



Développement Mobile sous Android



Marwa Hammami Maître-assistante en Informatique de Gestion
3 ème année Licence LIG

2022/2023

Plan



- 1. Chapitre 1 : Introduction générale**
2. Chapitre 2 : Le système Android
3. Chapitre 3 : Les ressources
4. Chapitre 4 : Les activités Android
5. Chapitre 5 : Les intents
6. Chapitre 6 : Gestion des données persistantes
7. Chapitre 7 : Le système multitâche dans Android
8. Chapitre 8 : Gestion des réseaux Dans Android

Premier Cours



- 1) Comprendre les éléments de base pour le développement sous d'Android**

- 2) Pouvoir créer une première application simple**

Premier Cours : Première partie



Android :
Le quoi,
Le pourquoi,
Le contexte,
...

Introduction : Téléphonie mobile



- **SmartPhone [wikipedia]**

- « Un smartphone, ordiphone ou téléphone intelligent, est un téléphone mobile disposant aussi des fonctions d'un assistant numérique personnel
- La saisie des données se fait par le biais d'un écran tactile ou d'un clavier
- Il fournit des fonctionnalités basiques comme : l'agenda, le calendrier, la navigation sur le web, la consultation de courrier électronique, de messagerie instantanée, le GPS, etc ».



Dispositifs mobiles

- **PDA**: Personal Digital Assistant appelé aussi organisateur ou ordinateur de poche.
- Il fournit les applications suivantes :
 - Un agenda
 - Un gestionnaire de tâches (aide-mémoire)
 - Un carnet d'adresses
 - Un logiciel de messagerie
 - Des outils de bureautiques allégés
 - Géolocalisation (GPS)



Dispositifs mobiles

- **Smartphone:** un téléphone mobile disposant en général d'un écran tactile et d'un appareil photographique numérique, et des fonctions d'un PDA et de certaines fonctions d'un ordinateur portable.
- Ils permettent de
 - téléphoner
 - envoyer des messages,
 - envoyer des mails,
 - écouter de la musique,
 - regarder des films,
 - jouer,
 - prendre des photos et vidéos...
 - L'ajout d'applications ce qui permet de personnaliser l'appareil selon les besoins.



Dispositifs mobiles

- **Tablette**: est un **ordinateur portable** ultraplat qui se présente sous la forme d'un **écran tactile** sans clavier et qui offre à peu près les mêmes fonctionnalités qu'un ordinateur personnel.
- La tablette tactile est un intermédiaire entre l'ordinateur portable et le smartphone.
- Elle permet de :
 - Accéder à des contenus multimédias (TV),
 - Naviguer sur le web,
 - Consulter et envoyer des courrier électroniques, l'agenda,
 - Consulter calendrier et la bureautique simple.
 - Il est possible d'installer des applications supplémentaires depuis une boutique d'applications en ligne.



Dispositifs mobiles

- **Consoles de jeu mobile:** C'est une console de jeux qui est conçue de façon à pouvoir être transportée.
- La principale différence entre les consoles dites portables et les consoles « classiques » est que l'ensemble du matériel nécessaire pour jouer, c'est-à-dire l'écran, les touches de contrôle et les haut-parleurs sont regroupés dans un seul ensemble léger. De cette façon, la console peut être prise en main.



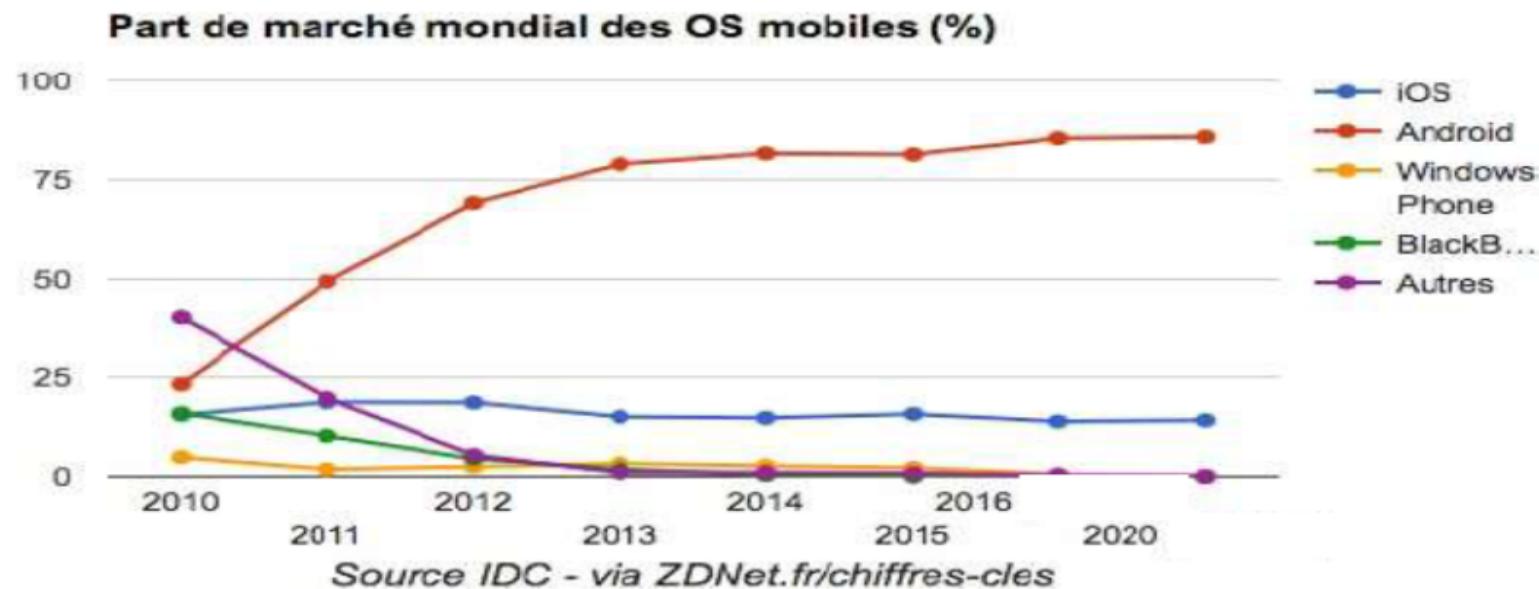
Dispositifs mobiles

- **Smartwatch:** est une montre bracelet informatisée avec des fonctionnalités allant au-delà du simple affichage de l'heure et du chronométrage,
- Elle présente des caractéristiques comparables à celles d'un PDA.
- Il faut les considérer comme des **ordinateurs de poignet**.



Introduction : Téléphonie mobile

- Vente de "terminaux mobiles" évolués
 - Un téléphone mobile sur deux vendu dans le monde



Source : <https://www.leptidigital.fr/technologie/parts-de-marche-systemes-exploitation-mobiles-ios-android-windows-12957/>

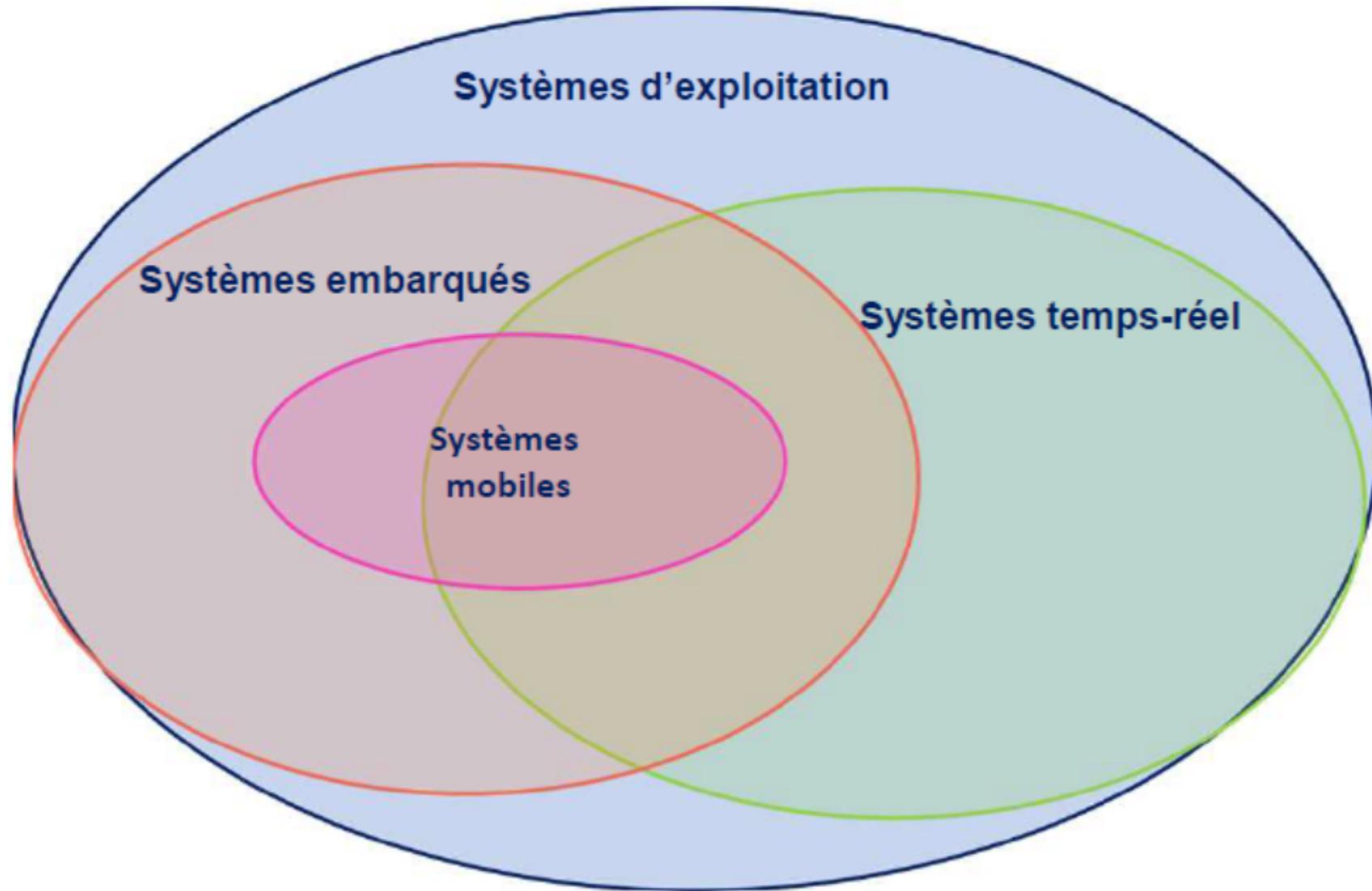
On note que l'**Android** est le système d'exploitation le plus utilisé dans le monde.

Introduction : Les principaux OS mobiles



- **Android** (google, ...)
- **Iphone OS** (Apple) sur des téléphones IPhone et sur les tablettes d'Apple
- **Windows Mobile** (Microsoft) ; système propriétaire
- **Symbian** (Nokia) ; récemment passé en open source
- **BlackBerry OS**. Présent sur tous les téléphones de la marque RIM (Research In Motion) ;
- **Palm Web OS** (successeur de Palm Os)
- **LiMo** (Linux Mobile), Système ouvert basé sur Linux (<http://www.tizenassociation.org/en/>)
- **MeeGo**, Intel et Nokia (<https://meego.com/>)
- **Bada**, Samsung (<http://www.bada.com/whatisbada/index.html>)
- Etc.

Classification des Systèmes d'exploitation



Source : <http://socialcompare.com/>

Systèmes d'exploitation mobiles



Un **système d'exploitation mobile** (SEM) est un système d'exploitation embarqué conçu pour fonctionner sur un **appareil mobile**. Ce type de système d'exploitation se concentre entre autres sur la gestion de la **connectivité sans fil** et celle des différents types d'**interface**.

- Un système d'exploitation mobile est l'ensemble des programmes responsables de:
 - la gestion des opérations (processus, ordonnancement, E/S, fichiers, multimédias...)
 - la coordination,
 - l'utilisation du matériel
 - le partage des ressources d'un dispositif mobile entre divers programmes tournant sur ce dispositif

Contraintes des systèmes d'exploitation mobiles



- Les **dispositifs mobiles** ont des **contraintes** et des restrictions sur leurs caractéristiques physiques telles que:
 - la taille de l'écran,
 - la mémoire,
 - la puissance de traitement et etc.
 - Faible disponibilité de batterie
 - Quantité limitée de capacités informatiques et de communication.
- Ainsi, ils ont **besoin** de différents types de **systèmes d'exploitation** en fonction des capacités qu'ils supportent. **par exemple**, un OS PDA est différent d'un système d'exploitation de Smartphone.
- Un **système d'exploitation mobile** est un **logiciel embarqué** responsable de la gestion des opérations, du contrôle, de la coordination de l'utilisation du matériel entre les différents programmes d'application et le partage des ressources d'un dispositif mobile.

Applications mobiles



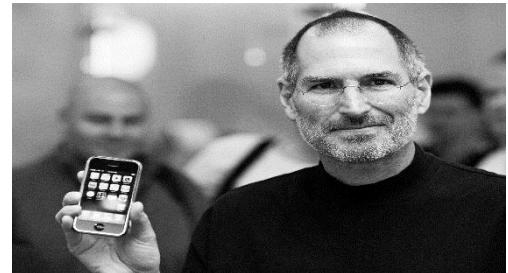
- Une **application mobile** est un **programme** téléchargeable de façon gratuite ou payante et **exécutable** à partir du **système d'exploitation** d'un dispositif mobile (smartphone, tablette...).
- **Elles** sont pour la plupart **distribuées** depuis des plateformes de **téléchargement** (parfois elles-mêmes contrôlées par les fabricants de smartphones) telles que l'**App Store** (plateforme d'Apple), le **Google Play** de Google/Android), ou encore (plateforme le **Windows Phone Store**(plateforme de Microsoft).

La plate-forme Android : Historique



- **L'iphone d'Apple**

- A bouleversé le paysage des systèmes d'exploitation mobiles par :
 - Son ergonomie et les capacités du matériel
 - Les usages proposés
 - Les possibilités offertes avec l'Apple Store



- **Handset Alliance**

- Est une coalition qui a vu le jour fin 2007
 - A pour objectif de créer et de promouvoir le système Android comme système ouvert et gratuit dans le monde du mobile
 - Google est l'acteur majeur
 - Adresse web : <http://www.openhandsetalliance.com>

La plate-forme Android : Historique



• Les versions de la plate-forme

- Première version d'Android en septembre 2008, 1.1 (février 2009), 1.5 (Avril 2009), 1.6 (septembre 2009), 2.0 (octobre 2009), 2.0.1 (Octobre 2009) ...



La plate-forme Android : Caractéristiques



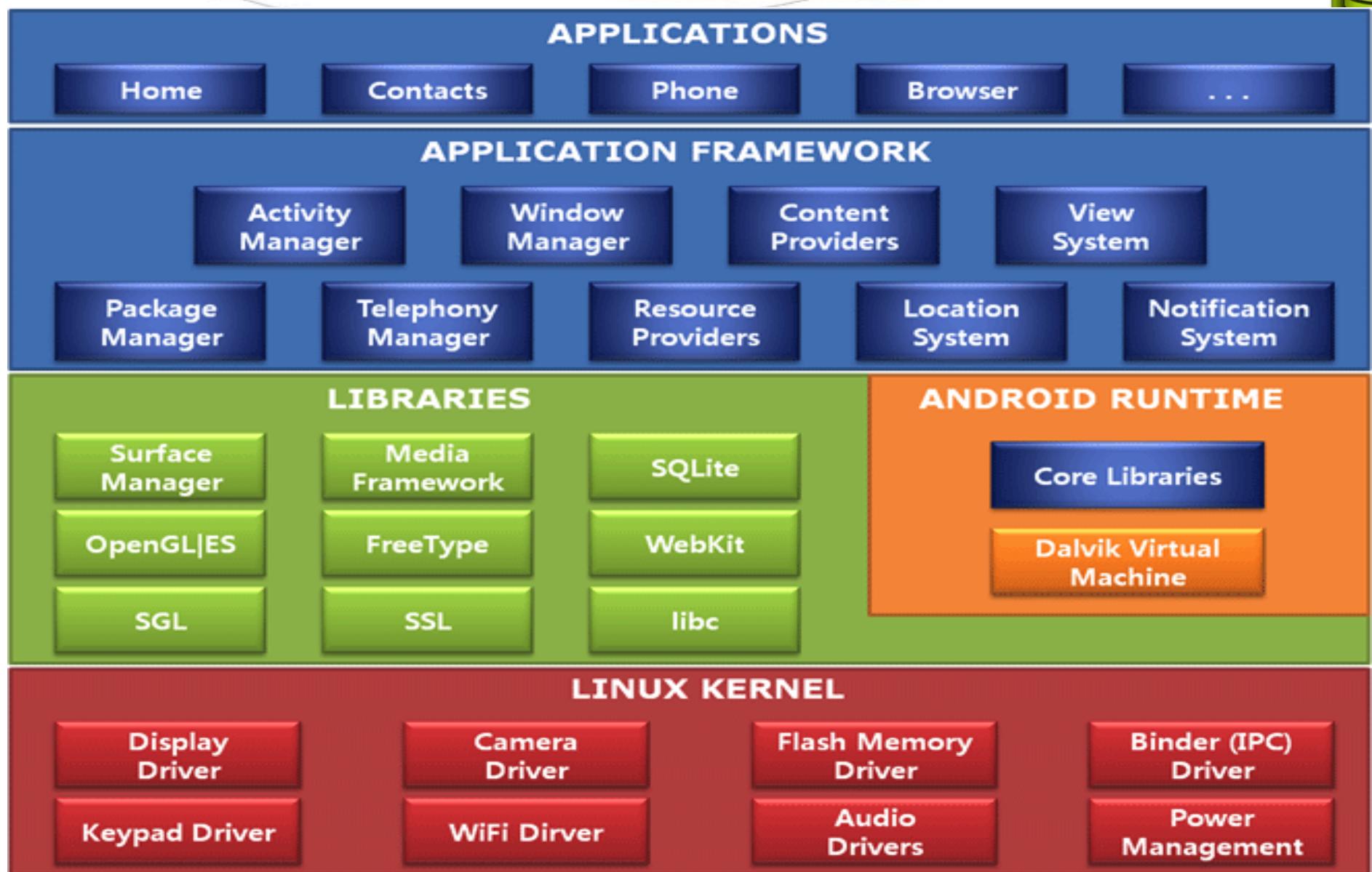
- **Elle est innovante**
 - Toutes les dernières technologies de téléphonie y sont intégrées : écran tactile, accéléromètre, GPS, appareil photo numérique, etc.
- **Elle est accessible**
 - En tant que développeur, il n'y a pas besoin de matériel spécifique
 - Utilisation d'un émulateur
 - Pas d'apprentissage d'un langage spécifique. Le développement ce fait en *Java*
- **Elle est ouverte**
 - Elle est fournie sous licence *open source*, permettant aux développeurs et constructeurs de consulter les sources et d'effectuer les modifications qu'ils souhaitent
 - Utilisation de la *licence Apache* ce qui permet la redistribution du code sous forme libre ou non et d'en faire un usage commercial

La plate-forme Android : Architecture



- **Android est conçue pour des appareils mobiles au sens large**
 - Téléphones mobiles, tablettes, ordinateurs portables, bornes interactives, baladeurs, Téléviseurs, machine à laver, interaction usagers voiture, ...
- **La plate-forme Android est composée de différentes couches**
 - Un noyau Linux permettant des caractéristiques multitâches
 - Des bibliothèques graphiques, multimédias
 - Une machine virtuelle Java open-source : la Davik Virtual Machine
 - Il existe un framework natif permettant le développement en C/C++ NDK (Native Development Kit)
 - 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...

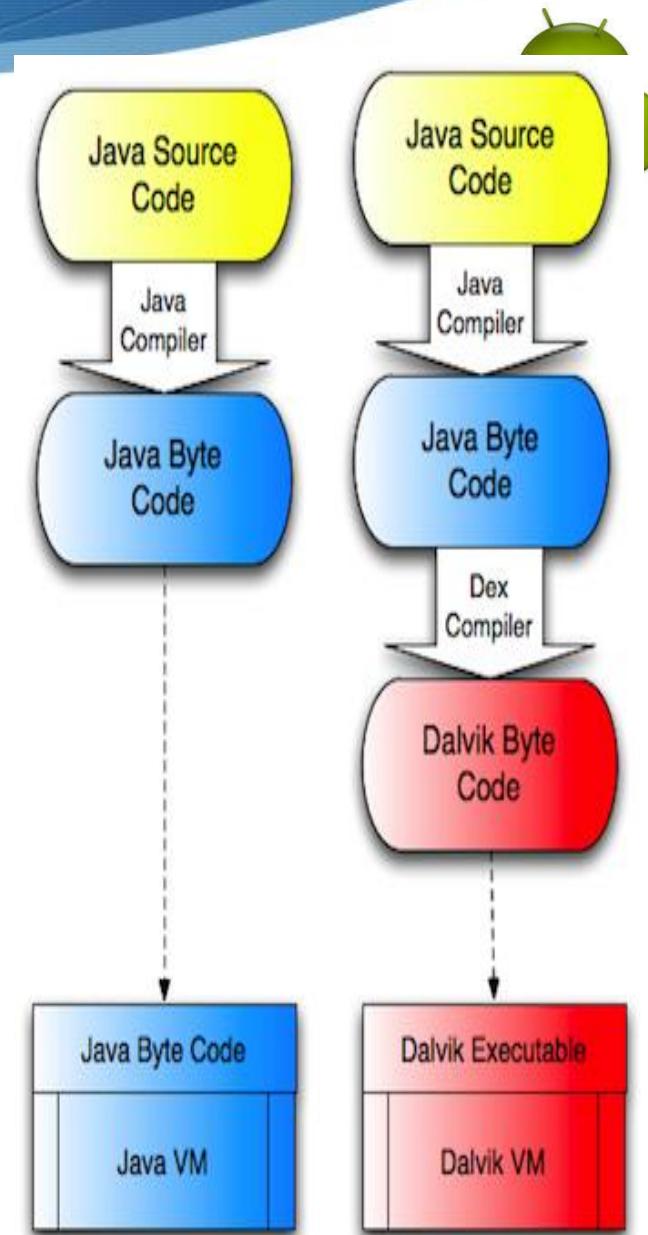
La plate-forme Android : Architecture



La plate-forme Android : kit de développement

- **Machine virtuelle "Dalvik"**

- Offre l'avantage de toute machine virtuelle
 - Couche d'abstraction entre le développeur d'applications et des implémentations matérielles particulières
- La VM Dalvik n'est pas une VM Java
 - Tient compte des contraintes de CPU et mémoire
 - Exécute des fichiers .dex (*Dalvik Executable*) optimisé
 - Les applications sont totalement indépendantes ("sandbox")



La plate-forme Android : kit de développement



- **Le SDK Android est composé de plusieurs éléments :**
 - Des API (Application Programming Interface)
 - Un certain nombre d'exemples illustrant les possibilités du SDK
 - De la documentation
 - Des outils -parmi lesquels un émulateur
- **Le SDK Android est disponible sur le site de Google :<http://developer.android.com>**
- **Autres briques logicielles :**
 - ADT : Android Development Tools Plugin
 - Outil s'intégrant directement à Eclipse
 - Propose des interfaces et des assistants pour la création et le débogage des applications Android
 - Android Studio : <https://developer.android.com/studio/index.html>



Cours1-Partie 2

Présentation des différents
composants d'une application
Android



Fichier de configuration Android (manifest)



- **Qu'est que c'est ?**
 - Une application Android est un assemblage de composants liées grâce à un fichier de configuration
 - Décrit entre autres :
 - Le point d'entrée de l'application : quel code doit être exécuté au démarrage de l'application
 - Quels composants constituent ce programme : les activités, les services, ...
 - Les permissions nécessaires à l'exécution du programme
- **Comment ?**
 - Fichier XML : AndroidManifest.xml

Fichier de configuration Android

- ## Exemple



```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="andro.jf"
    android:versionCode="1"
    android:versionName="1.0">
    <application android:icon="@drawable/icon"
        android:label="@string/app_name">
        <activity android:name=".Main"
            android:label="@string/app_name">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>

        <service>...</service>
        <receiver>...</receiver>
        <provider>...</provider>
    </application>
</manifest>
```

Composants d'une application Android

- Les composants peuvent être classés en éléments applicatifs et éléments d'interaction
 - Eléments applicatifs
 - Activité
 - Service
 - Fournisseur de contenu
 - Gadget (widget)
 - Eléments d'interaction
 - Objet Intent
 - Récepteur d'Intents
 - Notification



Composants d'une application Android



Activités de l'application

Application

Activité au premier plan

Vue2

Vue1

Vue3

Vue n

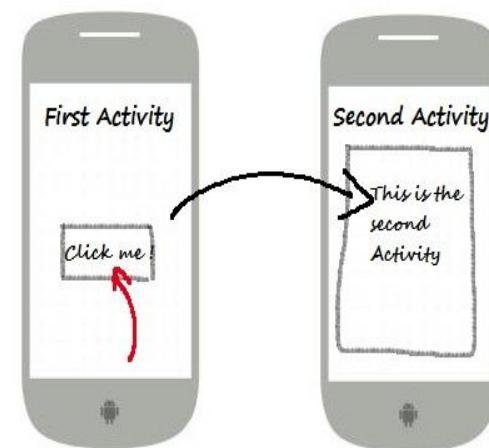
Ressources

Manifeste : fichier de configuration

Activité

• Présentation

- Elle correspond à la partie présentation de l'application : *correspond à un écran*
 - Représente le bloc de base d'une application
- Fonctionne par le biais de *vues* qui affichent des interfaces graphiques et répondent aux actions utilisateur
 - Elle est composée d'une hiérarchie de vues contenant elles-mêmes d'autres vues
 - Un formulaire d'ajout de contacts ou encore un plan Google Maps sur lequel on peut ajouter de l'information
 - Une application comportant plusieurs écrans, possédera donc autant d'activités



Activité



- **Utilisation**

- Une activité est composée de deux volets :
 - Sa logique métier et la gestion de son cycle de vie
 - Implémentés en Java dans une classe héritant de *Activity*
 - Son interface utilisateur
 - Deux façons alternatives pour sa définition:
 - Programmative : dans le code de l'activité
 - Déclarative : dans un fichier XML

Activité

- **Logique métier d'une activité : Squelette minimal**



```
import android.app.Activity;  
import android.os.Bundle;  
  
public class ActiviteBasic extends Activity {  
  
    //méthode OnCreate appelée à la création de l'activité  
    public void onCreate(Bundle etatSauvegarde){  
        super.onCreate(etatSauvegarde);  
  
    }  
}
```

Activité



- La balise `<activity>` déclare une activité
 - Les paramètres généralement utilisés sont :
 - `name` qui désigne la classe de l'activité
 - `label` qui désigne le nom sous lequel elle apparaîtra sur le terminal
 - `icon` qui désigne l'icône à afficher sur le terminal
 - Structure

```
<application ...>
<activity android:name=".ClasseDeLActivite"
          android:label="nom_de_l_activite"
          android:icon="@drawable/nom_du_fichier_icone">

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

Activité



- **Cycle de vie d'une activité**

- Les états principaux d'une activité sont les suivants :
 - Active (active)
 - Activité visible qui détient le focus utilisateur et attend les entrées utilisateur
 - Appel à la méthode *onResume()*
 - Suspendue (Paused)
 - Activité au moins en partie visible à l'écran mais qui ne détient pas le focus
 - Appel à la méthode *onPause()* pour entrer dans cet état
 - Arrêté (stopped)
 - Activité non visible
 - Appel de la méthode *onStop()*

Activité

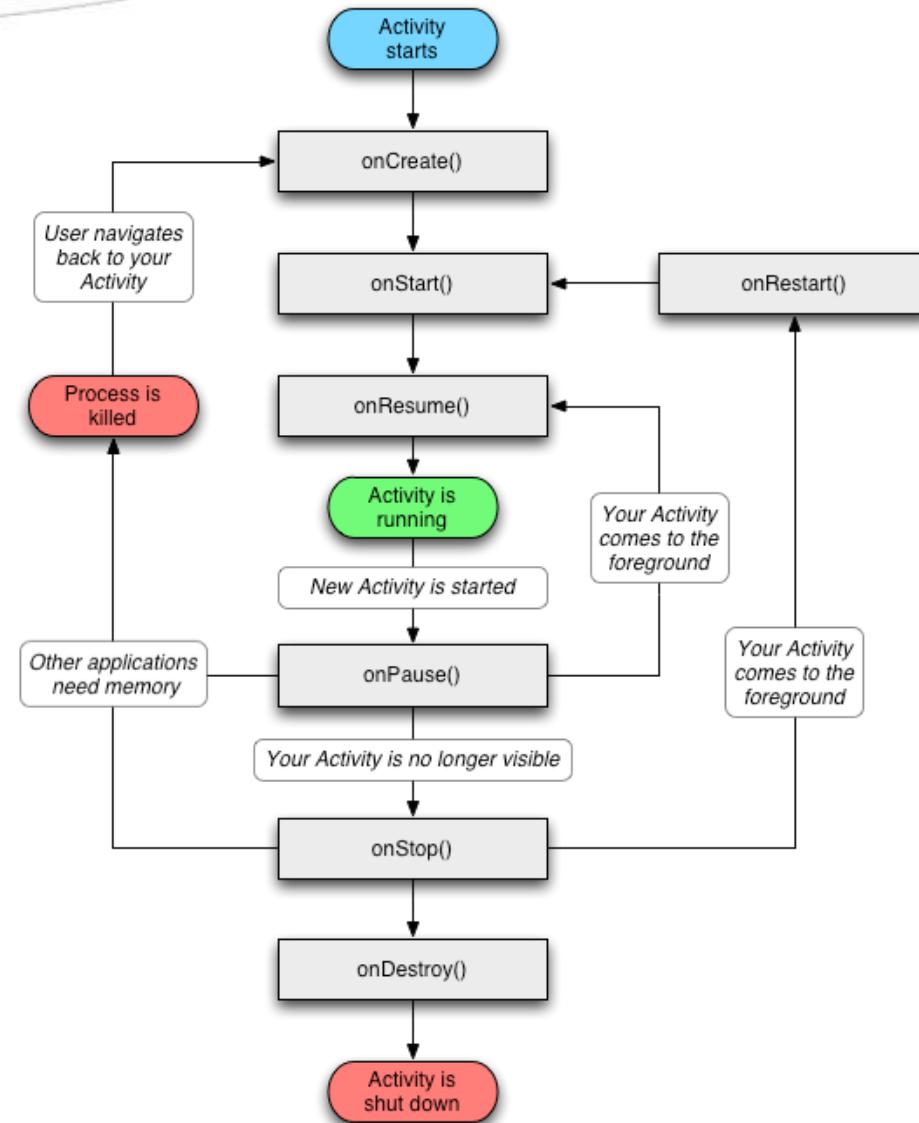
- **Cycle de vie d'une activité**

```
public class Main extends Activity {  
  
    public void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        setContentView(R.layout.acceuil);  
  
    protected void onDestroy() {  
        super.onDestroy();  
  
    protected void onPause() {  
        super.onPause();  
  
    protected void onResume() {  
        super.onResume();  
  
    protected void onStart() {  
        super.onStart();  
  
    protected void onStop() {  
        super.onStop();  
    }  
}
```



Activité

● Cycle de vie d'une activité



Les interfaces d'applications : Les Vues

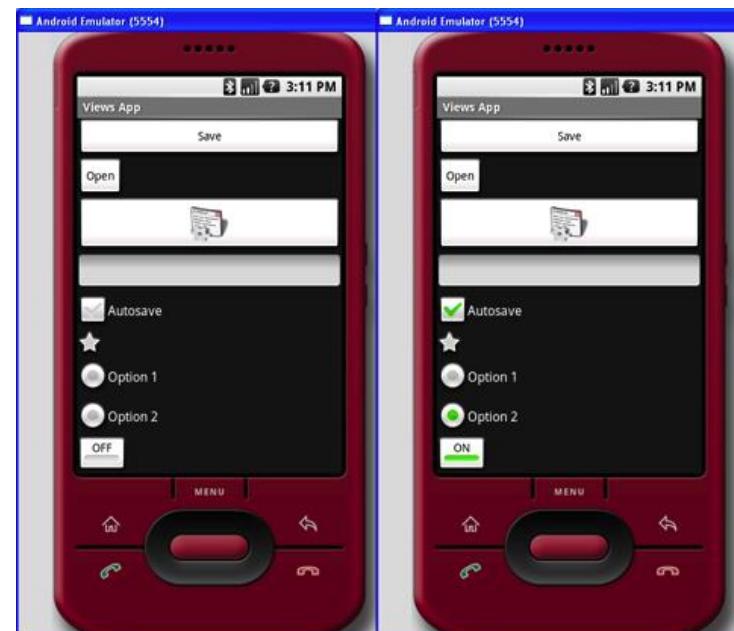


• Présentation

- Sont les briques de construction de l'interface graphique d'une activité Android

• Utilisation

- Les vues sont soient prédéfinies par la plateforme -**textes, boutons, ...** ou créées comme des éléments personnalisés
- Chaque écran Android contient un arbre d'éléments de type *View*
- Les vues peuvent être disposées dans une activité (objet *Activity*) et donc à l'écran soit par une description XML, soit par un morceau de code Java



Création d'une interface utilisateur



- **Deux possibilités pour créer une interface**
 - Directement dans le code : instancier les vues dans le code
 - La création en deux étapes en séparant la présentation de la logique fonctionnelle de l'application
 - Définition de l'interface utilisateur (gabarit, etc.) de façon déclarative dans un fichier XML
 - Définition de la logique utilisateur (comportement de l'interface) dans une classe d'activité

Définition de l'interface en XML

- Les fichiers de définition d'interface en XML sont enregistrés dans le dossier **res/layout** du projet
- Chaque fichier XML définissant une interface graphique est associé à un **identifiant unique** généré automatiquement qui peut être référencé dans le code de l'application
 - Exemple : **R.layout.monLayout**



```
<?xml version="1.0" encoding = "utf-8"?>
<LinearLayout xmlns:android=
    "http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width = "fill_perent"
    android:layout_height="fill_parent"
    >
    <TextView
        android:layout_width="fill_perent"
        android:layout_height="wrap_content"
        android:id="@+id/monText" />
</LinearLayout>
```

Association entre activité et interface



- Une interface est affichée par l'intermédiaire d'une activité
- Le chargement du contenu de l'interface s'effectue à linstanciation de lactivité
 - Redéfinition de la méthode *OnCreate()* de lactivité pour y spécifier la définition de linterface à afficher via la méthode
 - Affichage de linterface par la méthode *setContentView()*
 - Prend en paramètre un identifiant qui spécifie quelle ressource de type interface doit être chargée et affichée

Création une interface dans le code source (sans définition XML)

- **Exemple**

```
import android.app.Activity;
import android.os.Bundle;
import android.widget.TextView;

public class Main extends Activity {

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        TextView monTextView = new TextView(this);
        setContentView(monTextView);

        monTextView.setText(" Notre premier cours Android");

    }

}
```



Utilisation des gabarits



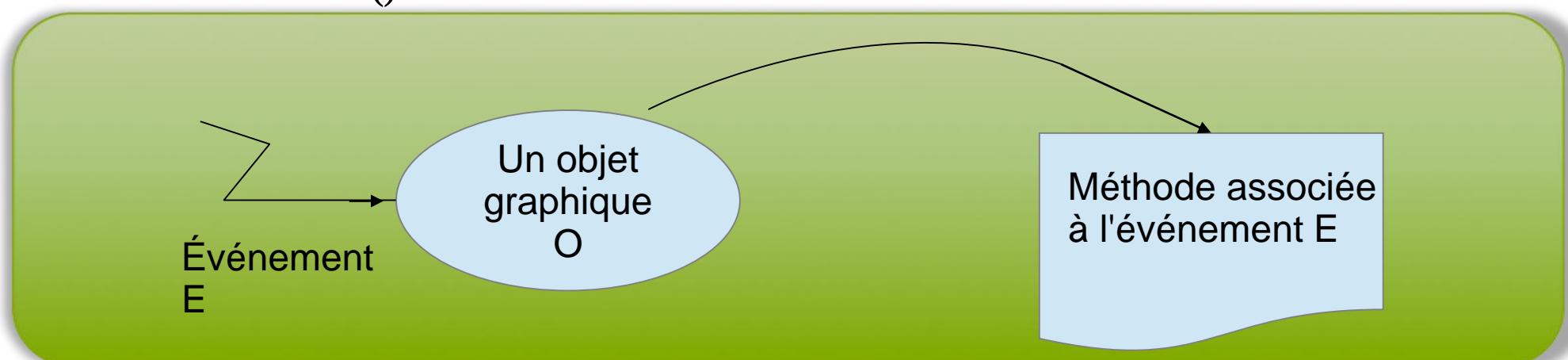
- Pour intégrer plus d'une vue à une activité : réunir tous ces vues dans un gabarit de vues
 - De manière programatique : Directement dans le code
 - De manière déclarative : dans un fichier XML

```
<?xml version="1.0" encoding = "utf-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width = "fill_perent"
    android:layout_height="fill_parent" />
<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:id="@+id/monText"
    android:text="Texte en haut à droite"
    android:gravity="topright" >
<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:id="@+id/monText"
    android:text="Texte en bas au centre"
    android:gravity="bottomlcenter_horizontal"/>
</LinearLayout>
```

Gestion des événements



- Sous Android, toutes les actions de l'utilisateur sont perçues comme un événement
- Les événements sont interceptés par les éléments d'une interface en utilisant des écouteurs (listeners)
 - Association entre un événement et une méthode à appeler en cas d'apparition de cet événement
 - Exemple : pour un événement OnClick, la méthode associée est OnClick()



Gestion des événements

- **Exemple**

Insertion d'un bouton dans l'interface

```
<?xml version="1.0" encoding = "utf-8"?>

<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width = "fill_perent"
    android:layout_height="fill_parent"
    android:gravity="center_vertical | center_horizontal"
    >

    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/monBouton"
        android:text="Cliquez ici !"
        >
    </Button>
</LinearLayout>
```



Gestion des événements

- **Exemple**

création d'un écouteur sur un bouton

```
import android.app.Activity;
import android.os.Bundle;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;
import android.widget.Toast;

public class Main extends Activity {
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        ((Button) findViewById(R.id.monBouton)).
            setOnClickListener(new OnClickListener() {
                @Override
                public void onClick(View v) {
                    Toast.makeText(Main.this, "Bouton cliqué !", Toast.LENGTH_LONG).show();
                }
            });
    }
}
```





Les ressources

Les ressources



- **Présentation**

- Sont des **fichiers externes** ne contenant pas d'instructions qui sont utilisés par le code
 - Les fichiers images JPEG et PNG, les fichiers XML...

- **Utilisation**

- L'**externalisation** des ressources permet une meilleure gestion de ces ressources ainsi qu'une maintenance plus aisée
 - Les ressources de l'application sont déposées dans le *répertoire res* du projet
 - Android crée *une classe nommée R* utilisée pour référer aux ressources dans le code
 - Toutes les ressources sont placées, converties ou non, dans **un fichier de type APK** qui constituera le programme distribuable de l'application

Les ressources



Type de ressources	Répertoire associé	Description
Valeurs simples	res/values	définitions en XML de valeurs : chaînes, tableaux, valeurs numériques
Drawables	res/drawable	Des ressources images
Layouts	res/layout	description en XML des interfaces
Ressources XML	res/xml	Fichier XML qui peuvent être lus et convertis à l'exécution par la méthode <code>ressources.getXML</code>

Les ressources

• Cration de ressources

- Les ressources de type valeur (entiers, boolens, chanes de caracteres, etc. et des tableaux) peuvent tre dcrites dans des fichiers xml :

```
<?xml version="1.0" encoding= "utf-8"?>
<resources>
    <color name= "coulfond">          #AA7B03      </color>
    <integer name= "limite">          567         </integer>
    <integer-array      name= "codes_postaux">
        <item>34100</item>
        ...
        <item>30000</item>
    </integer-array>
    <string name "nom_de_mon_application" > mon premier exemple Android </string>
    <string-array name= "planetes">
        <item>Mercure</item>
        ...
        <item>Venus</item>
    </string-array>
    <bool name="actif">          true       </bool>
    <dimen name "taille">          55px      </dimen>
</resources>
```



Les ressources



- **Utilisation des ressources**

- Les ressources peuvent être utilisées dans les fichiers XML ou dans le code java
- Utilisation des ressources dans le code Java
 - Les ressources peuvent être utilisées via leurs identifiants :utilisation de *la classe statique R* automatiquement générée

```
android.R.type_ressource.nom_ressource
```

Les ressources

- Utilisation des ressources

```
public final class R {  
  
    public static final class string {  
        public static final int invitation = 0x7f040001;  
        public static final int texte_titre_ecran = 0x7f040002;  
    };  
  
    public static final class layout {  
        public static final int ecrain_de_demarrage= 0x7f030001;  
        public static final int ecran_principal= 0x7f030002;  
    };  
  
    public static final class drawable {  
        public static final int image_android = 0x7f020000;  
    };  
};
```

Utilisation de la ressource dans le code Java



Android.R.string.invitation



Les ressources



- Utilisation des ressources

- Les ressources peuvent être utilisées dans les fichiers XML ou dans le code java
- Utilisation des ressources dans le code Java
 - Les ressources peuvent être utilisées via leurs identifiants :utilisation de *la classe statique R* automatiquement générée
 - Les ressources peuvent être utilisées en récupérant l'instance de la ressource en utilisant la classe **Resources**

```
Resources res = getResources();
String hw = res.getString(R.string.hello);
-----
XXX o = res.getXXX(id);
```

- Une méthode spécifique pour les objets graphiques permet de les récupérer à partir de leur id:

```
TextView texte = (TextView) findViewById(R.id.le_texte);
texte.setText("Here we go !");
```

Les ressources



- **Utilisation de ressources**

- Référencement d'une ressource dans un fichier XML
 - On référence une ressource dans un fichier XML par

```
"@[paquetage:]type/identificateur"
```

- Exemple

```
@string/nom_de_mon_application
```

- fait référence à une chaîne décrite dans un fichier XML placé dans le répertoire res/values

```
<string name "nom_de_mon_application" > mon premier exemple Android </string>
```

Les ressources



- Utilisation des ressources
 - Ressources référencées par d'autres ressources
 - Les ressources définies peuvent être utilisées comme valeurs d'attributs dans d'autres ressources sous forme XML

```
<?xml version="1.0" encoding = "utf-8"?>

<TableLayout xmlns:android="http://schema.android.com/apk/res/android"

    android:layout_width = "fill_parent"
    android:layout_height="fill_parent"
    android:stretchColumns="1">

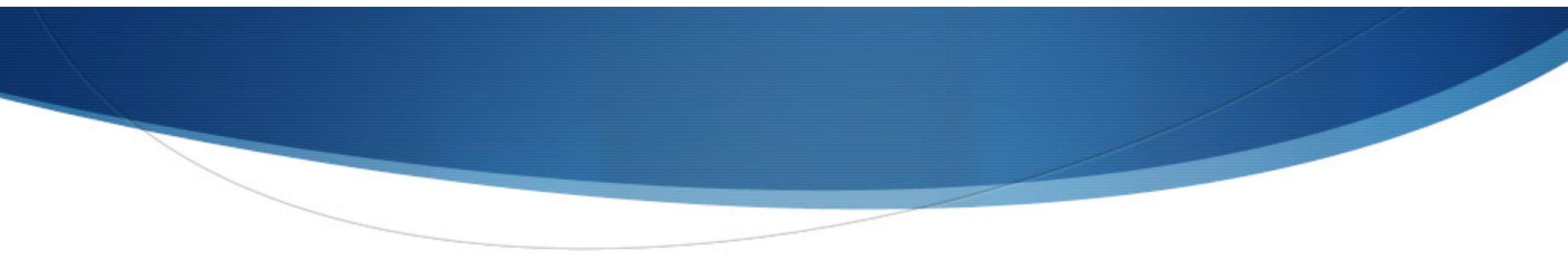
    <TableRow>
        <TextView
            android:text="@string/table_contenu_cellule_gauche" />
        <TextView
            android:text="@string/table_contenu_cellule_droite" />
    </TableRow>
</TableLayout>
```

Référencement
d'une autre
ressource

Autres composants

- **Service**
 - Est un composant qui fonctionne en tâche de fond, de manière invisible
 - Ses principales utilisations sont la mise à jour de sources de données ainsi que d'activités visibles et le déclenchement de notifications
- **Gadget**
 - Est un composant graphique qui s'installe sur le bureau Android
 - Exemples :
 - Le calendrier qui affiche de l'information
- **Fournisseur de contenu**
 - Permet de gérer et de partager des informations
 - Un même fournisseur permet d'accéder à des données au sein d'une application et entre applications





Les intents

Intent

- Un intent est une sous classe de **android.content.Intent**
- Le moyen de lier 2 composants
- Un événement (intent) est une intention à faire quelque chose à un composant Android
- Len intents pour communiquer entre Activités, Services.
- Attribut d'un intent

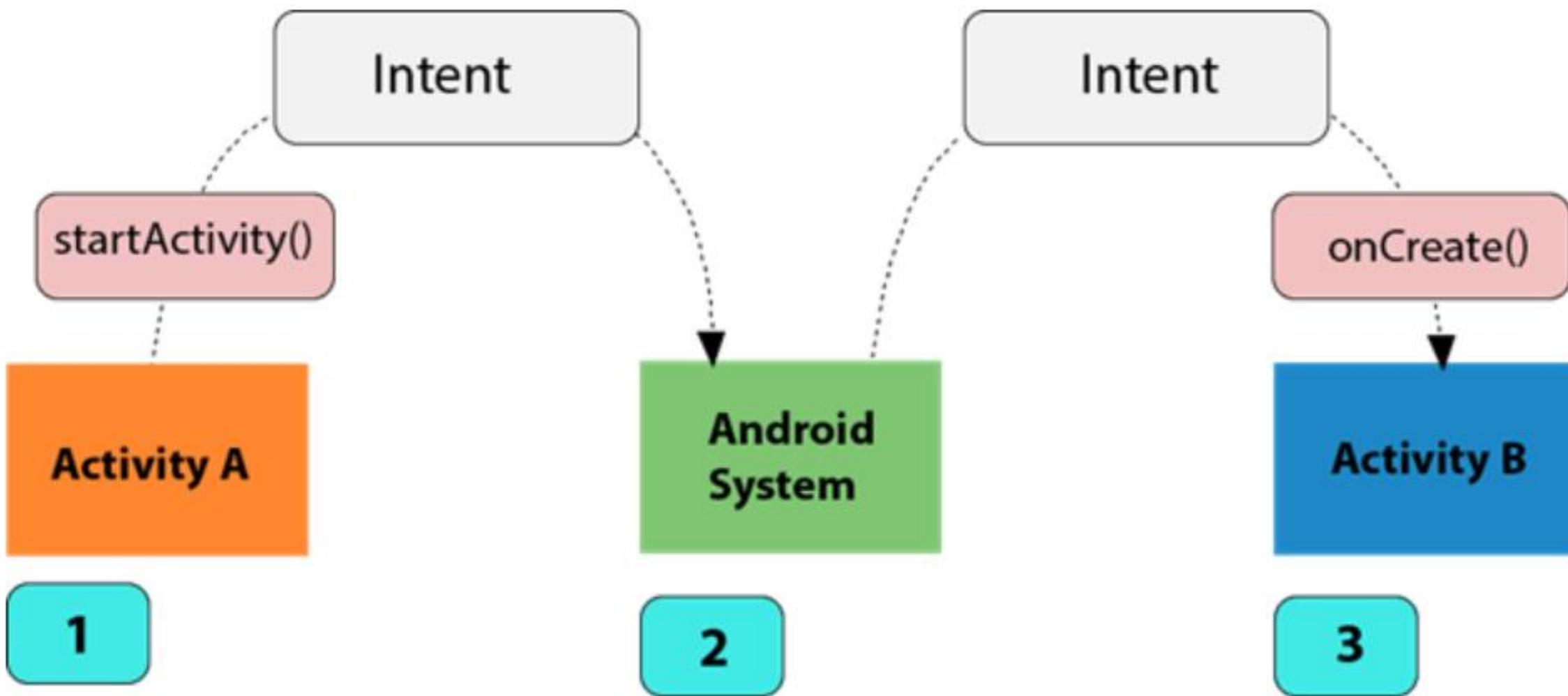
Nom - Action - Données (Data) - Catégorie (Category) – Type – Composant – Extras

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

Intent

- Intent explicite
 - Permet de lancer une activité particulière
- Intent implicite
 - Permet de demander au système de réaliser une action particulière sans viser une activité spécifique
 - Lors de l'utilisation, Android cherche parmi les activités qui se sont enregistrées comme capable de gérer cette demande (manifest)
 - Si plusieurs activités sont trouvées, il est automatiquement demandé à l'utilisateur de choisir (« ouvrir avec »)
- Exemple: affichage d'une page web

```
Uri webpage = Uri.parse("http://www.android.com");
Intent webIntent = new Intent(Intent.ACTION_VIEW, webpage);
startActivity(webIntent);
```



Intent implicite



Intent implicite

Activity_Main.xml

```
<Button  
    android:id="@+id	btnCamera"  
    android:layout_width="match_parent"  
    android:textColor="@android:color/holo_orange_light"  
    android:layout_height="55dp"  
    android:text="Camera" />
```

MainActivity.java

```
import android.app.Activity;  
import android.content.Intent;  
import android.net.Uri;  
import android.os.Bundle;  
import android.provider.MediaStore;  
import android.view.View;  
import android.widget.EditText;  
  
//extends Activity  
public class MainActivity extends Activity {  
  
    @Override  
    protected void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        setContentView(R.layout.activity_main);  
  
        //set click listener to camera button  
        findViewById(R.id.btnCamera).setOnClickListener(new View.OnClickListener() {  
  
            //perform camera open action  
            @Override  
            public void onClick(View v) {  
  
                Intent i = new Intent();  
                i.setAction(MediaStore.ACTION_VIDEO_CAPTURE);  
                startActivityForResult(i);  
            }  
        });  
    }  
}
```

Les services

Présentation

- Un service est un composant qui peut effectuer des opérations en arrière plan et ne fournit pas d'interface utilisateur
 - Par exemple: gérer les transactions réseau, lire de la musique

Peut prendre une longue période d'exécution

- Déclaration

```
<manifest ... >
...
<application ... >
    <service android:name=".ExampleService" />
    ...
</application>
</manifest>
```

Modes d'activation d'un service

- Deux modes pour lancer un service :
 - Démarré par un autre composant (mode Started)
 - Quand un composant d'application (par exemple, une activité) démarre ce service en appelant **startService()**
 - Lié à d'autres composants (mode Bound) **Binder IPC**
 - Quand un composant d'application se lie à ce service en appelant **bindService()**

Créer une activité qui nous permet (via l'intent) de soit de lancer un nouveau service, soit de relier (binder) un service qui est déjà existant.

Modes d'activation d'un service

- Démarré (Started)
 - Une fois démarré, un service peut s'exécuter en arrière-plan indéfiniment, même si le composant qui a commencé est détruit
 - Un service démarré effectue une opération unique et ne retourne pas de résultat à l'appelant
 - Par exemple,
 - télécharger un fichier sur le réseau
 - Lorsque l'opération est terminée, le service doit s'arrêter

Lancer un service

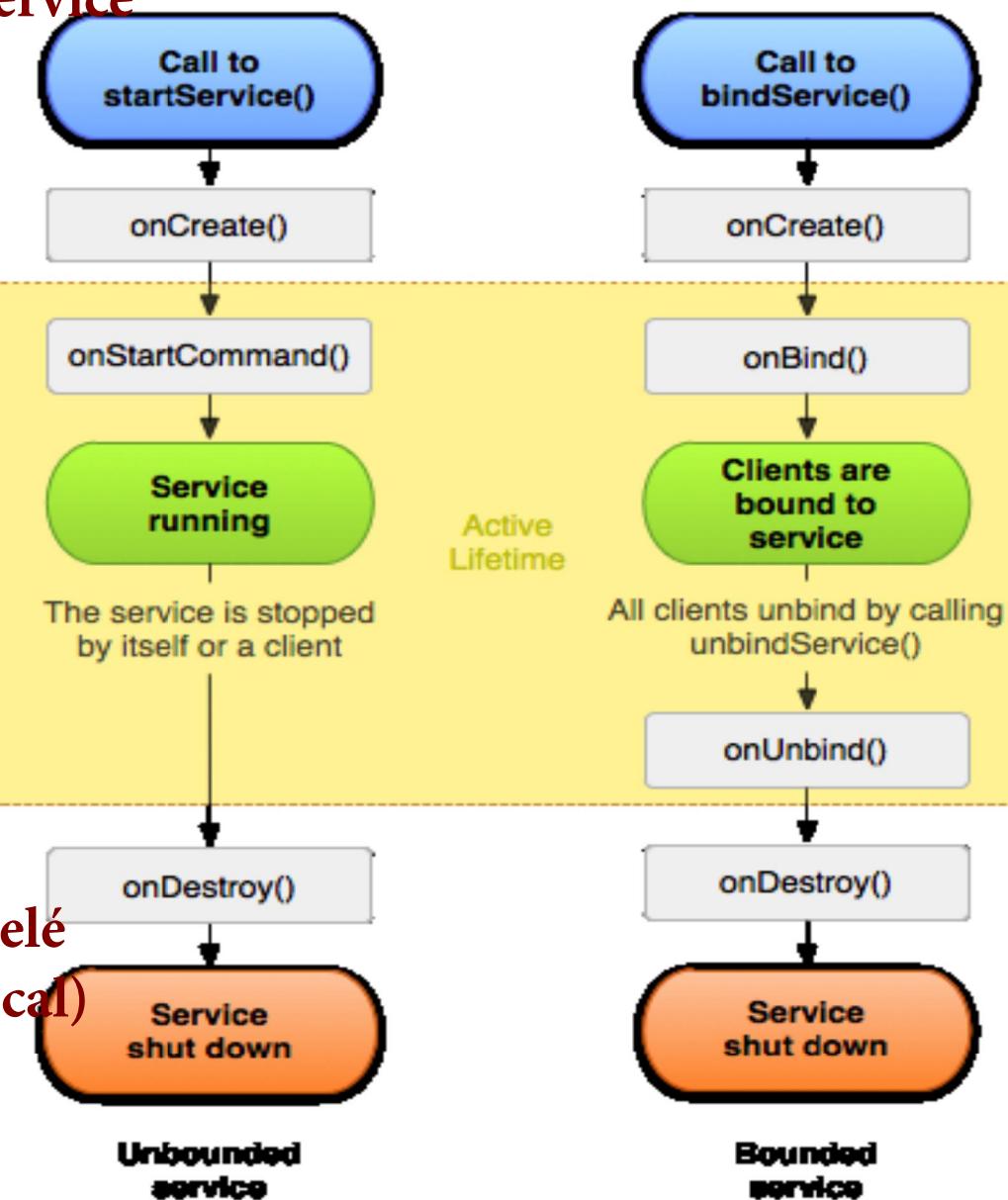
- Démarrer un service en passant un Intent à startService ()
 - Android appelle la méthode **onStartCommand ()** du service et lui transmet l'intention
- ```
Intent intent = new Intent(this, HelloService.class);
startService(intent);
```
- Par exemple, une activité doit enregistrer des données dans une base de données en ligne. L'activité peut démarrer un service et lui transmettre les données à sauvegarder en lui passant une intention via **startService ()**
  - Un service peut définir des filtres qui permettent de capter les appels avec intentions (intents) implicites

# Modes d'activation d'un service

- Lié (Bound)
  - Un service lié propose une interface client-serveur qui permet aux composants d'interagir avec le service
    - Envoyer des demandes
    - Obtenir des résultats
    - Faire à travers des processus de communication inter-processus (IPC).
  - Un service lié ne fonctionne que tant qu'un ou plusieurs composants d'autres applications sont liés à celui-ci
    - Quand tous les liaisons sont détruites, le service est détruit

# Cycle de vie d'un service

on fait appel à un service



service qui est déjà lancé

Service rejoint/bind par une activité

pour se déconnecter  
Bounded service

Unbounded service

service qui a été appelé par une activité (local)

service s'éteint

# Création de service

- Deux classes peuvent être étendues pour créer un service démarré:
  - Service
    - Il s'agit de la classe de base pour tous les services
  - IntentService
    - Il s'agit d'une sous-classe de Service qui utilise un même thread pour traiter toutes les demandes, une à la fois

# Création de service

- **onStartCommand ()**
  - Le système appelle cette méthode lorsqu'un autre composant tel une activité, active un service en appelant **startService()**
  - Une fois que cette méthode lancée, le service est démarré et peut fonctionner en tâche de fond indéfiniment
    - C'est au développeur d'arrêter le service lorsque son travail est fait, en appelant **stopSelf ()** ou **StopService ()**

# Création de service

- onBind ()
  - Le système appelle cette méthode quand un autre composant veut se lier avec le service
    - Par exemple, pour effectuer RPC en appelant **bindService()**
  - Utilisation d'une interface par les clients pour communiquer avec le service, en renvoyant un IBinder.
  - Si un service est appelé par **bindService ()**, une fois dissocié de tous ses clients, le système le détruit

# Création de service

- **Étendre la classe IntentService**

- Permet de traiter les demandes du service dans l'ordre de leur arrivée
  - Le IntentService effectue les opérations suivantes:
    - Crée un thread par défaut qui exécute toutes les intentions passées à **onStartCommand()**
    - Crée une file d'attente des demandes vers le service
    - Arrête le service après que toutes les demandes ont été traitées
    - Fournit une implémentation par défaut de **onBind ()** qui renvoie la valeur null.
    - Fournit une implémentation par défaut de onStartCommand ()
  - Le développeur doit implémenter **onHandleIntent()**

```
public class HelloIntentService extends IntentService {

 /**
 * A constructor is required, and must call the super IntentService(String)
 * constructor with a name for the worker thread.
 */
 public HelloIntentService() {
 super("HelloIntentService");
 }
 /**
 * The IntentService calls this method from the default worker thread with the intent that *started
 * the service. When this method returns, IntentService stops the service, as appropriate.
 */
 @Override
 protected void onHandleIntent(Intent intent) {
 // Normally we would do some work here, like download a file.
 // For our sample, we just sleep for 5 seconds.

 long endTime = System.currentTimeMillis() + 5*1000;
 while (System.currentTimeMillis() < endTime) {
 synchronized (this) {
 try {
 wait(endTime - System.currentTimeMillis());
 } catch (Exception e) {
 }
 }
 }
 }
}
```

# Création de service

- Étendre la classe de service
  - Permet au service d'effectuer un multi-threading des tâches
  - Possibilité de traiter plusieurs demandes en même temps
    - Créer un nouveau thread pour chaque requête et l'exécuter immédiatement, au lieu d'attendre la fin de la requête précédente
      - Chaque appel à **onStartCommand()** peut être traité à part

```
public class HelloService extends Service {
 private Looper mServiceLooper;
 private ServiceHandler mServiceHandler;

 // Handler that receives messages from the thread
 private final class ServiceHandler extends Handler {
 public ServiceHandler(Looper looper) {
 super(looper);
 }
 @Override
 public void handleMessage(Message msg) { //... }
 }

 @Override
 public void onCreate() { //... }

 @Override
 public int onStartCommand(Intent intent, int flags, int startId)
 { //... }

 @Override
 public IBinder onBind(Intent intent) { //... }

 @Override
 public void onDestroy() { //... }
}
```

# Création d'un service

```
@Override
 public void handleMessage(Message msg) {
 // Normally we would do some work here, like download a file.
 // For our sample, we just sleep for 5 seconds.

 long endTime = System.currentTimeMillis() + 5*1000;
 while (System.currentTimeMillis() < endTime) {
 synchronized (this) {
 try {
 wait(endTime - System.currentTimeMillis());
 } catch (Exception e) {
 }
 }
 }

 // Stop the service using the startId, so that we don't stop
 // the service in the middle of handling another job

 stopSelf(msg.arg1);
 }
}
```

# Création d'un service

```
@Override
 public void onCreate() {
 // Start up the thread running the service. Note that we create a
 // separate thread because the service normally runs in the
 // process's main thread, which we don't want to block.
 // We also make it background priority so CPU-intensive
 // work will not disrupt our UI.

 HandlerThread thread = new
 HandlerThread("ServiceStartArguments",
 Process.THREAD_PRIORITY_BACKGROUND);
 thread.start();

 // Get the HandlerThread's Looper and use it for our Handler

 mServiceLooper = thread.getLooper();
 mServiceHandler = new ServiceHandler(mServiceLooper);
 }
```

# Création d'un service

```
@Override

public int onStartCommand(Intent intent, int flags, int startId) {
 Toast.makeText(this, "service starting", Toast.LENGTH_SHORT).show();

 // For each start request, send a message to start a job and
 // deliver the start ID so we know which request we're stopping
 // when we finish the job

 Message msg = mServiceHandler.obtainMessage();
 msg.arg1 = startId;
 mServiceHandler.sendMessage(msg);

 // If we get killed, after returning from here, restart
 return START_STICKY;
}
```

# Création d'un service

```
@Override
public IBinder onBind(Intent intent) {
 // We don't provide binding, so return null
 return null;
}

@Override
public void onDestroy() {
 Toast.makeText(this, "service done",
 Toast.LENGTH_SHORT).show();
}
```