Disclaimer

Portions of this presentation are modifications based on work created and shared by :

- The Android Open Source Project
  > http://code.google.com/policies.html
- The Java Passion
  > http://www.javapasion.com
- Reto Meire auteur of the book “Android 2”
- They are used according to terms described in the Creative Commons 2.5 Attribution License

> http://creativecommons.org/licenses/by/2.5/

Android : Développement des applications mobiles

Max Bonbhel

Senior Software Consultant

Java / Android Trainner

Pourquoi faire ?

AGENDA

Agenda

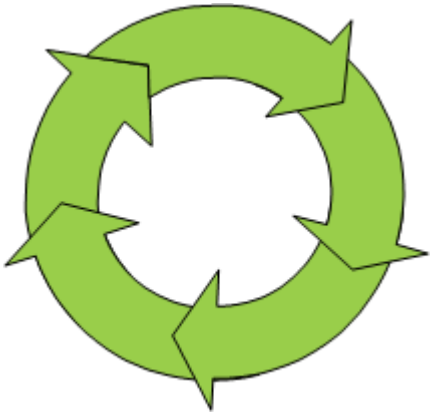
- Où trouver Android ?
- Qui utilise Android
- UI Layout, Menu, Dialog and Notification
- Activity
- Resources and Assets
- Data storage
- Content Providers
- Multimedia (Audio and Video)
- Services
- Location and Maps
- Web services, XML, Cloud-computing
- Security and Permissions

OÙ TROUVER ANDROID ?

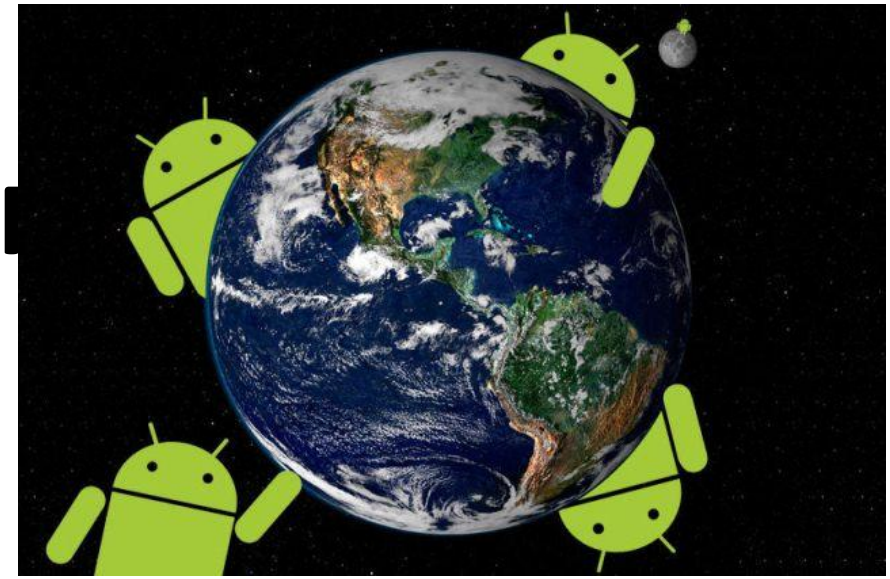
It’s everywhere !

Où trouver Android ?

- Téléphones mobiles
- NetBook
- RIM BlackBerry PlayBook ta
- Tablette PC
- Des voitures
- Smart TV
- Bref…Presque

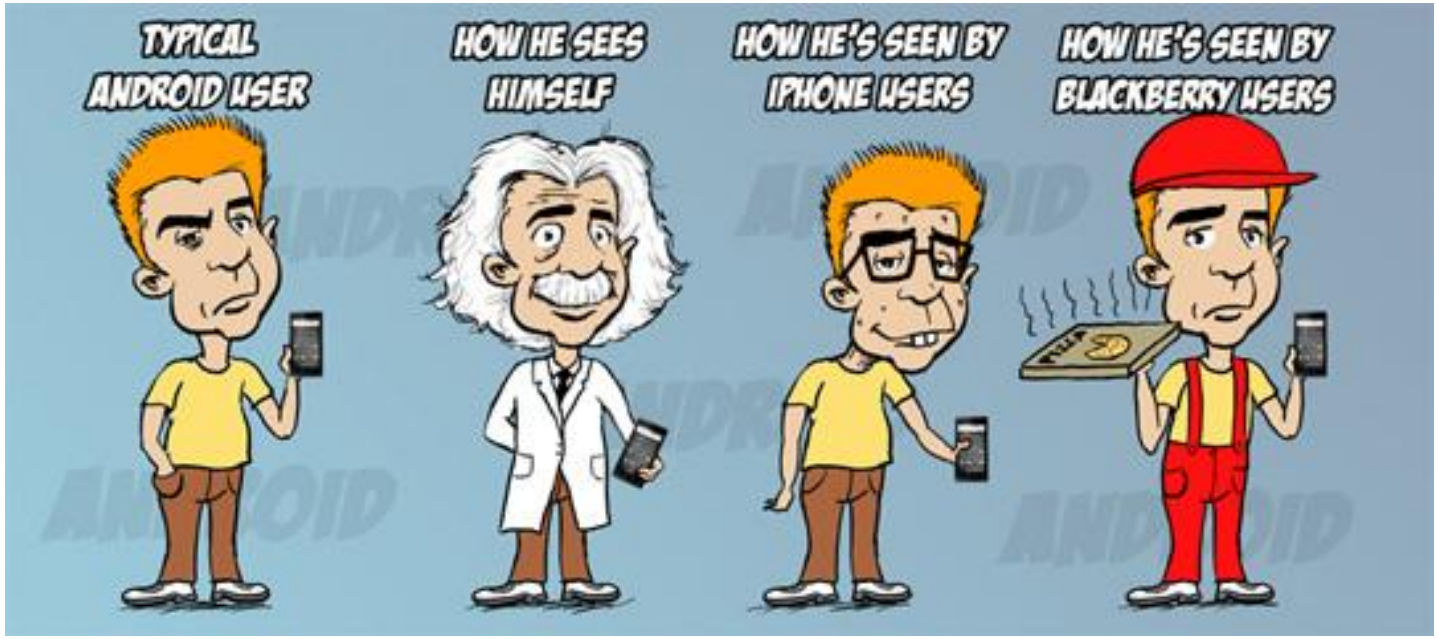


QUI UTILISE ANDR



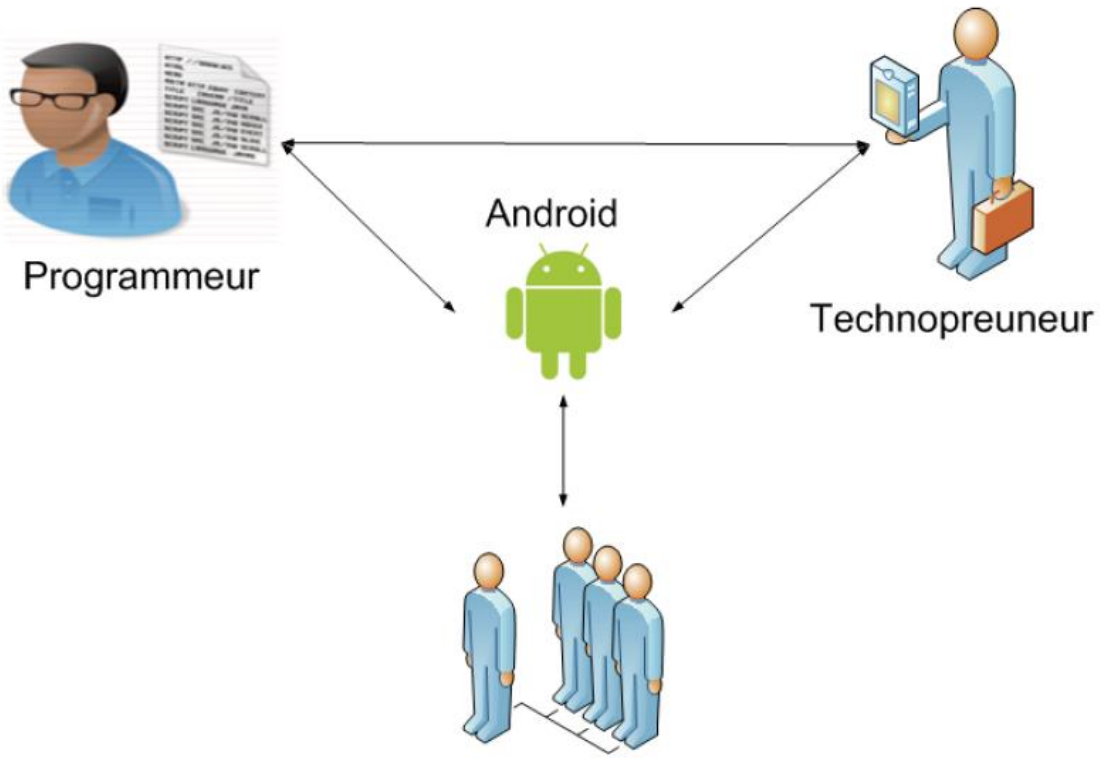
Everyone !

Qui utilise Android



Où trouver Android ?

- Programmeurs
- Tech
- Mon



CE QUE ANDROID N'EST PAS

Android vs IPhone

Deux plateformes 2 concepts

Ce que Android n'est pas

- Un outil pour construire des robots humanoïdes (…du moins pas pour l'instant)
- Une implémentation de Java ME
  - Les applications Android ne sont pas exécutées sur la machine virtuelle Java ME
- Un élément de Linux Phone Standard Forum (LiPS) ou d'Open Mobile Alliance (OMA)
  - Bien qu'exécuté par un noyau Lunix open source, Android va au-delà des préoccupations de standardisation de LiPS et OMA
- Une simple couche applicative (tel qu'UIQ ou S60)
  - Android = l'OS sous-jacent + les APIs + Applications(natives et tierces)
- Un téléphone mobile
  - Plateforme de référence pour les fabricants. Pas **UN** téléphone Android mais **DES** téléphones Android
- Une réponse à l'IPhone (..hum. Pas sûr!)
  - IPhone = Un fabricant , Un OS (fermé), Un matériel

CE QU'EST VRAIMENT ANDROID

Une plateforme ouverte pour le développement sur mobiles

Ce qu'est vraiment Android

- Un système d'exploitation open-source Linux assurant l'interface de bas-niveau avec le matériel :
  - Gestion de la mémoire
  - Contrôle des processus
- Des bibliothèques open-source pour le développement d'applications
  - SQLite, WebKit, OpenGL
  - Un gestionnaire des médias
- Un moteur d'exécution des applications Android incluant :
  - La VM Dalvik
  - Les bibliothèques de base
- Un Framework applicatif qui expose les services du système
  - Window Manager (gestionnaire des fenêtres)
  - Location Manager(gestionnaire de géolocalisation)
  - Content Provider (gestionnaire de contenu)

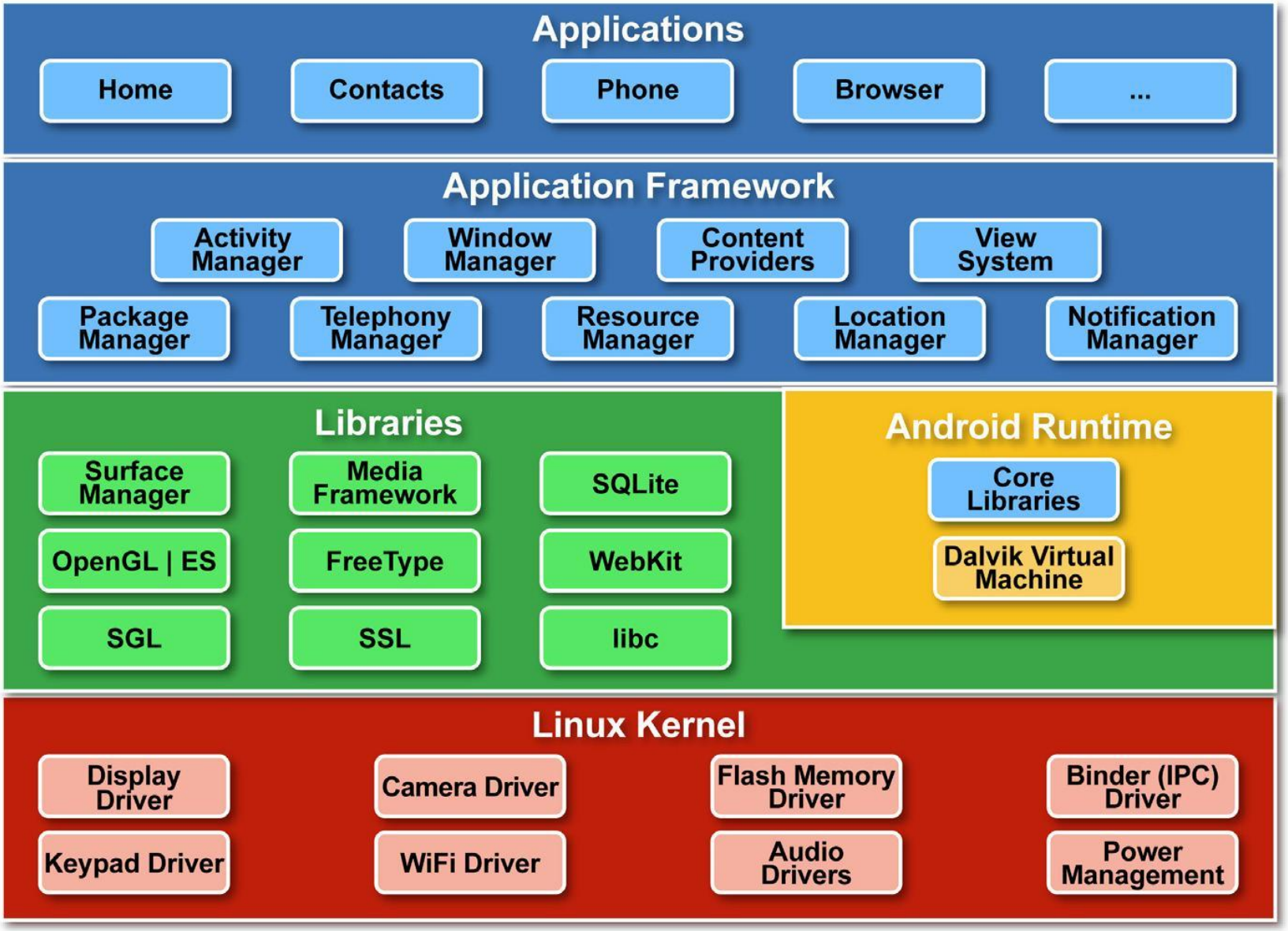
SDK ANDROID

La boîte à outils : Possibilité de créer des applications qui seront intégrées au téléphone exactement comme les applications natives d'Android

VM Dalvik n'est pas une JVM comme Java ME

Possibilité d'écrire les applications en C#C++ pour Dalvik

Le SDK Android



Le SDK Android

- Aucun frais de licence (développement, distribution)
- Pas de processus d'approbation des applications tierces
- Accès directe aux composants matériels
  - Wi-Fi, Réseaux (GSM, EDGE, et 3G)
  - SMS
- APIs complètes pour les services de géolocalisation comme le GPS
- Contrôle complet du matériel multimédia
  - Lecture et enregistrement vidéo et audio
- Contrôle de capteurs
  - Accéléromètre
  - Boussole
- Moteur graphique
  - Graphique 2D
  - Graphique 3D avec OpenGL ES 2.0

DÉVELOPPEURS : PAR OÙ COMMENCER ?

Sur toutes les plateforme

Par où commencer

- OS supportés
  - Microsoft Wondows (XP ou supérieur)
  - Mac OS X 10.4.8 ou supérieur (sur processeur Intel uniquement)
  - Linux
- Environnement de développement
  - SDK Android (http://developer.android.com/sdk/index.html)
  - JDK5 ou 6 (http://sun.java.com/javase/downloads/index.jsp)
  - Eclipse 3.4 ou 3.5 (de préférence Galileo)
  - Le plugun Eclipse ADT (Android Developer Tool)
- Le plugin ADT n'est pas indispensable mais facilite grandement le développement

Déclarer un Layout

Structure du fichier Layout

Types de Layout

INTERFACE UTILISATEUR (LAYOU¯`

Déclarer un Layout

- Deux façons de déclarer un Layout
  - Option #1: Déclarer les éléments de l'interface utilisateur dans un fichier XML (facile à structurer, à visualiser etc.) . Très ***Recommandé***
    - Android fourni un vocabulaire assez simplifié des éléments XML correspondant à des classes et sous-classes de type View.
  - Option #2: Instancier les éléments du Layout à l'exécution (dans le code Java)
    - L'application peut alors générer des objets de type View (Vue) ou ViewGroup (Groupe de vue) et manipuler ainsi ses propriétés par programmation dans le code Java.
- Il est aussi possible d'utiliser les deux options en même temps
    - Utiliser le mode déclaration pour les éléments d'interface utilisateur par défaut et puis utiliser du code Java pour modifier les propriétés à l'exécution .

Structure du fichier Layout

- Chaque fichier ne peut contenir qu'un seul élément racine (root) qui doit être un object de type View (Ex : un Button) ou de type ViewGroup (Ex : un LinearLayout).
- Une fois l'élément racine (root) défini, il est possible d'ajouter d'autres éléments enfants pour construire graduellement son Layout.

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
xmlns:android="http://schemas.android.com/apk/res/android"
   android:layout_width="fill_parent"
   android:layout_height="fill_parent"
   android:orientation="vertical" >
<TextView android:id="@+id/text"
          android:layout_width="wrap_content"
          android:layout_height="wrap_content"
          android:text="Hello, I am a TextView" />
   <Button android:id="@+id/button"
          android:layout_width="wrap_content"
          android:layout_height="wrap_content"
          android:text="Hello, I am a Button" />
</LinearLayout>
```

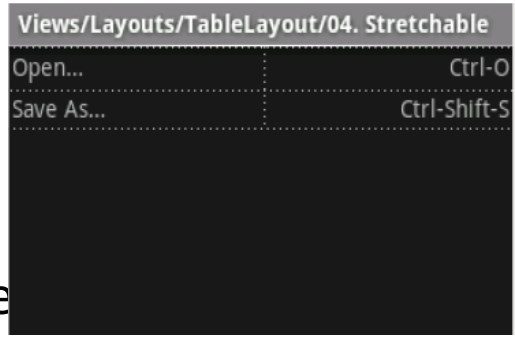
Structure du fichier Layout

- Chaque fichier Layout sera compilé dans un objet View ressource.
- La ressource générée est accessible dans le code de l'application via *Activity.onCreate()*

```
public void onCreate(Bundle savedInstanceState)
{
    super.onCreate(savedInstanceState);
    setContentView(R.layout.main_layout);
}
```

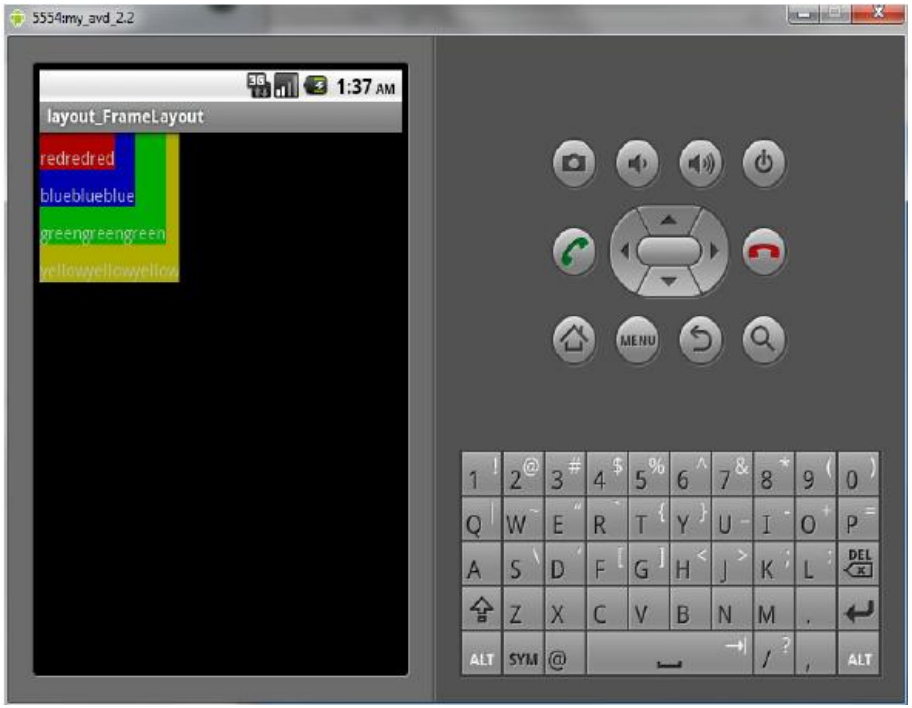
Type de Layout

- Tous les types de Layout sont des sous-classes de ViewGroup class
  - LinearLayout
    - Aligne ou empile tous les éléments enfants dans une seule direction (horizontale ou verticale).
    - La distance entre les éléments est respectée ainsi que la gravité (alignement à droite, au centre ou à gauche)
  - RelativeLayout
    - Chaque élément enfant peut avoir sa propre position relativement au parent ou aux autres éléments enfants.
    - Si par exemple le premier élément enfant est centré, alors, tous les autres éléments dont les positions n'ont pas été spécifiées seront alignés au centre aussi.
  - TableLayout
    - Les éléments enfants sont positionnés dans des lignes (TableRow) et des colonnes
    - Chaque ligne peut avoir zéro ou plusieurs cellule



– FrameLayout Type de Layout

- FrameLayout le plus simple des objets de type Layout. C'est simplement un espace vide sur l'écran où l'on peut remplir avec un **seul** objet. Ex : Une image de fond.
- Tous les éléments enfants seront automatiquement positionnés en haut à gauche de l'écran.



INTERFACE UTILISATEUR (MENU)

Types de menus

Menu Context

Menu Options

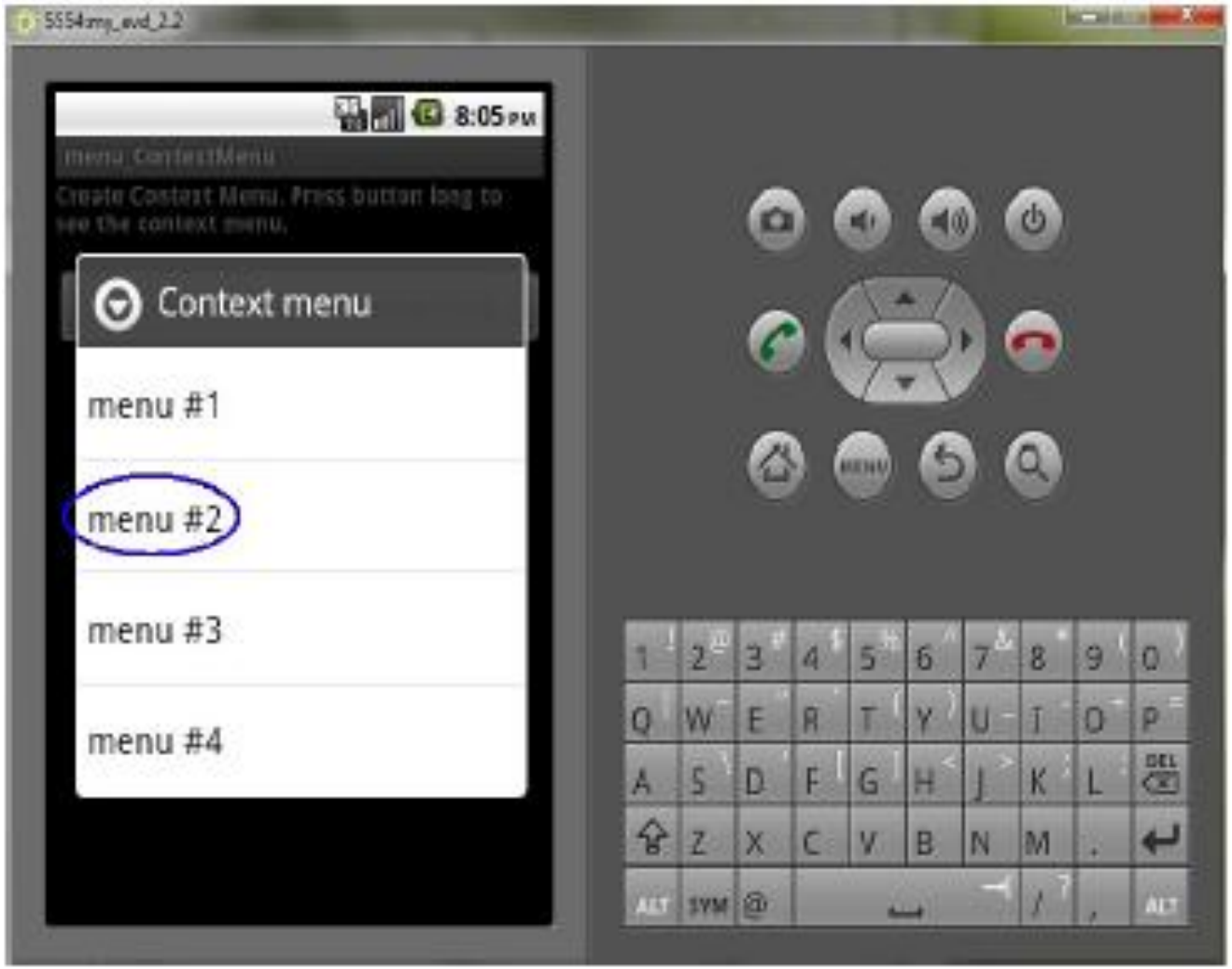
Sous Menu

Menu (Context Menu)

- Types de menu
  - Context Menu :
    - Liste flottante des items de menu qui apparait lorsqu'on appuie longtemps sur une View
    - Ne supporte pas des icones (juste du texte).
    - Les items peuvent être ajoutés via : le code Java *add();* ou via le .XML (recommandé).

```
// Override this method of Activity class in order to create menu items.
@Override
public void onCreateContextMenu(
      ContextMenu menu, // le Context menu à constuire
      View view, // La View pour laquelle le Context menu est construit
      ContextMenuInfo menuInfo) {
      super.onCreateContextMenu(menu, view, menuInfo);
      menu.setHeaderTitle("Context menu");
      menu.add(0, Menu.FIRST , Menu.NONE, "menu #1");
      menu.add(0, Menu.FIRST + 1, Menu.NONE, "menu #2");
      menu.add(0, Menu.FIRST + 2, Menu.NONE, "menu #3");
      menu.add(0, Menu.FIRST + 3, Menu.NONE, "menu #4");
}
```

Menu (Context)



Menu (Options)

- Options Menu :
  - S'affiche lorsque qu'on appuie sur le bouton «Menu» de l'appareil.
  - Options Menu est généralement l'endroit indiqué pour inclure les fonctionnalités basiques ainsi que les principaux items de navigation.
  - Supporte bien les icones.
  - Seuls les six (6) premiers items sont affichés par défaut. Il faudrait cliquer sur «More» pour voir apparaitre le reste.

```
/* Creates the menu items without Icons */
public boolean onCreateOptionsMenu(Menu menu) {
    // The add() method used in this sample takes four
arguments:
    // groupId, itemId, order, and title.
    menu.add(0, MENU_NEW_GAME, 0, "New Game");
    menu.add(0, MENU_QUIT, 0, "Quit");
    return true;
}
```

Menu (Sous Menu)

- SubMenu :
  - Permet d'organiser les items par groupes.
  - Ne supporte pas les sous-menus imbriqués

```
public boolean onCreateOptionsMenu(Menu menu) {
     boolean result = super.onCreateOptionsMenu(menu);
     // Create submenu "File"
     SubMenu fileMenu = menu.addSubMenu("File");
     fileMenu.add("New");
     fileMenu.add("Open File");
     fileMenu.add("Close");
     fileMenu.add("Close All");
     // Create submenu "Edit"
     SubMenu editMenu = menu.addSubMenu("Edit");
     editMenu.add("Undo Typing");
     editMenu.add("Redo");
     editMenu.add("Cut");
     return result;
}
```

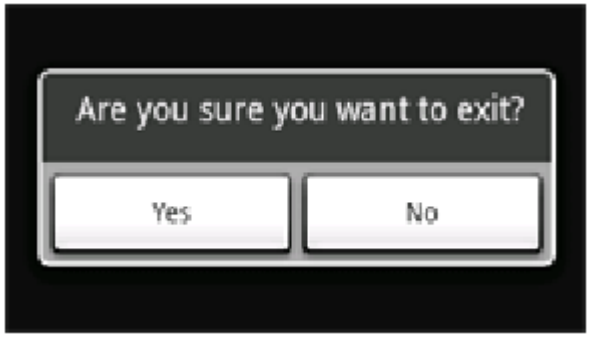
AlertDialog

ProgressDialog

INTERFACE UTILISATEUR (DIALOG)

Dialog (AlertDialog)

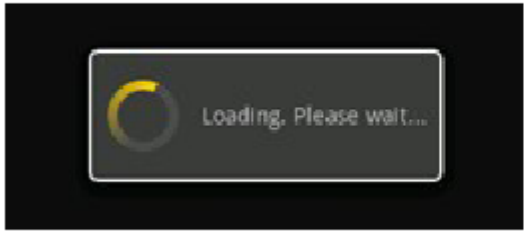
- Une *AlertDialog* est une extension de la classe *Dialog*.
- Souvent constituée d'un :
  - Titre
  - Message
  - 1, 2 ou 3 boutons

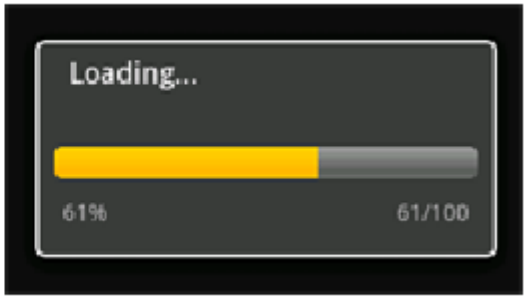



```
AlertDialog.Builder builder = new AlertDialog.Builder(this);
builder.setMessage("Are you sure you want to exit?").setCancelable(false)
.setPositiveButton("Yes", new DialogInterface.OnClickListener() {
      public void onClick(DialogInterface dialog, int id) {
                MyActivity.this.finish();
      }
})
.setNegativeButton("No", new DialogInterface.OnClickListener() {
      public void onClick(DialogInterface dialog, int id) {
                dialog.cancel();
      }
});
AlertDialog alert = builder.create();
```

Dialog (ProgressDialog)

- Un *ProgressDialog* est une extension de la classe *AlertDialog*.
- On peut y associer des boutons. Comme par exemple un bouton *Annuler* pour interrompre le chargement.






```
ProgressDialog progressDialog;
progressDialog = new ProgressDialog(mContext);
progressDialog.setProgressStyle(

    ProgressDialog.STYLE_HORIZONTAL);
progressDialog.setMessage("Loading...");
progressDialog.setCancelable(false);
```

Toast notification

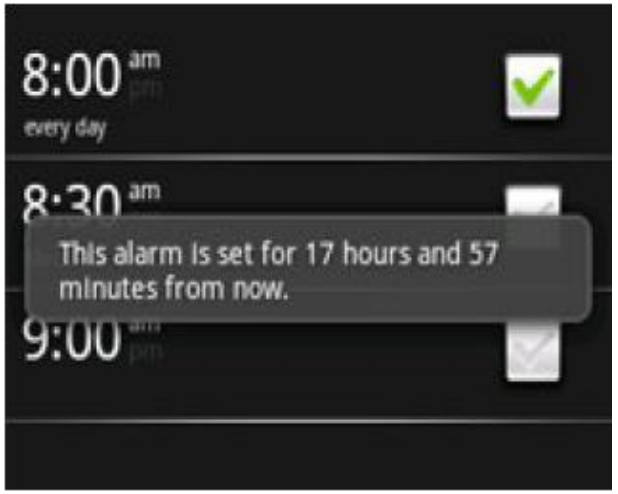
Status bar notification

Alarm manager

INTERFACE UTILISATEUR (NOTIFICATION)

Notification (Toast notification)

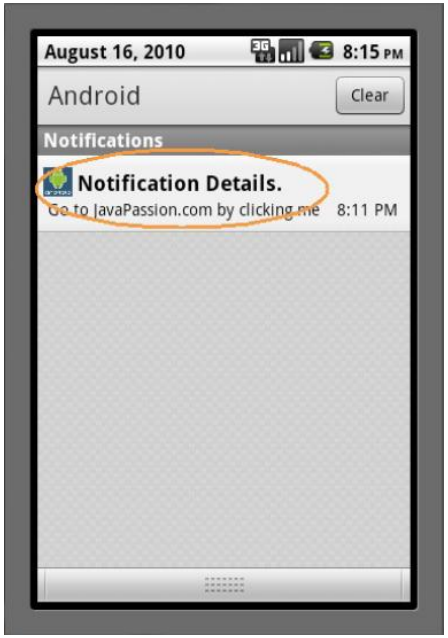
- Une *Toast notification* est un message qui s'affiche directement en premier plan à l'écran. Il est comparable à un *mini pop-up*
- Ne requière aucune action de l'utilisateur
- Fini par s'estamper progressivement puis disparaître




```
Toast.makeText(
getApplicationContext(), // Context
R.string.toast_message, //Obtenir la ressource à afficher
Toast.LENGTH_LONG).show(); //Afficher le message show()
```

Notification (Status Bar Notification)

- S'exécute souvent en trois étapes et en arrière plan :
  - Étape #1 : *Status Bar Notification* ajoute une petite icones (avec ou sans message) à la barre de statut du Système.
  - Étape #2 : Lorsque l'utilisateur clique sur l'icône ou message, le détail de la notification s'affiche.
  - Étape #3 : Lorsque l'utilisateur clique sur le détail, l'action définie (généralement un Activity) se déclenche.
  - On peut y associer une sonnerie, une vibration ou tout autre option offerte par l'appareil utilisé.



Notification (Alarm Manager)

- La class *AlarmManager* donne accès aux services de gestion des alertes du Système.
- Il est possible de céduler l'heure et la fréquence d'envoi des alertes
- Mais aussi préciser le composant cible qui sera exécuté le cas échéant

C’est quoi une Activity?

Déclaration des Activity dans le fichier ***manifest***

Activity et application

Le cycle de vie d’une Activity

ACTIVITY

Activity (C’est quoi ?)

- Une Activity représente un écran (semblable à un From) qu’une application peut présenter à l’utilisateur.
- Plus l’application est complexe, plus il y aura des écrans donc des Ativity.
- Ex : Une application pour l’envoi des SMS va présenter :
    - Une Activity pour la liste des contacts
    - Une Activity pour écrire le message
    - Une Activity pour lister les anciens messages etc.
- Activity vs View
    - Les Views sont des composants graphiques qui permettent l’interaction entre l’utilisateur et l’activité.
- Activity vs Application
    - Une Application est constituée d’une ou plusieurs Activity
    - Dans le cas de plusieurs Activity, une d’entre elle sera marquée pour être présentée en premier à l’utilisateur lors du lancement de l’application.

Activity (Déclaration dans le fichier Manifest)

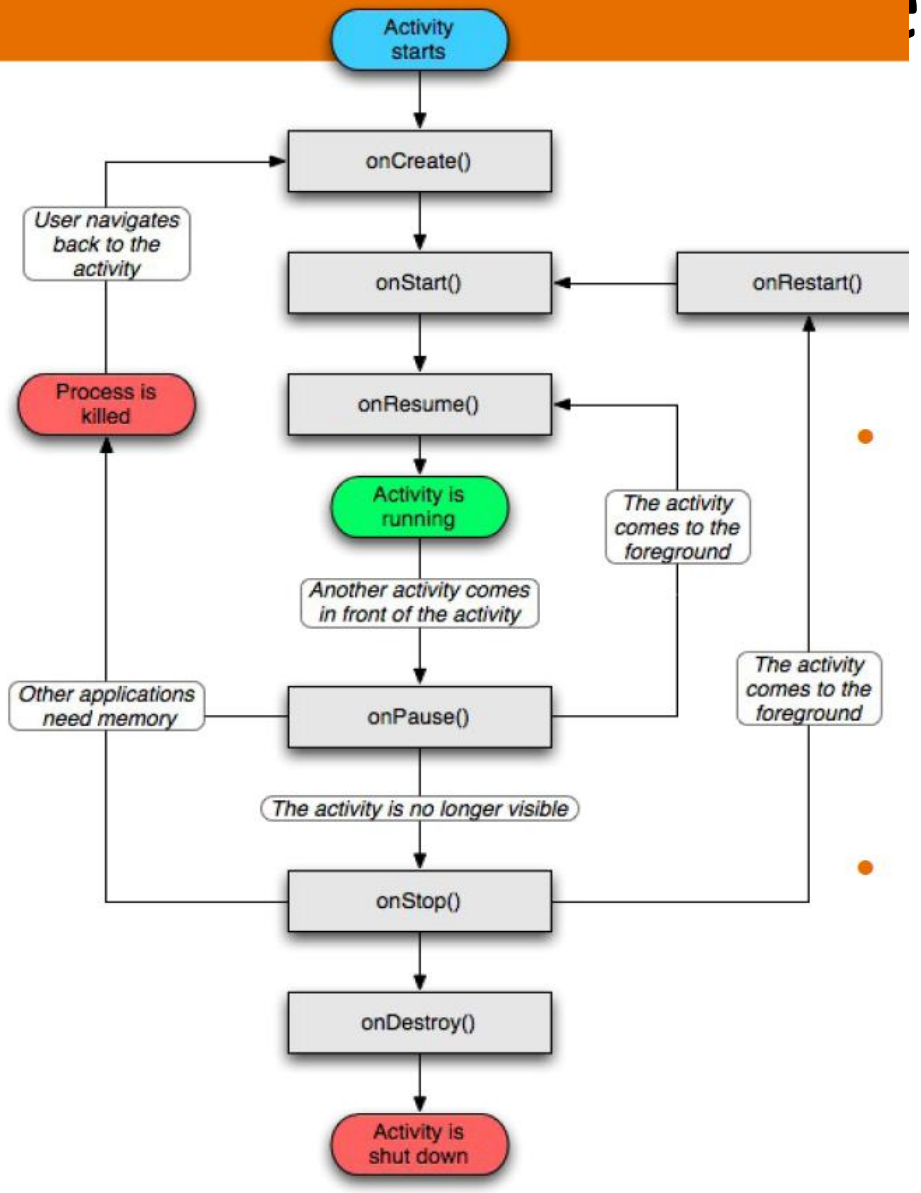
- Chaque Activity doit être déclarée dans le fichier *manifest*
- Sinon, une erreur sera affichée (Activity Not Found)

```
<?xml version="1.0" encoding="utf-8"?>
<manifest . . . >
     <application . . . >
             <activity android:name="com.project.MyActivity"
                     android:icon="@drawable/small_pic.png"
                     android:label="@string/freneticLabel"
                     . . . >
             </activity>
     . .
     </application>
</manifest>
```

Activity (Cycle de vie)

- Il ya quatre états possibles dans le cycle de vie d'une Activity :
  - **Avctive** : Lorsque l'Activity est en premier plan, est visible, à le focus et reçoit les entrées de l'utilisateur
  - **Paused** : Lorsque l'Activity n'a plus le focus mais reste visible à l'écran. C'est le cas où, une autre Activity transparente ou n'occupant pas tout l'écran devenait active.
  - **Stopped** : Lorsqu'une activité n'est plus visible à l'écran. Elle reste en mémoire et garde toutes ses informations d'état. Cette activité est susceptible d'être tuée par le système pour libérer les ressources.
  - **Inactive** : Une fois tuée ou avant d'être lancée, une Activity est inactive. Les Activity inactives sont supprimées de la pile.
- La pile des Activity est gérée par le système.
  - Une Activity nouvellement lancée est systématiquement placée en haut de la pile.

Activity (Ciclo de vida)



APPLICATION ET RESSOURCES

C’est quoi une ressource?

Externalisations des ressources

Default vs. ressources Alternatives

Où placer mes ressources

Resources (C’est quoi ?)

- Toutes les données statiques qui peuvent être externalisées du code Java
  - Layouts
  - String
  - Images
  - Vidéo et audio
  - Etc.

Resources (Externalisation)

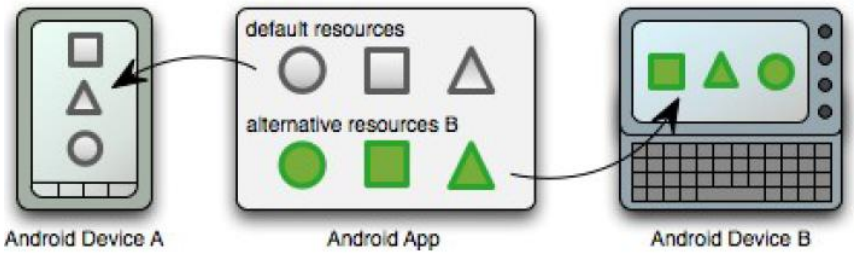
- Permet de maintenir les ressources en dehors du code Java.
- Permet de fournir des ressources alternatives en fonction de l'utilisateur ou du matériel utilisé :
  - Gestion du Multilingue
  - Gestion des différentes tailles d'écrans
- C'est encore plus important vu la multiplication des appareils supportant Android

Resources (Alternative)

- Pour tous les types de ressources, il est possible d'en spécifier une par défaut et plusieurs autres alternatives pour la même application.
  - Les ressources alternatives seront utilisées sans égard à la configuration de l'appareil ou bien s'il n'y a pas de ressources alternatives pour une config donnée.
  - Les ressources alternatives seront utilisées pour des cas spécifiques et prédéterminés.

Deux configurations utilisant Les ressources par défaut

Deux configurations dont un utilise les ressources alternatives



Resources (res/)

- Chaque type de ressources doit être placé dans un sous-répertoire du dossier res/
- Ex :

```
Myproject/
    src/
      MyActivity.java
    res/
      drawable/
                icon.png
      layout/
                main.xml
                info.xml
      values/
                strings.xml
```

Système de stockage de données
Les Préférences (Shared Preference)
Stockage interne
Stockage externe
Base de données

STOCKAGE DE DONNÉES

Données (système de stockage)

- Android offre plusieurs façons pour stocker les données d'une application en fonction des besoins.
- Exemple :
  - Les données doivent être privées pour une application
  - Les données doivent être accessibles par d'autres applications
  - La taille des données à stocker etc…

Données (Les préférence)

- Données de type ***primitive*** stockées sous le mode Clé-Valeurs.
- La classe *SharedPreferences* fourni un ensemble d'outils pour sauvegarder et retrouver le couple Clé-Valeurs.
- On peut utiliser la classe *SharedPreferences* pour stocker toutes les données de type *primitive :*
  - boolean, float, int, long et string
- Les données ainsi stockées seront persistées à travers les sessions utilisateurs (même lorsque l'application est fermée)

Données (Stockage Interne)

- Permet de stocker des données privées directement dans la mémoire du téléphone
- Les fichiers stocker dans le téléphone ne peuvent être utilisés **que par votre application**
- Lorsque l'application est désinstallée, les fichiers stockés sont automatiquement supprimés.

```
String FILENAME = "hello_file";
String string = "hello world!";
FileOutputStream fos = openFileOutput(FILENAME,
Context.MODE_PRIVATE);
fos.write(string.getBytes());
fos.close();
```

Données (Stockage Externe)

- Permet de stocker des données publiques dans des espaces partagés
- Tous les téléphones Android supportent ce mode stockage partagé via :
  - Une clé USB
  - Une carte mémoire
  - Le système de fichier interne.
- Les fichiers stockés avec ce mode sont accessibles à tous (matériels, logiciels, utilisateurs) et peuvent être modifiés par l'utilisateur par exemple lors du transfert vers son ordinateur.

Données (Base de données)

- Permet de stocker des données de façon structurée dans une base de données privée
- Android fourni par défaut un support complet de la base de données SQLite **(Zéro configuration)**
  - SQLite est aussi équipé pour gérer les requêtes SQL Transactionnelles.
- Toutes les bases créées et les données stockées seront accessibles par toutes les class de l'application **(mais pas en dehors de l'application)**.

Données (Base de données)

```
private static final String DATABASE_NAME = "myDB.db";
private static final String DATABASE_TABLE_NAME = "COUNTRY";
private static final String DATABASE_CREATE_TABLE =
"create table " + DATABASE_TABLE_NAME +
" (_id integer primary key autoincrement, " +
" country_name text not null, " +
" capital_city text not null)";
    // Ouvrir une nouvelle SQLiteDatabase associée à ce Context de l'application
    // Crée le fichier de base de données s'il n'existe pas encore.
    SQLiteDatabase myDB = openOrCreateDatabase(DATABASE_NAME,

          Context.MODE_PRIVATE,
                                                                    null);
    // créer la table
    myDB.execSQL(DATABASE_CREATE_TABLE);
============================================================
    SQLiteDatabase myDB = openOrCreateDatabase(DATABASE_NAME,
    Context.MODE_PRIVATE, null);
    // Créer une nouvelle ligne dans la base.
    ContentValues newRow = new ContentValues();
    newRow.put("country_name", "U.S.A.");
    newRow.put("capital_city", "Washington D.C.");
    myDB.insert(DATABASE_TABLE_NAME, null, newRow);
```

Plateforme Android pour l’audio et la vidéo

Lecture Audio and Vidéo

Formats de média supportés

MEDIA (AUDIO ET VIDÉO)

Média (Audio et Vidéo)

- Android offre une plateforme complète pour encoder/décoder plusieurs types de médias.
- La même mécanique d'accès que celle utilisée pour tous les autres types de ressources.
- Il facilite l'intégration des images, des vidéos et de la musique dans les applications tierces.

Média (Lecture Audio et Vidéo)

- Le média à lire peut être obtenu depuis plusieurs sources
  - Le répertoire des ressources : /res/raw

```
MediaPlayer mp = new MediaPlayer();
MediaPlayer mp = MediaPlayer.create(context,
R.raw.sound_file_1);
mp.start();
```

  - Le système de fichiers : Local (PATH) ou Réseau (URL)

```
MediaPlayer mp = new MediaPlayer();
mp.setDataSource(PATH_TO_FILE);
mp.prepare();
mp.start();
```

Média (Formats supportés)

- Audio
  - AAC LC/LTP, HE-AACv1 (AAC+), HE-AACv2 (enhanced AAC+), AMR-NB, AMR-WB, MP3, MIDI, Ogg Vorbis, PCM/WAVE
- Vidéo
  - H.263, H.264 AVC, MPEG-4 SP

C’est quoi un service?

Services locaux

Services distants

SERVICES D’ARRIÈRE PLAN

Services (C’est quoi ?)

- Certaines applications ont besoin de continuer à fonctionner en arrière plan sans l’intervention de l’utilisateur.
- Ces processus d’arrière plan continuent de fonctionner même lorsque le téléphone est occupé avec d’autres tâches ou d’autres Activity.

Services (Locaux et Distants)

- Services Locaux
  - Ne sont pas accessibles depuis d'autres applications
  - S'exécute dans le même processus que l'application qui l'exécute.
- Services distants
  - Sont accessibles par d'autres applications
  - S'exposent à d'autres applications via AIDL (Android Interface Definition Language).

Services de géolocalisation

Simuler les données de géolocalisation

Google Maps (external library)

GÉOLOCALISATION ET MAP

Géolocalisation et Map (Services)

- Les principales techniques utilisées par Android pour déterminer la position courante d'un appareil sont les suivantes :
  - Location Manager : Fournit des hooks (points d'entrée) vers les services de géolocalisation pour :
    - Obtenir votre position courante
    - Suivre des déplacements sur une période donnée
    - Enregistrer ou supprimer les données de localisation périodiques via ***LocationProvider***
    - Déclencher des alertes de proximité à l'approche d'une zone spécifique
  - Location Providers : Chaque Location Provider représente une technologie de localisation de la position de l'appareil

Géolocalisation et Map (Simuler une position)

- Utiliser DDMS (Dalvik Debug Monitor Service) sous Eclipe dans la perspective DDMS pour :
  - Envoyer manuellement les données de localisation (longitude/latitude) dans le *LocationProvider*
- Utiliser un fichier GPX (GPS Exchange Format) décrivant la route à «simuler». GPX est fichier XML permettant d'inter-changer les données GPS
- Utiliser les fichier KLM décrivant les points de repère pour la séquence de lecture.
- Utiliser «GEO» en ligne de commande. Cet outil accessible depuis la console de l'émulateur Android permet de simuler les positions

Géolocalisation et Map (Google Map)

- Basé sur la classe : **com.google.android.maps.MapView**
- Permet d'ajouter les capacités offertes par Google Map directement dans votre application
- Afficher une carte avec des données obtenues depuis Google Map Service
- Gestion du zoom (***Touch gestures)***
- Fourni tous les éléments d'interface graphique nécessaires à un utilisateur pour contrôler la les cartes géographiques
- Un enregistrement et l'acceptation d'un agrément au service Google Map est nécessaire avant de pouvoir obtenir les données de Google Map

HttpClient API

Invocation styles (in HttpClient API)

Format de réponse

WEB SERVICE ET CLOUD

Web services (HttpClient API)

- Basé sur Apache HTTP package
- Fourni une interface pour un client HTTP
- Supporte le multitâche
- Supporte le mode Synchronous et Asynchronous
    - public abstract HttpResponse execute (HttpUriRequest request)
- Response Formats
    - XML
    - JSON
    - RSS, Atom
- XML Parsing
    - SAX
    - DOM
    - Pull-parser

Web services (HttpClient API)

Example Code - Synchronous

```
HttpClient httpclient = new DefaultHttpClient();
// Prepare a request object
HttpGet httpget = new HttpGet(url);
// Execute the request
HttpResponse response;
try {
    response = httpclient.execute(httpget);
    // Get hold of the response entity
    HttpEntity entity = response.getEntity();
    if (entity != null) {
    // A Simple JSON Response Read
    InputStream instream = entity.getContent();
    result = convertStreamToString(instream);
```

Web services (HttpClient API)

Example Code - Asynchronous

```
HttpClient httpclient = new DefaultHttpClient();
// Prepare a request object
HttpGet httpget = new HttpGet(url);
try {
    ResponseHandler<String> mResponseHandler =
                                        new
BasicResponseHandler();
    result = httpclient.execute(httpget,
mResponseHandler);
...
```

Les principes de sécurité d'Android

Les permissions

SÉCURITÉ ET PERMISSIONS

Sécurité et permissions

- Aucune application n’est autorisée par défaut à effectuer des opérations qui auront un effet négatif sur d’autres applications ou sur le système d'exploitation ou encore altérer l’expérience utilisateur.
- Chaque application s'exécute dans un processus propre à lui et fermé.
  - Elle ne peut pas perturber les autres applications, sauf si elle est explicitement autorisée à utiliser les capacités non prévues par défaut. EX : Gérer les SMS

```
<manifest
   xmlns:android="http://schemas.android.com/apk/res/android"
   package="com.android.app.myapp" >
   <uses-permission android:name="android.permission.RECEIVE_SMS"
   />
</manifest>
```