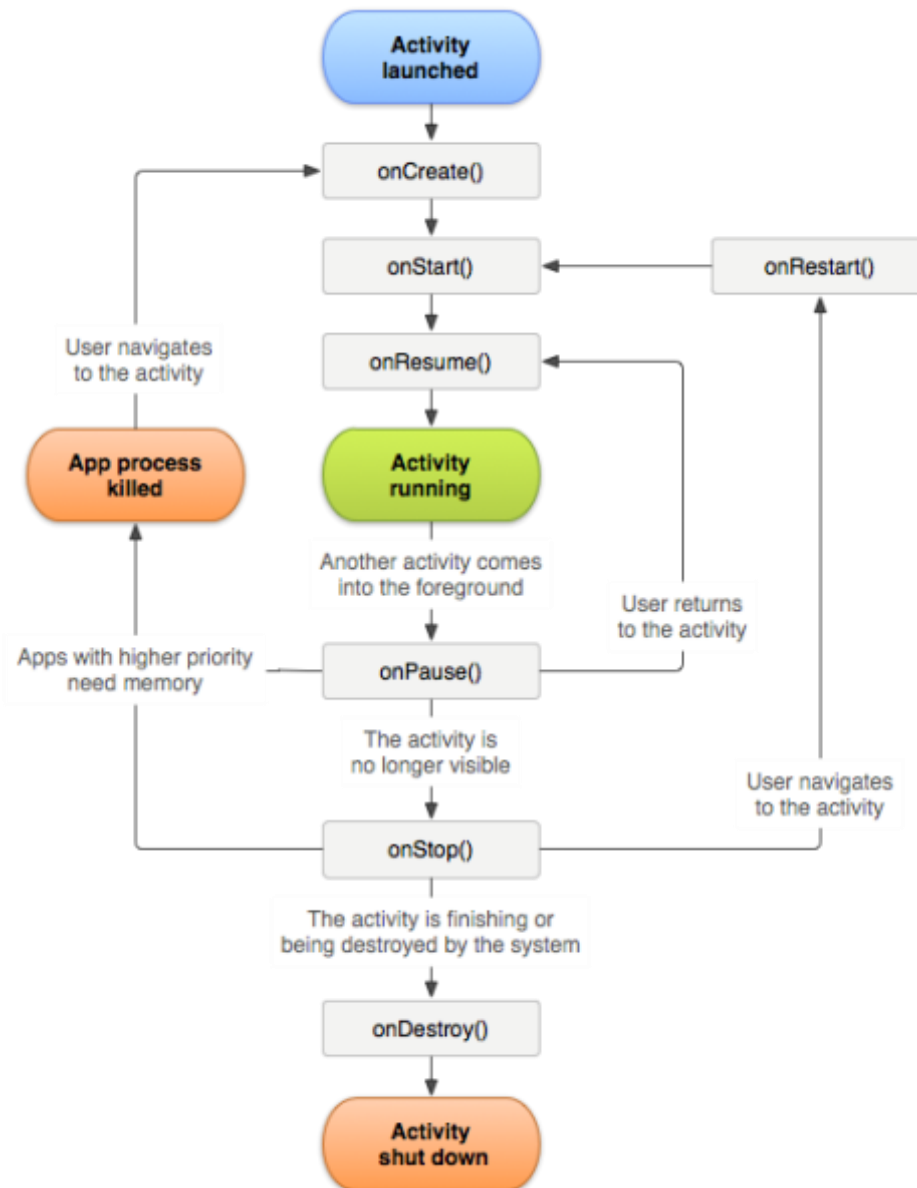
- Gestion des erreurs
  - Log
  - Messages techniques et logiques.
  - Donner la possibilité de recommencer
- Gérer les attentes augmente la fluidité
  - Un spinner ou message d'attente
  - Un préaffichage de l'écran
  - Chargement partiel des informations



# 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 systeme
    - reçoit des événements à différents moments de sa vie

## LES ACTIVITÉS



- 4 états :

- **Active** : Au 1<sup>er</sup> 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(final 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();  
    }  
}
```

# 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

```
final Intent intent = new Intent(this, SecondActivity.class);  
// Paramètre  
intent.putExtra("Cle", "Hello from MainActivity");  
startActivity(intent);  
// Tuer l'activité courante  
finish()
```

# 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 sûr d'avoir toujours le même dans la console.
  - Définir une variable permettant de savoir si on est en prod.

```
protected void onResume() {  
    //Log.e("tag" , "text");  
    if (Constante.LOG_DEV_MODE) {  
        Log.v(LogUtils.SET_FRONT_LOGTAG, "activity: main_activity ");  
    }  
}
```

# TP

- Créez un nouveau projet
- Ajouter une seconde Activity
- Créer un bouton sur la 1<sup>er</sup> qui redirige sur la seconde
- Surchargez chacun des événements (OnCreate, onResume...) correspondant au changement d'état dans le cycle de vie de l'activity et y mettre des logs.
- «Jouer» avec le simulateur pour comprendre à quel moment ces messages sont reçus
- Même chose en ajoutant un `SystemClock.sleep` dans chacun des états.
- Ajouter un point d'arrêt pour utiliser le mode debug





**IHM**



**81**



# ANDROID

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

```
//On peut afficher les info
runOnUiThread(new Runnable() {

    @Override
    public void run() {
        //traitement IHM
        tv_prenom.setText("Bob");
    }
});
```

# 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 :
  - Linear
  - Relatif
  - Table
  - ...

# LES LAYOUTS ET LE XML

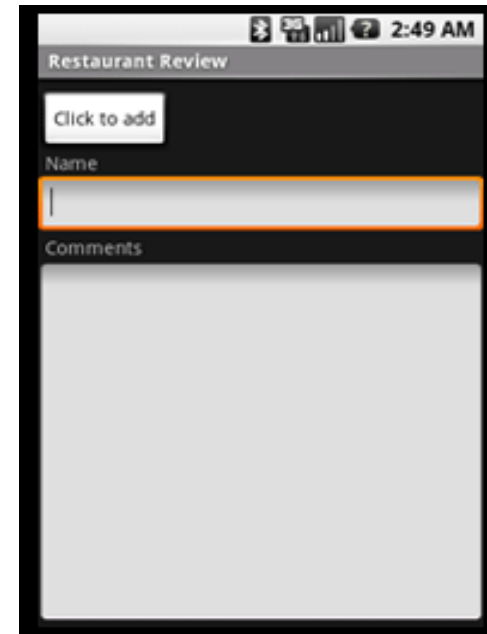
```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    // Attributs du layout
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical" >

    <!-- Contenu du layout -->

</LinearLayout>
```

# LINEAR LAYOUT

- Layout le plus utilisé
- Container permettant de placer les éléments en ligne
- Constructeur :
  - `LinearLayout(context,[object arg])`
- Méthodes :
  - `setOrientation(LinearLayout.VERTICAL)` : changer l'orientation des lignes du container
  - `setGravity(Gravity.RIGHT)` : place les éléments selon un côté.
  - `setPadding(left,top,right,bottom)` : marge des composants
  - `addView(View)` : ajouter des composants graphiques
  - `setBackgroundDrawable(BitmapDrawable)`: definit un arrière plan

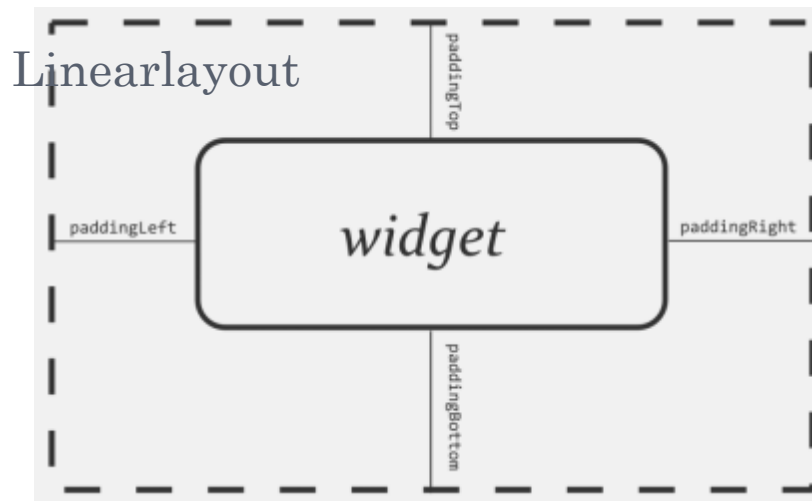


# ATTRIBUTS DU LINEARLAYOUT

- Orientation : indique si le layout ajoute les composants par ligne ou par colonne
  - android:orientation
- Width et height : 2 propriétés
  - "wrap-content" et "fill\_parent" (match\_parent)
- 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





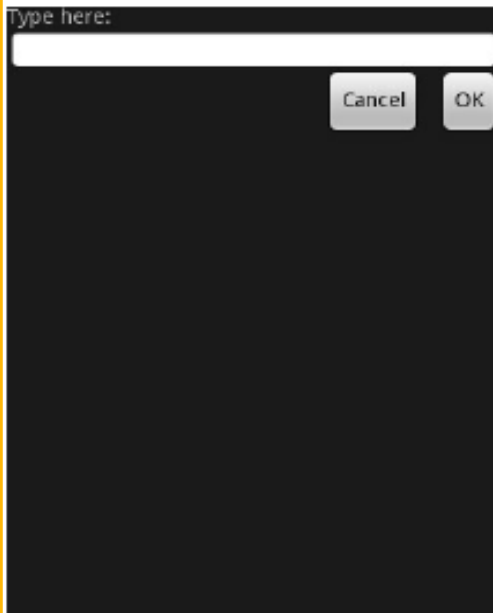
# RELATIVE LAYOUT

- 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

# Exemple

Type here:

# exemple

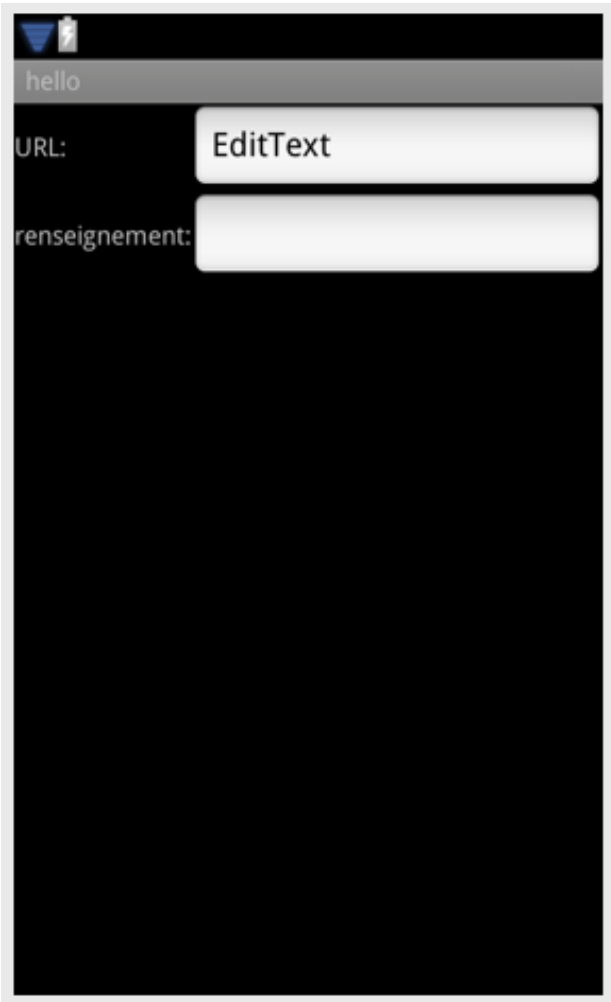


```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent">
    <TextView
        android:id="@+id/label"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="Type here:"/>
    <EditText
        android:id="@+id/entry"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:background="@android:drawable/editbox_background"
        android:layout_below="@id/label"/>
    <Button
        android:id="@+id/ok"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_below="@id/entry"
        android:layout_alignParentRight="true"
        android:layout_marginLeft="10dip"
        android:text="OK" />
    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_toLeftOf="@id/ok"
        android:layout_alignTop="@id/ok"
        android:text="Cancel" />
</RelativeLayout>
```

# 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>
```

# BONNES PRATIQUES

- Attention aux nombres de layout utilisés.
- Si la support library est utilisée au moins une fois, alors elle doit l'être partout. Si version min < IceCream.
- Utiliser les fichiers de config (dimen) pour définir les tailles des composants en fonction du device.
- Un galaxy S2 et un nexus 10 n'ont pas la même taille d'écran ni de densité mais ils lanceront la même application, mais pas les mêmes fichiers de value.

# 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

The screenshot shows an Android application interface with a dark theme. At the top, the status bar displays various icons including a warning triangle, a USB icon, a 3G signal icon, and a battery icon, along with the time 3:27. Below the status bar, the app's title bar reads "mes musiques". The main title of the screen is "Ajout/Edit musique". A text input field labeled "titre" is highlighted with a red border. Below this, there are two radio buttons: "j'aime" (which is selected, indicated by a green dot) and "je n'aime pas". Underneath the radio buttons, the label "Categories" is followed by a text input field containing the text "une categorie". A large, empty text area labeled "description" is positioned below the categories field. At the bottom of the screen, there are two buttons: "ok" and "annuler".

# 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/Button01"  
android:layout_width="wrap_content"  
android:layout_height="wrap_content"/>
```

- Dans le code de l'activity (au moment du onCreate)

- On récupère le composant via son identifiant

```
Button b= (Button)findViewById(R.id.Button01) ;
```

# GESTION DES ÉVÉNEMENTS

- Un écouteur d'événements est appelé en java un « Listener ».
- Certains composants peuvent réagir à des événements
  - Boutons réagissent au clic
- Ces composants proposent une API pour leur ajouter un écouteur de l'événement X
  - `public void setOnXListner(OnXListener listener)`

```
Button myButton = ...;  
  
myButton.setOnClickListener(new View.OnClickListener() {  
    public void onClick(View view) {  
        // Exécuter quand on clique sur myButton  
    }  
});
```

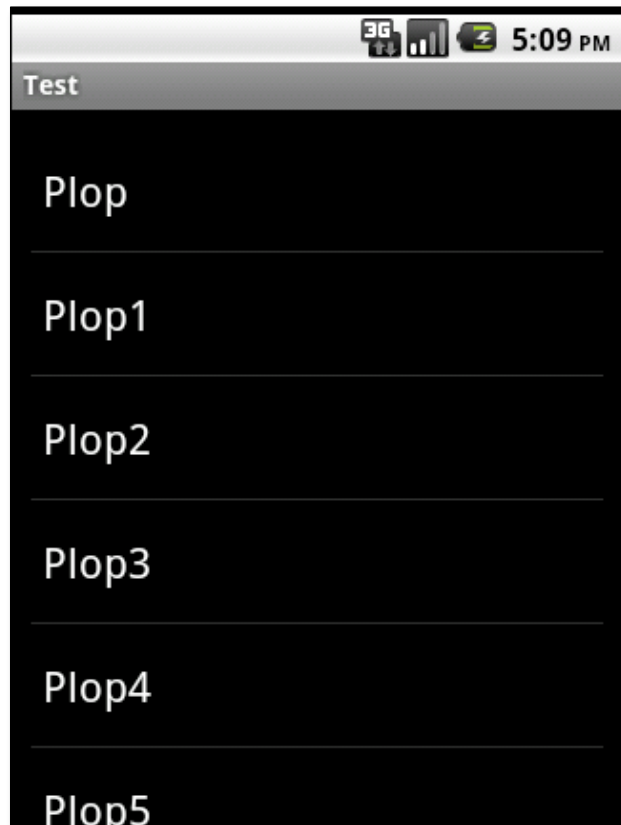


# QUELQUES APIs

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

# ANDROID

- Afficher les données sous forme de liste



# LISTVIEW

- Utilise un «adapter» pour savoir quoi afficher
  - ArrayAdapter
    - Construit simplement avec une ArrayList
    - Affiche le toString() de chaque objet
  - BaseAdapter (90% du temps)
    - Plus complexe
    - Permet un design plus évolué
      - design d'une cellule en XML
- Recycler les cellules, pour le bien de votre device.

# Exemple : BaseAdapter 1/4

```
public class EleveAdapter extends BaseAdapter {  
    private LayoutInflater mInflater;  
    private List<Eleve> eleveList;  
  
    public EleveAdapter(Context context, List<Eleve> eleveList) {  
        mInflater = (LayoutInflater)  
            context.getSystemService(Context.LAYOUT_INFLATER_SERVICE);  
        this.eleveList = eleveList;  
    }  
  
    @Override  
    public int getCount() {  
        return eleveList.size();  
    }  
  
    @Override  
    public Eleve getItem(int position) {  
        return eleveList.get(position);  
    }  
  
    @Override  
    public long getItemId(int position) {  
        return position;  
    }  
}
```

# Exemple : BaseAdapter 2/4

- Le View Holder represente les composants à modifier entre chaque cellule.
- Le but est de reutiliser une cellule existante et de ne faire que modifier les valeurs.

```
//-----  
// View Holder  
//-----  
public static class ViewHolder {  
    public TextView ec_tv_nom, ec_tv_prenom;  
    public ImageView ec_iv;  
    public Eleve eleve;  
}
```

# Exemple : BaseAdapter 3/4

```
@Override
public View getView(int position, View rowView, ViewGroup parent) {
    //-----
    // inflate
    //-----
    final ViewHolder viewHolder;
    if (rowView == null) {
        //création
        rowView = mInflater.inflate(R.layout.eleve_cellule, null);

        viewHolder = new ViewHolder();
        viewHolder.ec_tv_nom = (TextView) rowView.findViewById(R.id.ec_tv_nom);
        viewHolder.ec_tv_prenom = (TextView) rowView.findViewById(R.id.ec_tv_prenom);
        viewHolder.ec_iv = (ImageView) rowView.findViewById(R.id.ec_iv);

        rowView.setTag(viewHolder);
    }
    else {
        //recyclage
        viewHolder = (ViewHolder) rowView.getTag();
    }
}
```

# Exemple : BaseAdapter 4/4

```
//on remplit avec l'objet voulu
final Eleve eleve = getItem(position);

viewHolder.ec_tv_nom.setText(eleve.getNom());
viewHolder.ec_tv_prenom.setText(eleve.getPrenom());
viewHolder.eleve = eleve;

return rowView;
}
```

# Exemple : BaseAdapter (Coté Activity)

//Création

```
private ListView lv;  
private List<Eleve> eleveList;  
private EleveAdapter eleveAdapter;
```

```
eleveList = new ArrayList<Eleve>();  
eleveAdapter = new EleveAdapter(this, eleveList);  
lv = (ListView) findViewById(R.id.lv);  
lv.setAdapter(eleveAdapter);
```

//Mise à jour

//on vide la liste

```
eleveList.clear();
```

//on la remplit, attention à ne pas casser le pointeur.

```
eleveList.addAll(getEleves());
```

//on previent la liste que les données ont changés

```
eleveAdapter.notifyDataSetChanged();
```



# XML TO CODE

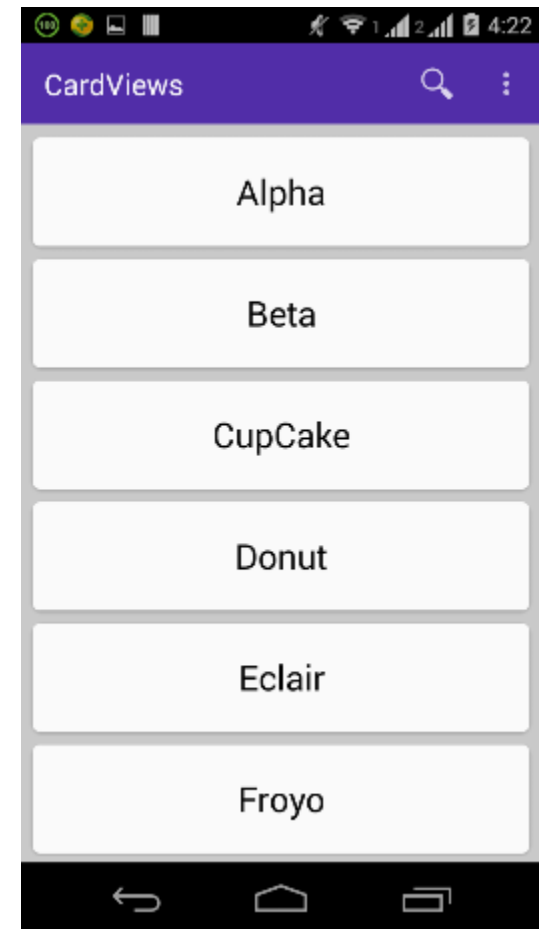
- Un site permettant de générer le code à partir du xml
  - <https://www.buzzingandroid.com/tools/android-layout-finder/>

# TP – LISTVIEW

- Créer une ListView d'élève.
  - Une cellule sera définie par le sexe, un prénom et un nom
- Ajouter un bouton pour ajouter un élève
- Pour les filles le nom sera en rose.
- Passer le device en mode paysage, qu'est ce qu'il se passe?
- Solution TP : Module listViewRecyclage

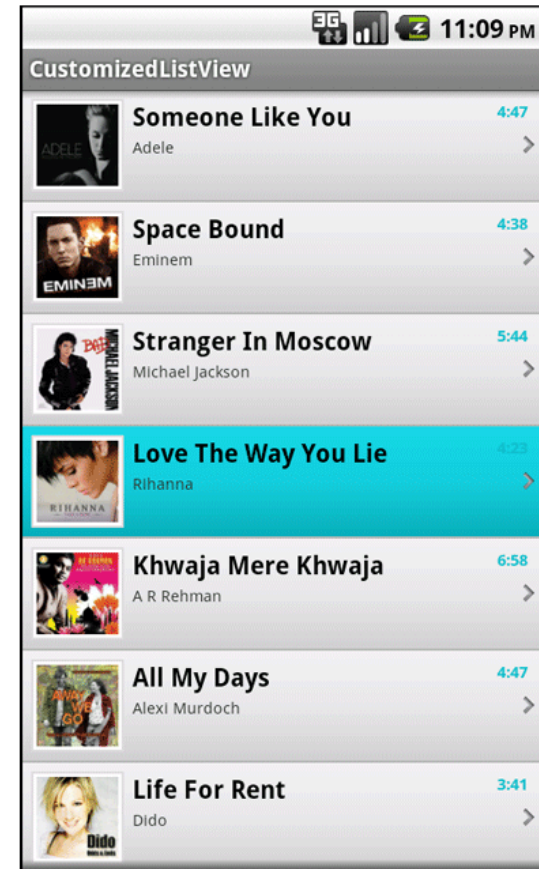
# NOUVEAU COMPOSANT

- RecyclerView
  - La nouvelle ListView de Lollipop
  - Permet des animations à l'utilisation
- CardView
  - Composant XML
  - Créer un bloc avec une ombre.



# GESTION DYNAMIQUE DES IMAGES

- Librairie Picasso
  - [square.github.io/picasso/](https://square.github.io/picasso/)
- Chargement en arrière plan des images à partir d'une url
- Gestion d'un cache mémoire et disque.



# ANDROID

Communication entre 2 activities

## LES ACTIVITÉS ET LEURS INTERACTIONS

- Android offre un système de communication très ingénieux permettant de faire passer l'information entre Activity ou plus généralement entre composants applicatifs
- Ce système est connu sous le nom d'Intent
- Il est possible de passer des informations entre activités grâce aux intents

# UTILISATION DES INTENTS

- Nous mettons directement le nom de notre activité sur l'intent
- Nous l'utilisons pour lancer nos propres activités
  - `new Intent(this, HelpActivity.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 mere
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 mere on recupere 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 à annulé
        }
    }
}
```

# Serializable et Parcelable

- En implementant serializable ou Parcelable, l'objet peut être transmit par intent.
- Parcelable est plus optimisé que serialisable 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);  
        }  
    }  
}
```

# 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(final Bundle outState) {  
    super.onSaveInstanceState(outState);  
    outState.putParcelableArrayList(Constants.EXTRA_LIST_ELEVE, eleveList);  
}
```

- Charger les éléments en mémoire.

```
protected void onRestoreInstanceState(final Bundle savedInstanceState) {  
    if (savedInstanceState != null) {  
        eleveList = savedInstanceState.getParcelableArrayList(Constants.EXTRA_LIST_ELEVE);  
        //On passe par une variable intermediaire, sinon le cast ne passe pas.  
        final ArrayList<Eleve> temp=savedInstanceState.getParcelableArrayList(Constants.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
final String callUrl = "tel:" + friend.getPhoneNumber();
startActivity(new Intent(Intent.ACTION_DIAL, Uri.parse(callUrl)));

// Sms
final 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
final String url_a_parser = getIntent().getData().toString();

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



# MENUS ET BOÎTES DE DIALOGUE

132

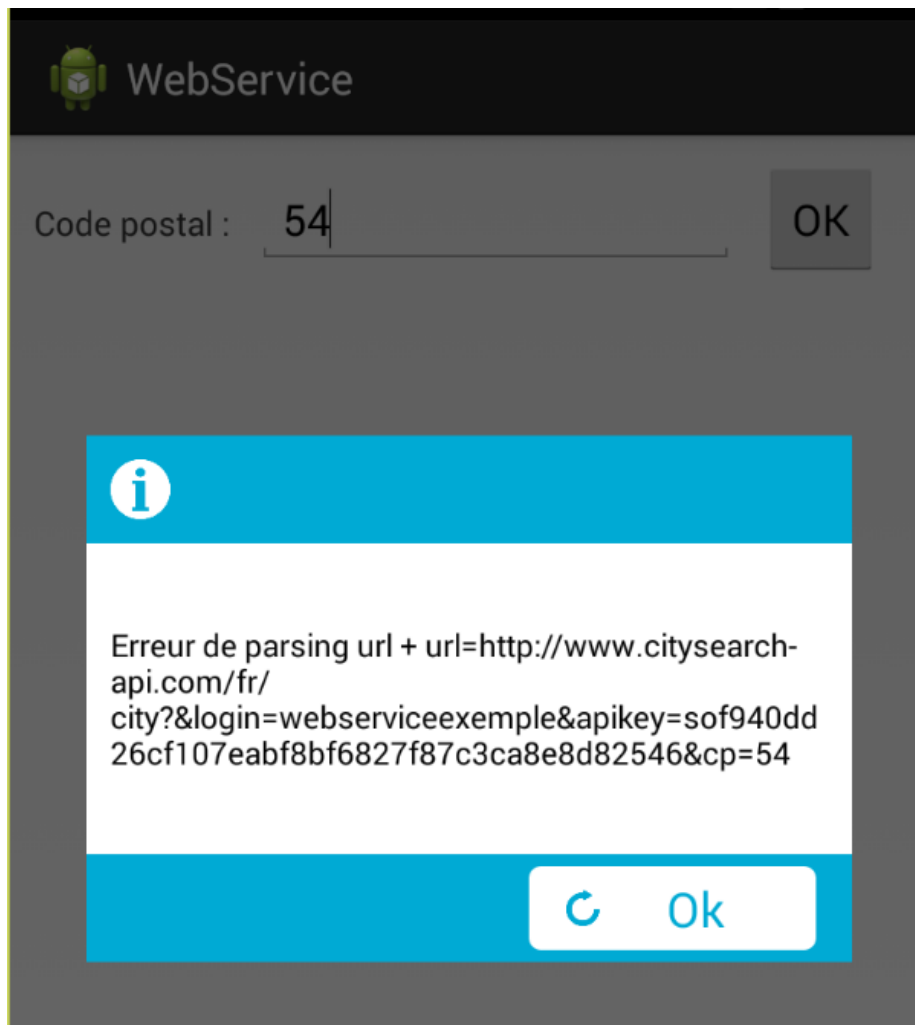
# LES BOÎTES DE DIALOGUES

- 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...

# Exemple simple (FormationUtils)

```
Dialog dialog = new Dialog(activity);  
// On implémente le XML  
View promptsView = activity.getLayoutInflater().inflate(R.layout.my_alerte_dialog, null);  
TextView tv_message = (TextView) promptsView.findViewById(R.id.tv_message);  
Button bt_valider = (Button) promptsView.findViewById(R.id.bt_valider);  
  
tv_message.setText(message);  
bt_valider.setOnClickListener(new View.OnClickListener() {  
  
    @Override  
    public void onClick(final View v) {  
        //Ferme la fenêtre  
        dialog.dismiss();  
    }  
});  
  
//Un clic en dehors ne ferme pas la fenêtre.  
dialog.setCanceledOnTouchOutside(false);  
//Pas de titre pour la fenêtre  
dialog.requestWindowFeature(Window.FEATURE_NO_TITLE);  
dialog.setContentView(promptsView);  
dialog.show();
```

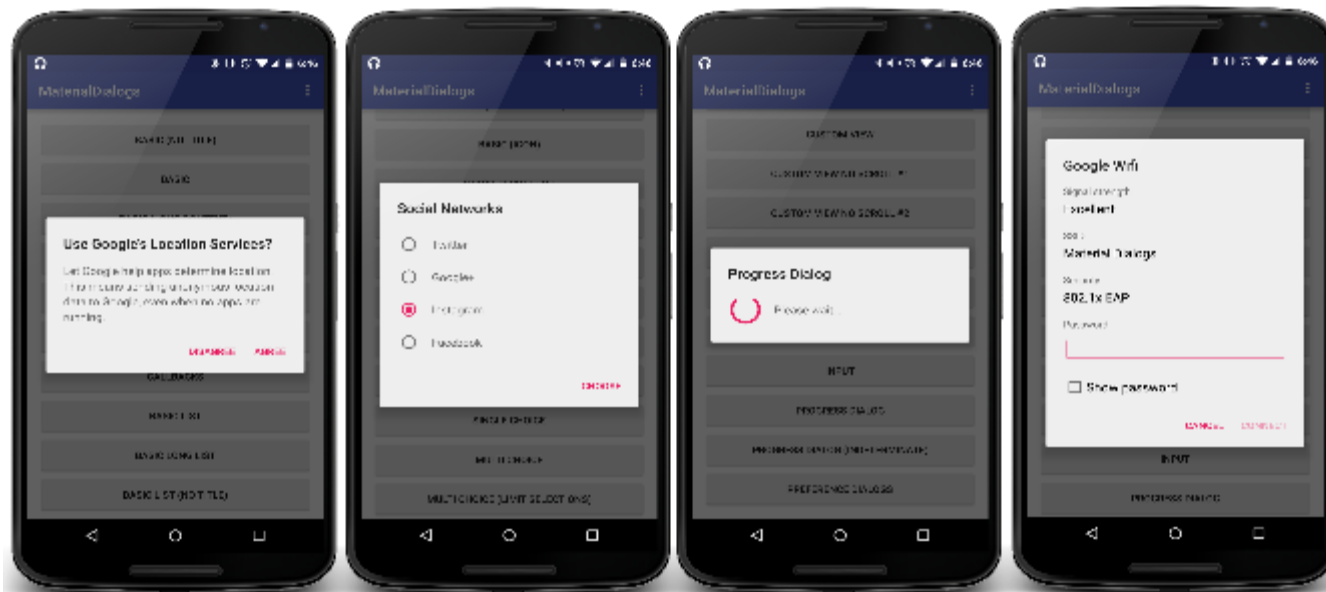
# Exemple simple (FormationUtils)



# MaterialDialog

o <https://github.com/afollestad/material-dialogs>

```
new MaterialDialog.Builder(this).title(R.string.title)
    .content(R.string.content).positiveText(R.string.agree)
    .negativeText(R.string.disagree).show();
```





# TOAST

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

```
Toast monToast;  
monToast = Toast.makeText(context, "Ceci est un Toast", Toast.LENGTH_LONG);  
monToast.show();
```

# OPTIONMENU

- Créé lors de l'appui sur le bouton Menu du téléphone
- Événement reçu par l'activité en cours
  - public boolean onCreateOptionsMenu(Menu menu)
- Possibilité d'ajouter des MenuItem à ce menu
- Callback reçu sur sélection d'un élément
  - public boolean onOptionsItemSelected(MenuItem i)

# Exemple

```
/* Crée le menu */
public boolean onCreateOptionsMenu(Menu menu)
{
    menu.add(0, MENU_NEW_GAME, 0, "New Game");
    menu.add(0, MENU_QUIT, 0, "Quit");
    return super.onCreateOptionsMenu(Menu menu);
}

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

# MENUITEM AVEC ICÔNES

- Possibilité d'ajouter une icône à un menuItem
  - `public void setIcon(Drawable icon)`
- Possibilité d'utiliser nos propres drawables ou ceux d'Android
  - accessible via `android.R.drawable.ic_*`

```
menu.add(...).setIcon(R.drawable.myIcon);  
menu.add(...).setIcon(android.R.drawable.ic_menu_agenda)
```

# SOUS MENU

- Possibilité d'ajouter un sous menu à chaque élément d'un menu
- Méthode de la classe Menu
  - public SubMenu addSubMenu(String menuTitle)
- Possibilité d'ajouter des items au sous menu retourné

```
final SubMenu fileMenu = menu.addSubMenu("File");  
fileMenu.add("new");  
fileMenu.add("open");  
fileMenu.add("save");
```

# EXAMPLE

```
public boolean onCreateOptionsMenu(Menu menu) {  
    boolean result = super.onCreateOptionsMenu(menu);  
    SubMenu fileMenu = menu.addSubMenu("File");  
    fileMenu.add("new");  
    fileMenu.add("open");  
    fileMenu.add("save");  
    return result;  
}
```

# ACTION BAR

- À partir de la version 3.0 d'Android, l'affichage du menu d'une activité s'appelle à l'intérieur d'une ActionBar
- C'est une barre de menu se situant au plus haut niveau de notre activité.

```
public class GeneriqueActivity extends AppCompatActivity { }
```

# AJOUTER DES ICONES SUR L'ACTIONBAR

- Nous ajoutons un menu de manière classique puis nous utilisons la fonction “setShowAsAction” avec plusieurs constantes :

```
public boolean onCreateOptionsMenu(Menu menu) {  
  
    MenuItem mnu1 = menu.add(0, 0, 0, "save");  
    mnu1.setIcon(R.drawable.save);  
    mnu1 setShowAsAction(MenuItem.SHOW_AS_ACTION_ALWAYS |  
        MenuItem.SHOW_AS_ACTION_WITH_TEXT);  
  
    MenuItem mnu2 = menu.add(0, 1, 0, "delete");  
    mnu2.setIcon(R.drawable.delete);  
    mnu2 setShowAsAction(MenuItem.SHOW_AS_ACTION_IF_ROOM |  
        MenuItem.SHOW_AS_ACTION_WITH_TEXT);  
  
    return true;  
  
    //Version chargement du fichier menu à parti d'un XML  
    getMenuInflater().inflate(R.menu.menu_main_activity, menu);  
}
```



# ACTIONBAR

## ○ Chargement du menu depuis l'XML

```
public boolean onCreateOptionsMenu(Menu menu) {  
    //Version chargement du fichier menu à parti d'un XML  
    getMenuInflater().inflate(R.menu.menu_main_activity, 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>
```

# DÉTAILS

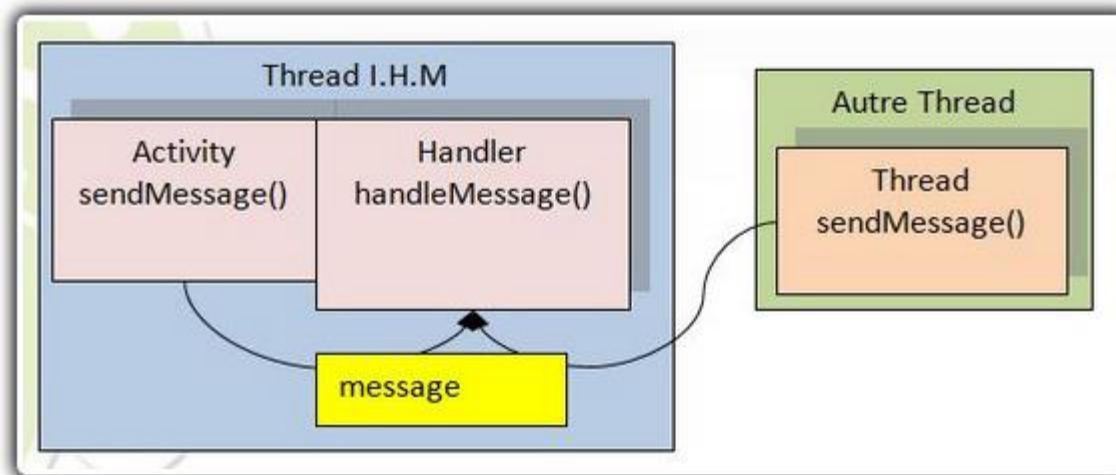
- MenuItem.SHOW\_AS\_ACTION\_ALWAYS :
  - Nous permet d'afficher l'icone même s'il y a peu de place
- MenuItem.SHOW\_AS\_ACTION\_IF\_ROOM :
  - Affiche l'icone seulement s'il y a de la place dans la barre
- MenuItem.SHOW\_AS\_ACTION\_WITH\_TEXT :
  - Affiche le texte à côté de l'icone

The left side of the slide features a series of vertical stripes in various shades of green and grey. Overlaid on these stripes are several green circles of different sizes. One large circle is positioned near the top left, and a smaller circle below it contains the number 147. Other smaller circles are scattered further down the vertical axis.

# HANDLER ET ASYNCTASK

147

# Handler



# Handler

- Permet de réaliser une FIFO d'événements vers un Thread donné (Par exemple l'UIThread).
- Pour de la mise à jour de l'IHM, ou d'événement à retardement
- La Thread envoie le message au Handler en utilisant l'une des méthodes suivantes :
  - **sendMessage** (envoie le message et le place à la fin de la queue)
  - **sendMessageToFrontOfQueue** (envoie le message et le place au début de la queue)
  - **sendMessageAtTime** (envoie le message au moment donné en paramètre et le place à la fin de la queue)
  - **sendMessageDelayed** (envoie le message après un temps d'attente passé en paramètre et le place à la fin de la queue)

# Handler

## ○ Utilisation

```
private static final int MSG_START_PROGRESS = 0;  
handler.sendMessage(MSG_START_PROGRESS);
```

# Handler

## ○ Définition du handler

```
private final Handler handler = new Handler() {  
    @Override  
    public void handleMessage(final Message msg) {  
        switch (msg.what) {  
            case MSG_START_PROGRESS:  
                if (!progressDialog.isShowing()) {  
                    progressDialog.show();  
                }  
                break;  
            case MSG_STOP_PROGRESS:  
                if (progressDialog.isShowing()) {  
                    progressDialog.cancel();  
                }  
                break;  
        }  
    }  
};
```

# TP

- Reprendre le TP sur CommonActivity et ajouter la possibilité de faire apparaître et disparaître une fenêtre d'attente pour toutes les activités qui en héritent.

```
public void startProgress() {  
    // à remplir  
}
```

```
public void stopProgress() {  
    // à remplir  
}
```



# Handler

- Chaque vue possède son propre handler.
- Action à retardement

```
view.getHandler().postDelayed(new Runnable() {  
    @Override  
    public void run() {  
        //TODO in 6 seconds  
    }  
}, 6000);
```

# 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
- 5 AsyncTask maximum simultanément, après cela se complique.
- Une tache ne peut être exécutée qu'une seule fois.

# AsyncTask (Simple)

```
public class ChargementEleveAT extends AsyncTask {  
    public ChargementEleveAT() {  
    }  
  
    // Garantie en dehors de l'UIThread  
    protected Object doInBackground(final Object... params) {  
        // traitement  
        return null;  
    }  
  
    // UIThread  
    protected void onPostExecute(final Object object) {  
  
    }  
}
```

# AsyncTask (Avancé)

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

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

    // UIThread
    protected void onPostExecute(final ExceptionA exception) {
    }
}
```

# AsyncTask (Simple)

## ○ Utilisation

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

- Si l'activité qui a lancée l'AT est détruite, elle l'est aussi.
- Une tâche 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*)

# AsyncTask (Avancé)

## ○ Utilisation

```
private ChargementEleveAT chargementEleveAT = null;

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

if (chargementEleveAT.getStatus() == Status.PENDING) {
    chargementEleveAT.execute();
};
```

# AsyncTask

- Arrêter une AsyncTask

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

- Si le « doInBackground » a commencé il finira son exécution et on ne passera pas par le « onPostExecute »

- Faire stopper le doInBackground dans celui-ci

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

# TP

- Reprendre le TP de l'affichage des élèves sur une ListView ou le module AsyncTask
- Faire en sorte que le clic sur le bouton lance une AsyncTask qui chargera une liste d'élève.
- L'AsyncTask devra rendre son résultat à l'aide de l'interface suivante :

```
public interface Callback {  
    //succes  
    void eleveLoaded(List<Eleve> eleve);  
  
    //fail  
    void loadFail(String message);  
  
    void updateChargement(int max, int current);  
}
```

- Solution : Module AsyncTaskSolution



# TP

```
o protected String doInBackground(final Void... arg0) {  
    //Appel WS avec attente  
    if (random.nextInt(2) > 0) {  
        for (int i = 0; i < nbrEleve; i++) {  
            eleveList.add(new Eleve("Jean" + i, "Pierre" + i, i % 2 == 0));  
            publishProgress(i);  
            SystemClock.sleep(1000);  
        }  
        return null;  
    }  
    else {  
        //Simulation de l'echec  
        SystemClock.sleep(2000);  
        return "Pas de chance cela a échoué";  
    }  
};
```



# BROADCAST

162

# 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

## ○ 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 envoie un toast à chaque réception de sms.
  - Doc :  
<https://developer.android.com/reference/android/provider/Telephony.Sms.Intents.html>
- Solution : Module smsBroadcast



# 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`

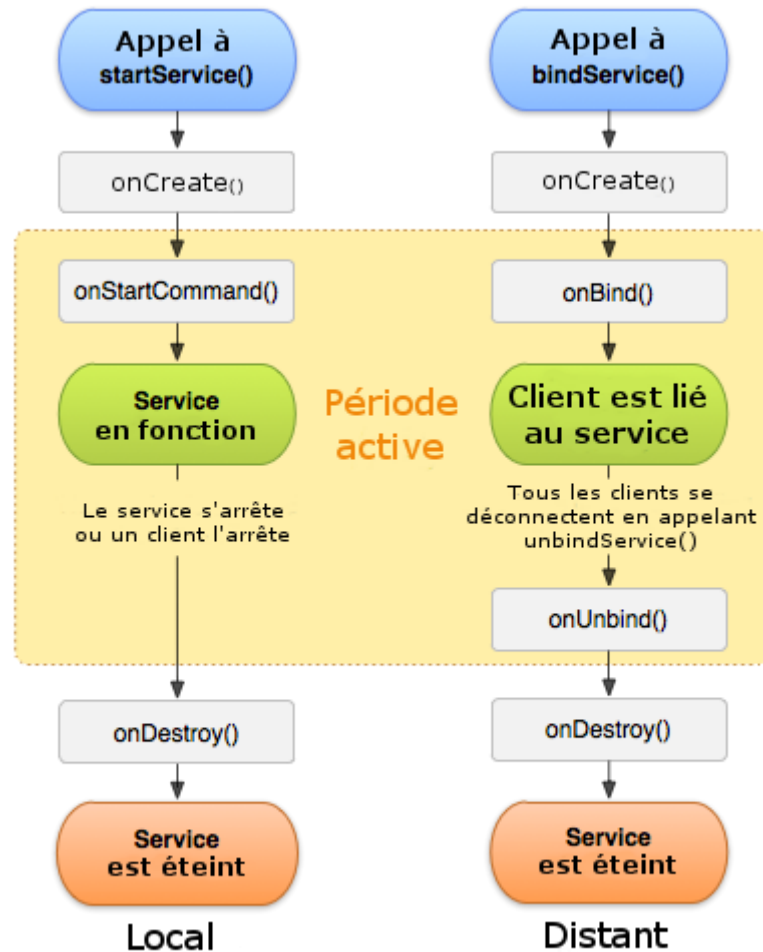
`void onDestroy(); // libération des ressources`

`IBinder onBind(Intent intent); // connexion client distant`

`boolean onUnbind(Intent intent); // déconnexion d'un client`

`void onRebind(Intent intent); // Récupérer un service`

# Services



# Exemple

```
public class BackgroundService extends Service {

    private Timer timer;

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

    @Override
    public int onStartCommand(final Intent intent, final int flags, final 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(final Intent intent) {
        return null;
    }
}
```

# SERVICE

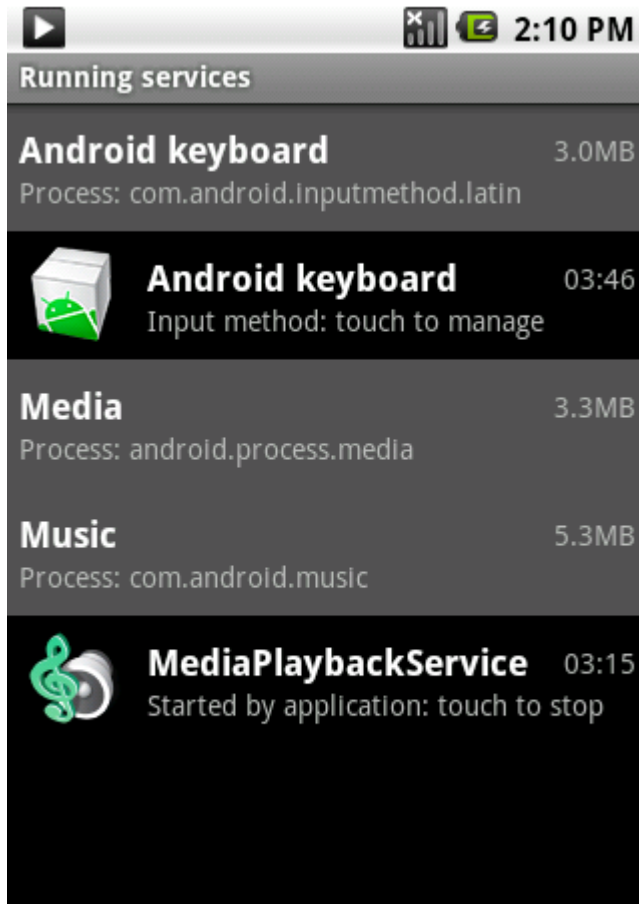
- Si le service existe déjà, lors d'un second appel, on ne passera pas par la méthode onCreate!!

# ONSTARTCOMMAND

OnStartCommand(Intent intent, int flags, int startId)

- startId : Au 1<sup>er</sup> lancement vaut 1, au 2eme 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



Avail: 20MB+24MB in 7

Other: 23MB in 3

- 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, MyService.class));
```

```
// stop le service  
stopService(new Intent(this, MyService.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.

```
//Minimum (et non égale, c'est Android qui gère) 5 secondes et 200m de difference
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(final Location location) {
    final Double latitude = location.getLatitude();
    final Double longitude = location.getLongitude();
    //do something
}

//Désabonnement
locationMgr.removeUpdate(this);
```

- Permission

```
<uses-permission android:name="android.permission.INTERNET"/>
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION"/>
<uses-permission android:name="android.permission.ACCESS_GPS"/>
<uses-permission android:name="android.permission.ACCESS_ASSISTED_GPS"/>
<uses-permission android:name="android.permission.LOCATION"/>
```

- Solution : Module serviceWithOtto



# COMMUNICATION INTERTHREAD

- Librairie otto

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

- Classe Application

```
Bus eventBus = new Bus();  
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
```

# TP

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

# 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  
    }  
}
```

# MYSERVICEBINDER

- Permet de récupérer l'instance d'un service

```
public class MyServiceBinder extends Binder {  
  
    private final BackgroundService backgroundService;  
  
    //on recoit l'instance du service  
    public MyServiceBinder(final 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(final Intent intent, final int flags, final 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(final 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();
                }
            };
        }

        //démarré le service si il n'est pas démarré
        startService(new Intent(this, MyService.class));
        Intent intent = new Intent(this, MyService.class);
        //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 detruit 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



The left side of the slide features a series of vertical stripes in various shades of green and grey. Overlaid on these stripes are several green circles of different sizes, some of which are partially cut off by the left edge of the frame.

185

## AFFICHAGE DYNAMIQUE

S'adapter à l'orientation et à la plateforme

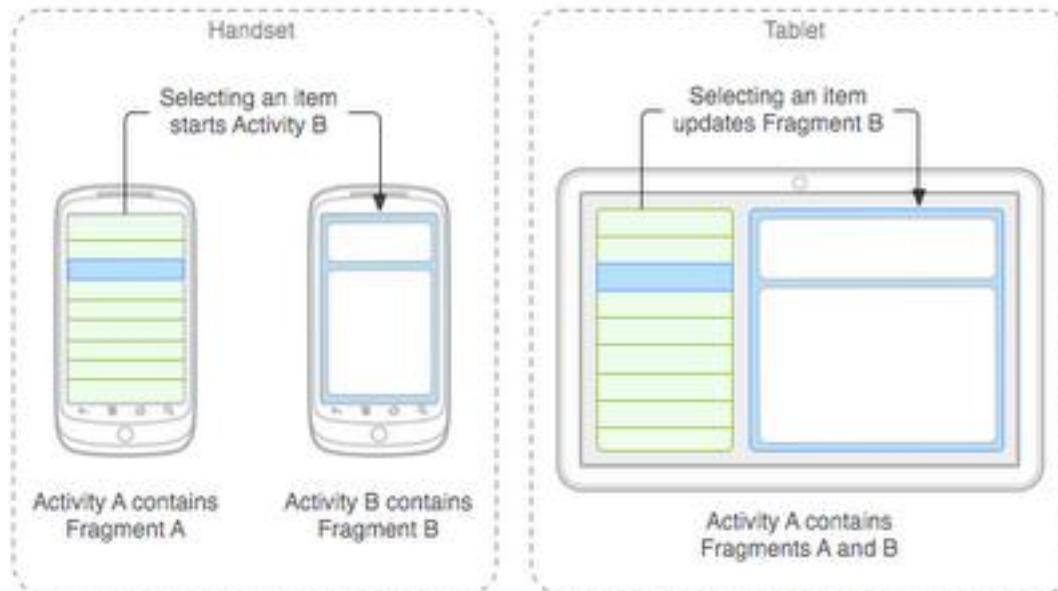
# PROBLÉMATIQUE

- Différentes tailles d'écran
  - Tablette vs telephone
- Différentes orientation
  - 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 côté 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 fonctionnalités (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 être géré par une activité
- Une activité peut gérer 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/fl_2"
        android:layout_width="0dp"
        android:layout_height="match_parent"
        android:layout_weight="3" />

</LinearLayout>
```

# VERS UNE SINGLEACTIVITY 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

# TP

- 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

# EXAMPLE: 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

# EXAMPLE: 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) {
    final View rootView = inflater.inflate(R.layout.list_fragment, container, false);

    lv = (ListView) rootView.findViewById(R.id.lv);
    if (eleve == null) {
        eleveList = new ArrayList<Eleve>();
    }
    eleveAdapter = new EleveAdapter(getActivity(), eleveList);
    lv.setAdapter(eleveAdapter);

    return rootView;
}
```



# EXAMPLE: DETAILFRAGMENT

```
public View onCreateView(final LayoutInflater inflater, final ViewGroup container, final Bundle savedInstanceState) {
    final 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(final Eleve eleve) {
    this.eleve = eleve;
}

public void refreshScreen() {
    getActivity().runOnUiThread(new Runnable() {
        @Override
        public void run() {
            if (eleve == null) {
                tv.setText("Aucun élève");
            }
            else {
                tv.setText(eleve.getPrenom() + " " + eleve.getNom());
            }
        }
    })
}
```

# EXAMPLE: MainActivity

```
public class MainActivity extends Activity implements OnClickListener {

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

    @Override
    protected void onCreate(final 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();
}
```

# TP

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

# EXAMPLE: LISTFRAGMENT

## ○ Ajouter une interface dans ListFragment

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

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

```
@Override  
public void onClick(final AdapterView<?> parent, final View view, final int position, final long id) {  
  
    final 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 (final Eleve eleve) {  
    if (MyApplication.getInstance().isTwoPane()) {  
        detailFragment.setEleve(eleve);  
        detailFragment.refreshScreen();  
    }  
    else {  
        //on demande à l'OS de remplacer le fragment  
        final 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'ecran d'avant avec un back bouton  
ft.commit();  
}  
}
```



# COMMON ACTIVITY

207

# 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(final 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(final int layoutResID) {
    final View v = getLayoutInflater().inflate(layoutResID, null);
    final FrameLayout container = (FrameLayout) findViewById(R.id.container);
    container.removeAllViews();
    container.addView(v);
}
}
```

# COMMON ACTIVITY

## ○ Exemple d'Activity Fille

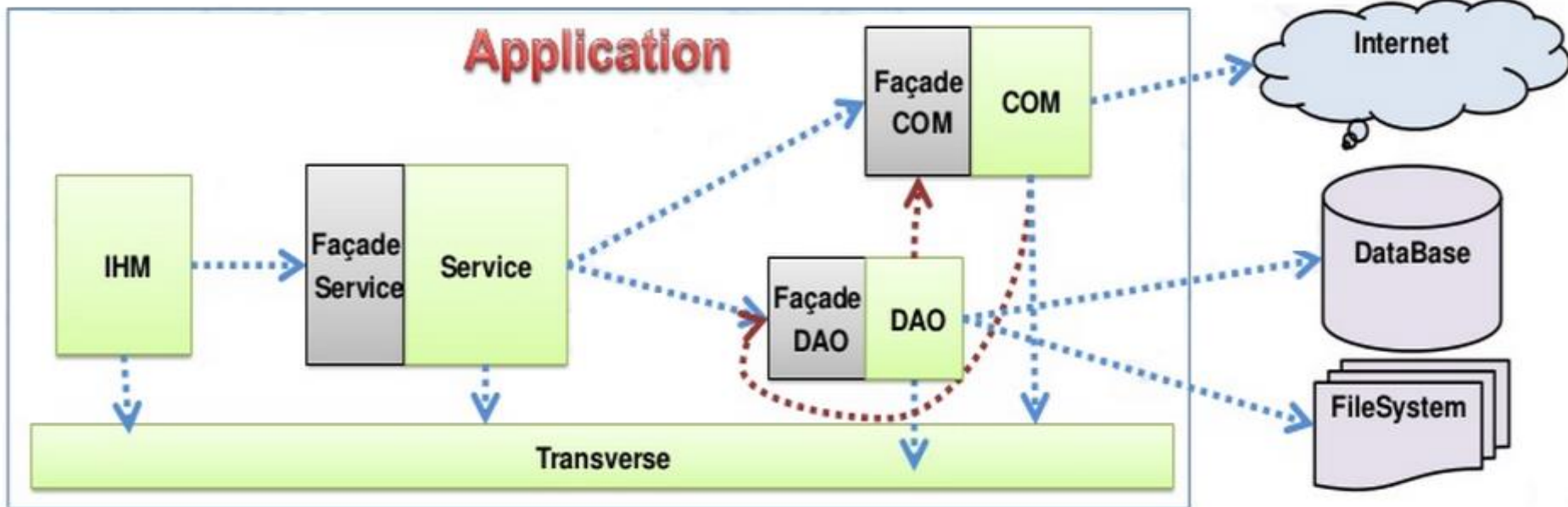
```
public class MainActivity extends CommonActivity {  
  
    private Button bt;  
    @Override  
    protected void onCreate(final Bundle savedInstanceState) {  
  
        super.onCreate(savedInstanceState);  
        super setContentView(R.layout.activity_main);  
  
        bt = (Button) findViewById(R.id.bt);  
    }  
}
```

# 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,

# TP

- 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="android:windowActionBar">false</item>  
    <item name="android:windowNoTitle">true</item>  
  
  </style>
```

# GESTION DES EXCEPTIONS

## ○ Intérêt

- Détecter 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 forcement 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**



# CRASHLYTICS ET FLURRY

## ○ CrashLytics

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

## ○ Flurry (Yahoo)

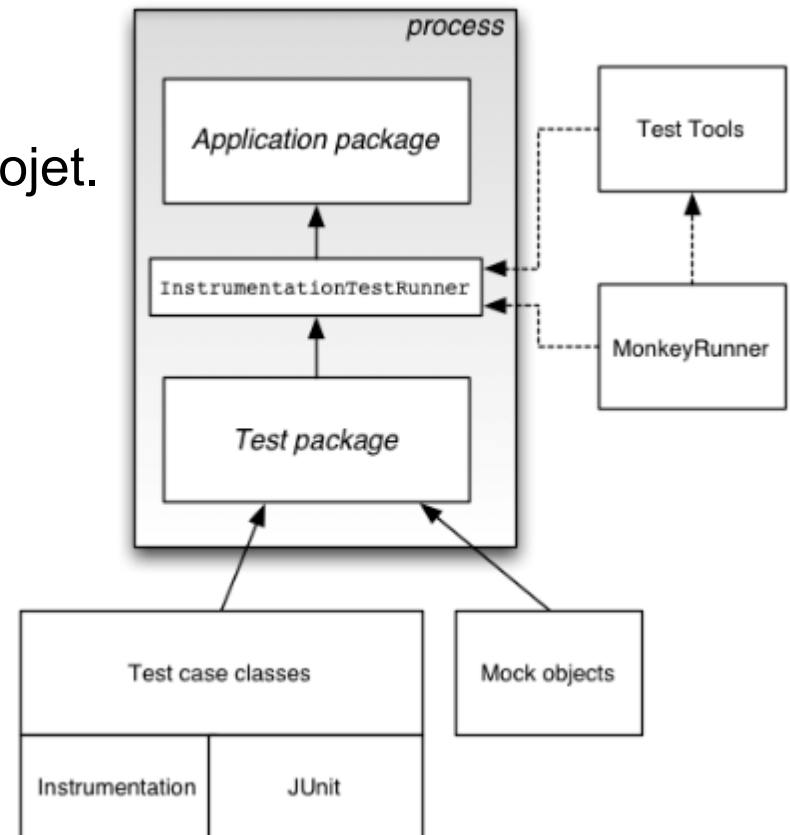
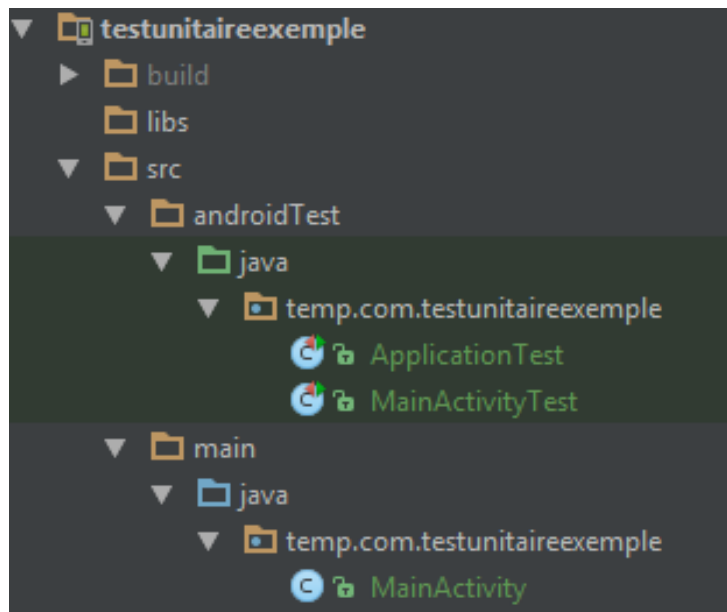
- <https://dev.flurry.com/secure/login.do>
- Gestion d'événements

## ○ 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>

# 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 méthodes 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 default
                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() {
    final 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);

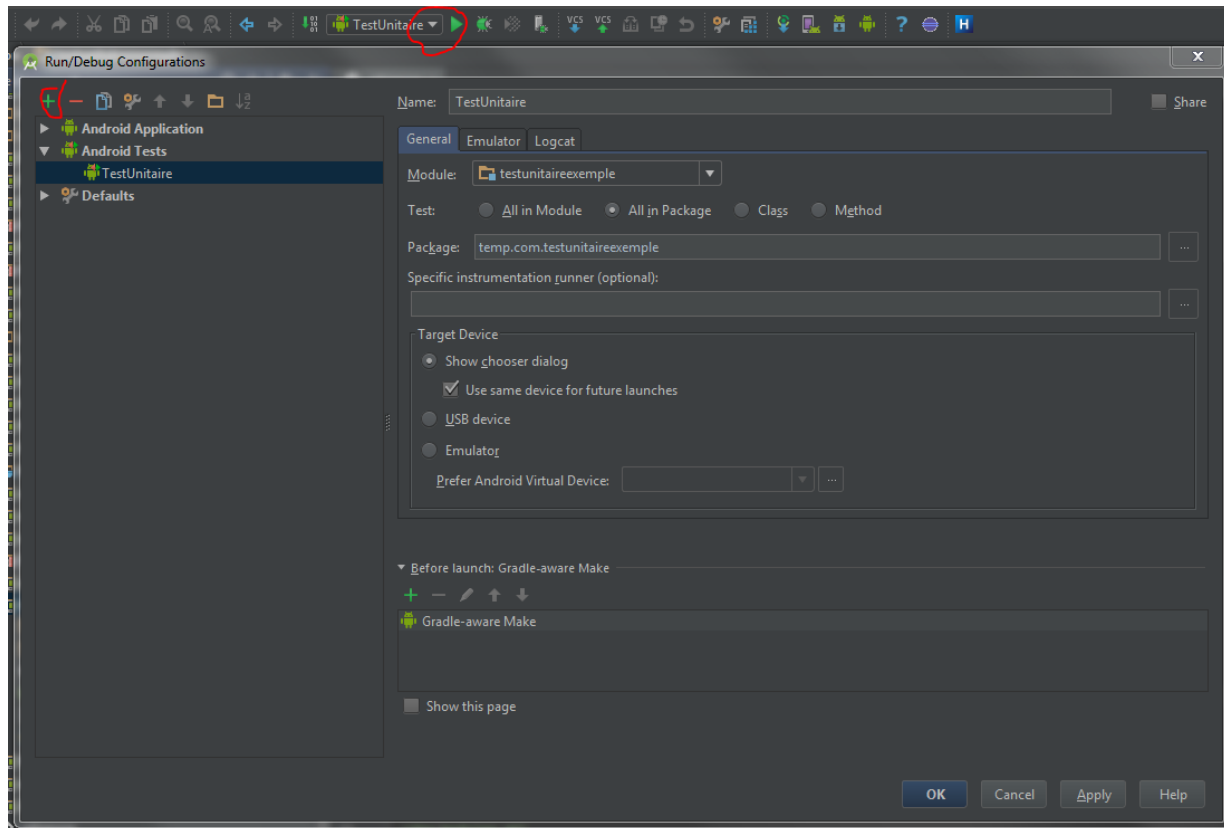
    mainActivity = getActivity();

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

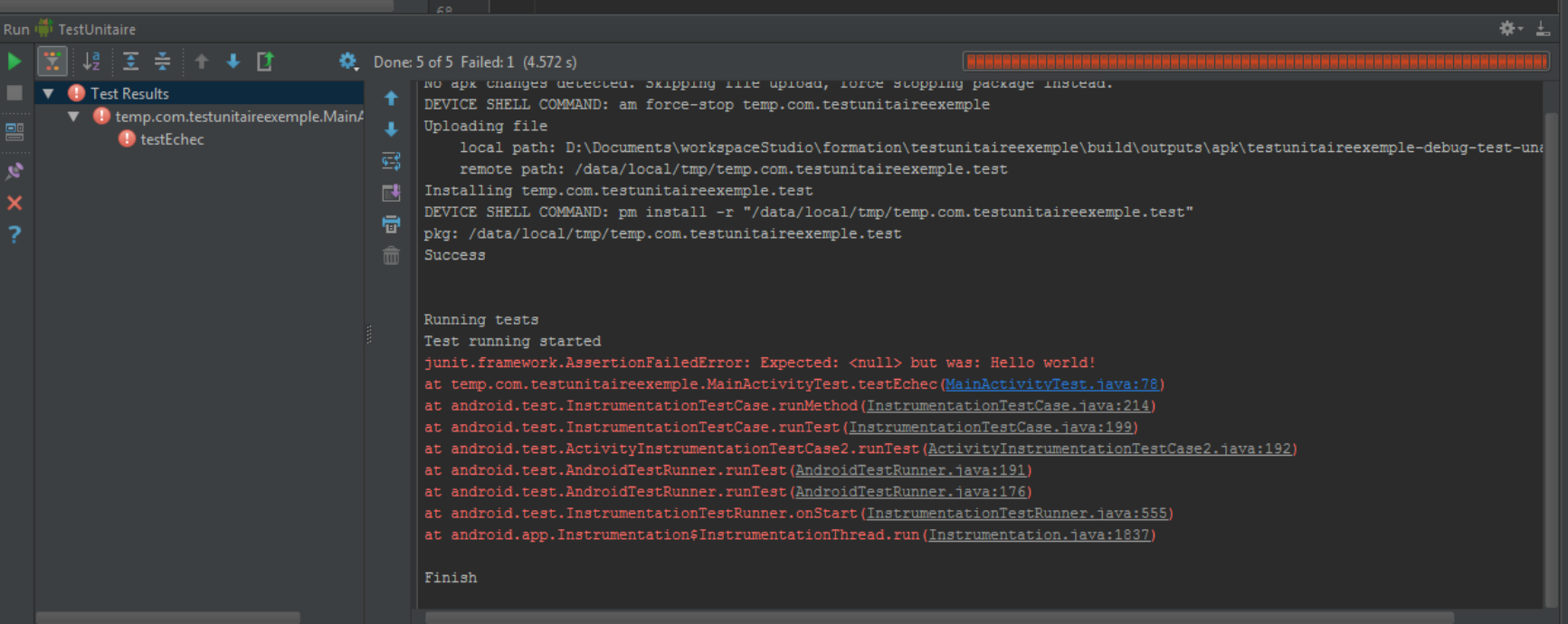
    assertEquals(saveValue, tv_hello_world.getText().toString());
}
```

# JUNIT : LANCER LES TESTS

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



# JUNIT : RÉSULTATS



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

Test Results
temp.com.testunitairexemple.MainActivity
testEchec

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

Running tests
Test running started
junit.framework.AssertionFailedError: Expected: <null> but was: Hello world!
at temp.com.testunitairexemple.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

The left side of the slide features a series of vertical stripes in various shades of green and grey. Overlaid on these stripes are several green circles of different sizes. One large circle is positioned near the top, and several smaller circles are scattered below it, some overlapping the stripes.

# USER EXPERIENCE

227

# 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+



The image shows a dark-themed login and registration interface for 'FANCY'. At the top, the word 'FANCY' is displayed in large white letters, with 'Se connecter' below it. On the left, there are three social login buttons: a red one for Google+, a blue one for Facebook, and a light blue one for Twitter. On the right, there are two white input fields for 'Email' and 'Mot de passe'. Below the password field is a blue 'Se connecter' button and a link for 'Mot de passe oublié ?'. At the bottom center, there is a link that says 'Besoin d'un compte ? S'inscrire'.

FANCY

Se connecter

g+ Sign in with Google

f Se connecter avec Facebook

Se connecter avec Twitter

Email

Mot de passe

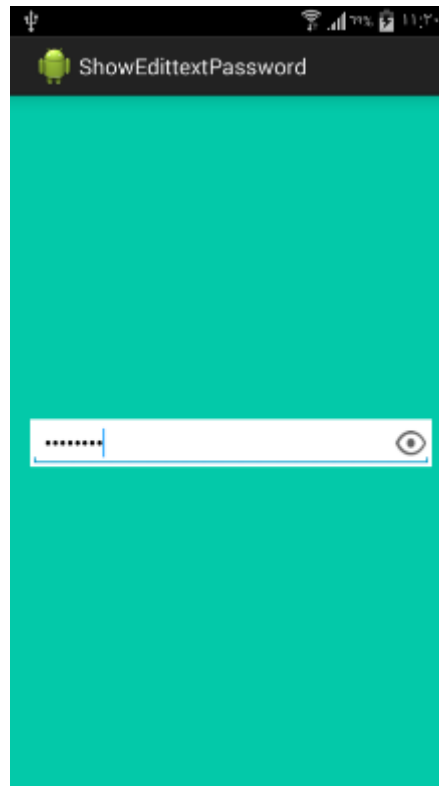
Se connecter Mot de passe oublié ?

Besoin d'un compte ? [S'inscrire](#)



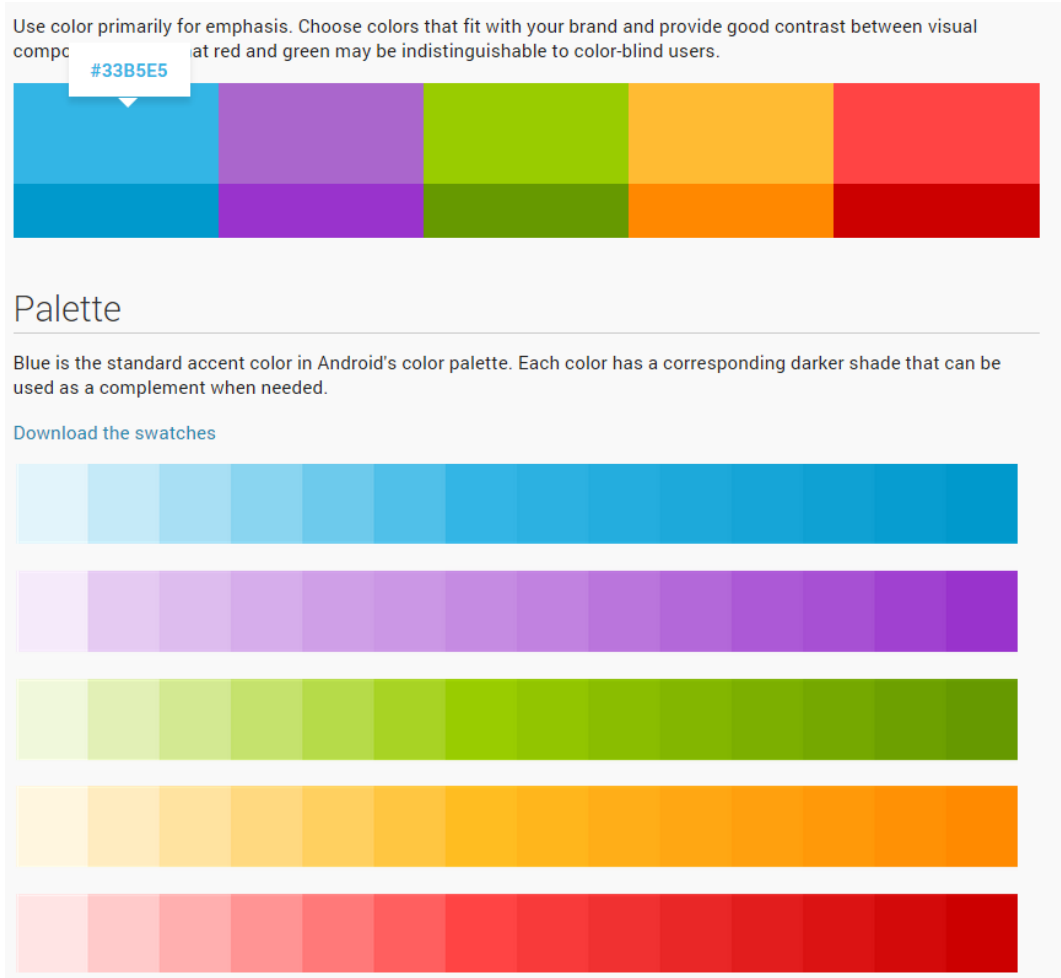
# Connexion

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



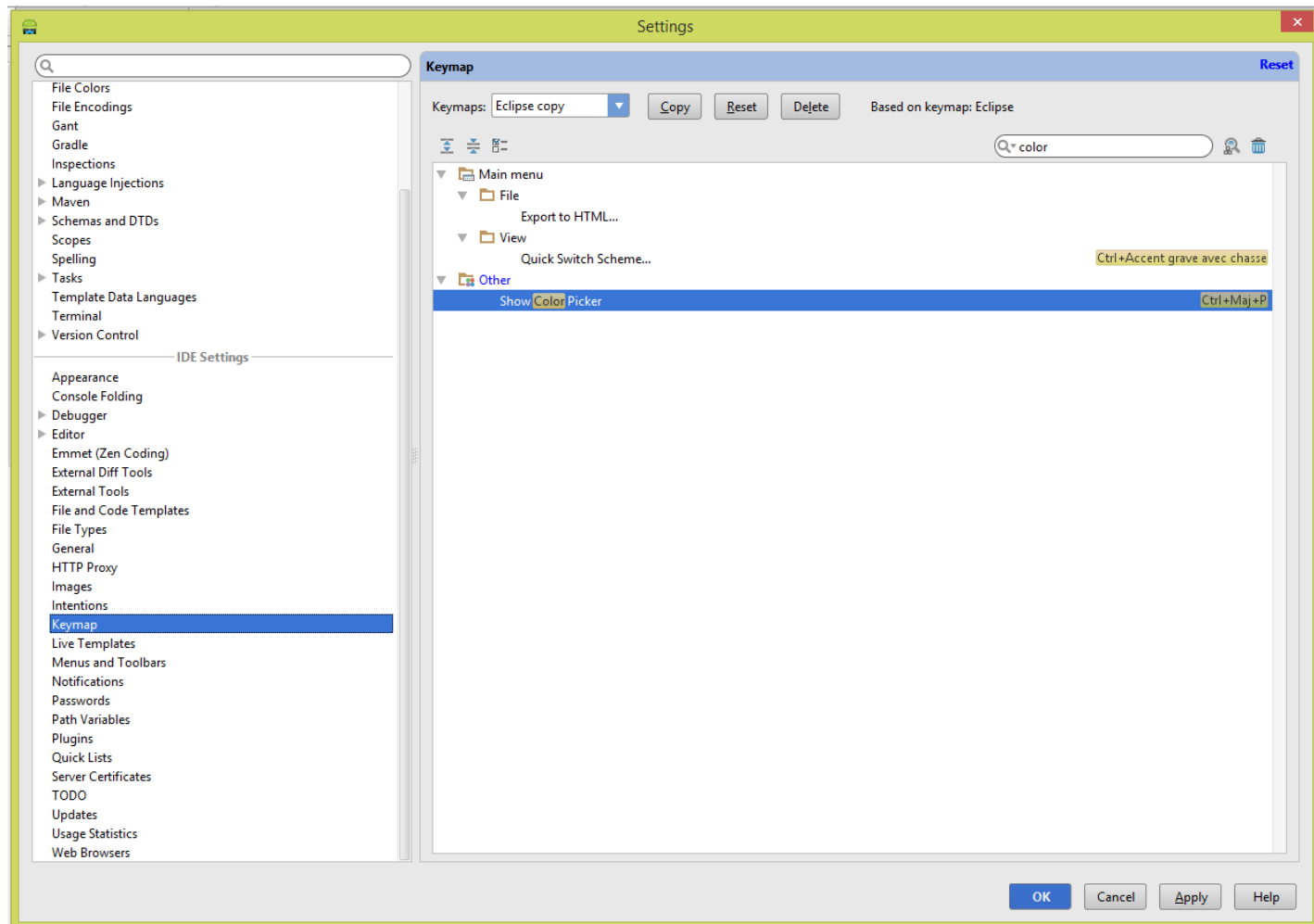
# Choix des couleurs

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



# COLORPICKER

## ○ Intégré à AndroidStudio



# 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="styleableButton">
        <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  
    LayoutInflater li = (LayoutInflater) 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.stylableButton);
```



# 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(final View view, final MotionEvent motionEvent) {  
    if (motionEvent.getAction() == MotionEvent.ACTION_DOWN) {  
        root.setBackground().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.setBackground().setColorFilter(backGroundColor, PorterDuff.Mode.MULTIPLY);  
        iv.setColorFilter(textColor, PorterDuff.Mode.MULTIPLY);  
        tv.setTextColor(textColor);  
    }  
    return false;  
}
```

# TP

## ○ 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

# TP

- 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

# TP

- 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

# INKSCAPE

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





249

# PERSISTENCE

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();
- Ecrire

```
SharedPreferences mPrefs = activity.getSharedPreferences("App_preference",  
MODE_PRIVATE);  
Editor editor = mPrefs.edit();  
editor.putBoolean("first_start", false);  
editor.apply(); //commit()
```

- Lire

```
return prefs.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"  
+ ");");
```

- Il y a certaines fonctionnalités non disponibles sur SQLite:
  - Les jointures externes
  - Les foreign key

# 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 : Création 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 : Création 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 créé la table à partir de la requête écrite 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);
        }
    }
}
```

# CRÉATION DES TABLES

- Nous utiliserons la méthode `db.execSQL(String)` afin de lancer une requête de création.
- `db.execSQL("CREATE TABLE constants ( id INTEGER PRIMARY KEY AUTOINCREMENT,title TEXT, value REAL);");`

# OUVRIR UNE CONNEXION

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

# Exemple : Création de la table

```
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)";

    private SQLiteDatabase bdd;
    private MaBaseSQLite maBaseSQLite;

    public EleveBDDManager(Context context) {
        //On créer la BDD et sa table
        maBaseSQLite = new MaBaseSQLite(context);
    }

    private void open() {
        bdd = maBaseSQLite.getWritableDatabase();
    }

    private void close() {
        bdd.close();
    }

    ...
}
```

# Exemple : Insert - Update

```
public void insertEleve(Eleve eleve) {
    open();
    //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
    }
    close();
}

public int updateEleve(Eleve eleve) {
    open();
    //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);
    close();
    return result;
}
```

# Exemple : Remove - Get

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

```
public List<Eleve> getAllEleves() {  
    open();  
    Cursor c = bdd.query(TABLE_ELEVE,  
        new String[] { COL_ID, COL_PRENOM, COL_NOM },  
        null, null, null, null, null);  
    List<Eleve> result = cursorToEleves(c);  
    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 : Cursor

```
//Cette méthode permet de convertir un cursor en list d'Eleve
private 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)), false);
                eleveListe.add(eleveBean);
            } while (c.moveToNext());
        }
    }

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

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

# TP : SQLITE

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

# 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.
  - FormationUtils
    - Utils.CopySQLiteBaseToDownload()

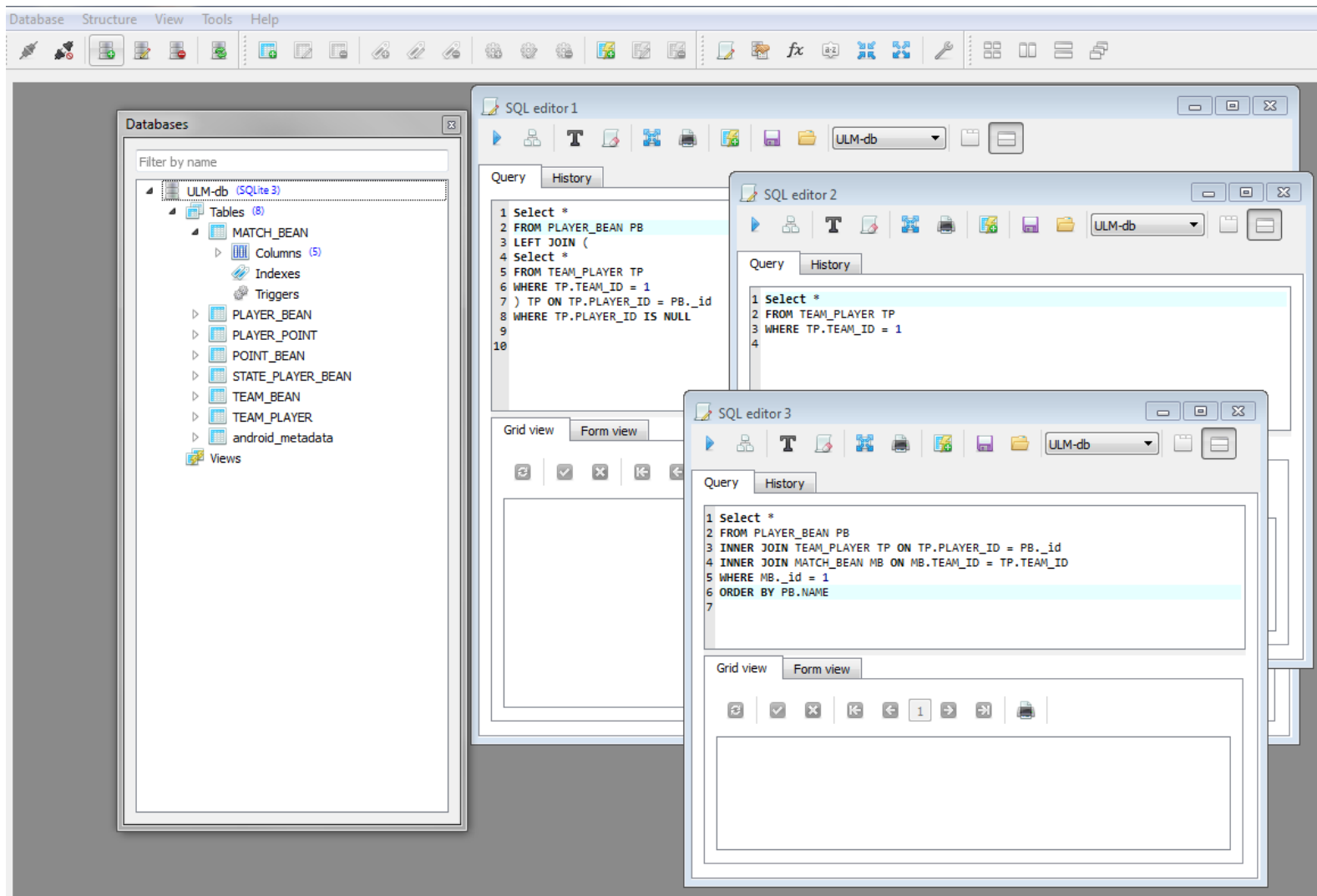
# VISUALISER SA BASE DE DONNÉES

- SQLiteStudio

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

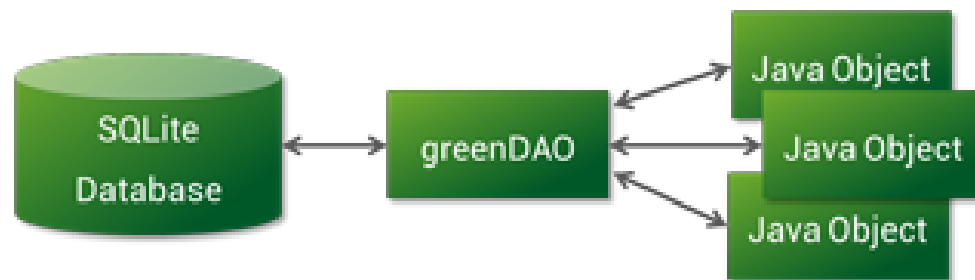
Il suffit d'importer le fichier .sqlite pour voir sa base de données, ses tables, leurs contenues et effectuer des requêtes dessus.

# VISUALISER SA BASE DE DONNÉES



# GREENDAO

- Un projet open source s'occupant du mapping (ORM).
- <http://greendao-orm.com/>



# GREENDAO DAOGENERATOR

- Exemple facile de mise en place
  - <https://github.com/SureCase/GreenDaoForAndroidStudio>
- Importer le module MyDaoGenerator
- Définir sa table dans MyDAOGenerator

```
public class MyDaoGenerator {  
  
    public static void main(String args[]) throws Exception {  
        Schema schema = new Schema(versionNumber, "javapackage");  
  
        //Table Eleve  
        Entity eleve = schema.addEntity("Eleve");  
        eleve.addIdProperty();  
        eleve.addStringProperty("Nom");  
        eleve.addStringProperty("Prenom");  
        new DaoGenerator().generateAll(schema, args[0]);  
    }  
}
```



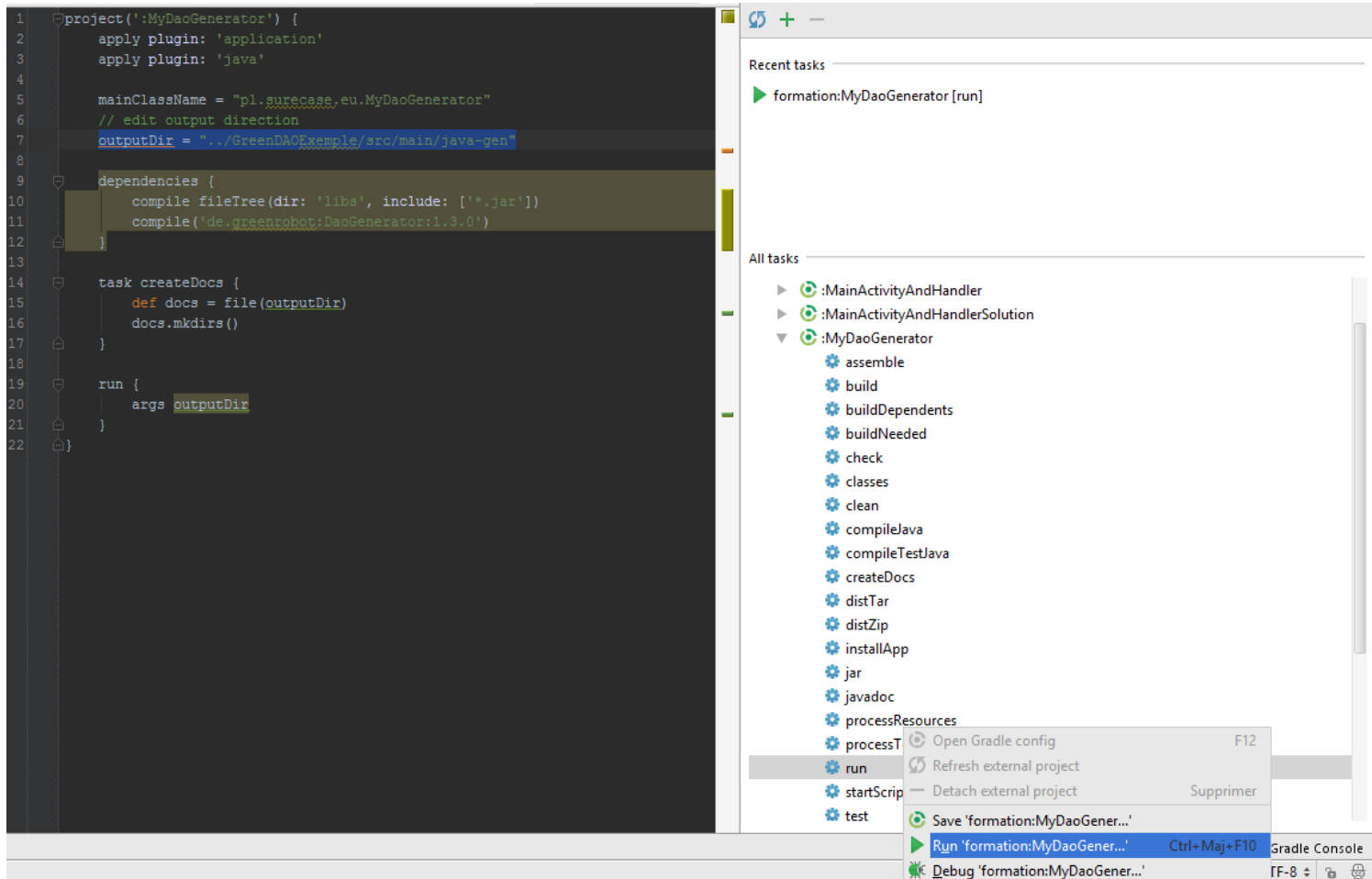
# GREENDAO DAOGENERATOR

- Modifier le répertoire des fichiers générées dans le gradle de MyDaoGenerator

```
task createDocs {  
    def docs = file("../DAOGreenSolution/src/main/java-gen")  
    docs.mkdirs()  
}  
  
run {  
    args "../DAOGreenSolution/src/main/java-gen"  
}
```

# GREENDAO DAOGENERATOR

- Lancer la génération dans la fenêtre de Gradle



# GREENDAO UTILISATION

- Les classes du mapping apparaissent dans java-gen

- Dans notre projet ajouter GreenDAO à Gradle

```
dependencies {  
    compile 'de.greenrobot:greendao:1.3.7'  
}
```

- Ajouter le répertoire générer aux sources compilées

```
android {  
    sourceSets {  
        main {  
            java.srcDirs = ['src/main/java', 'src/main/java-gen']  
        }  
    }  
}
```

# GREENDAO SETUP

```
public class MyApplication extends Application {  
    private DaoSession daoSession;  
    private static MyApplication instance;  
  
    public static MyApplication getInstance() {  
        return instance;  
    }  
    public void onCreate() {  
        instance = this;  
        setupDatabase();  
    }  
  
    private void setupDatabase() {  
        final DaoMaster.DevOpenHelper helper = new DaoMaster.DevOpenHelper(this, "mytable-  
db", null);  
        final SQLiteDatabase db = helper.getWritableDatabase();  
        final DaoMaster daoMaster = new DaoMaster(db);  
        daoSession = daoMaster.newSession();  
    }  
    public 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().delete(getEleveForId(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.getInstance().getDaoSession().getEleveDao();  
    }  
}
```

# GREENDAO UTILISATION

## ○ Le constructeur de requete.

```
/**
 * 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
- Module de départ : DAOExemple
- Solution : Module DAOGreenSolution

# GREENDAO : RELATIONNELLE

```
// Table Equipe
Entity teamBean = schema.addEntity("TeamBean");
teamBean.addIdProperty();

// Table Joueur
Entity playerBean = schema.addEntity("PlayerBean");
playerBean.addIdProperty();

// Table Equipe - Joueur
//On ne peut pas mettre des primaryKey sur plusieurs valeurs
Entity teamPlayer = schema.addEntity("TeamPlayer");
teamPlayer.addIdProperty();
//Relation : Team* teamPlayer
Property teamId = teamPlayer.addLongProperty("teamId").NotNull().getProperty();
teamBean.addToMany(teamPlayer, teamId);
//Relation : TeamPlayer 1 TeamBean
teamPlayer.addToOne(teamBean, teamId);
//Relation : Player * teamPlayer
Property playerId = teamPlayer.addLongProperty("playerId").NotNull().getProperty();
playerBean.addToMany(teamPlayer, playerId);
//Relation : TeamPlayer 1 Player
teamPlayer.addToOne(playerBean, playerId);
```



# GREENDAO : RELATIONNELLE

- Utilisation

```
List<TeamPlayer> list = teamBean.getTeamPlayerList();  
TeamPlayer tp = teamPlayer.getTeamBean()
```

- Chargement à la demande.
- Vider le cache

```
teamBean.resetTeamPlayerList();
```

# GREENDAO : REQUETE

```
/* Retourne la liste des joueurs non inscrits dans l'equipe*/
public static List<PlayerBean> getPlayerNotInTeam(long teamId) {
    //Sous requete contenant l'ensemble des joueurs de l'équipe
    String allTeamPlayer = "(Select * FROM " + TeamPlayerDao.TABLERNAME + " TP WHERE TP." +
TeamPlayerDao.Properties.TeamId.columnName + "=? )";

    //Jointure indiquant si les joueurs sont de cette equipe ou non
    String query = " LEFT JOIN " + allTeamPlayer + " TP ON TP."
        + TeamPlayerDao.Properties.PlayerId.columnName + " = T."
        + PlayerBeanDao.Properties.Id.columnName;

    //On ne prend que ceux qui ne le sont pas
    query += " WHERE TP." + TeamPlayerDao.Properties.PlayerId.columnName + " IS NULL";

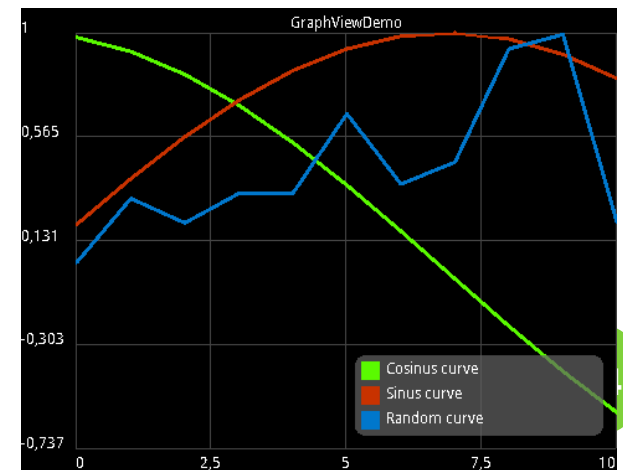
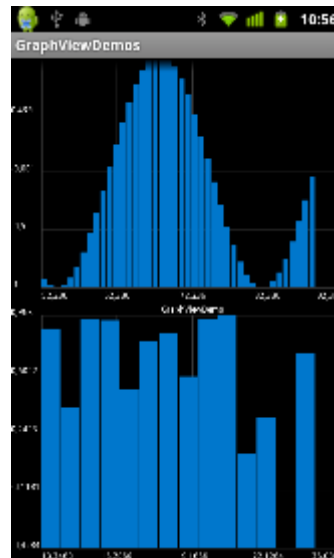
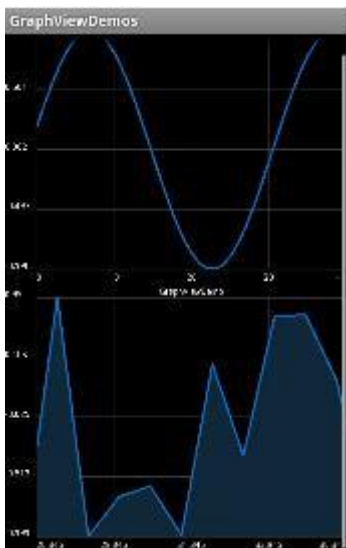
    try {
        return getPlayerDAO().queryRawCreate(query, teamId).list();
    }
    catch (Throwable e) {
        LogUtils.logException(PlayerBeanDao.class, e, true);
        return new ArrayList<>();
    }
}
```

# LIBRAIRIES ANDROID

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

# GRAPHVIEW

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



# 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
```

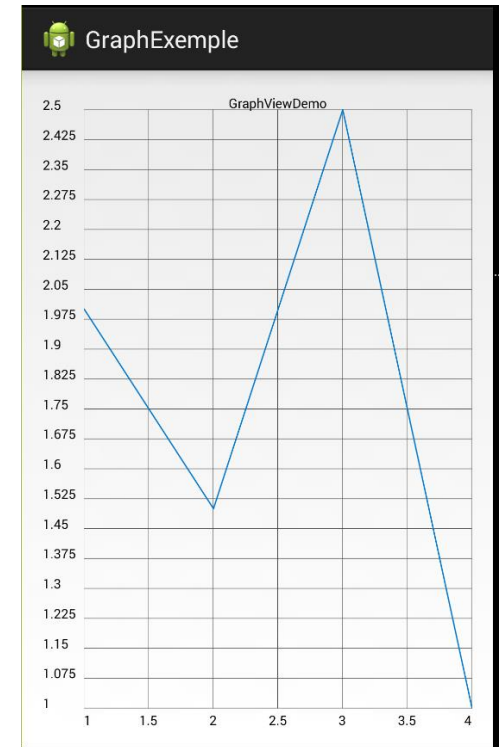
```
public double getY() {  
    return y;  
}
```

```
}
```

# 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(final String reason) {}  
        public void onException(final Throwable throwable) {}  
        public void onThinking() {}  
        public void onLogout() {}  
    });  
}
```



# UTILISATION

## ○ PostOnWall

```
final 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(final String reason) {}

    @Override
    public void onException(final Throwable throwable) {}

    @Override
    public void onThinking() {}

    @Override
    public void onComplete(final String postId) {}

});
```

# TP - FACEBOOK

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

# OAUTH AVEC SCRIBE

- Le github

- <https://github.com/fernandezpablo85/scribe-java>

- Le gradle

- Compile 'org.scribe:scribe:1.3.6'

```
repositories {  
    maven { url 'https://raw.github.com/fernandezpablo85/scribe-java/mvn-repo/' }  
}
```

# OAUTH AVEC SCRIBE

- Exemple avec Twitter, récupérer l'api\_key de Twitter

- <https://apps.twitter.com/app/new>

## Create an application

### Application Details

**Name \***

Your application name. This is used to attribute the source of a tweet and in user-facing authorization screens. 32 characters max.

**Description \***

Your application description, which will be shown in user-facing authorization screens. Between 10 and 200 characters max.

**Website \***

Your application's publicly accessible home page, where users can go to download, make use of, or find out more information about your application. This fully-qualified URL is used in the source attribution for tweets created by your application and will be shown in user-facing authorization screens.

(If you don't have a URL yet, just put a placeholder here but remember to change it later.)

**Callback URL**

Where should we return after successfully authenticating? OAuth 1.0a applications should explicitly specify their oauth\_callback URL on the request token step, regardless of the value given here. To restrict your application from using callbacks, leave this field blank.



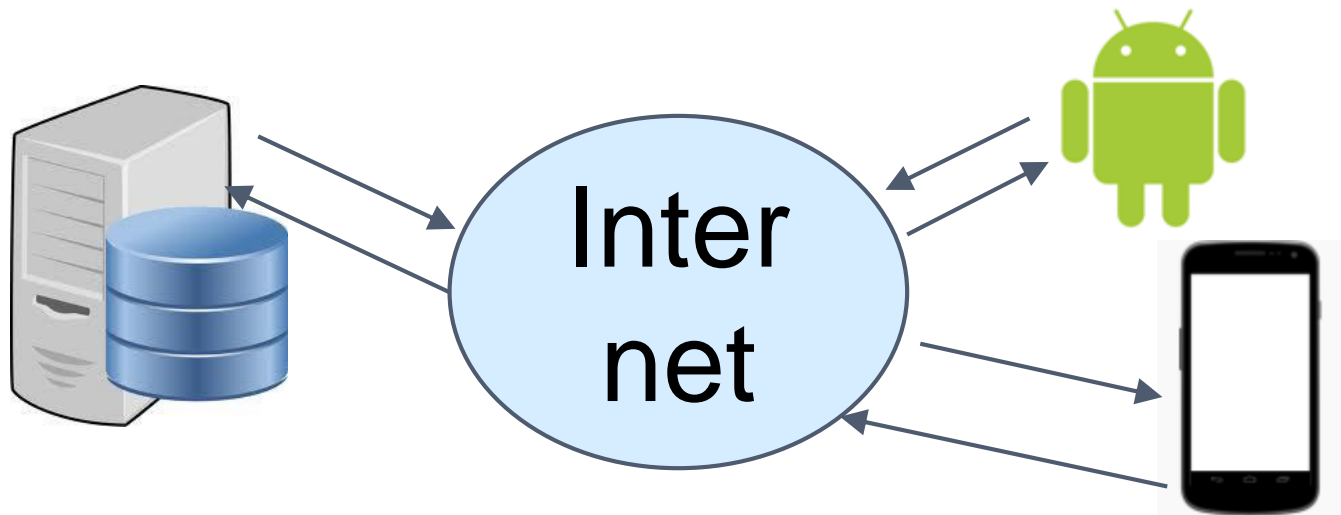
# NOTIFICATION

301

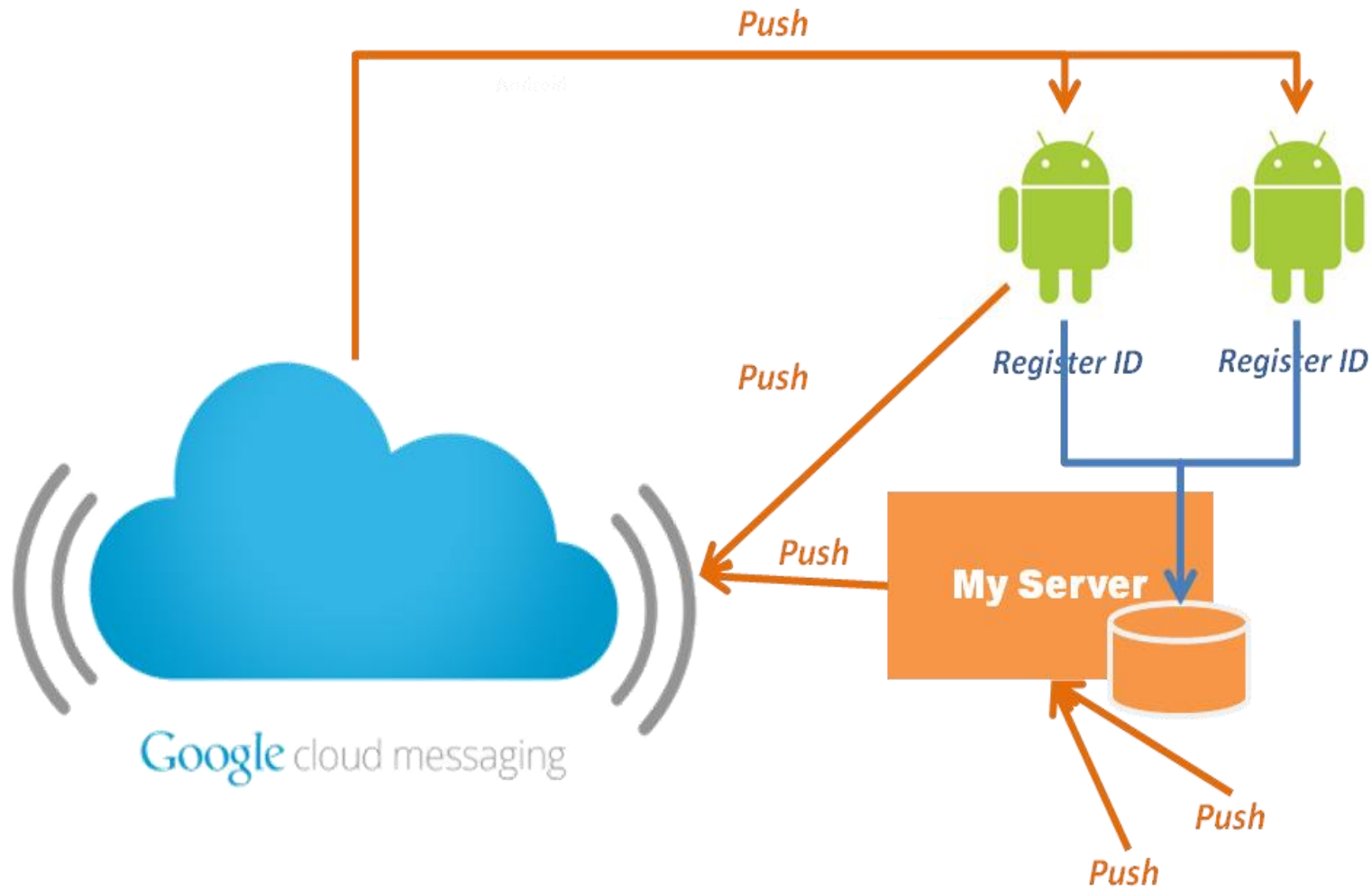
# Notifications

- Les notifications permettent d'alerter l'utilisateur et de garder un historique des dernières notifications dont l'utilisateur n'a pas encore pris connaissance.
- Les notifications sont gérées par un gestionnaire central de notifications. Les applications utilisent la barre de statut du système pour afficher les notifications.
- 2 types de notifications
  - Google cloud messaging
  - NotificationManager

# Google cloud messaging



# Google cloud messaging





# NotificationManager

- Les notifications font partie des services proposés par Android. Nous récupérons ce service en utilisant la méthode `getSystemService` avec le paramètre `Context.NOTIFICATION_SERVICE`.
- La création d'une notification se fait tout d'abord en créant une instance de type `Notification`.
- Une notification possède trois caractéristiques
  - une icône
  - un texte défilant
  - une heure (qui déterminera quand la notification sera affichée)

# 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, NOTIFICATION_REQUEST_CODE,  
    new Intent(this, MainActivity.class), PendingIntent.FLAG_ONE_SHOT);
```

# Notifications

<http://developer.android.com/reference/android/app/PendingIntent.html>

```
public class NotificationHelper {  
    private static final int NOTIFICATION_REQUEST_CODE = 13; //au hasard pour l'activity  
    private static final int NOTIFICATION_ID = 14; //au hasard pour retrouver la notif  
  
    //Uniquemeent pour JellyBean et +  
    public static void createNotification(final Context context) {  
        //COMMENT ON L'ENVOIE  
        NotificationManager mNotification = (NotificationManager)  
            context.getSystemService(Context.NOTIFICATION_SERVICE);  
  
        Intent launchNotificationIntent = new Intent(context, MainActivity.class);  
        PendingIntent pendingIntent = PendingIntent.getActivity(context, NOTIFICATION_REQUEST_CODE,  
            launchNotificationIntent, PendingIntent.FLAG_ONE_SHOT);  
  
        //CE QU'ON ENVOIE  
        final Notification.Builder builder = new Notification.Builder(context)  
            .setWhen(System.currentTimeMillis()).setTicker("Ticker").setSmallIcon(R.drawable.ic_launcher)  
            .setContentTitle("ContentTitle").setContentText("ContentText")  
            .setContentIntent(pendingIntent);  
  
        //ENVOIE  
        mNotification.notify(NOTIFICATION_ID, builder.build())  
    }  
}
```

# TP : notifications

- Reprendre l'exercice sur le listener de sms et y ajouter une notification en plus du Toast.
- Un clic sur la notification relance l'application.
- Solution : projet SMSBroadcast



309

# WEB

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();  
}
```

# CONNEXION HTTP

## ○ Préparer la requête

```
URL url = new URL(myurl);  
URLConnection conn = (URLConnection) url.openConnection();  
conn.setReadTimeout(10000 /* milliseconds */);  
conn.setConnectTimeout(15000 /* milliseconds */);  
conn.setRequestMethod("GET");
```

## ○ Lancer la requête (Nouveau thread)

```
conn.connect();
```

## ○ Lire la réponse

```
int response = conn.getResponseCode();  
InputStream is = conn.getInputStream();  
// Convert the InputStream into a string  
return readIt(is);
```

## ○ Fermer la requête.

```
if (is != null) {  
    is.close();  
    conn.disconnect();  
}
```



# CONNEXION HTTP

## ○ OkHttp

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

```
OkHttpClient client = new OkHttpClient();

public String run(String url) throws IOException {
    Request request = new Request.Builder().url(url).build();

    Response response = client.newCall(request).execute();

    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  
mymyWebView.loadUrl("file:///android_asset/index.html");  
//charger une page  
myWebView.loadData("<html><body>Bonjour !</body></html>« , "text/html", "UTF-8");
```

## ○ Activer le Javascript

- Attention aux performances!!!

```
WebSettings webviewSettings = myWebView.getSettings();  
webviewSettings.setJavaScriptEnabled(true);
```

# WEBVIEWCLIENT

## ○ Interagir avec les pages grâce au WebViewClient

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

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

# TP

- Créer un projet chargeant une page avec une webView et avec HttpURLConnection
  - Solution TP : httpExemple

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

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

# 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'



# GSON FONCTIONNEMENT

- Créer les beans de réception
- Passer d'un InputStream à un InputStreamReader

```
new InputStreamReader(inputStream) ;
```

- Parser le flux

```
//Création de l'objet  
  
private Gson = new Gson() ;  
  
//parsing du flux ou du String contenant du JSON  
ResultBean result = gson.fromJson((String | InputStreamReader), ResultBean.class);
```

- Traiter le résultat

# GSON BEAN

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

```
public class ResultBean {  
  
    private CityBean[] results;  
    private int nbr;  
    private ErrorBean errors;  
  
}
```

# CLIENT SERVEUR : BONNES PRATIQUES

- Créer une librairie de Bean commune sur un repository Maven
  - Rien à faire coté 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

# TP

- 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



# GOOGLE MAPS

# 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
- Mettre en place
  - <https://console.developers.google.com>

# GOOGLE MAPS : RÉFÉRENCEMENT

< Projets

## FormationMaps

Présentation

Autorisations

Facturation et paramètres

## API et authentification

API

Identifiants

Écran d'autorisation

Push

## Surveillance

## Code source

## Compute

## Réseau

## Stockage

## Big Data

Assistance

Besoin d'aide ?

Règles de confidentialité et ...

Bienvenue dans la nouvelle version d

## OAuth

OAuth 2.0 permet aux utilisateurs de partager des données spécifiques avec vous (par exemple, des listes de contacts) tout en préservant la confidentialité de leur nom d'utilisateur, de leur mot de passe et d'autres informations.

[En savoir plus](#)

Créer un identifiant client

## Accès à l'API publique

L'utilisation de cette clé ne nécessite aucune action de l'utilisateur ni son consentement, ne donne pas accès aux informations du compte ni ne sert à des fins d'autorisation.

[En savoir plus](#)

Créer une clé

# GOOGLE MAPS : RÉFÉRENCEMENT

- Générer une clé dans une console
  - `C:\Program Files\Java\jdk1.8.0_20\bin\keytool -list -v -keystore mystore.keystore`
- Si aucun fichier keystore
  - `C:\Program Files\Java\jdk1.8.0_20\bin\keytool -list -alias androiddebugkey -keystore C:\Users\Anthony\.android\debug.keystore -storepass android -keypass android`
- Ajouter le package de l'application à la clé reçu



# GOOGLE MAPS : RÉFÉRENCEMENT

## Modifier les applications Android autorisées

**Cette clé peut être déployée dans votre application Android.**

Les requêtes API sont envoyées directement à Google à partir des appareils Android de vos clients. Google vérifie que chaque requête provient d'une application Android qui correspond à l'une des empreintes de certificat SHA1 et à l'un des noms de package énumérés ci-dessous. Vous pouvez obtenir l'empreinte SHA1 de votre certificat développeur à l'aide de la commande suivante :

```
keytool -list -v -keystore mystore.keystore
```

[En savoir plus](#)

### ACCEPTER LES REQUÊTES PROVENANT D'UNE APPLICATION ANDROID AVEC L'UNE DES EMPREINTES DE CERTIFICAT ET L'UN DES NOMS DE PACKAGE ÉNUMÉRÉS CI-DESSOUS

Une empreinte de certificat SHA1 et un nom de package par ligne (séparés par un point-virgule). Exemple :  
45:B5:E4:6F:36:AD:0A:98:94:B4:02:66:2B:12:17:F2:56:26:A0:E0;com.example

```
FC:A8:F5:89:78:F5:FF:92:A0:A5:57:F9:D7:94:E3:17:2A:50:7E:37;com.formation.googlemap
```

Mettre à jour

Annuler

# GOOGLE MAPS

## OAuth

OAuth 2.0 permet aux utilisateurs de partager des données spécifiques avec vous (par exemple, des listes de contacts) tout en préservant la confidentialité de leur nom d'utilisateur, de leur mot de passe et d'autres informations.

[En savoir plus](#)

Créer un identifiant client

## Accès à l'API publique

L'utilisation de cette clé ne nécessite aucune action de l'utilisateur ni son consentement, ne donne pas accès aux informations du compte ni ne sert à des fins d'autorisation.

[En savoir plus](#)

Créer une clé

## Clé pour les applications Android

CLÉ DE L'API	AlzaSyCyklSb3raiiyVz12q-ZtPT6yVC8igsLNA
APPLICATIONS ANDROID	FC:A8:F5:89:78:F5:FF:92:A0:A5:57:F9:D7:94:E3:17:2A:50:7E:37;com.formation.googlemap
DATE D'ACTIVATION	30 nov. 2014 08:22:00
ACTIVÉE PAR	anth06ny@gmail.com (vous)

Modifier les applications Android autorisées

Regénérer la clé

Supprimer

# GOOGLE MAPS : RÉFÉRENCEMENT

```
<string name="google_maps_key" templateMergeStrategy="preserve">  
    AIzaSyB9oARLFPe4wRGyeuWJq0IqZ0g84TjmjVI  
</string>  
<integer name="google_play_services_version">6171000</integer>
```

# GOOGLE MAPS : DÉCLARATION

```
<application>
    <meta-data
        android:name="com.google.android.gms.version"
        android:value="@integer/google_play_services_version" />

    <meta-data
        android:name="com.google.android.maps.v2.API_KEY"
        android:value="@string/google_maps_key" />
</application>

<uses-permission android:name="android.permission.INTERNET" />
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
<uses-permission android:name="com.google.android.providers.gsf.permission.READ_GSERVICES" />

<!--
    The ACCESS_COARSE/FINE_LOCATION permissions are not required to use
    Google Maps Android API v2, but are recommended.
-->

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

# GOOGLE MAPS : UTILISATION

```
<fragment
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:id="@+id/map"
    android:name="com.google.android.gms.maps.SupportMapFragment"/>
```

# GOOGLE MAPS : UTILISATION

```
public class MapsActivity extends FragmentActivity {  
    private GoogleMap mMap; // Might be null if Google Play services APK is not available.  
  
    protected void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        setContentView(R.layout.activity_maps);  
        setUpMapIfNeeded();  
    }  
  
    protected void onResume() {  
        super.onResume();  
        setUpMapIfNeeded();  
    }  
  
    private void setUpMapIfNeeded() {  
        // Do a null check to confirm that we have not already instantiated the map.  
        if (mMap == null) {  
            // Try to obtain the map from the SupportMapFragment.  
            mMap = ((SupportMapFragment) getSupportFragmentManager().findFragmentById(R.id.map)).getMap();  
        }  
    }  
}
```

# GOOGLE MAPS : MARKER

## ○ Afficher un marker

```
if (mMap != null) {  
    mMap.addMarker(new MarkerOptions().position(new LatLng(45.5837, 0.094452)).title("Angoulême"));  
}
```

## ○ Afficher sa position

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

# GOOGLE MAPS

## ○ Quelques possibilités

- Dessiner
- Zoom / Dézoom
- Déplacer la Map
- Intercepter un clic sur un marker
- Connaitre la position d'un élément de la carte sur l'écran
- Ajouter un tracer (Polyline)