

Technologies et développement Mobile



Pr. Slimane Bah, ing. PhD

Génie Informatique option TI & IQL

Semaine 9

Séance 1

Organisation du cours

- **Cours :** Mardi 10h à 12h
 Vendredi 14h à 16h
- **Site du cours:**
www.emi.ac.ma/bah/cours/dev_mobile.html

Organisation du cours

- Cours $\approx 14\text{h}$
- TPs $\approx 8\text{h}$
- Évaluations :
 - ✓ Participation
 - ✓ Un examen en fin de semestre
 - ✗ Contrôle **surprise** en classe
 - ✓ TPs sous Android Studio

Question d'entrée de jeu

- Qui a déjà programmé une application mobile ?



A propos du cours

■ Vos connaissances ?

- Qu'est ce qu'une application mobile ?
- Quels outils pour développer une App. mobile ?
- Existe-t-il des différences avec une application « normale »
- Qu'est ce qu'Android ?

Objectifs du cours

- Connaître les approches du dev. mobile
 - Comprendre les **concepts** de la programmation sous Android
 - Se familiariser avec l'environnement de développement Android
- ⇒ Être capable de programmer correctement des applications mobiles

Plan

- Introduction Dév. Mobile
- Architecture et principes d'Android
- Système des Permissions
- Activités et cycle de vie
- Threads
- Persistance de données

Devoir

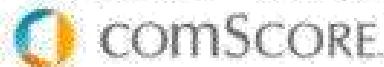
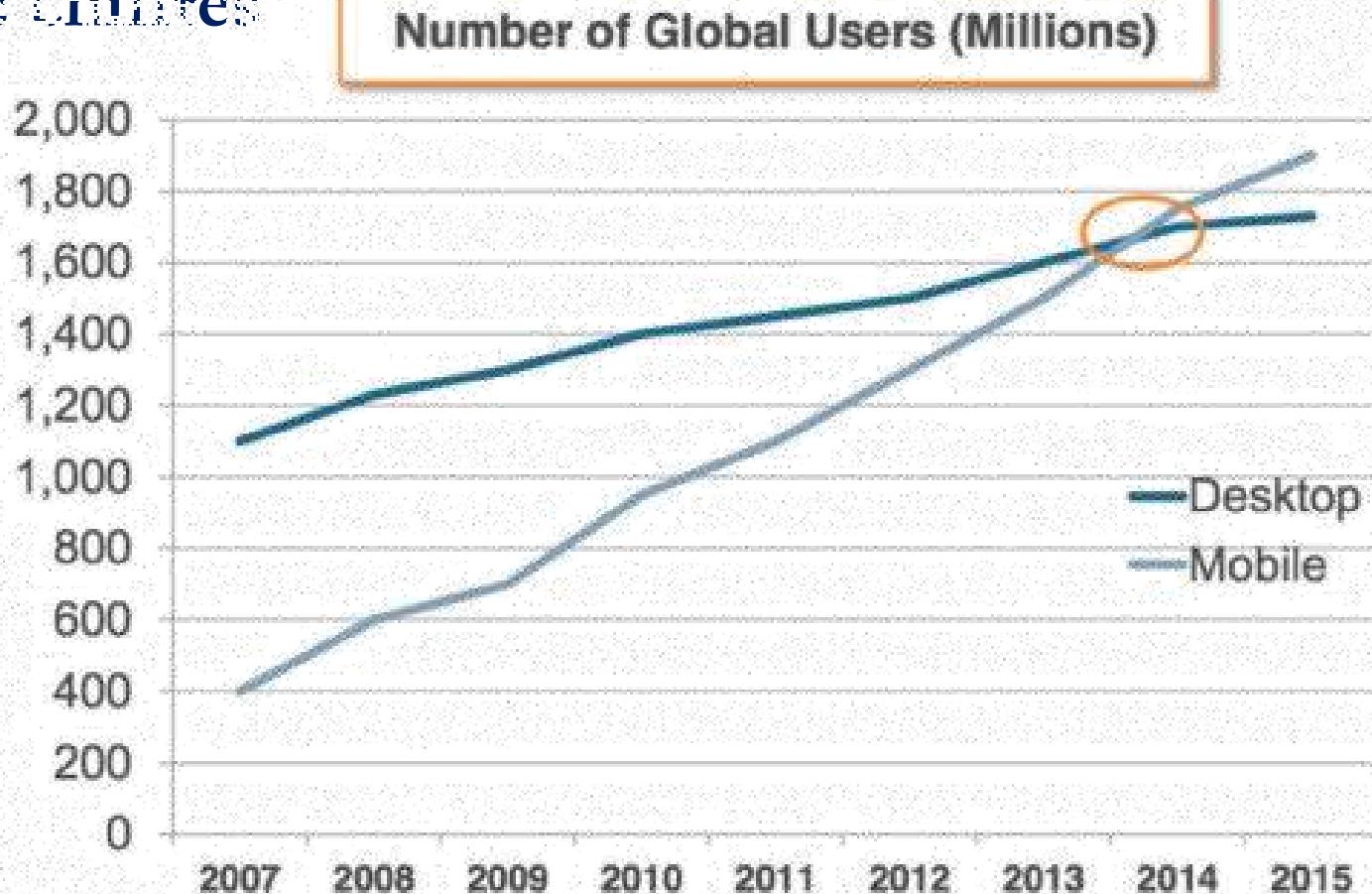
■ **Android studio ready : setup, update and run**

1. Installer Android studio
2. Installer et mettre à jour des packages
3. Démarrer studio, créer un émulateur et le tester

~~Suivre le guide (à titre indicatif) sur le site~~

Introduction

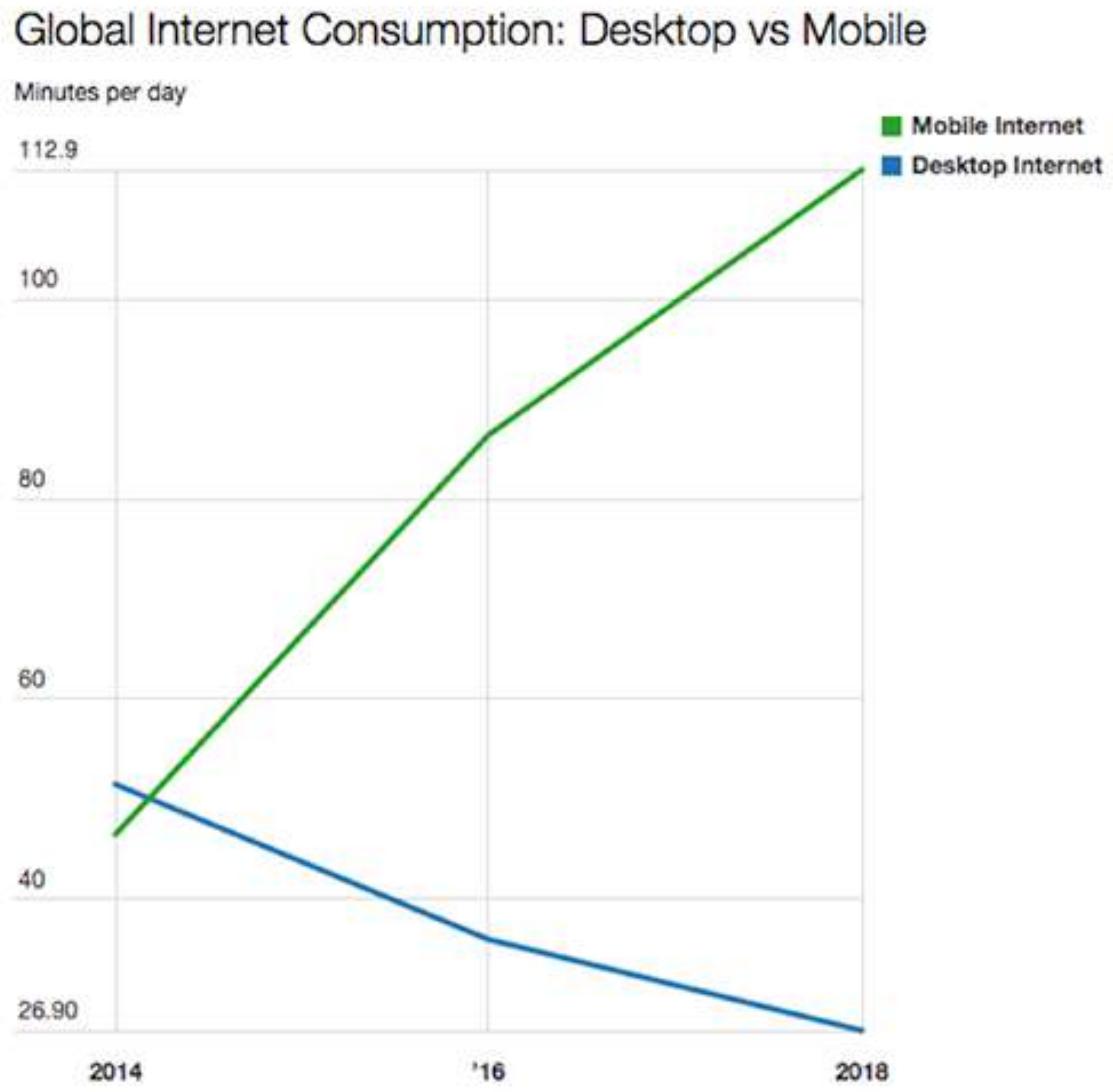
■ Quelques chiffres



Source: Morgan Stanley Research

Introduction

■ Quelques chiffres



Introduction

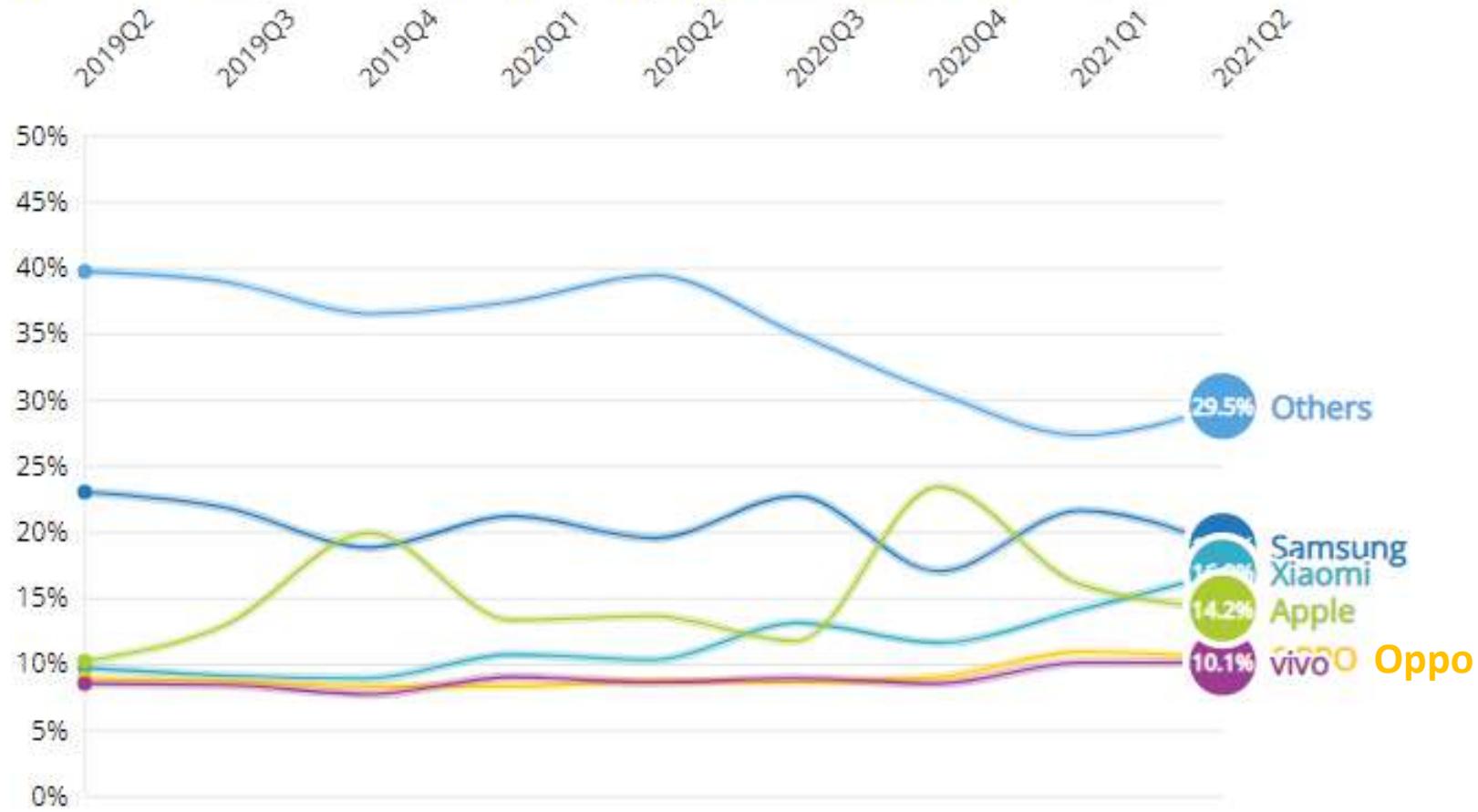
■ Quelques chiffres

- En 2018 les clients ont dépensé sur les stores : **97\$ Milliards**
- Le nombre moyen d'App par smartphone : **80**
- Nombre d'utilisateurs de smartphone en 2020 : **5.7 Milliards**
(5 Milliards en 2017)

Introduction

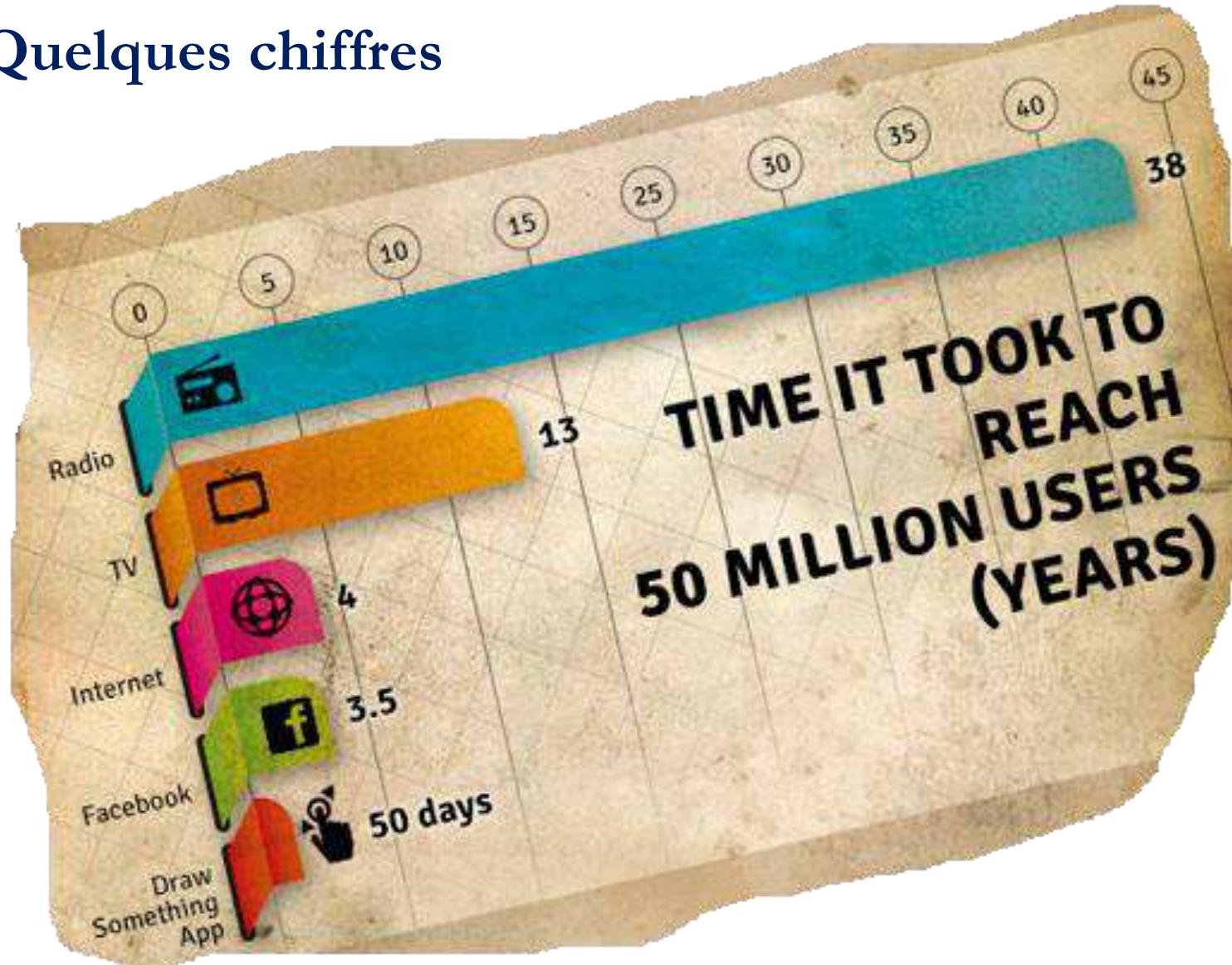
■ Quelques chiffres

Worldwide Top 5 Smartphone Company Unit Market Share (%)



Introduction

■ Quelques chiffres



Introduction

2021 This Is What Happens In An Internet Minute



Introduction

■ Histoire mobile



Graham Bell est le premier à obtenir la patente pour un téléphone électrique



Alfred Gross obtient la patente pour le Talkie-Walkie/pager/



Martin Cooper invente le 1^{er} téléphone portable commercial (Motorola)



Iphone
Android



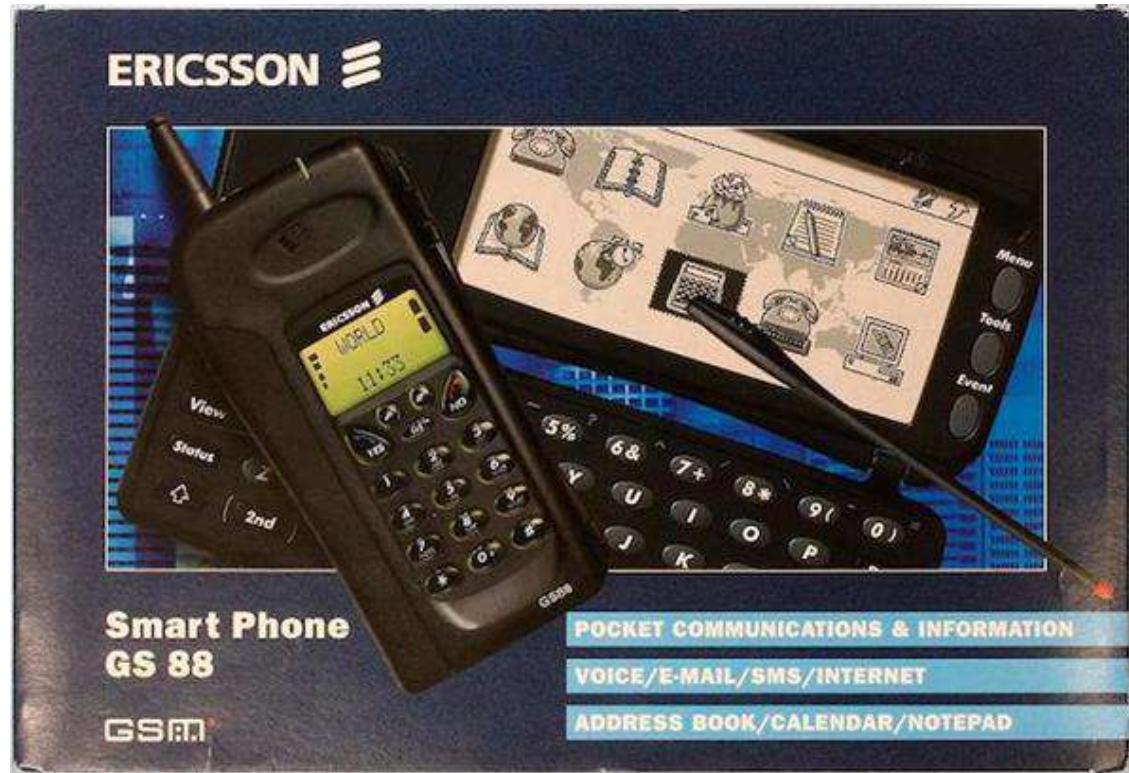
Introduction

■ Histoire des smartphones

Le terme Smartphone apparaît pour la 1^{ère} fois

1997 : Ericsson GS 88

Avec stylus



Introduction

■ Histoire des smartphones

- 1999 : premier mobile de RIM le Blackberry 850



Blackberry 850

- Les débuts des années 2000 : PalmOS, Pocket PC (windows), Symbian deviennent très populaire



Pocket PC 2002 : PDA + phone

Introduction

- Plusieurs modèles ont vu le jour
- Orienté entreprise
- De 2004 à 2007 la demande pour les smartphones a explosé
- Les logiciels et applications deviennent de plus en plus ergonomiques et disponibles



La révolution mobile



Introduction

■ La révolution mobile ?

Y a pas longtemps...	Aujourd'hui	Demain
<ul style="list-style-type: none">- Téléphone- Pager- PDA- Laptop- Lecteur Mp3- Modem filaire- Appareil photo- Peu d'Internet	<ul style="list-style-type: none">- Smartphone- Laptop (peut-être)- TV	<p>- Smartphone</p> <p>(appareil photo, télécommande, Play Station, TV, Livres, Argent \$, GPS, Laptop , Professeur, ...)</p>

Introduction

■ La révolution mobile ?

1. Évolution des réseaux mobiles
2. Évolution du hardware
3. Évolution des utilisateurs
4. Systèmes d'exploitation

Introduction

■ La révolution mobile ?

1. Évolution des technologies mobile

- Évolution des réseaux mobiles télécoms à travers plusieurs générations
- Maturité de ces technologies + performance



Introduction

■ La révolution mobile ?

2. Évolution du hardware (Smaller ? and smarter)

- Maturité des technologies sans fils
- Des appareils « consistants » en terme de ressources
(connexion, processeurs, mémoire, ..etc)
- Des appareils bien équipés en termes de gadgets
(capteurs, accéléromètres, gps, caméras,...)
- Divers types d'appareils

Introduction

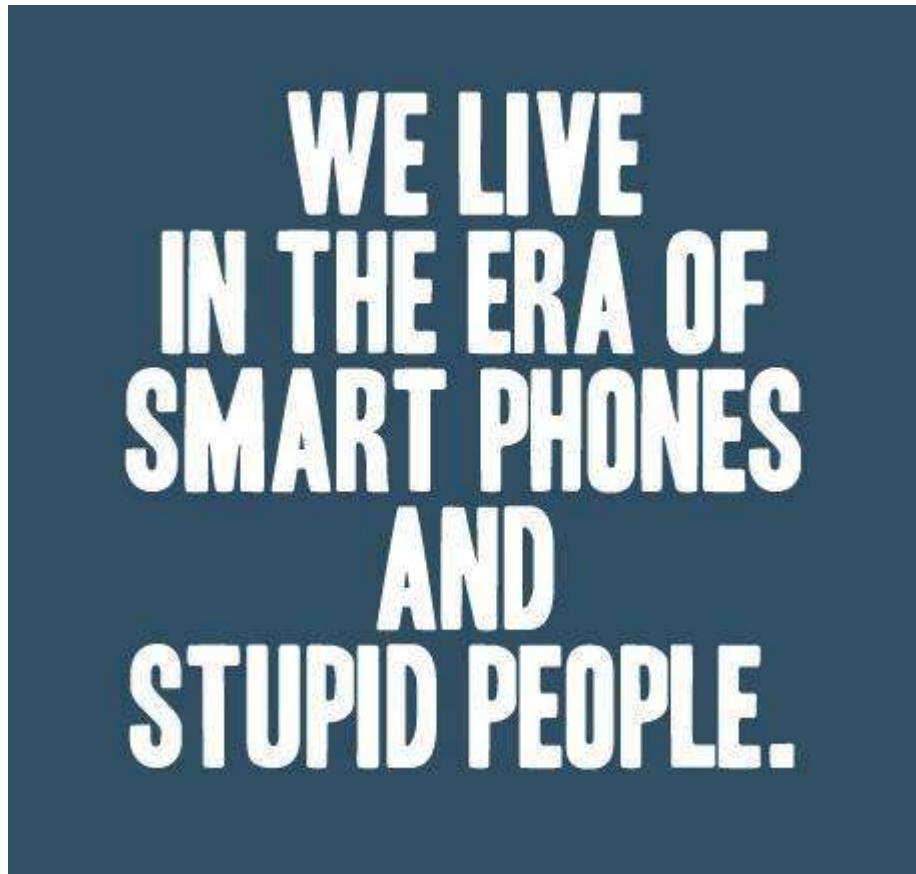
■ La révolution mobile ?

3. Évolution des utilisateurs

- Utilisateurs de plus en plus avertis
- Utilisateurs de plus en plus exigeants
- Utilisateurs de plus en plus mobile et de plus en plus dépendants

Introduction

■ La révolution mobile ? 3. Évolution des utilisateurs



**"Hold on. I just want to put on Facebook
that I'm actually in the process of being
mugged, then you can have my BlackBerry!"**

Introduction

■ La révolution mobile ? 3. Évolution des utilisateurs



Introduction

■ La révolution mobile ?

3. Évolution des utilisateurs

13 ways smartphones make us dumber

7. We don't live in the moment



Introduction

■ La révolution mobile ? 3. Évolution des utilisateurs



Introduction

■ La révolution mobile ?

4. Systèmes d'exploitation

Des SE qui sont capables de profiter de tout le potentiel technologique et matériel des smartphones

⇒ **Multiplication des SE et d'appareils mobiles**

Introduction

■ La révolution mobile ?



Introduction

■ La révolution mobile ?



Introduction

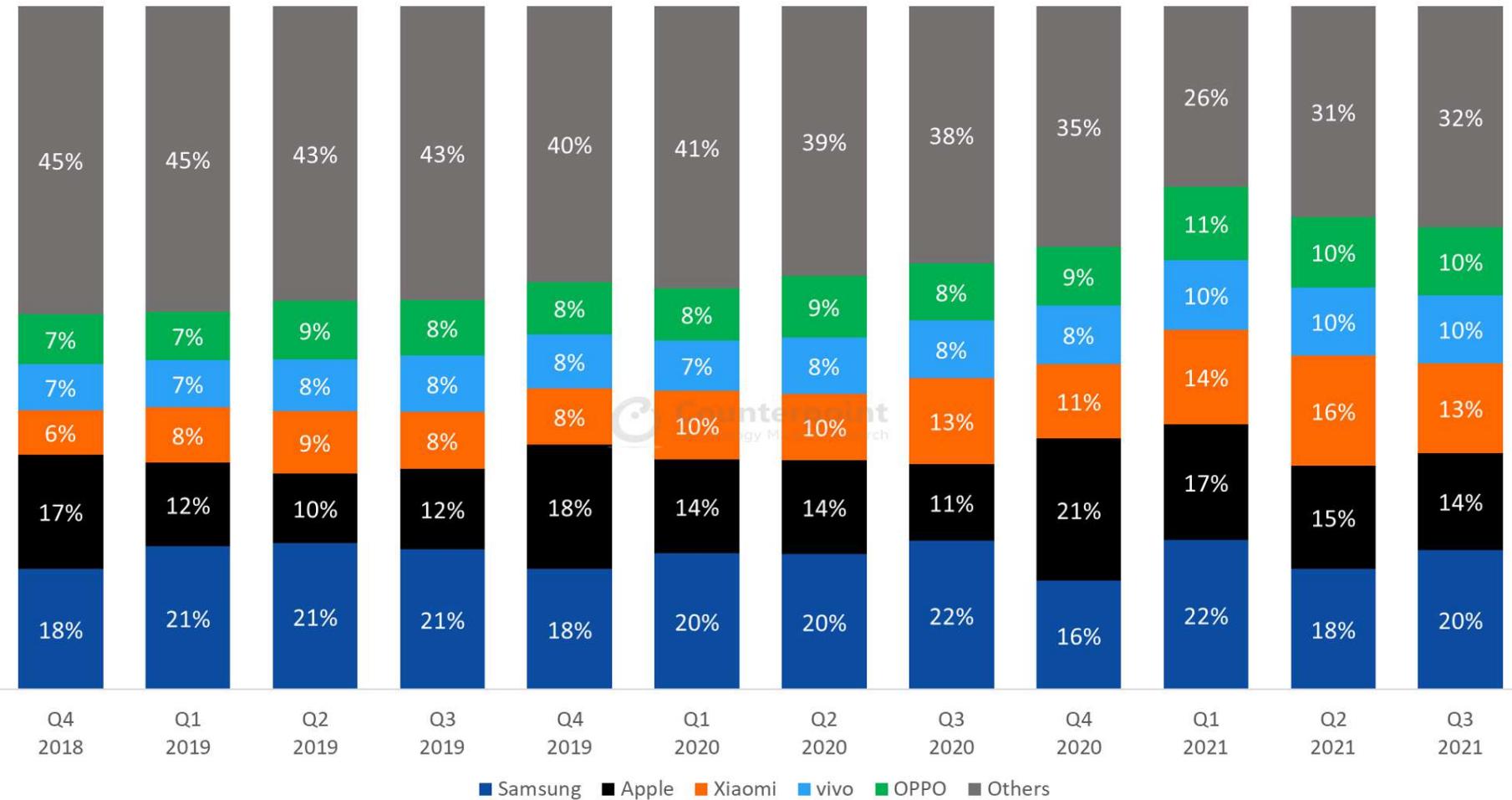
■ La révolution mobile ?

Plateforme	
Windows mobile	Visual studio .NET (C#, C++, VB .NET)
Symbian (open source)	Multi-langage: Python, C/C++, java ME, .NET
PalmOS (pour PDA)	C/C++
WebOS (pour smart TV - LG)	C/C++, HTML5
RIM Blackberry	Java
Iphone	Objective C, C, C++, Swift
Android	Java (un peu de C, C++), Kotlin
Windows Phone	.NET

Introduction

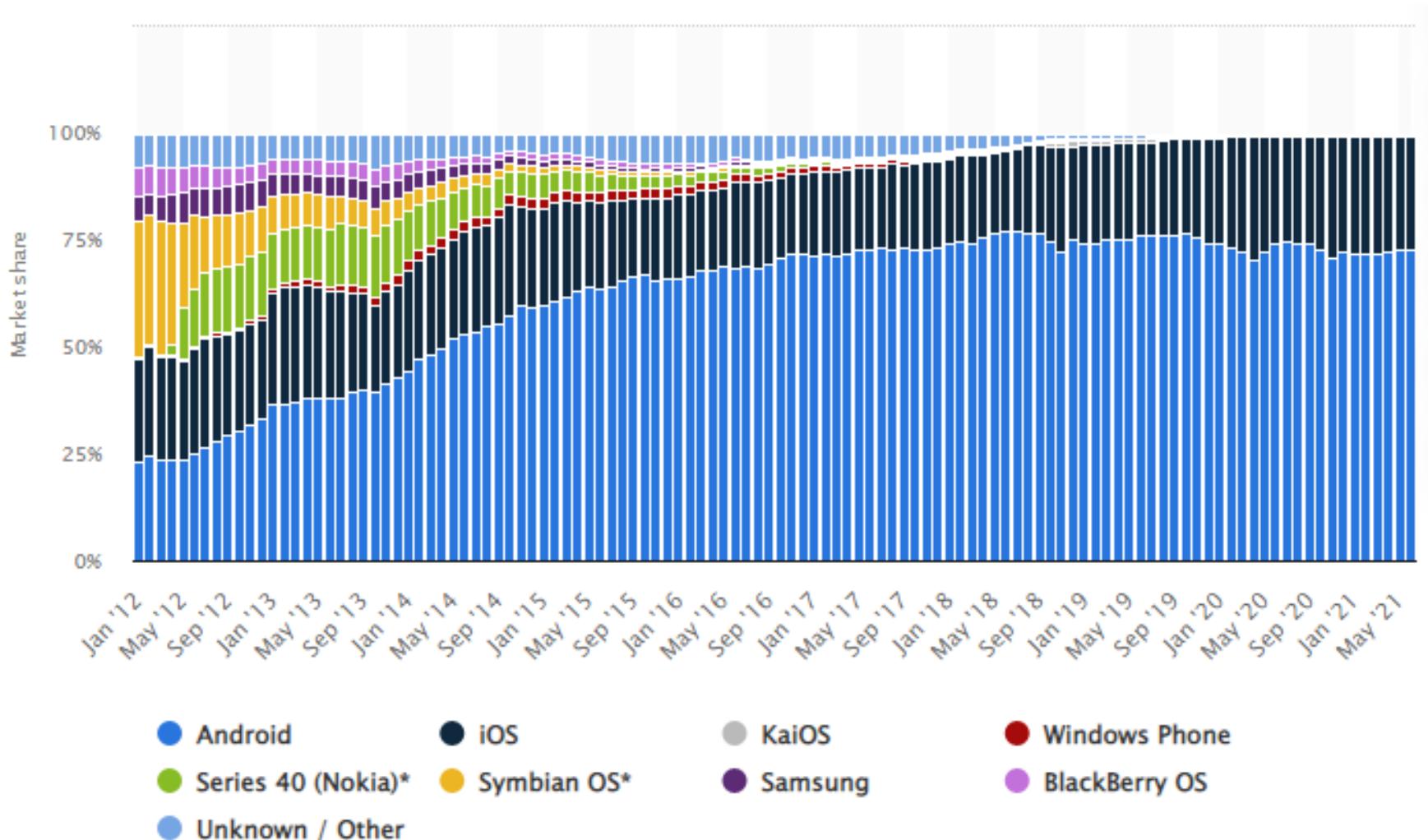
■ La révolution mobile ?

Global Smartphone Market Share (Q4 2018 - Q3 2021)



Introduction

■ La révolution mobile ?



Introduction

■ Spécificités du développement mobile

Écrire des applications pour des Smart Phones doit prendre en considération un nombre de contraintes :

Hardware :

- Petite taille : écran, clavier (s'il existe)
- Écran tactile : gros doigts et multitouches
- Vitesse processeur
- Taille mémoire
- Batterie
- Gadgets disponibles dépendent du constructeur

Introduction

■ Spécificités du développement mobile

- Magnetometer
- Barometer
- Thermometer
- Air humidity
- Pedometer
- Heart rate
- Harmful radiation (japan only)



Introduction

■ Spécificités du développement mobile

Software :

- Le langage de programmation dépend souvent du constructeur
- Les Apps sont conscientes qu'elles sont dans un téléphone
- L'App. ne doit pas empêcher le fonctionnement **normal** du téléphone
- L'application doit être compatible avec l'OS du téléphone
- Tenir compte des différentes versions de l'OS
- La surconsommation des ressources (CPU, RAM) fera cracher l'App
- L'application doit tenir compte de l'énergie (batterie)
- Coexistence avec d'autres applications

Introduction

■ Spécificités du développement mobile

	Desktop (java)	Mobile (android)
Création des interfaces utilisateurs	Exclusivement programmatique (SWING, JFace, AWT, SWT, etc.)	Programmative OU Déclarative (layout XML)
Communications entre applications	Pas de protocole dédié (en pratique on utilise les fonctions réseaux de Java)	Protocole d'intent permettant d'envoyer un message à une application sur la même machine
Cycle de vie des applications	L'OS hôte ne peut pas arrêter une application en dehors de la levée d'une exception (ex. fuite mémoire)	Android peut arrêter une application en pause pour libérer de l'espace mémoire, c'est au développeur de définir les éléments à sauvegarder en cas d'arrêt de l'application
Accès aux ressources (fichiers, périphériques...)	L'application a les mêmes droits de lecture et d'écriture que l'utilisateur qui la lance (ex. root) l'accès aux périphériques (webcam, micro...) dépend de librairie tierces	Les droits d'accès de l'application sur les fichiers sont déclaratifs (Manifest.xml), Android fournit nativement les librairies pour l'ensemble des périphériques
Philosophie de développement	Programmative et monolithique, la communication entre deux applications Java différente est l'exception	Déclarative et modulaire, l'absence de communication entre les applications est l'exception

Technologies et développement Mobile



Pr. Slimane Bah, ing. PhD

Génie Informatique option TI & IQL

Semaine 10.1

Séance 2

Introduction

■ Spécificités du développement mobile

	Desktop (java)	Mobile (android)
Création des interfaces utilisateurs	Exclusivement programmatique (SWING, JFace, AWT, SWT, etc.)	Programmative OU Déclarative (layout XML)
Communications entre applications	Pas de protocole dédié (en pratique on utilise les fonctions réseaux de Java)	Protocole d'intent permettant d'envoyer un message à une application sur la même machine
Cycle de vie des applications	L'OS hôte ne peut pas arrêter une application en dehors de la levée d'une exception (ex. fuite mémoire)	Android peut arrêter une application en pause pour libérer de l'espace mémoire, c'est au développeur de définir les éléments à sauvegarder en cas d'arrêt de l'application
Accès aux ressources (fichiers, périphériques...)	L'application a les mêmes droits de lecture et d'écriture que l'utilisateur qui la lance (ex. root) l'accès aux périphériques (webcam, micro...) dépend de librairie tierces	Les droits d'accès de l'application sur les fichiers sont déclaratifs (Manifest.xml), Android fournit nativement les bibliothèques pour l'ensemble des périphériques
Philosophie de développement	Programmative et monolithique, la communication entre deux applications Java différentes est l'exception	Déclarative et modulaire, l'absence de communication entre les applications est l'exception

Les approches du développement mobile

Approches dev. mobile

■ Les approches du développement mobile

- Il existe 3 façons d'écrire une application mobile :

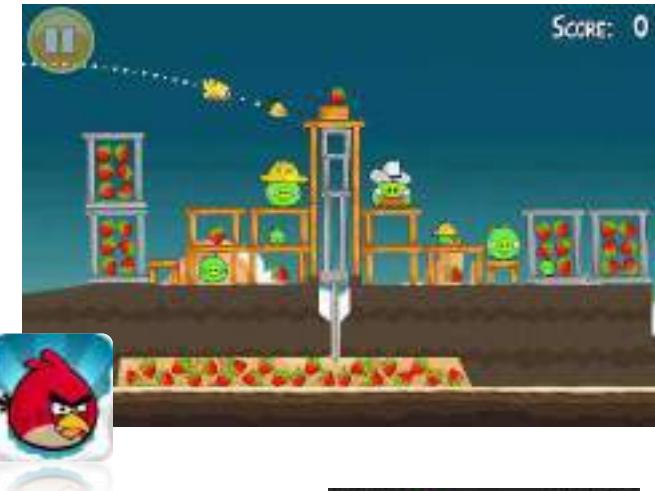


Approches dev. mobile

■ Les approches du développement mobile

1- Native Apps

- Écrite dans un **langage** lié au mobile
- Exécutable **binaire**, explicitement **téléchargée** et **stockée** dans le système de fichiers du mobile
- **Distribuée** via un store d'applications ou une entreprise de distribution
- Exécutée **directement** par l'OS
- Utilise explicitement les **API** de l'OS

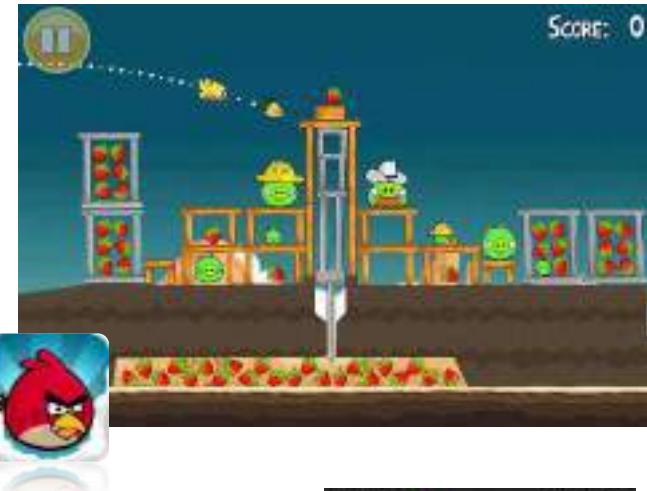


Approches dev. mobile

■ Les approches du développement mobile

1- Native Apps

- Écrite dans un **langage** lié à l'OS
- Exécutable **binaire**, chargé directement dans la mémoire de la machine
- Téléchargée et stockée dans la mémoire de l'appareil sous forme de fichiers du mobile

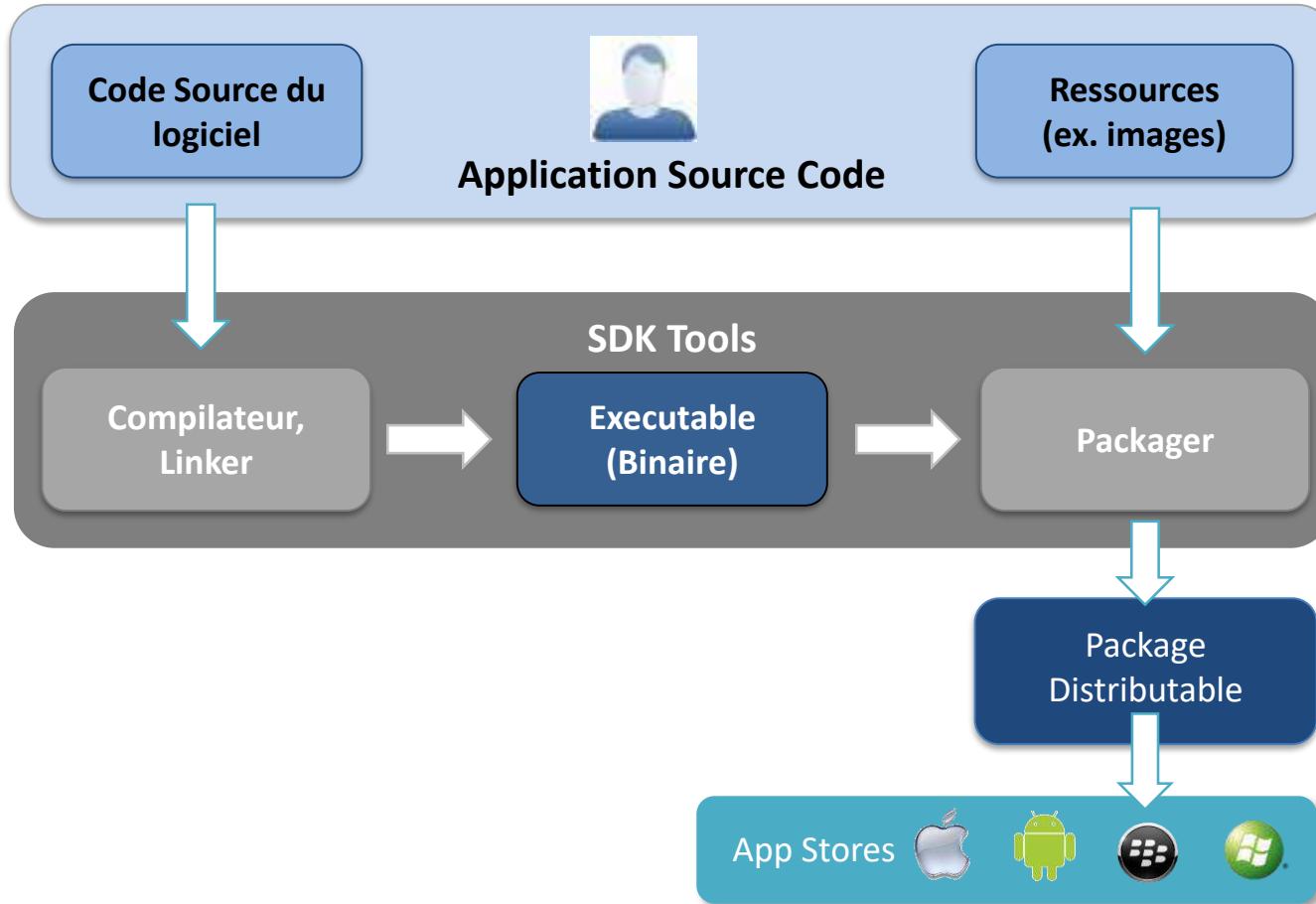


- Utilise les capacités d'applications ou de fonction
- Utilise les capacités de l'OS
- Utilise expérimentent les API de l'OS



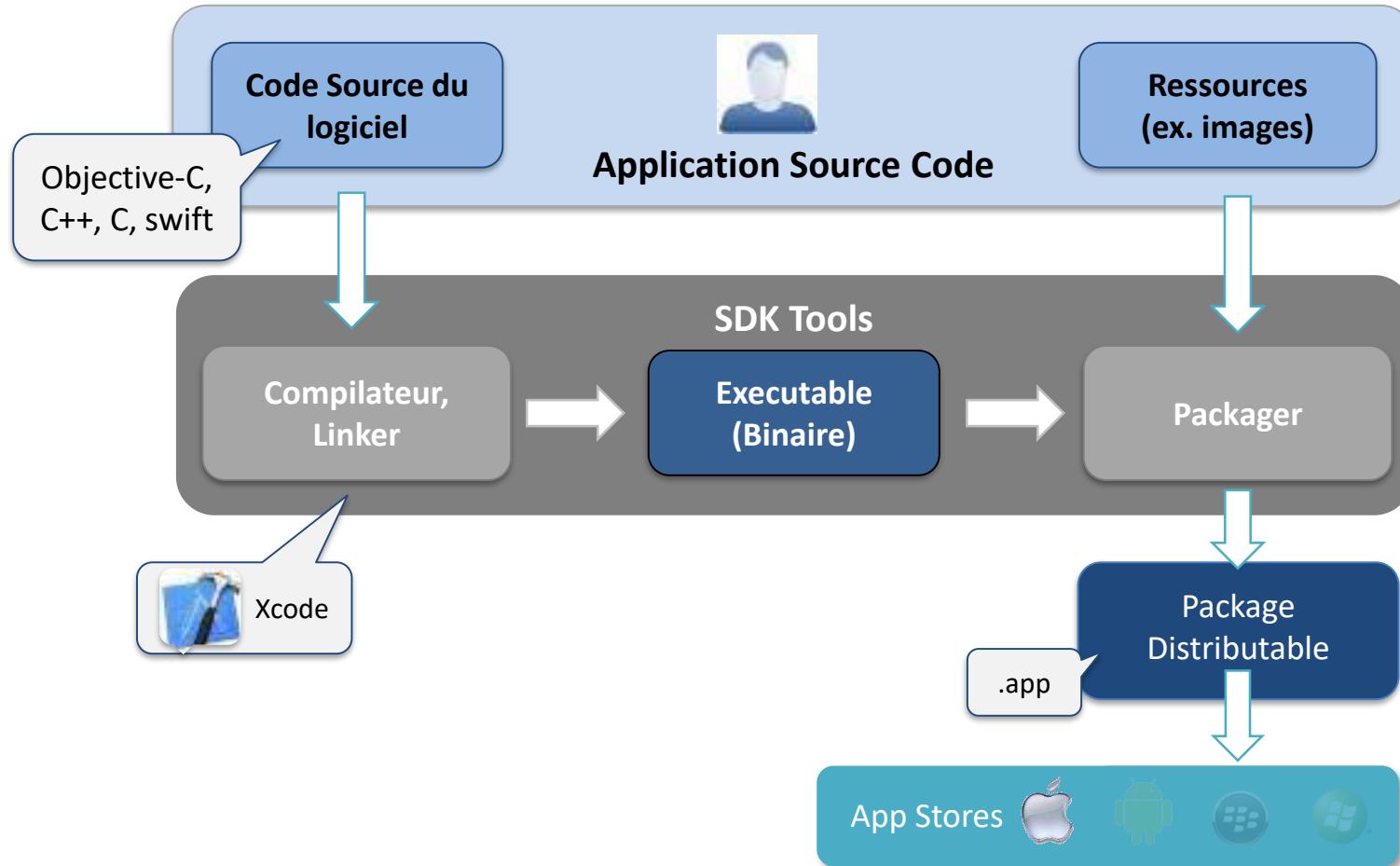
Approches dev. mobile

■ Les approches du développement mobile



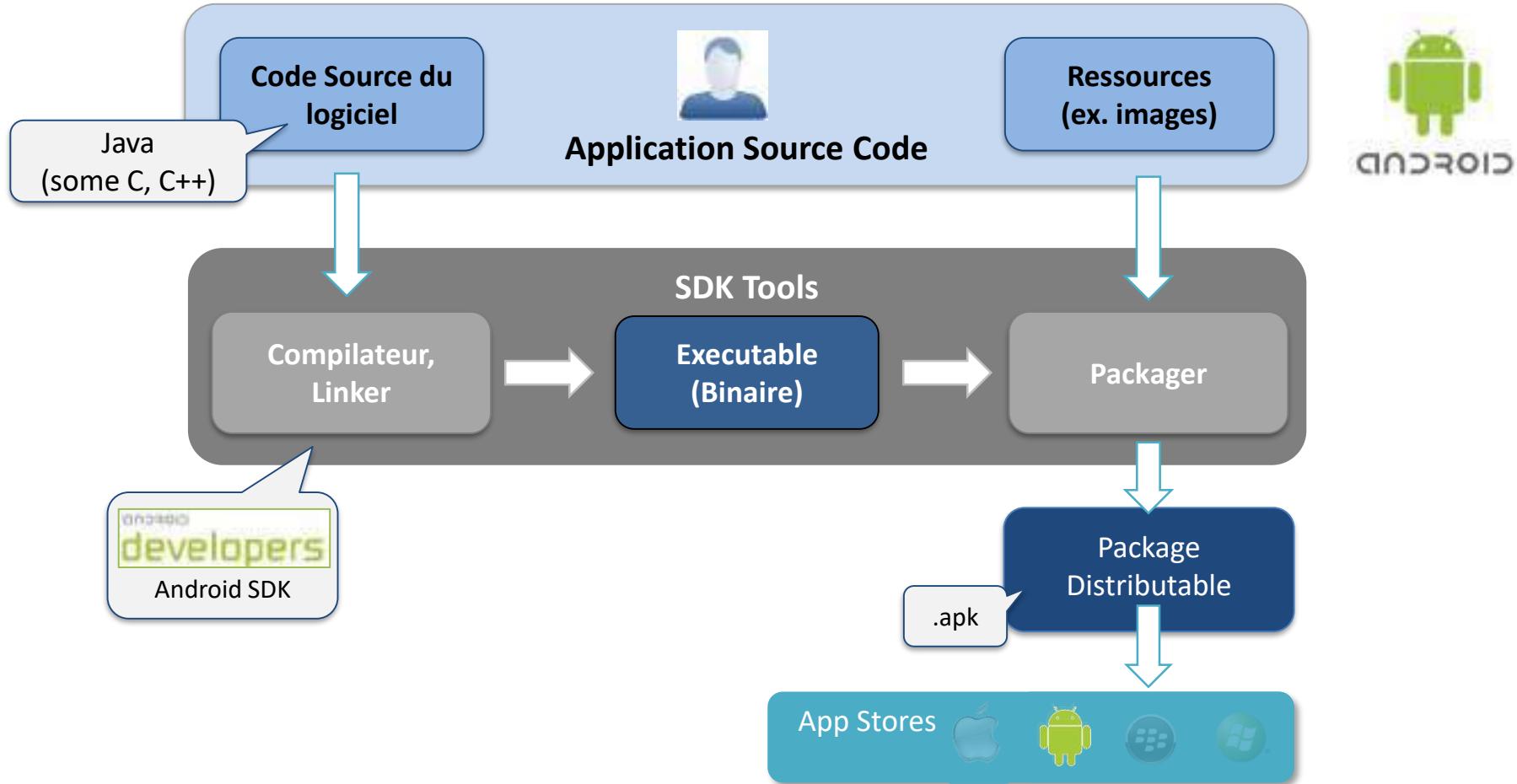
Approches dev. mobile

■ Les approches du développement mobile



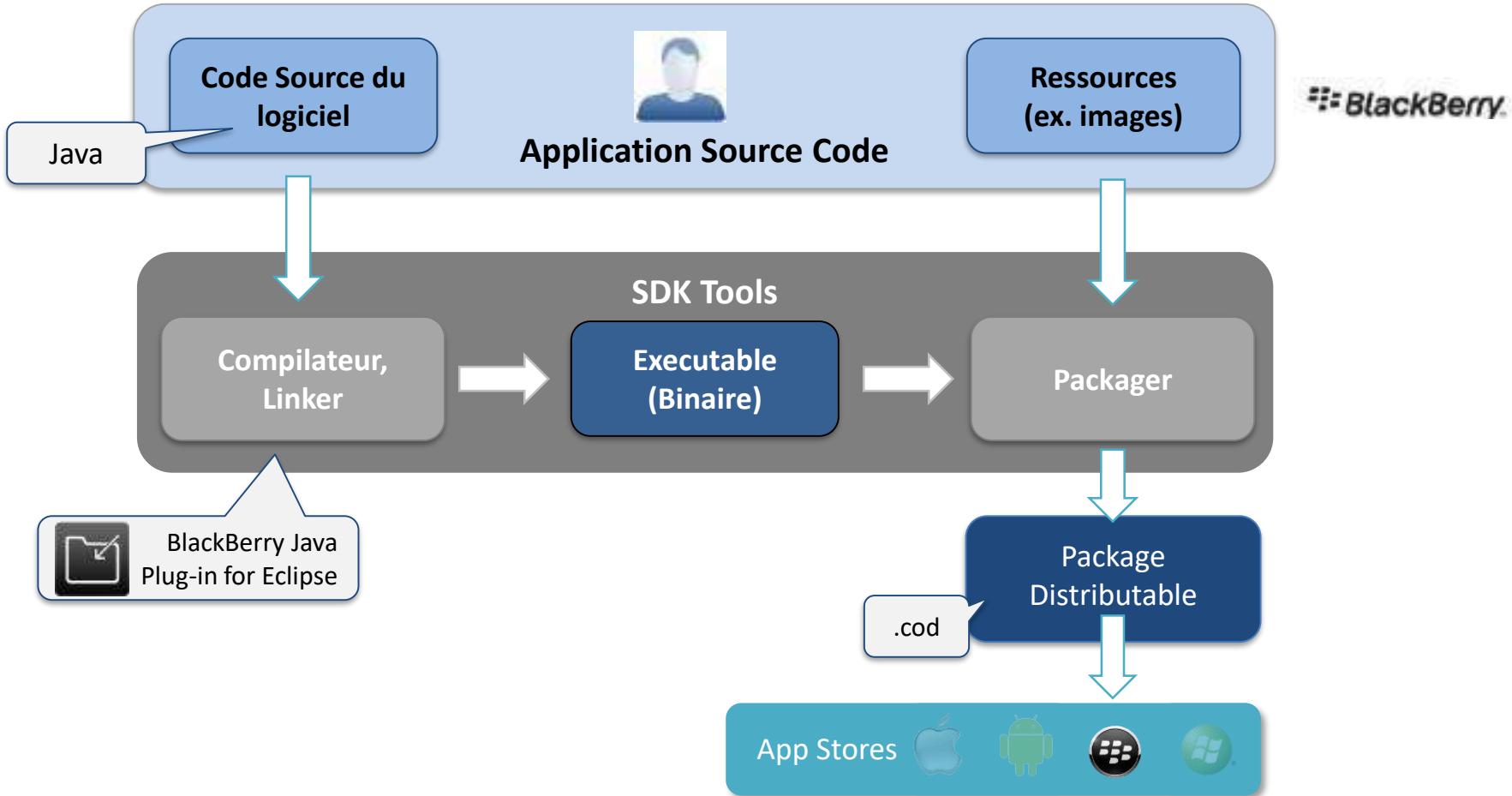
Approches dev. mobile

■ Les approches du développement mobile



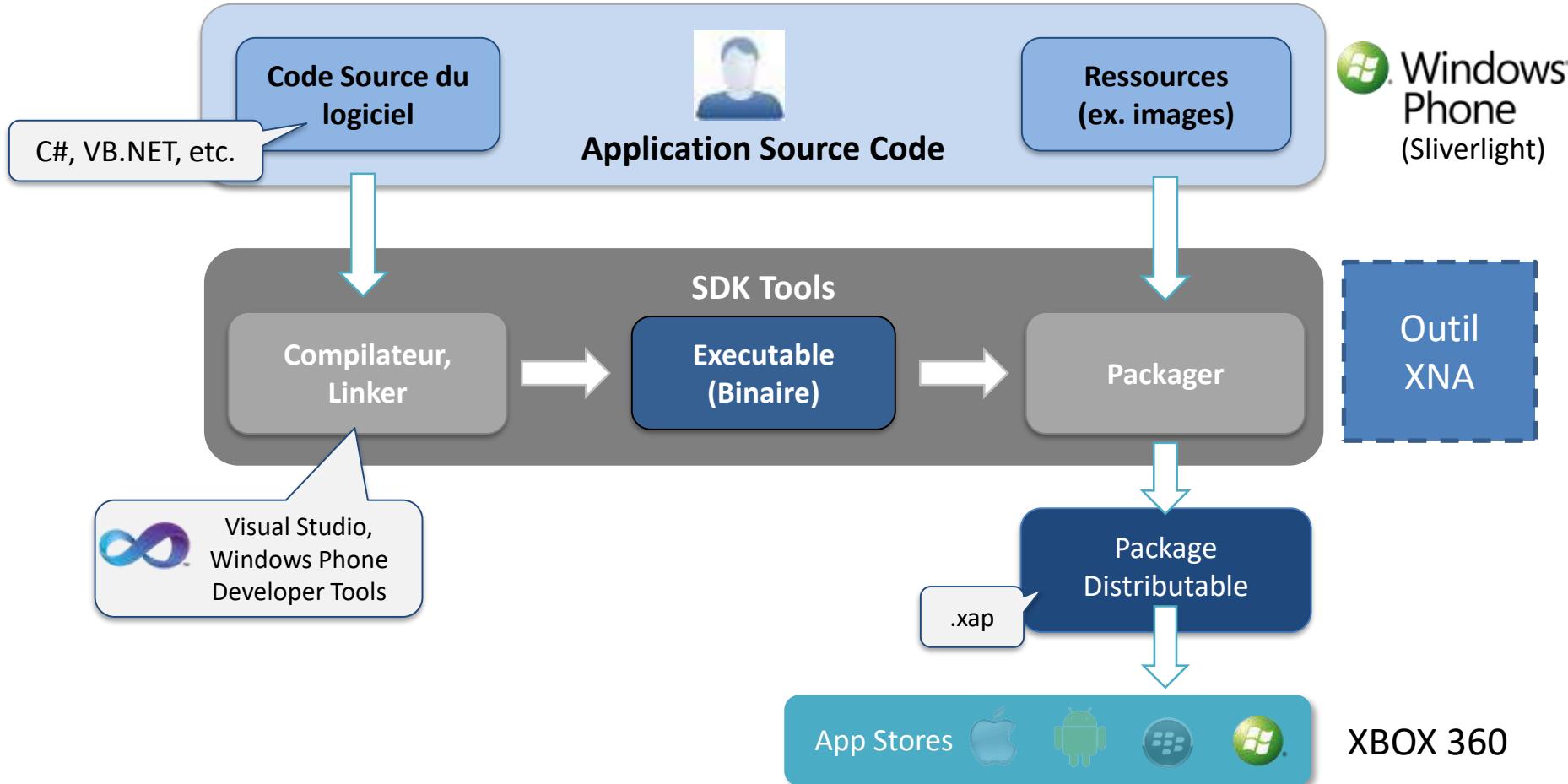
Approches dev. mobile

■ Les approches du développement mobile



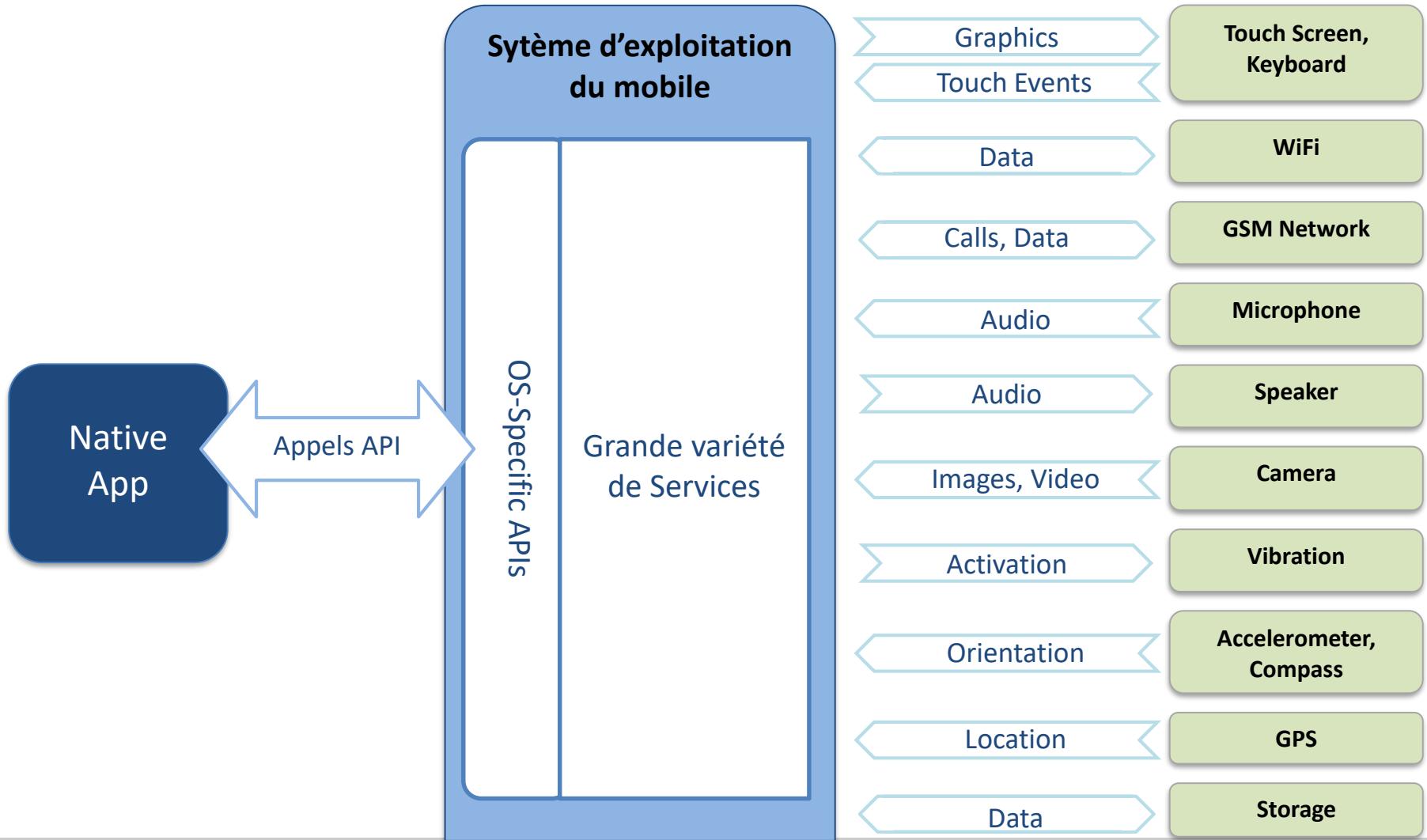
Approches dev. mobile

■ Les approches du développement mobile



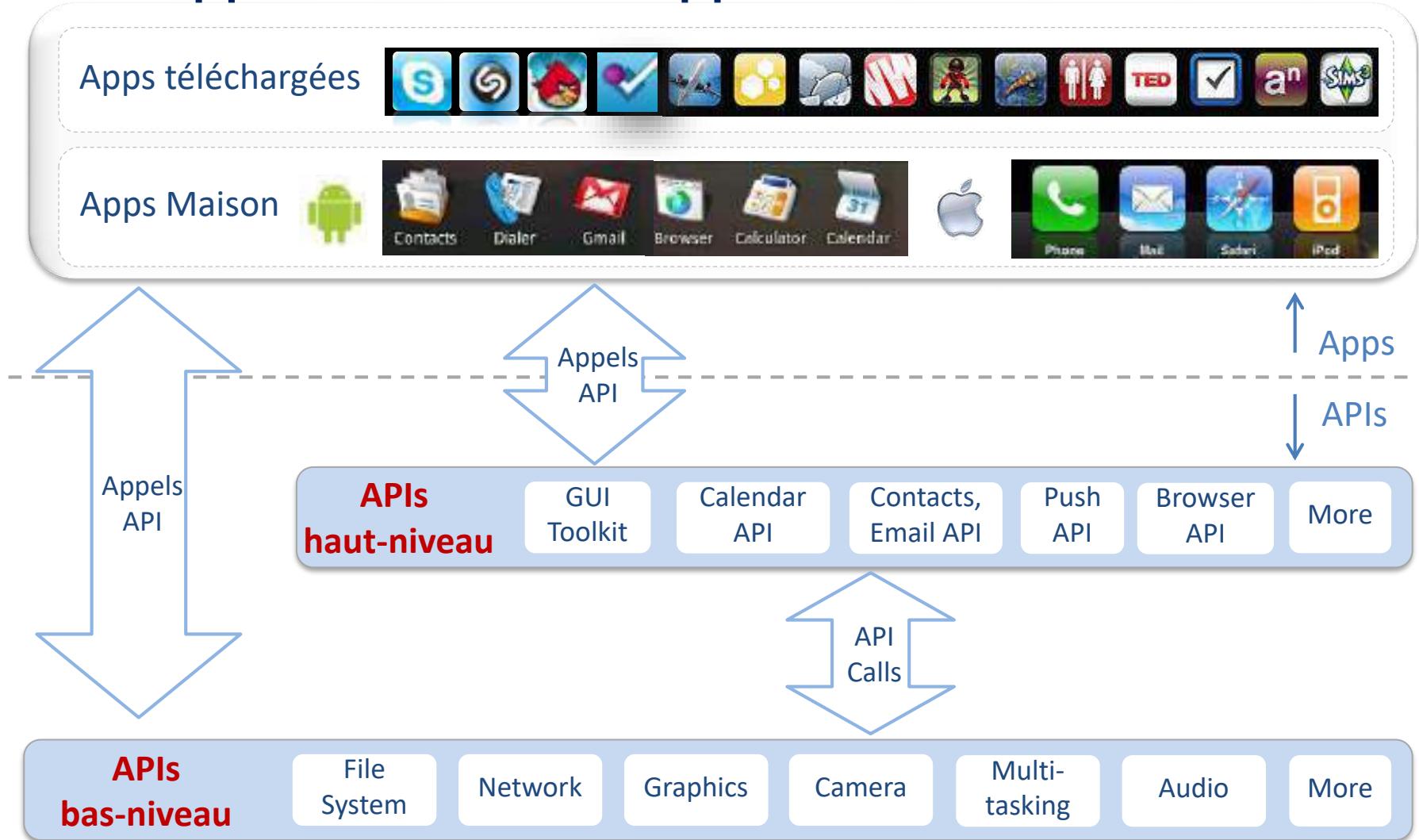
Approches dev. mobile

■ Les approches du développement mobile



Approches dev. mobile

■ Les approches du développement mobile



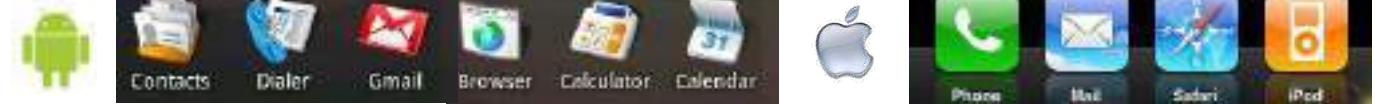
Approches dev. mobile

■ Les approches du développement mobile

Apps téléchargées

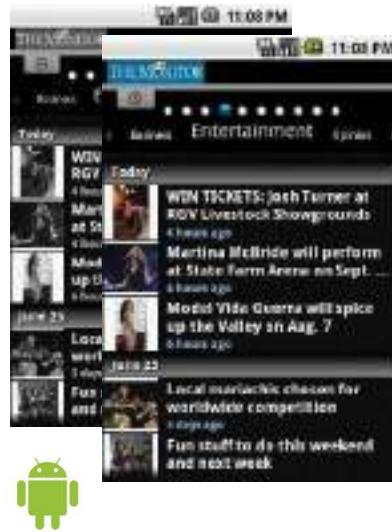


Apps Maison



Appels
API

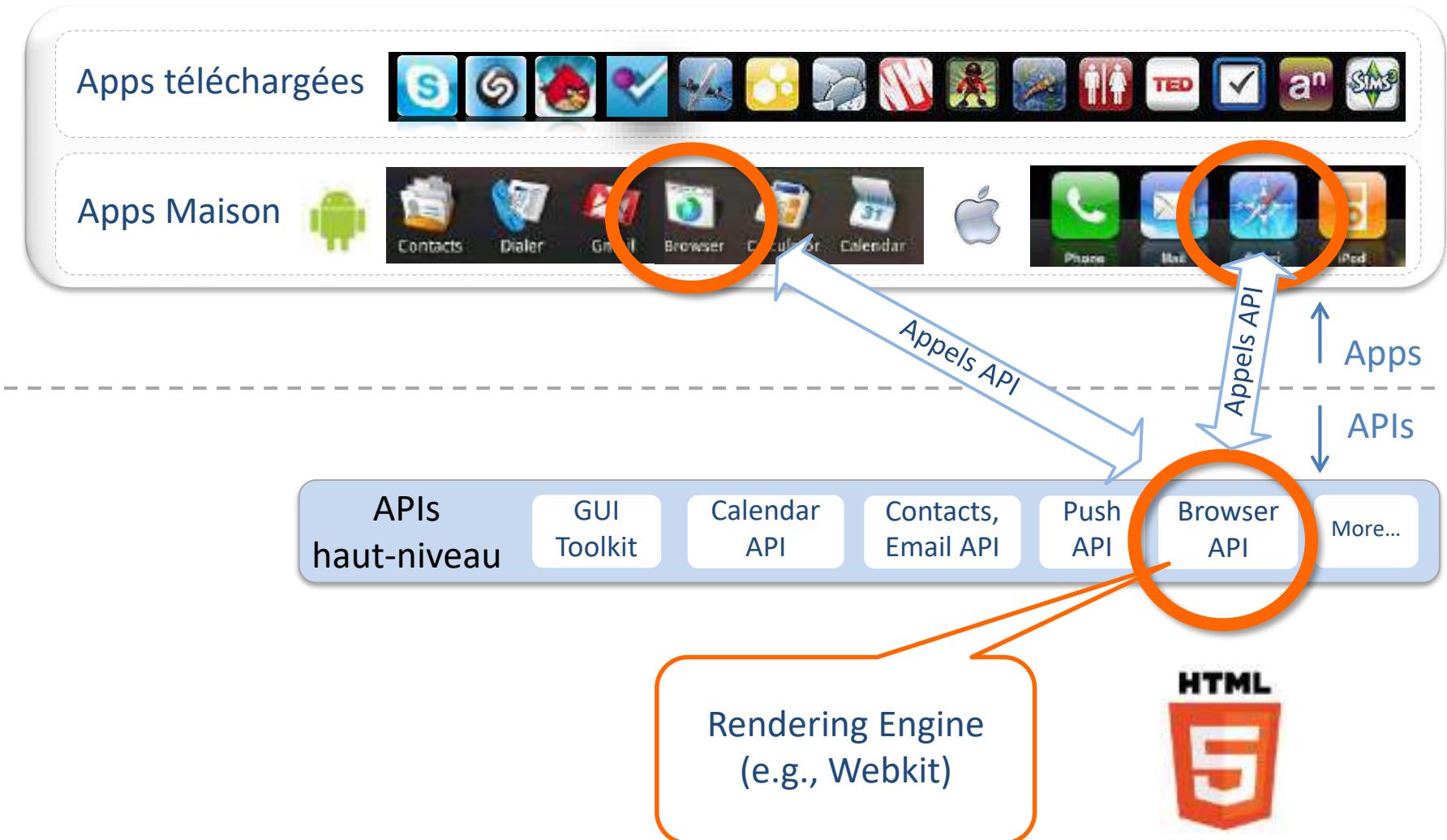
GUI
Toolkit



Apps
APIs

Approches dev. mobile

■ Les approches du développement mobile



Approches dev. mobile

■ Les approches du développement mobile

2- Apps web mobile



Google, Wikipedia:
sites web optimisés
pour mobile



Dremel:
s'exécute avec
QR-Codes



YouTube: Web App



2B: Web App

Approches dev. mobile

■ Les approches du développement mobile



Sites web mobile purs

- Accès via navigateur
- UI navigable, statique
- Look & feel Générique
- Rendering sur le serveur
- Besoin de connexion

... Zone Grise...

Apps web mobile pures

- Installée et exécutée
- UI interactive
- Optimisé pour tactile
- Rendering chez le Client
- Disponible offline

Approches dev. mobile

urise | new.dedy.tv/2b-webapp



2B

for iPhone/iPod Touch

From jotting down measurements at IKEA, to quick tic tac toe duels over who's paying for lunch, 2B lets your iPhone double as an impromptu scribble pad whenever you're in the mood for analog. Primary features are:

- Switch between pencil/eraser modes
- Supports portrait or landscape orientation, image rotates along to maintain its original direction
- Saves one scribble locally, automatically loaded when you start up the app. You can always take a screenshot of your scribble for more permanent storage on your device's camera roll (hold down the device's Home button then press the Power button)
- Works offline and in fullscreen mode for maximum available drawing space

Free

Get 2B - FREE

Visit the above link on your iPhone or iPod Touch. You can install 2B for offline use by tapping + then adding to your Homescreen.



Approches dev. mobile

	Safari iOS	Android Browser	Samsung Internet	Google Chrome	Amazon Silk	BlackBerry Browser	Nokia Browser	Internet Explorer	Opera Mobile	Opera mini	Firefox
File API <small>W3C API</small> Opening local files through input type	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
FileSystem API <small>W3C API</small> Virtual FileSystem for persistent storage			✓	✓	✓	✓	✓		✓	14+	
HTML Media Capture <small>W3C API</small> Taking pictures, record video and audio from an input file type	✓	✓	✓	✓		✓	✓		✓	14+	✓
Web Speech API <small>W3C API</small> Speech Recognition and Synthesizer	✓			✓					14+		11+
HomeScreen Webapp <small>NO API</small> Add icon to the home screen with fullscreen support	✓	meta tags		✓	32+		✓	Anna+	✓	14+	✓

Approches dev. mobile

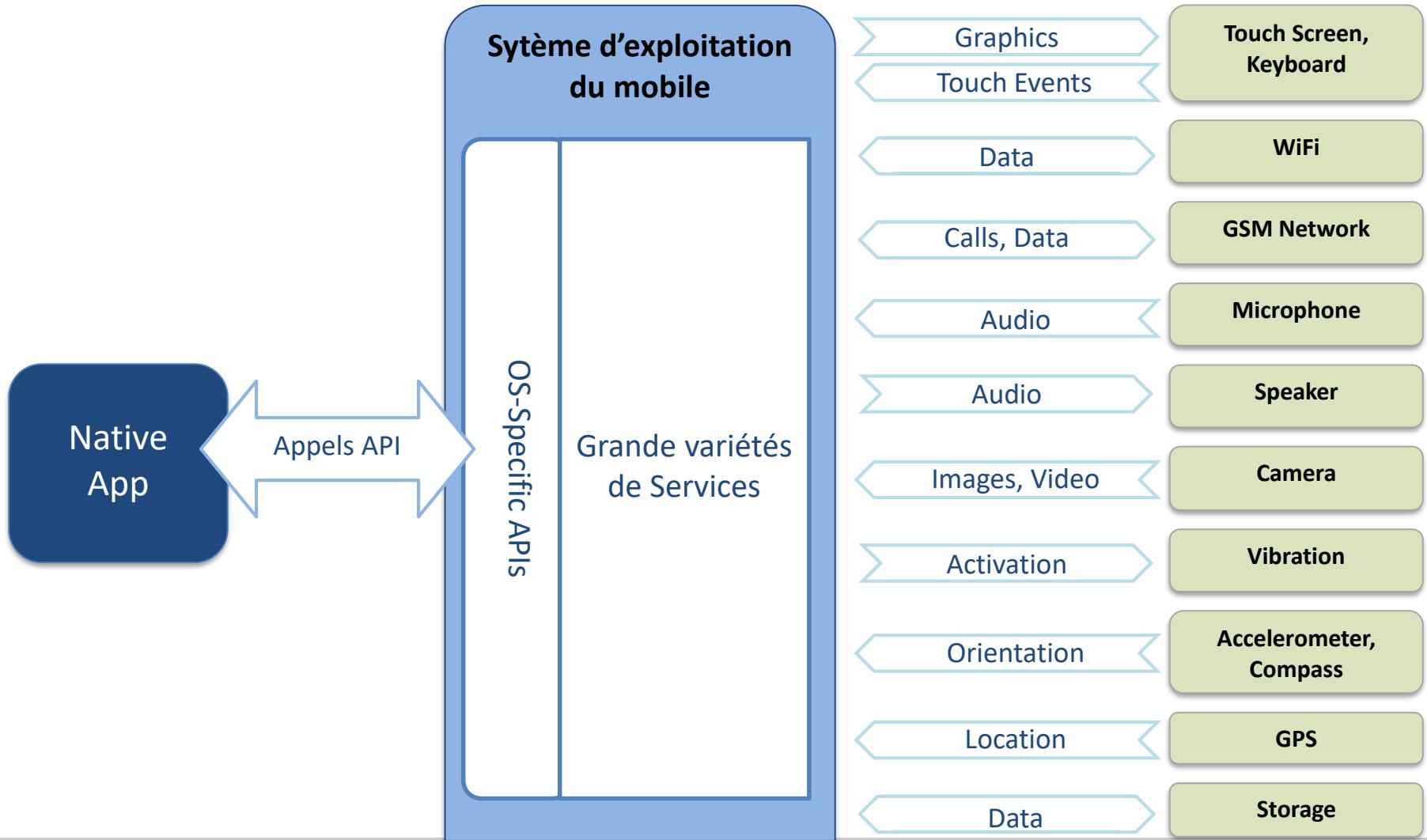
■ Les approches du développement mobile

2- Apps web mobile

- Développée entièrement avec les technologies Web (HTML, CSS, JavaScript)
- Exécutée **par le navigateur** et non par l'OS
- **Démarrée** de plusieurs façons: URL, code QR, lien HyperText, raccourci
- L'installation est **optionnelle**
- Utilise des fonctionnalités HTML5 **Indépendantes** de la plateforme + d'autres liées à la plateforme

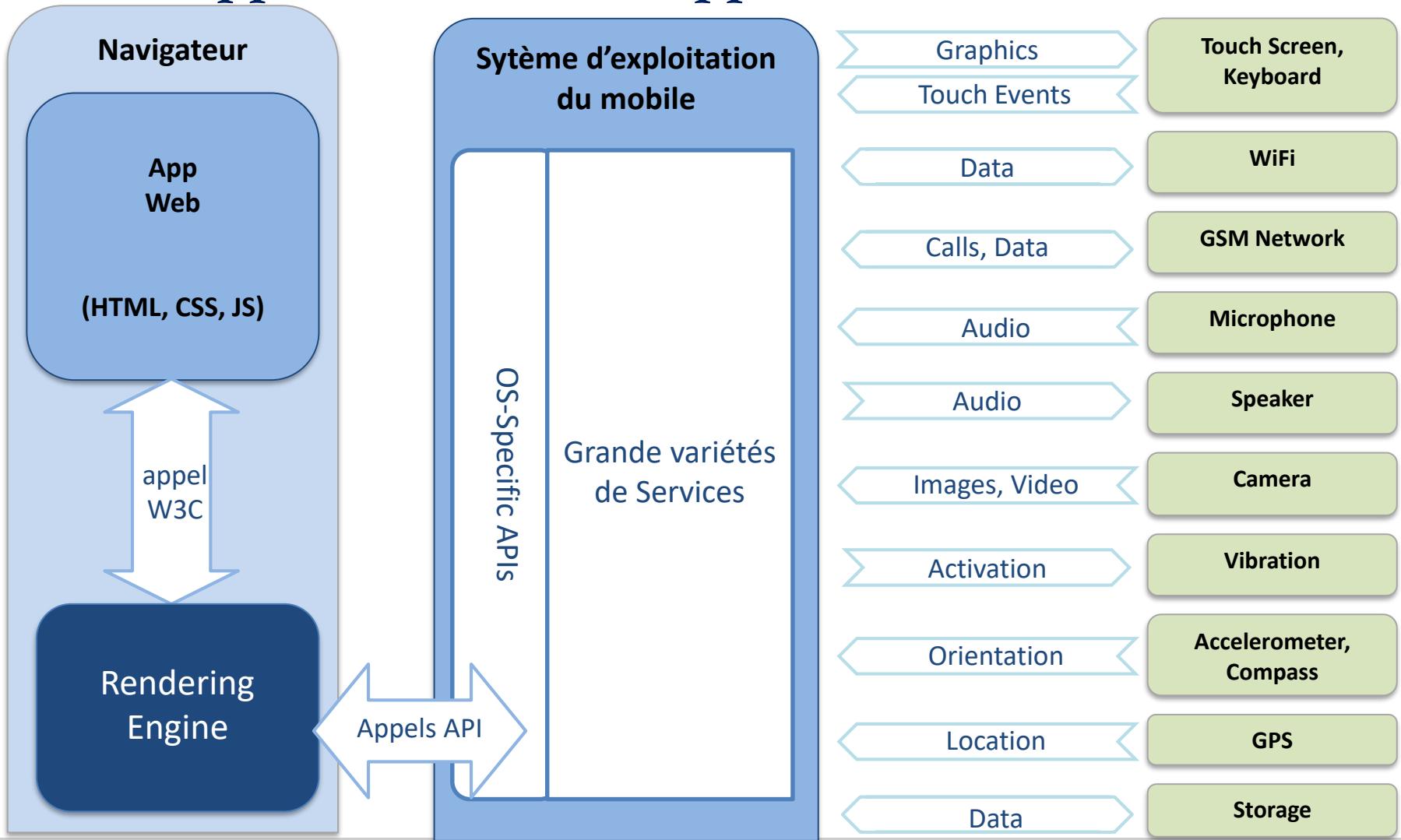
Approches dev. mobile

■ Les approches du développement mobile



Approches dev. mobile

■ Les approches du développement mobile



Approches dev. mobile

■ Les approches du développement mobile

Apps web mobile Vs Native Apps

Accès aux
ressources

Vitesse

Coût Dev.

App Store

Approbation

Native

complet

Très rapide

élevé

disponible

Obligatoire

Web

Partiel

Rapide

Raisonnables

Non

Aucune

Approches dev. mobile

■ Les approches du développement mobile

3- Apps Hybrides

Accès aux
ressources

Vitesse

Coût Dev.

App Store

Approbation

Native

complet

Très rapide

élevé

disponible

Obligatoire

Web

Partiel

Rapide

Raisonnables

Non

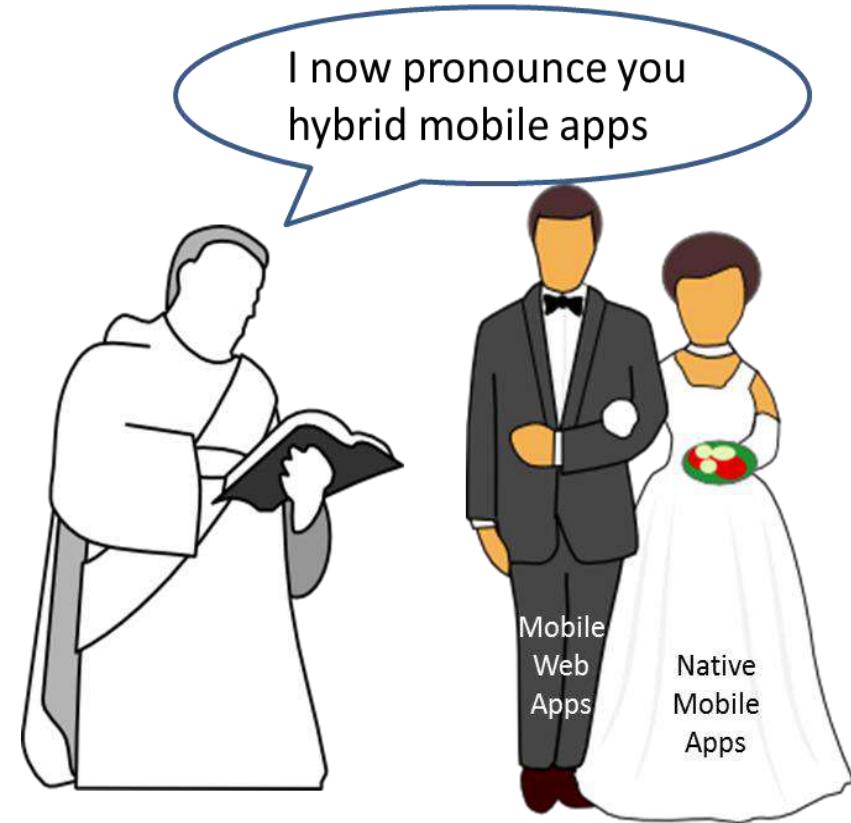
Aucune

Approches dev. mobile

■ Les approches du développement mobile

3- Apps Hybrides

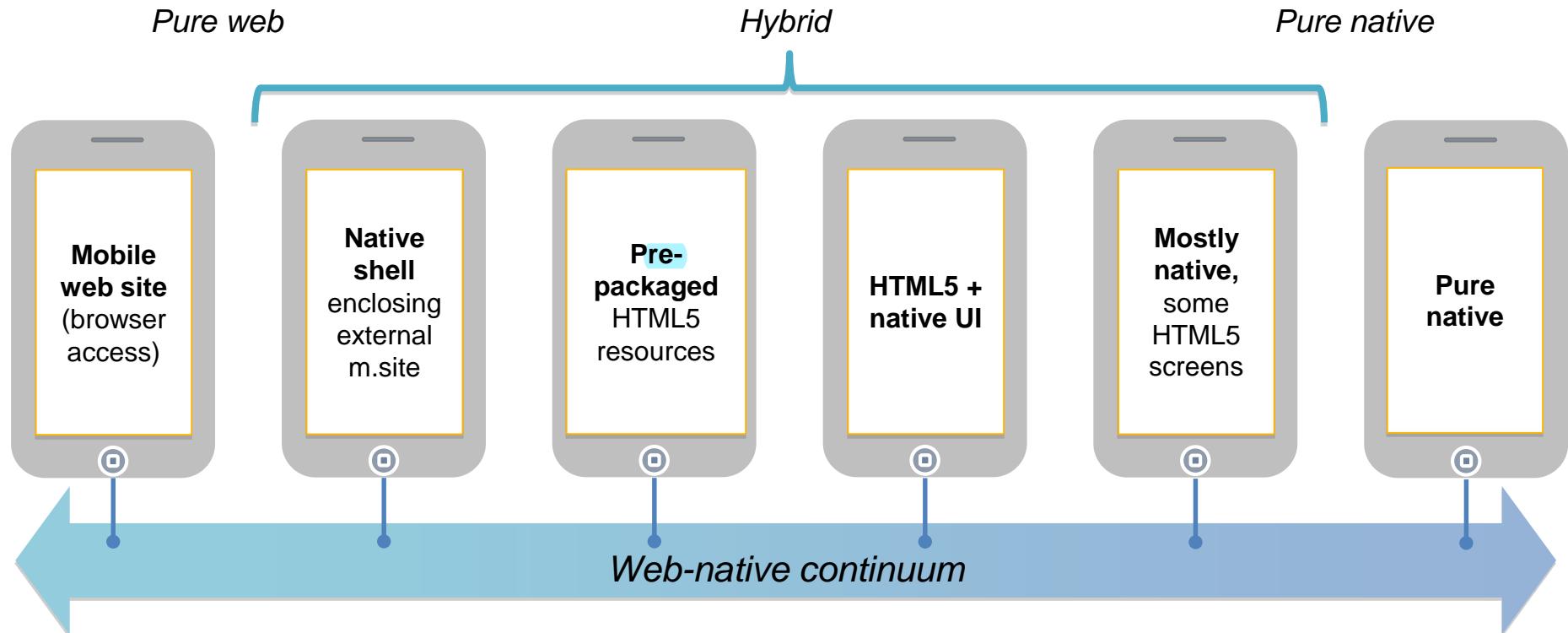
- Apps hybride = native apps + du HTML embarqué
- Elle a tous les avantages des Apps native
- Certaines parties de l'App sont développées avec les technologies Web
- La partie Web de l'App est soit téléchargée soit incluse dans le package



Approches dev. mobile

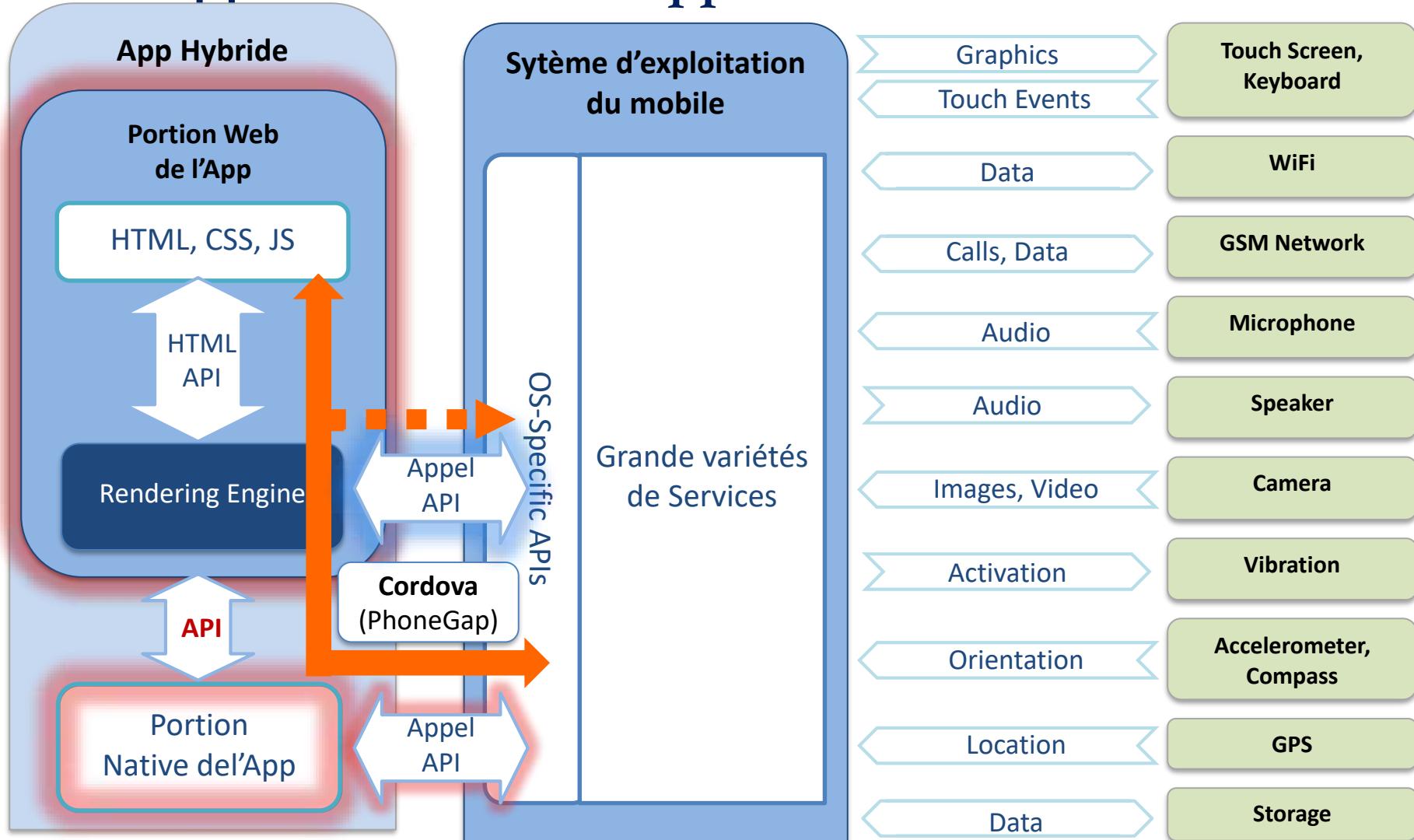
■ Les approches du développement mobile

3- Apps Hybrides



Approches dev. mobile

■ Les approches du développement mobile



Approches dev. mobile

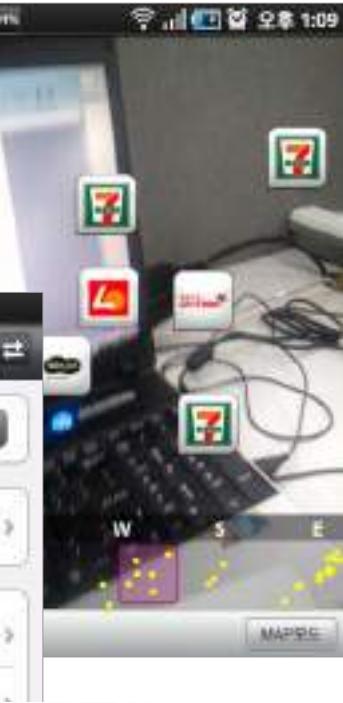
■ Les approches du développement mobile

3- Apps Hybrides

Bank of America



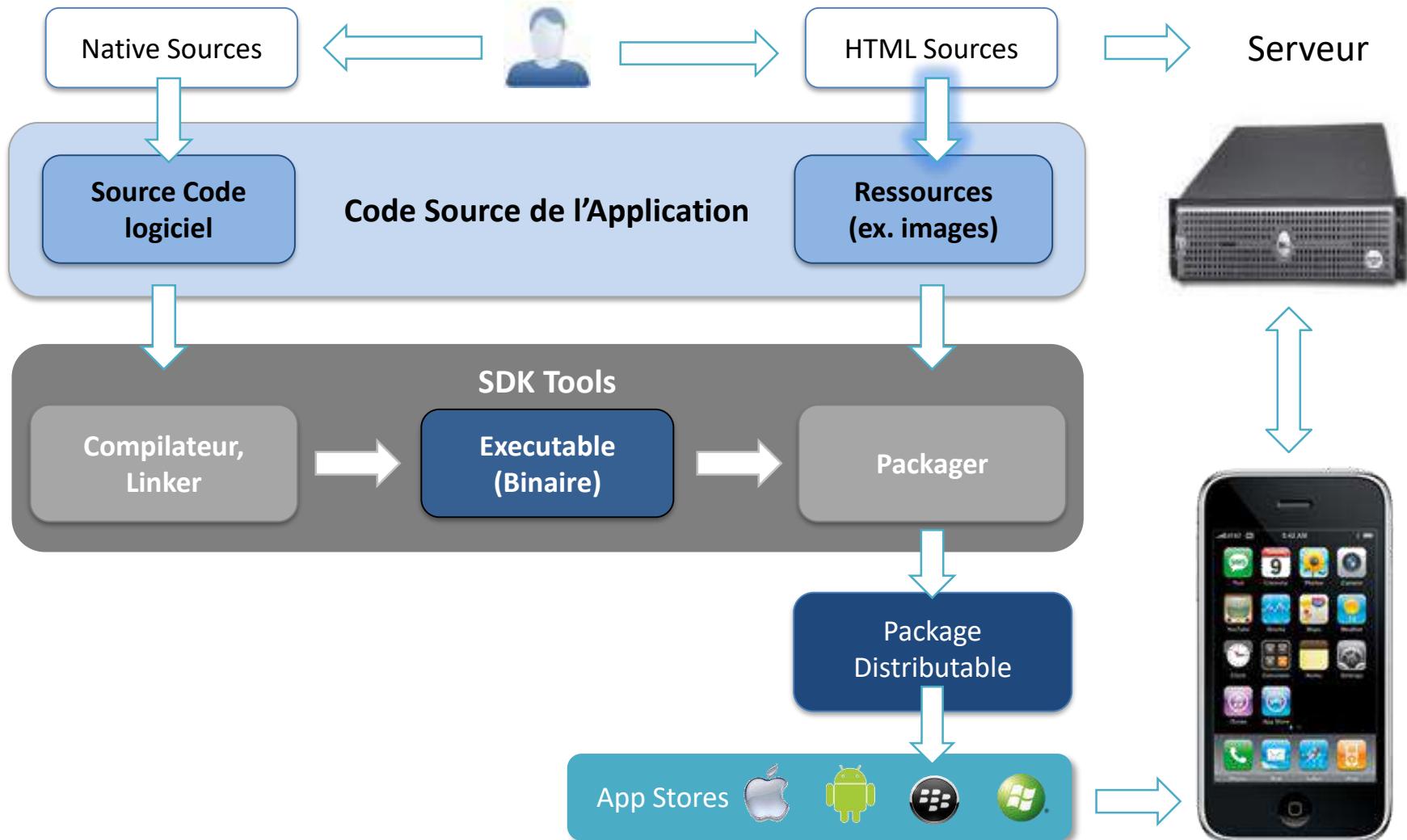
Morgan Stanley



Lotte Card (Korea)

Approches dev. mobile

■ Les approches du développement mobile



Approches dev. mobile

■ Les approches du développement mobile Comparaison

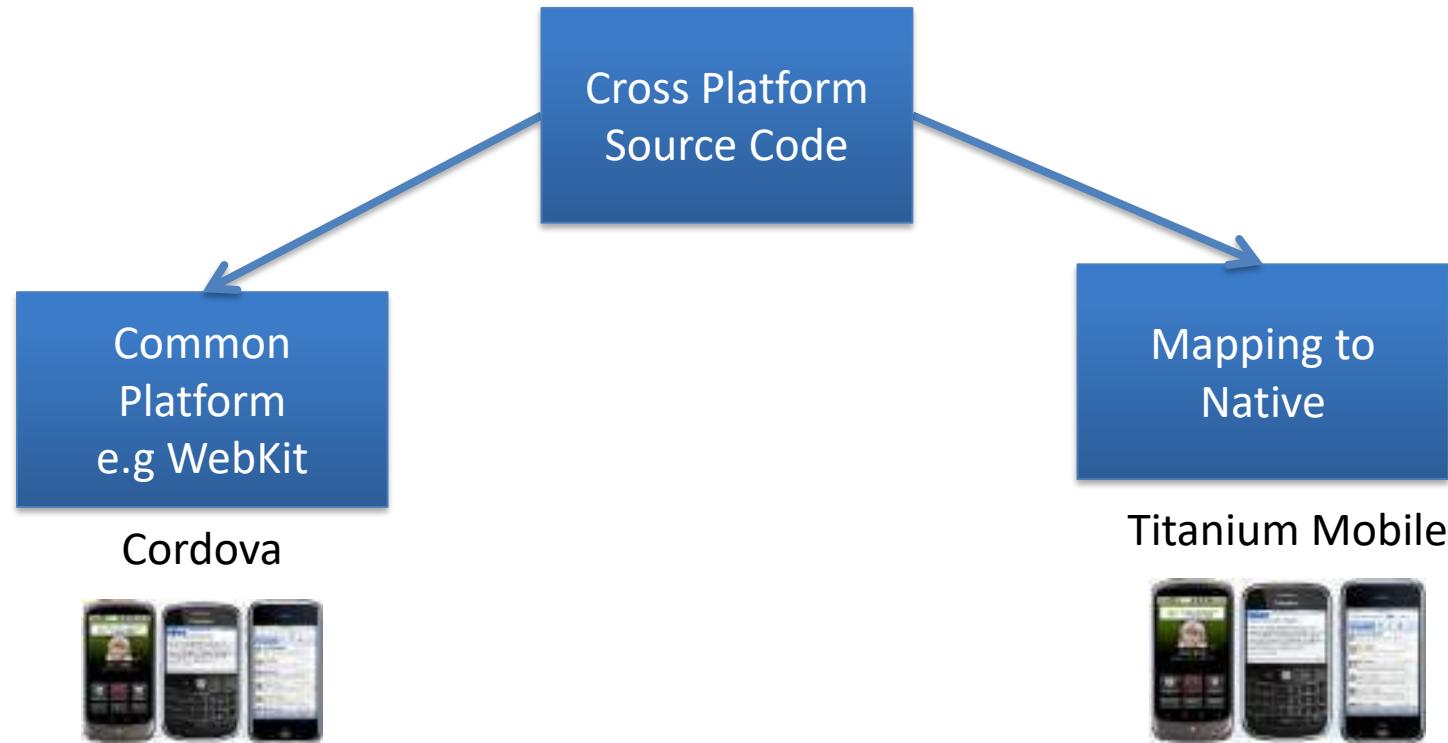
	Accès aux ressources	Vitesse	Coût Dev.	App Store	Approbation
Native	complet	Très rapide	élevé	disponible	Obligatoire
Hybride	complet	Bonne performance	Raisonnables	disponible	Moins complexe
Web	Partiel	Rapide	Raisonnables	Non	Aucune

Approches dev. mobile

■ Les approches du développement mobile

4- Cross-Platform

Plusieurs solutions : Flutter, Xamarin, Titanium, Ionic, ...



Approches dev. mobile

■ Les approches du développement mobile

4- Cross-Platform : common plateform



JSON packets

Native Code
(Java/C++/ObjC)

Approches dev. mobile

■ Les approches du développement mobile

4- Cross-Platform : common plateform



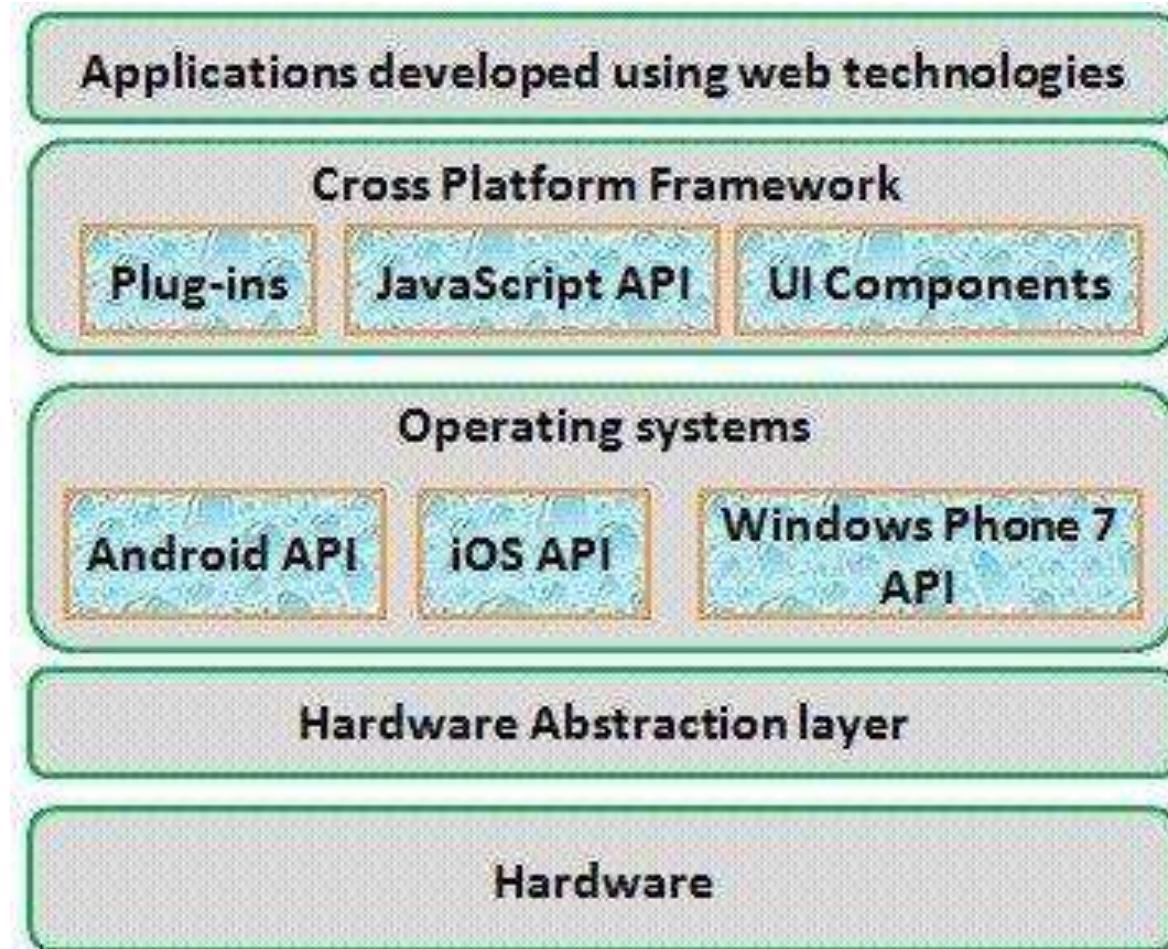
Native Code
(Java/C++/ObjC)

GPS Location

Approches dev. mobile

■ Les approches du développement mobile

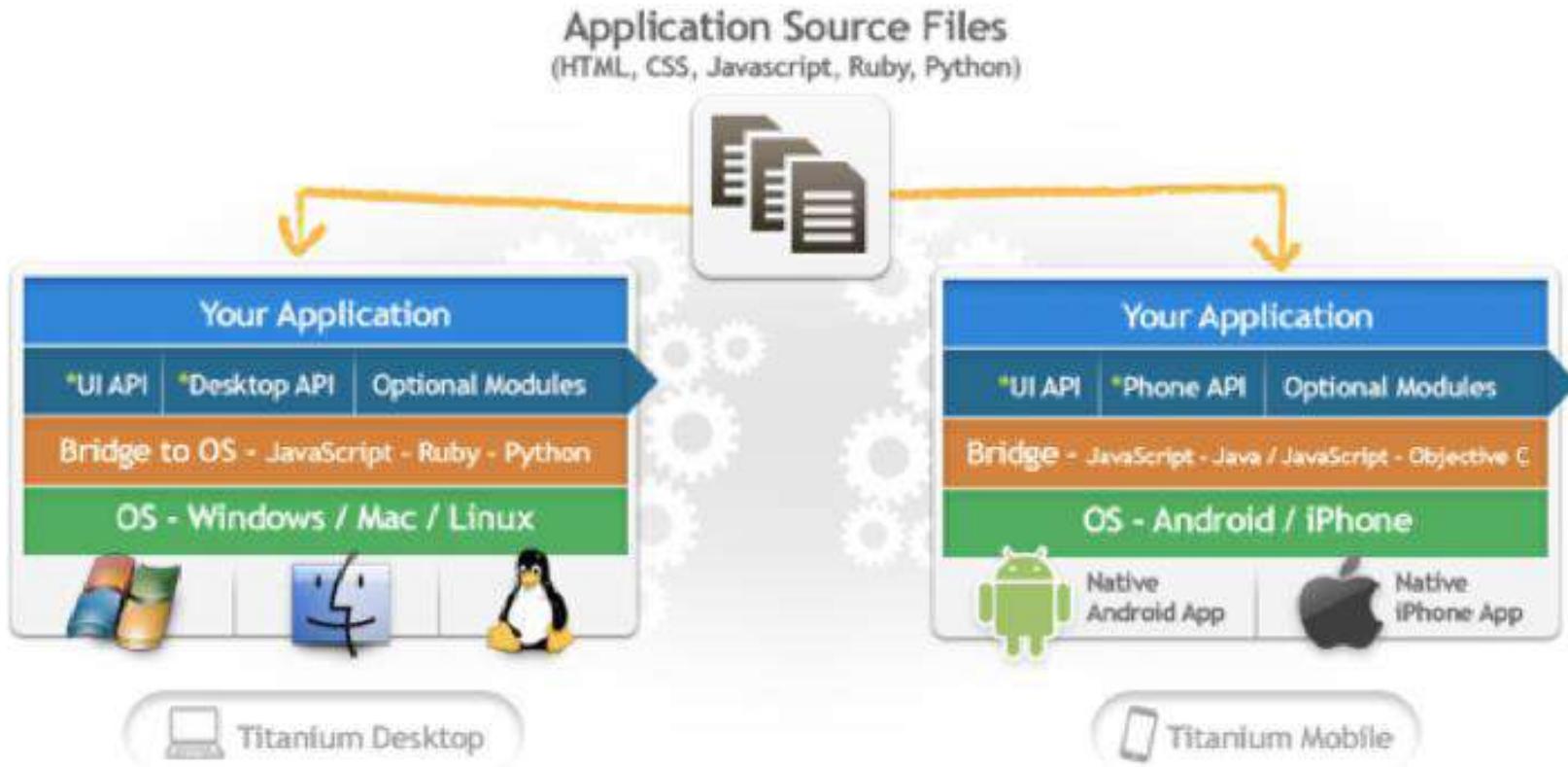
4- Cross-Platform : Mapping to Native



Approches dev. mobile

■ Les approches du développement mobile

4- Cross-Platform : Mapping to Native



Introduction

■ Introduction à Android : Historique

- Idée d'une petite Startup Californienne Android Inc. (fondée en 2003)
Andy Robin
- Pour les caméras numériques mais changé vers téléphones
- Au bord de la faillite
- En 2005 : Samsung refuse d'acheter la startup.

2 semaines plus tard Google la rachète pour 50 M\$

Introduction

■ Introduction à Android : Historique

- 2007 en réponse à Apple (iOS) :
Google et 34 compagnies forment l'Open Handset Alliance (OHA)
 - accélérer l'innovation dans le mobile.
 - ⇒ développer Android plateforme (ouvert et gratuit)
- Aujourd'hui OHA contient 84 acteurs : opérateurs mobile, fabricants d'appareils, développeur de logiciels, fournisseurs semi-conducteur, commercialisation de technologies

Technologies et développement Mobile



Pr. Slimane Bah, ing. PhD

Génie Informatique option TI & IQL

Semaine 3

Séance 3

Check Point

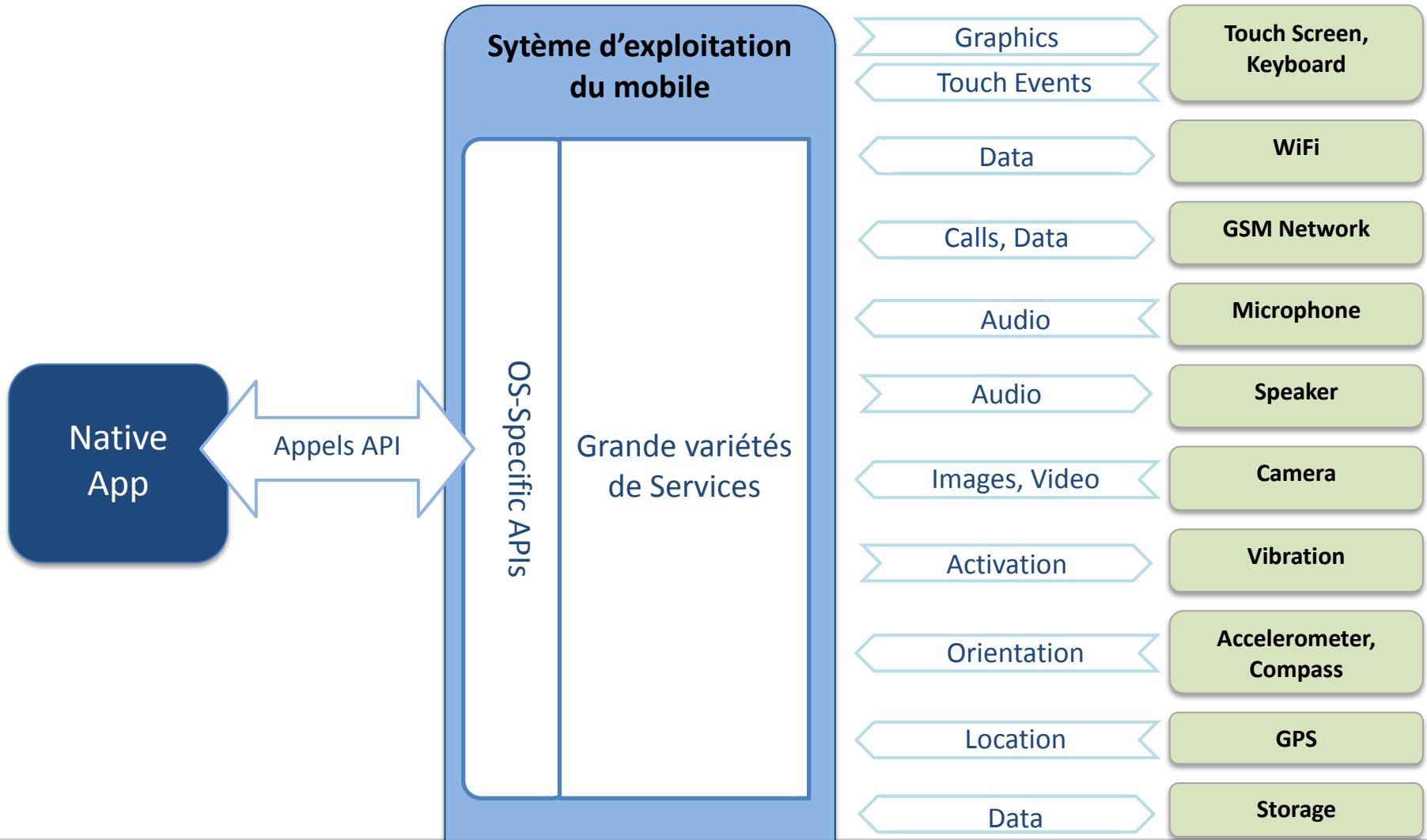
- Les approches du développement mobile ?
- Variantes d'une app web mobile?

Mise au point

Feature	Safari iOS	Android Browser	Samsung Internet	Google Chrome	Amazon Silk	BlackBerry Browser	Nokia Browser	Internet Explorer	Opera Mobile	Opera mini	Firefox
File API W3C API Opening local files through input type	✓ 6.0+	✓ 3.0+	✓	✓	✓ 2.0+	✓ 2.0+	✓ 2.0+	✓ 11+	✓ 12+ (partial)		✓ 11+
FileSystem API W3C API Virtual FileSystem for persistent storage			✓	✓	✓ 2.0+	✓	✓		✓ 14+		
HTML Media Capture W3C API Taking pictures, record video and audio from an input file type	✓ 6.0+	✓ 3.0+	✓	✓		✓	✓		✓ 14+	✓ 11+	✓ 11+
Web Speech API W3C API Speech Recognition and Synthesizer	✓ 7.0+			✓ 32+					✓ 14+		✓ 11+
HomeScreen Webapp No API Add icon to the home screen with fullscreen support	✓ meta tags			✓ 32+			✓ Anna+	✓ live tile	✓ 14+	✓ manifest	✓ manifest

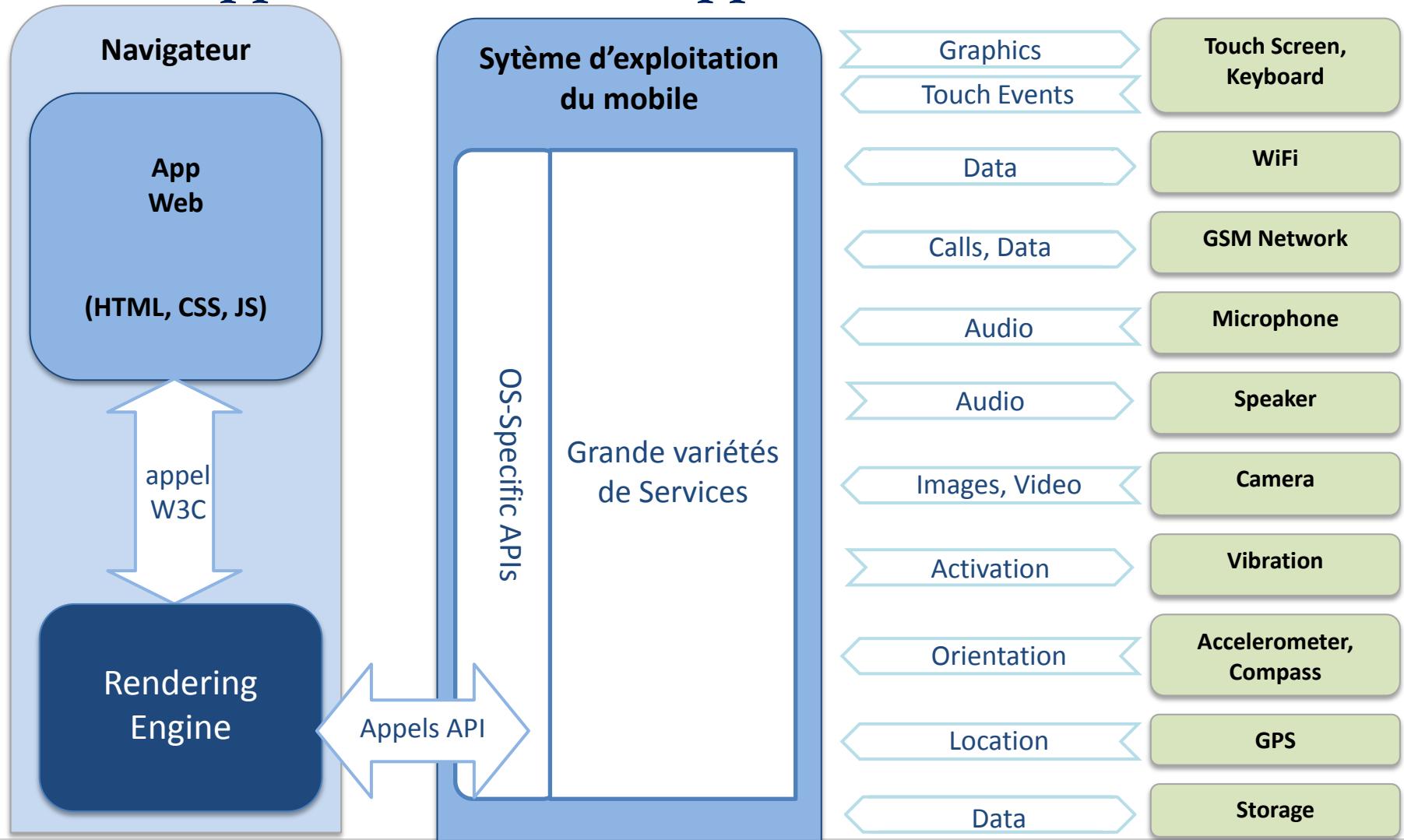
Introduction

■ Les approches du développement mobile



Introduction

■ Les approches du développement mobile



Introduction

■ Les approches du développement mobile

Apps web mobile Vs Native Apps

Accès aux
ressources

Vitesse

Coût Dev.

App Store

Approbation

Native

complet

Très rapide

élevé

disponible

Obligatoire

Web

Partiel

Rapide

Raisonnables

Non

Aucune

Introduction

■ Les approches du développement mobile

3- Apps Hybrides

Accès aux
ressources

Vitesse

Coût Dev.

App Store

Approbation

Native

complet

Très rapide

élevé

disponible

Obligatoire

Web

Partiel

Rapide

Raisonnables

Non

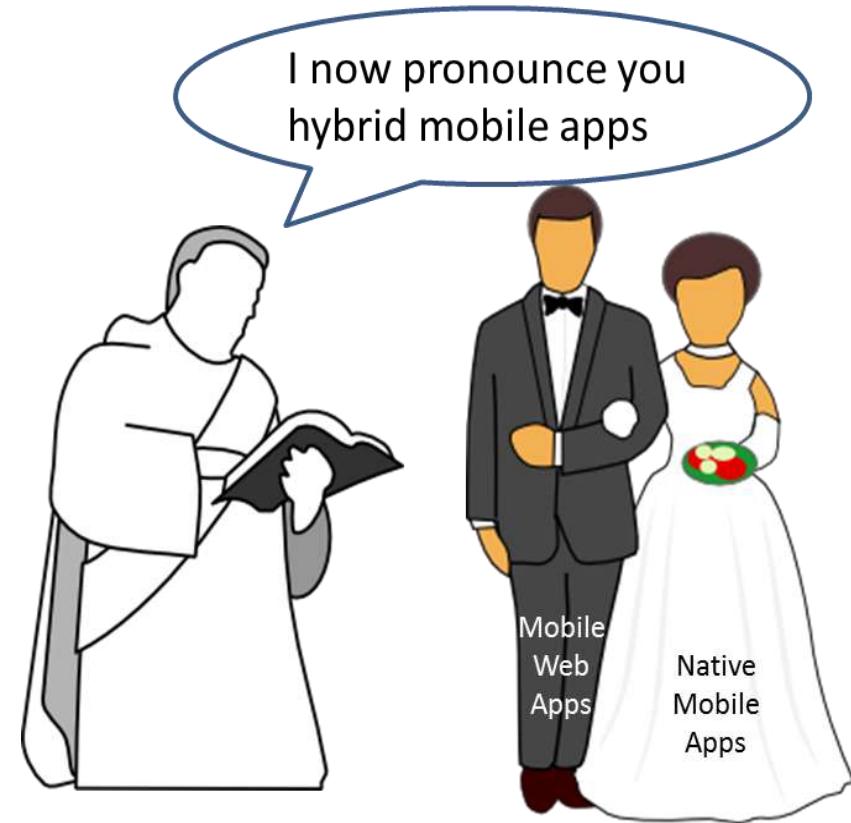
Aucune

Introduction

■ Les approches du développement mobile

3- Apps Hybrides

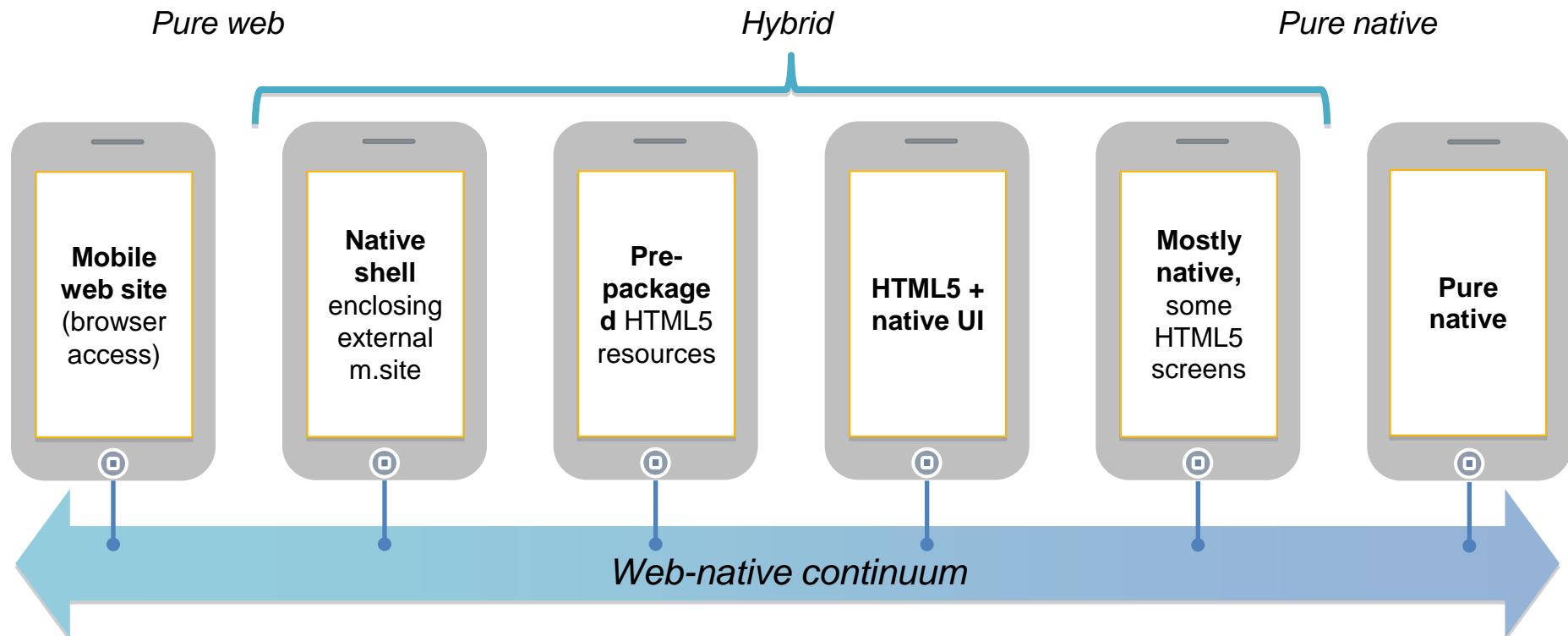
- Apps hybride = native apps + du HTML embarqué
- Elle a tous les avantages des Apps native
- Certaines parties de l'App sont développées avec les technologies Web
- La partie Web de l'App est soit téléchargée soit include dans le package



Introduction

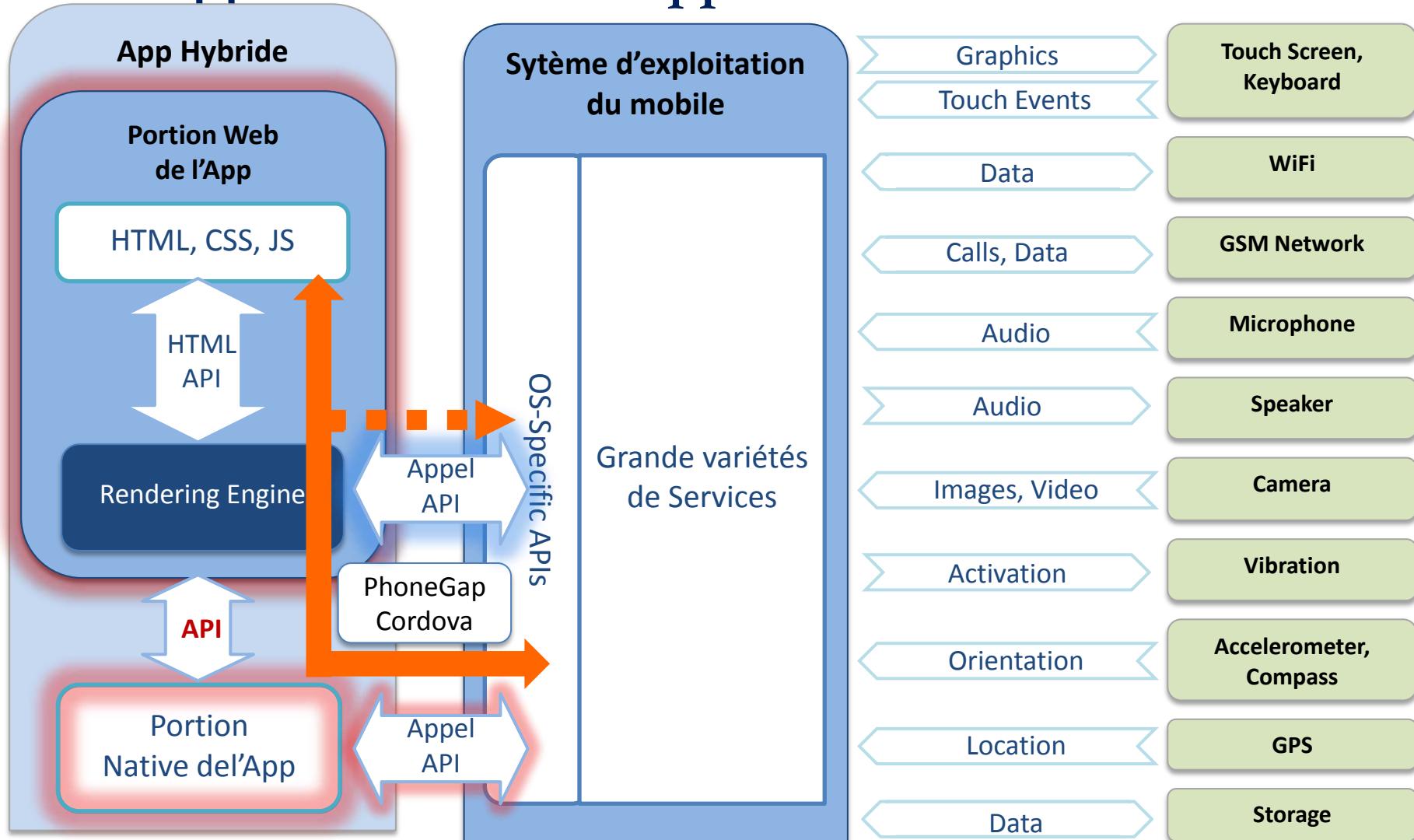
■ Les approches du développement mobile

3- Apps Hybrides



Introduction

■ Les approches du développement mobile



Introduction

■ Les approches du développement mobile

3- Apps Hybrides

Bank of America



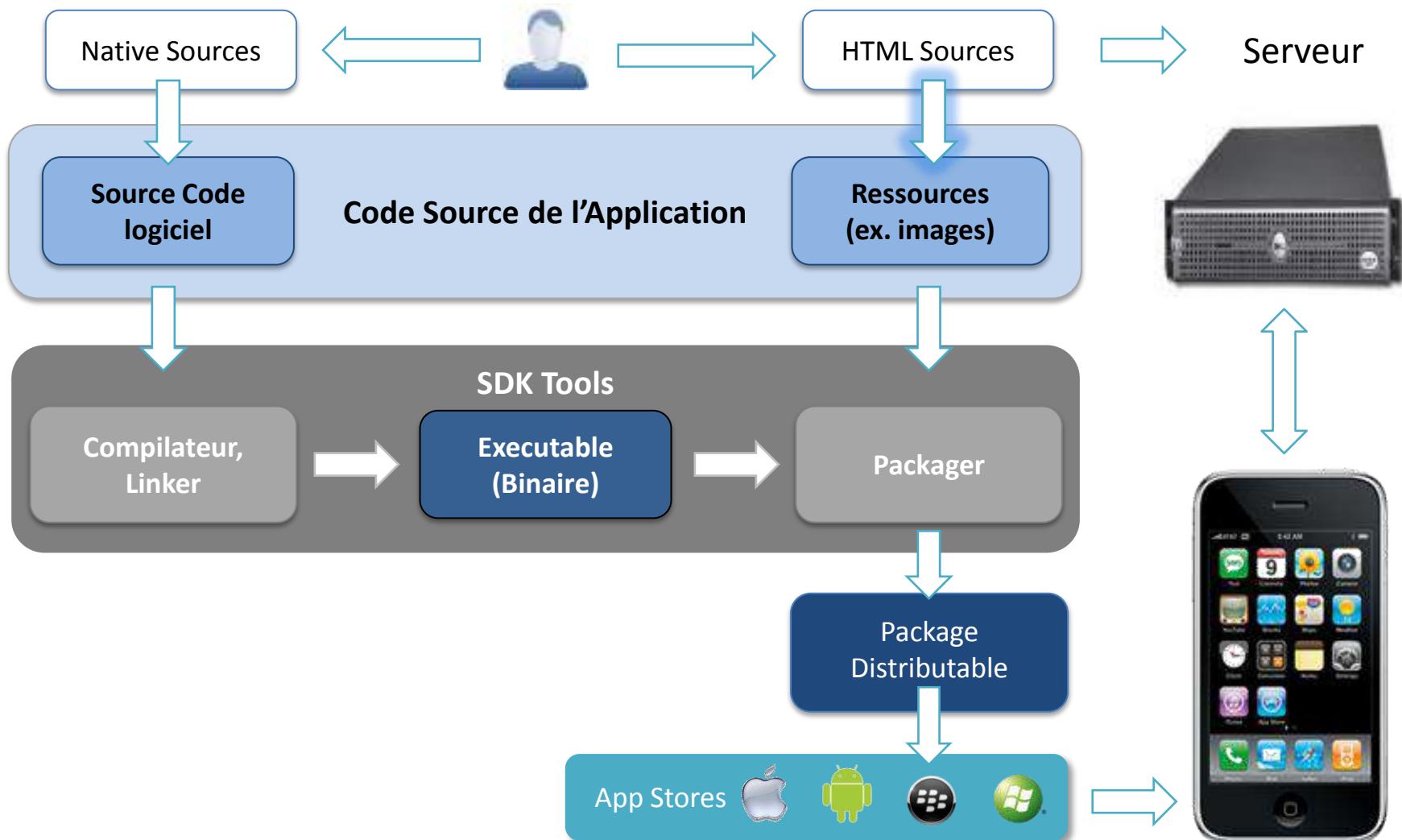
Morgan Stanley



Lotte Card (Korea)

Introduction

■ Les approches du développement mobile



Introduction

■ Les approches du développement mobile Comparaison

	Accès aux ressources	Vitesse	Coût Dev.	App Store	Approbation
Native	complet	Très rapide	élevé	disponible	Obligatoire
Hybride	complet	Bonne performance	Raisonnables	disponible	Moins complexe
Web	Partiel	Rapide	Raisonnables	Non	Aucune

Introduction

■ Introduction à Android : Historique

- Idée d'une petite Startup Californienne Android Inc. (fondée en 2003)
Andy Robin (AndRoid)
- Pour les caméras numériques mais changé vers téléphones
- Au bord de la faillite
- En 2005 : Samsung refuse d'acheter la startup.

2 semaines plus tard Google la rachète pour 50 M\$

Introduction

■ Introduction à Android : Historique

- 2007 en réponse à Apple (iOS) :
Google et 34 compagnies forment l'Open Handset Alliance (OHA)
 - accélérer l'innovation dans le mobile.
 - ⇒ développer Android plateforme (ouvert et gratuit)
- Aujourd'hui OHA contient 84 acteurs : opérateurs mobile, fabricants d'appareils, développeur de logiciels, fournisseurs semi-conducteur, commercialisation de technologies

Introduction

■ Introduction à Android : C'est quoi ?

- Une plate forme **ouverte** et **gratuite** : un ensemble de librairies, de composants logiciels pour les systèmes embarqués et mobiles
- Un SE, un middleware et des applications de bases
- Il offre des **outils** et des **APIs** aux développeurs pour étendre les fonctionnalités du mobile
- SE : Linux modifié / programmation : Java

Introduction

■ Introduction à Android : Versions



Cupcake
Android 1.5



Donut
Android 1.6



Eclair
Android 2.0/2.1



Froyo
Android 2.2.x



Gingerbread
Android 2.3.x



Honeycomb
Android 3.x



Ice Cream Sandwich
Android 4.0.x



Jelly Bean
Android 4.1.x



KitKat
Android 4.4.x



Lollipop
Android 5.0



Marshmallow
android 6.0



Nougat
android 7.0

Introduction

■ Introduction

Platform Version	API Level	VERSION_CODE	Notes
Android 4.4	19	KITKAT	Platform Highlights
Android 4.3	18	JELLY_BEAN_MR2	Platform Highlights
Android 4.2, 4.2.2	17	JELLY_BEAN_MR1	Platform Highlights
Android 4.1, 4.1.1	16	JELLY_BEAN	Platform Highlights
Android 4.0.3, 4.0.4	15	ICE_CREAM SANDWICH_MR1	Platform Highlights

Platform	Android Version	Released	API Level	Name
Android 7.0	Android 7.0	August 2016	24	Nougat
Android 6.0	Android 6.0	August 2015	23	Marshmallow
Android 5.1	Android 5.1	March 2015	22	Lollipop
Android 5.0	Android 5.0	November 2014	21	Lollipop
Android 4.4	4.4	CUPCAKE		
Android 4.1	4.1	BASE_1_1		
Android 1.0	1	BASE		

Introduction

- **Introduction à Android : Qu'est ce qu'il offre ?**
- **Pile logicielle pour les appareils mobile :**
 - Cœur du SE, librairies systèmes, Framework applicatif et applications de bases
- **SDK pour la création d'applications :**
 - Librairies et outils de développement

Technologies et développement Mobile



Pr. Slimane Bah, ing. PhD

Génie Informatique option TI & IQL

Semaine 11.1

Séance 4

Principes fondamentaux

■ Composantes d'une application Android

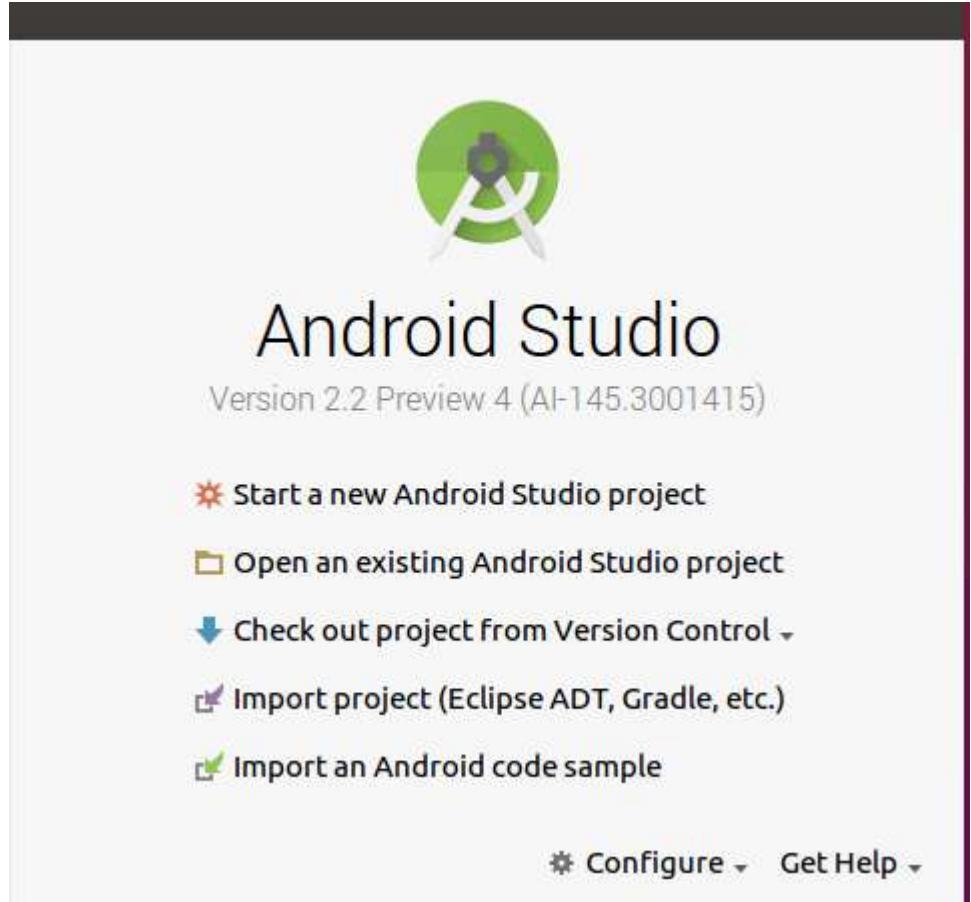
fonctionnalité	Classe Java	Exemple
Ce qu'un utilisateur peut faire	Activity	Éditer une note, composer un numéro
Processus d'arrière plan	Service	Jouer de la musique, MAJ de la météo
Réception de messages	BroadcastReceiver	Alerte à la réception d'un événement
Stocker et obtenir des données	ContentProvider	Ouvrir la liste des contacts

Principes fondamentaux

■ Environnement de développement Android

Android Studio

- Android SDK
- Compilateur (Gradle)
- Éditeur intelligent
- AVD
- SDK manager



Principes fondamentaux

■Android SDK

- Un ensemble d'outils intégrés permettant la création et le debugage des applications
- Il est contient les éléments suivants :
 1. Librairies Java
 2. Outils de packaging android virtual device____ tester dans pc
 3. Gestionnaire émulateurs AVD android debugging bridridge
 4. Un outil de communication avec les appareils ADB

Principes fondamentaux

■ Environnement de développement Android

SDK Manager

- Permet de choisir les composants à installer / à MAJ :
 - La version d'Android à installer
 - **SDK tools** : obligatoire (contient le gestionnaire)
 - **SDK platform-tools** : obligatoire (contient adb)
 - **SDK platform** : obligatoire (contient les libs)
 - **Intel/ARM System images** : pour les AVD
 - **Android Support** : divers outils pour créer des applications
 - Google play service

Principes fondamentaux

■ Environnement de développement Android

- Une application Android peut s'exécuter soit sur un appareil réel ou sur un émulateur
- Mobile réel :
Activer le mode **USB debugging** sur l'appareil
- Emulateur (Android Virtual Device) :
Plusieurs types sont supportés par la plateforme

Android Resources

■ Structure d'un projet Android

- **AndroidManifest.XML**

Configuration générale de l'app.

- **src/java**

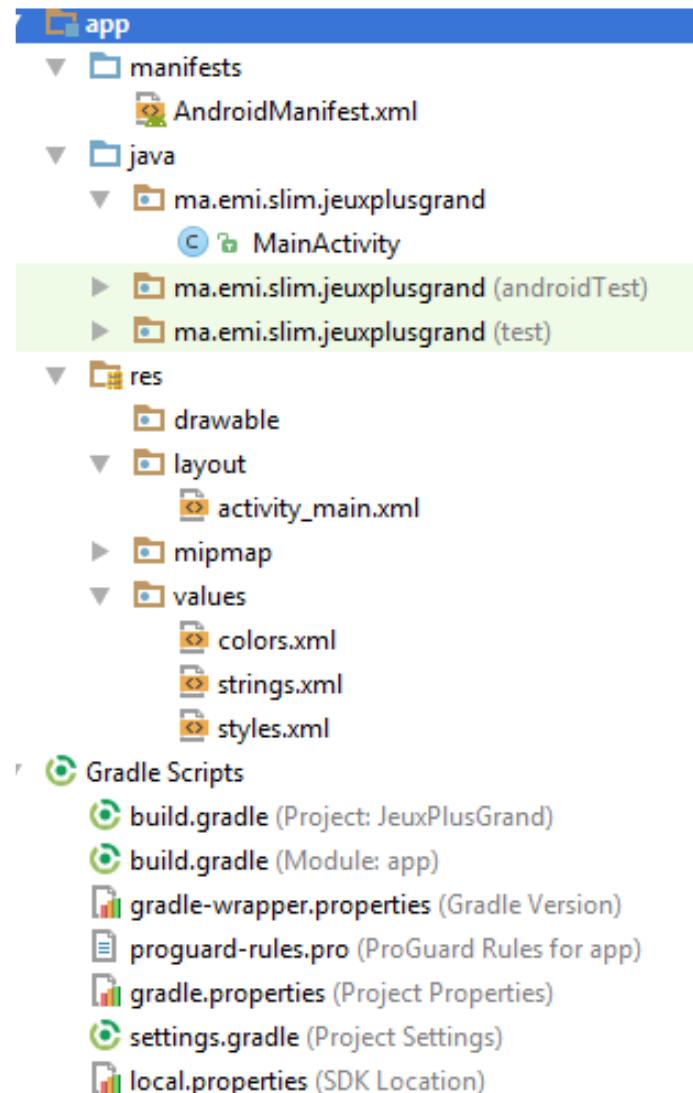
Code source des classes java

- **res/...**

Fichiers ressources : interfaces graphiques, menu constantes, couleurs...

- **Gradle**

Gestion et outils de compilation



Android Resources

■ Application Android : Ressources

- Une application Android est composée d'une partie compilable et une autre non compilable
- Des ressources peuvent être utilisés pour les besoins de l'application
- Les gestion des ressources se fait d'une manière centralisée
- Permet la réutilisation et la configuration des ressources

Android Resources

■ Ressources

Les différentes ressources possibles (sous répertoires de res/)

- /anim
- /color
- /drawable
- /mipmap
- /layout
- /menu
- /raw
- /xml
- /values (strings, arrays, color, dimen)

```
<?xml version="1.0" encoding="utf-8"?>
<!-- This XML file defines the colors to use for window
     decorations when the application is in focus -->

<?xml version="1.0" encoding="utf-8"?>
<selector xmlns:android="http://schemas.android.com/apk/res/android">
    <item android:state_pressed="true"
          android:color="#ffff0000"/> <!-- pressed -->
    <item android:state_focused="true"
          android:color="#ff0000ff"/> <!-- focused -->
    <item android:color="#ff000000"/> <!-- default -->
</selector>
<!-- android:state_window_focus -->
</selector>
```

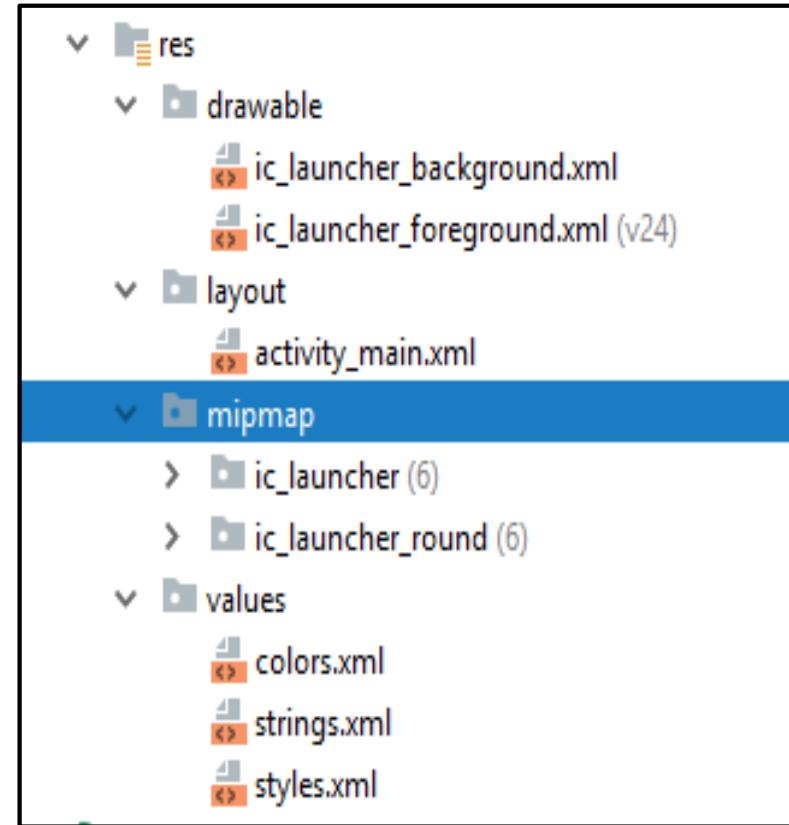
Button_text.xml

```
<Button
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="@string/button_text"
        android:textColor="@color/button_text" />
```

Android Resources

■ Ressources

- Sont gérés sous le répertoire **res/**
- Pour chaque ressource on peut spécifier : **la ressource par défaut** et la **ressource alternative**
- Le système détecte la configuration de l'appareil et charge les ressources correspondantes sinon celles par défaut.



Android Resources

■ Ressources

res/layout contient l'UI par défaut. Pour définir une UI pour l'affichage « paysage » il faut la stocker dans le dossier **res/layout-land**



2 différents appareils, utilisent les **ressources par défaut** (pas le choix)



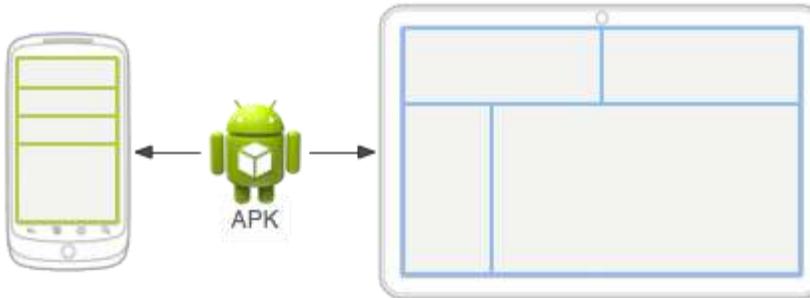
2 différents appareils avec l'un utilisant des **ressources alternatives**

Android Resources

■ Ressources



2 différents appareils, utilisent les ressources par défaut (pas le choix)



2 différents appareils avec l'un utilisant des ressources alternatives

Android Resources

■ Ressources

Définir des **ressources alternatives** :

- **Langage** ou language et région : en, fr, en-rUS, fr-rFR, fr-rCA...
- **Taille de l'écran** : small, normal, large, xlarge
- Orientation de l'écran : port, land
- Résolution d'écran (dpi) : ldpi, mdpi, hdpi, xhdpi, nodpi
- Disponibilité du clavier : keysexposed, keyshidden, keyssoft
- Version du SDK (API Level) : v3, v4, v7...
- Night mode : night, notnight...

Android Resources

■ Ressources

Définir des ressources alternatives :

- Langage ou language et région : en, fr, en-rUS, fr-rFR, fr-rCA...

```
res/
└── layout/
    └── main.xml (Default layout)
└── layout-ar/
    └── main.xml (Specific layout for Arabic)
└── layout-ldrtl/
    └── main.xml (Any "right-to-left" language, except
                    for Arabic, because the "ar" language qualifier
                    has a higher precedence.)
└── Disponibilité du clavier : keyboardPortrait, keyboardHidden, keyboard...
```

- Version du SDK (API Level) : v3, v4, v7...

Android Resources

■ Ressources

- Il est possible de combiner plusieurs qualificateurs de ressources séparés par un tiret –
- Il faut les donner dans l'ordre de précédence

Ex : drawable-port-hdpi, drawable-en-rUS-land

- Android trouve la bonne compatibilité en se basant sur la précédence des qualificateurs

Android Resources

■ Ressources

MCC and MNC	Examples: mcc310 (US), mcc310-mnc004 (US, Verizon) mcc208-mnc00 (France, orange)
Language and region	Examples: en, fr, en-rUS, fr-rFR, fr-rCA,
Layout Direction	Ldrtl, ldltr
smallestWidth	sw<N>dp Examples: sw320dp, sw600dp
Available width	w<N>dp Examples: w720dp, w1024dp
Available height	h<N>dp Examples: h720dp, h1024dp
Screen size	Small,normal,large, xlarge
Screen aspect	Long, notlong

Android Resources

■ Ressources

Round screen	Round, notround
Wide Color Gamut	Widecg, nowidecg
High Dynamic Range (HDR)	Highdr, lowdr
Screen orientation	Port, land
UI mode	Car, desk, television, appliance, watch, vrheadset
Night mode	Night, notnight
Screen pixel density (dpi)	Ldpi, mdpi, hdpi, xhdpi, xxhdpi, xxxhdpi, nodpi, tvdpi, anydpi, nnndpi
Touchscreen type	Notouch, finger
Keyboard availability	Keysexposed, keyshidden, keyssoft
Primary text input method	Nokeys, qwerty, 12key
Navigation key availability	Navexposed, navhidden
Primary non-touch navigation method	Nonav, dpad, trackball, wheel
Platform Version (API level)	V3, V4, V5

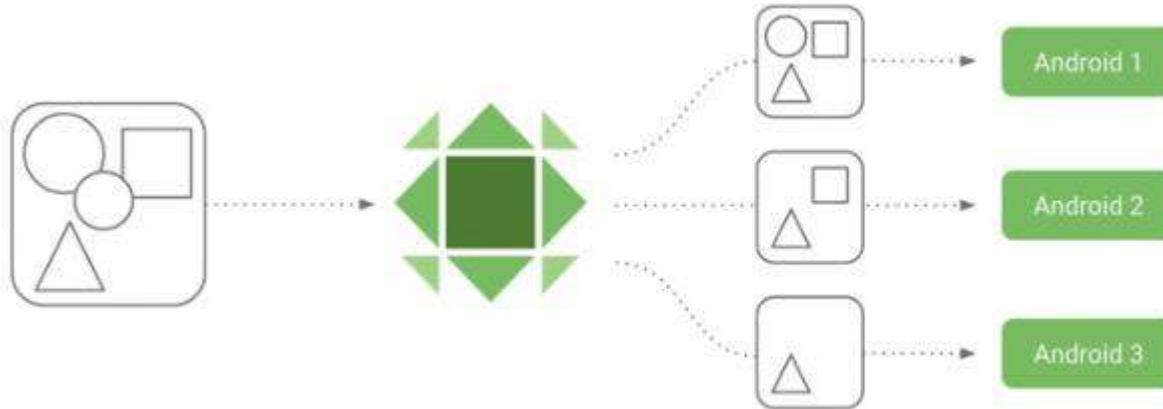
Android Resources

■ Ressources : APK vs Bundle

Lancé en 2021 : objectif Installation personnalisée

Le Bundle permet d'optimiser la taille de l'APK à installer

Il se base sur la configuration du terminal cible pour générer l'APK avec les ressources spécifiques à ce terminal



Android Resources

■ Ressources : APK vs Bundle



Android Resources

■ Ressources : APK vs Bundle

Autres fonctionnalités



In-app
updates



Conditional
delivery



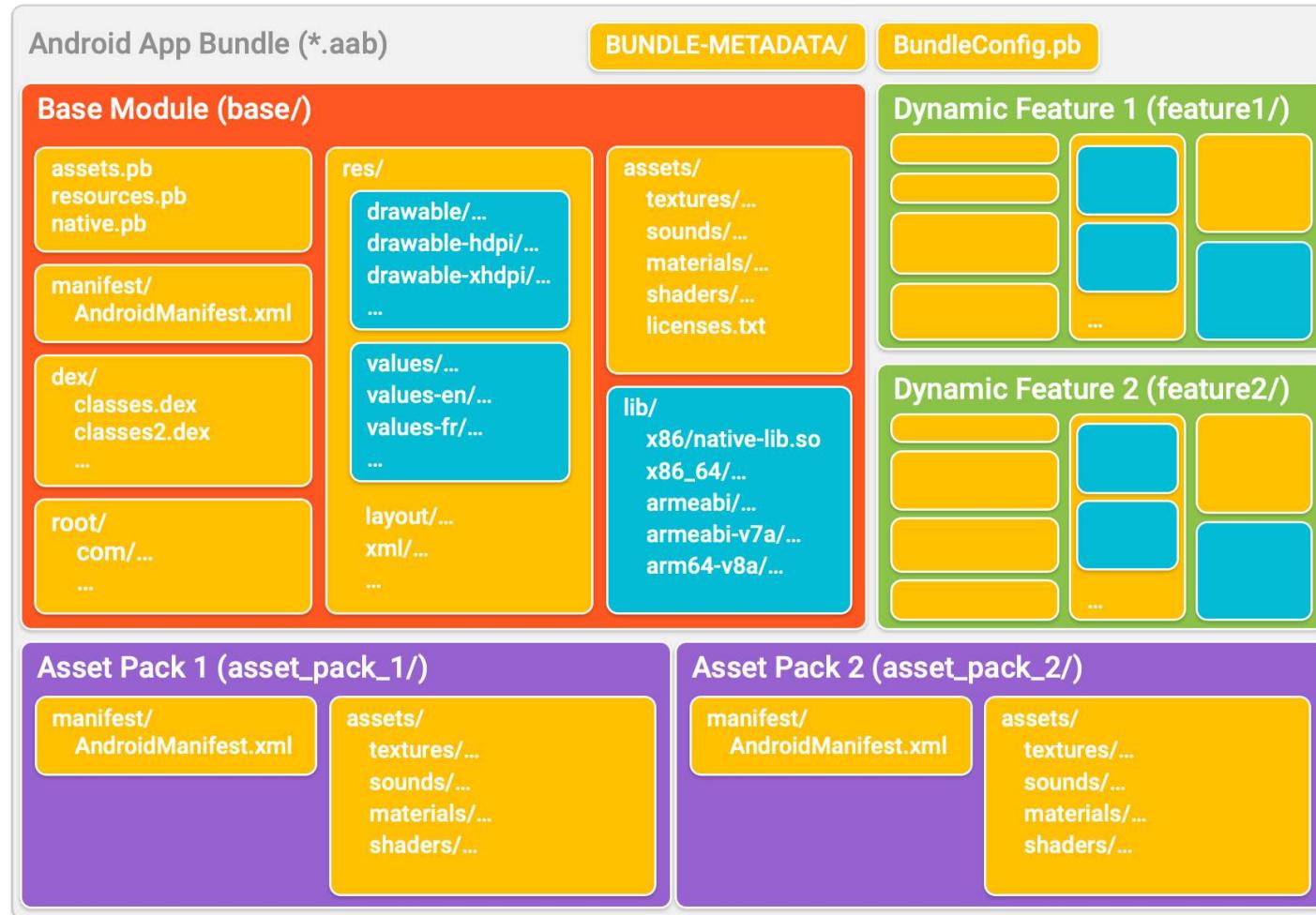
On-demand
delivery



Asset delivery
for games

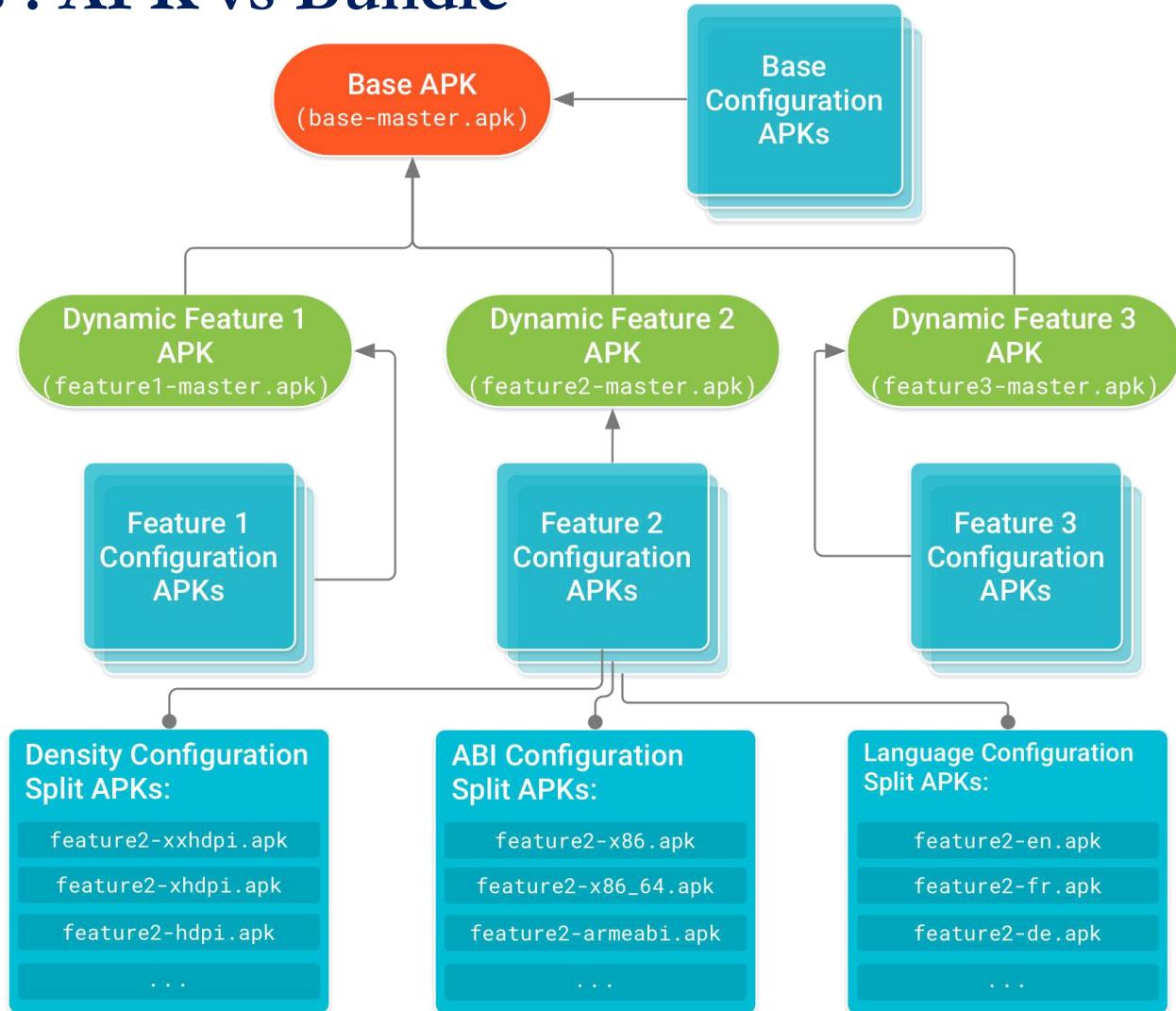
Android Resources

■ Ressources : APK vs Bundle



Android Resources

■ Ressources : APK vs Bundle



Android Resources

■ Ressources

- Lorsque l'application est compilée, Android (l'outil `aapt`) génère automatiquement une classe R (R.java) qui contient **les ID de toutes les ressources** qui se trouvent dans /res
- Ces IDs servent pour accéder aux ressources via le **code programme**
- Exemple :

R.drawable.icon = id qui permet d'obtenir une référence sur l'objet icon

R.strings.allo = id pour obtenir l'objet string hello

Ressource

- **Obtenir l'ID à partir du nom:**

```
int emId =  
    getResources().getIdentifier("emi","drawable", getPackageName());  
    //retourne R.drawable.emi
```

- **Obtenir le nom à partir de l'id:**

```
String emi = getResources().getResourceEntryName(R.drawable.emi);  
//retourne "emi"
```

```
String fullName = getResources().getResourceName(R.drawable.emi);  
//retourne "R.drawable.emi"
```

Android Resources

■ Ressources

- Utilisation dans Java :

Pour utiliser l'image : res/drawable/myimage.jpg

```
ImageView imageView = (ImageView) findViewById(R.id.myimageview);
```

```
imageView.setImageResource(R.drawable.myimage);
```

Il ne faut jamais modifier le fichier R

Démo Ressources alternatives

Technologies et développement Mobile



Pr. Slimane Bah, ing. PhD

Génie Informatique option TI & IQL

Semaine 11.2

Séance 5

Principes fondamentaux

■ Le fichier **manifest.xml**

- Le Manifest.xml est le **cœur** de toute application Android
- Il **décrit ce que contient l'application** : activités, services, broadcast receivers, content providers...Etc
- **Comment ces composantes sont liées et rattachées au système** : exemple quel composant est exécuté en premier
- Décrit les **permissions** requises par l'application
- Décrit les **besoins matériels et logiciels** de l'application

Principes fondamentaux

■ Le fichier manifest.xml

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.basicactivity"
    android:versionCode="1"
    android:versionName="1.0" >

    <uses-sdk android:minSdkVersion="8" android:targetSdkVersion="15" />

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

    <application
        ...
    </application>

</manifest>
```

Principes fondamentaux

■ Le manifest.xml

- <uses-sdk/>
- <uses-permission/>
- <permission/>
- <uses-feature/>

Android Permissions

■ Les permissions

- Le modèle de permission d'Android permet le contrôle d'accès aux :
 - Données importantes : ex. information sur les contacts
 - Ressources : ex. Caméra
 - Actions : ex. placer un appel
- Déclarées dans le Manifest

Android Permissions

■ Les permissions

Objectifs :

1. **Contrôle** : l'utilisateur est capable de prendre des décisions à la base des infos remontées
2. **Transparence** : l'utilisateur comprend quelles données sont utilisées et pourquoi
3. **Minimisation** : une App. n'accède qu'aux données requises par une action particulière (demandée par l'utilisateur)

Android Permissions

■ Les permissions

- Par défaut une application n'a aucune permission accordée.
- Une application peut déclarer :
 - Les permissions qu'elle désire utiliser
`<uses-permission/>`
 - Les permissions qu'elle exige des autres composants
`<permission/>`

Android Permissions

■ Les permissions

Android définit 4 niveaux de protection :

- **Normal**

sans danger pour le système, les autres Apps et l'utilisateur (ex. changer l'arrière plan, allumer flash)

- **Dangereux**

peuvent constituer un grand risque (ex. appels, internet)

- **Signature**

accordée aux Apps qui sont signées par le même certificat que l'app d'origine

- **SignatureOrSystem**

réservé aux fabricants. Idem que signature mais niveau système

Android Permissions

■ Les permissions

- Une permission normale est accordée automatiquement par le système

Sans déclaration

- Une permission dangereuse :

- Est déclarée dans le Manifest

- Le système demande explicitement à l'utilisateur d'accorder la permission

- `SET_ALARM`
- `SET_TIME_ZONE`
- `SET_WALLPAPER`
- `SET_WALLPAPER_HINTS`
- `TRANSMIT_IR`
- `UNINSTALL_SHORTCUT`
- `USE_FINGERPRINT`
- `VIBRATE`

Android Permissions

Permission Group	Permissions
CALENDAR	<ul style="list-style-type: none">• READ_CALENDAR• WRITE_CALENDAR
CAMERA	<ul style="list-style-type: none">• CAMERA
CONTACTS	<ul style="list-style-type: none">• READ_CONTACTS• WRITE_CONTACTS• GET_ACCOUNTS
LOCATION	<ul style="list-style-type: none">• ACCESS_FINE_LOCATION• ACCESS_COARSE_LOCATION
MICROPHONE	<ul style="list-style-type: none">• RECORD_AUDIO
PHONE	<ul style="list-style-type: none">• READ_PHONE_STATE• CALL_PHONE

Android Permissions

■ Les permissions

ACCESS_COARSE_LOCATION

ACCESS_FINE_LOCATION

ACCESS_NETWORK_STATE

ACCESS_WIFI_STATE

BATTERY_STATS

RECEIVE_BOOT_COMPLETED

RECEIVE_MMS

RECEIVE_SMS

RECEIVE_WAP_PUSH

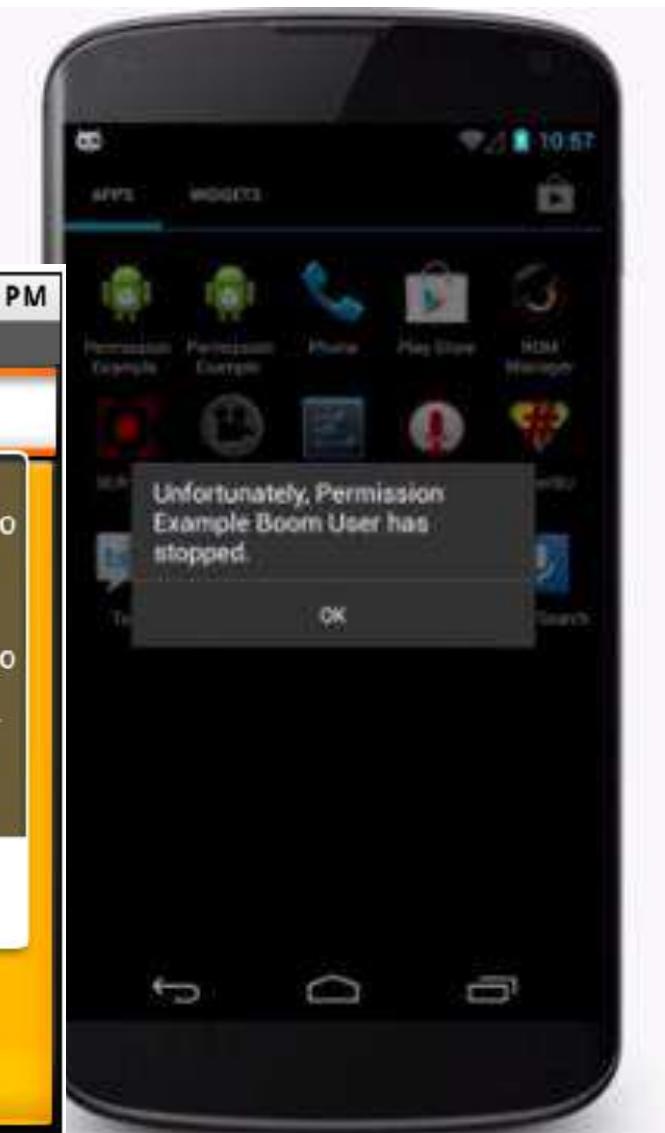
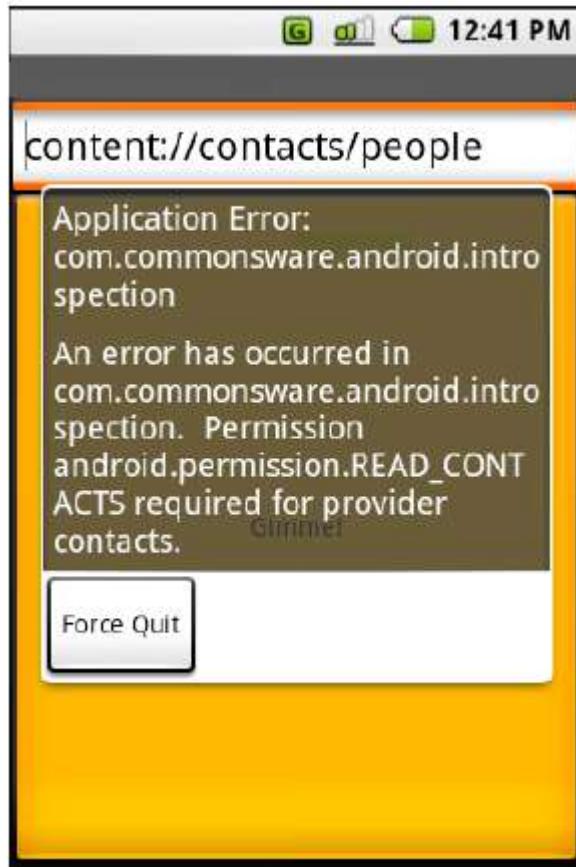
RECORD_AUDIO

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

Android Permissions

■ Les permissions

SecurityException



Android Permissions

■ Les permissions

Android définit 2 types de permissions:

- **Install-time permissions** (normale et signature) : ex. Internet
- **Runtime permissions** (dangereuse) : ex. Camera

Runtime permissions sont supportées par les terminaux avec une **API level 23 et plus.**

Android Permissions

■ Les permissions

- La demande à l'utilisateur dépend de la version :
- **Android V5.1 (API 22) et < :**
 - Permission demandée est listée dans le Manifest.xml
 - La permission dangereuse est accordée par l'utilisateur **une seule fois à l'installation**
 - L'utilisateur est informé du groupe de la permission demandée
 - La révocation se fait par désinstallation de l'application

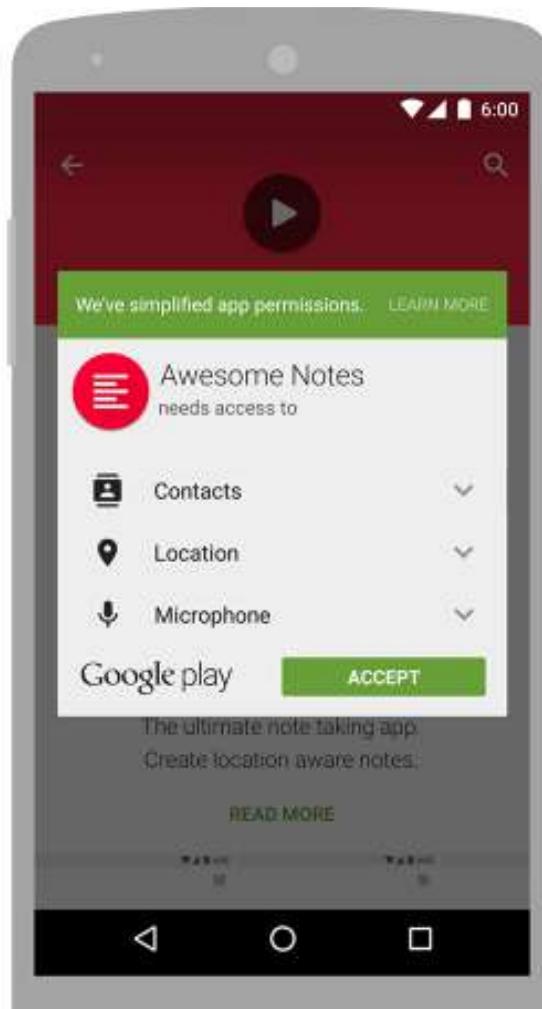
Android Permissions

■ Les permissions

- Android V5.1 (API 22) et < :

Tout ou rien

Idem pour les MAJ de l'App.



Android Permissions

■ Les permissions

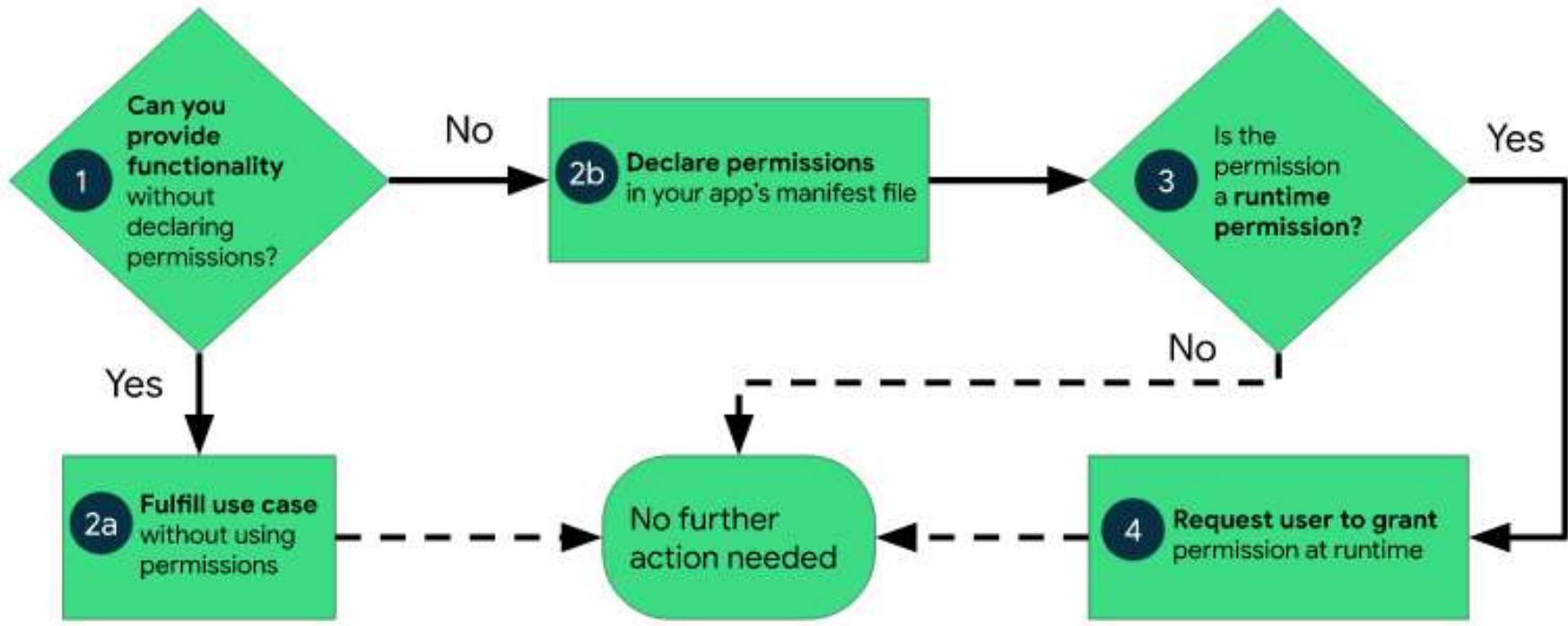
- **Android V6 (API 23 - Marshmallow) et > :**

- La demande de permission se fait à l'exécution
- Seule la permission demandée est accordée
- Permission du même groupe accordée **automatiquement** (read, write)
- L'utilisateur peut révoquer la permission **à tout moment**

Android Permissions

■ Les permissions

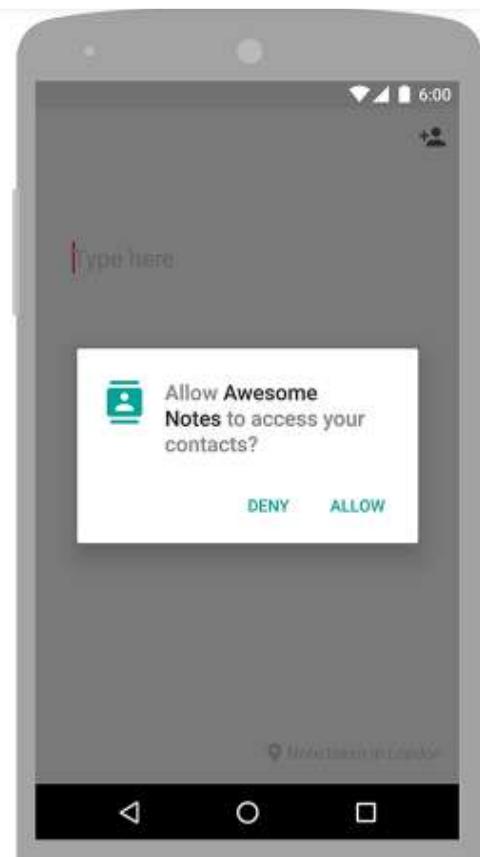
- Android V6 (API 23 - Marshmallow) et + :



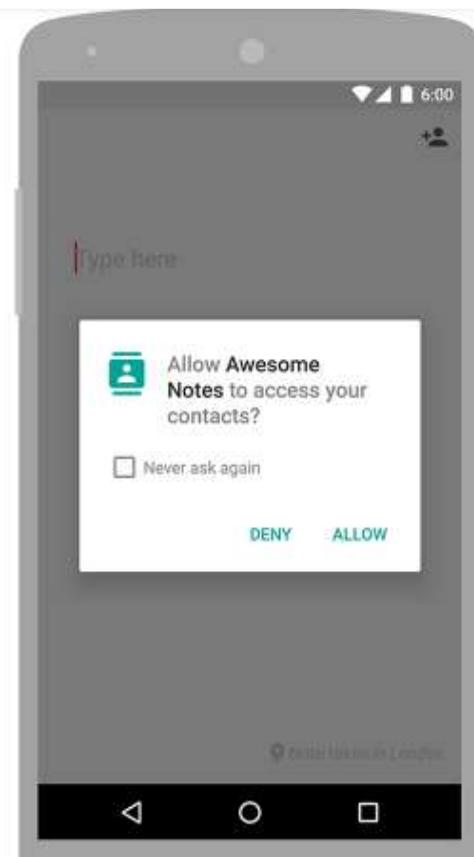
Android Permissions

■ Les permissions

Android V6 (API 23) et + :



Exec 1



Exec 2

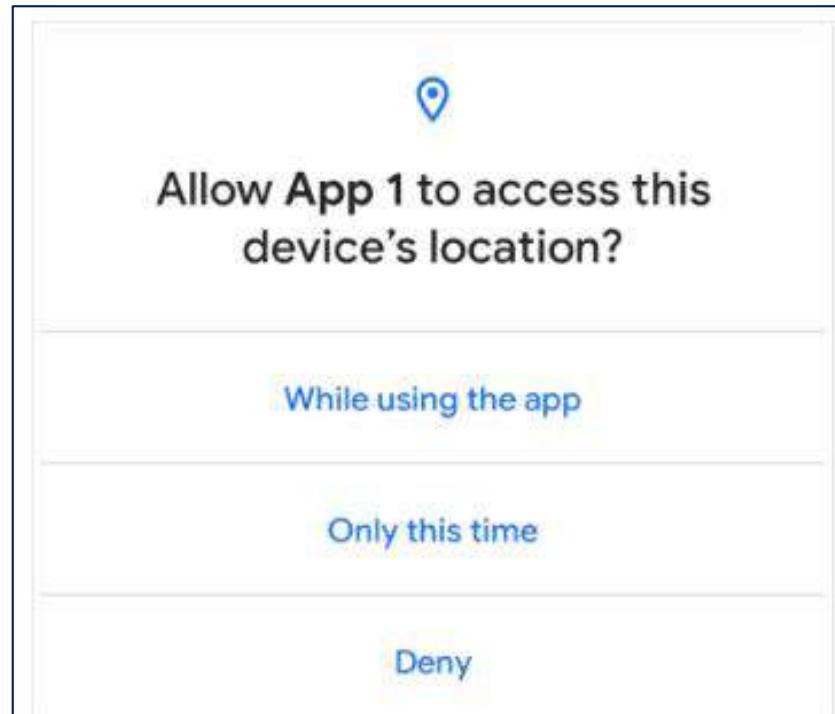
Android Permissions

■ Les permissions

Android V6 (API 23) et + :

Permission accordée

une seule fois



A partir de l'API level 30

Android Permissions

■ Les permissions

• Demander la permission à l'exécution

1. Vérifier si vous avez la permission à chaque exécution d'une opération nécessitant cette permission.

Utiliser : `ContextCompat` ou `ActivityCompat`

```
int permissionCheck = ContextCompat.checkSelfPermission(this,  
        Manifest.permission.WRITE_CALENDAR);
```

`PackageManager.PERMISSION_GRANTED` ou `PERMISSION_DENIED`

Android Permissions

■ Les permissions

- Demander la permission à l'exécution

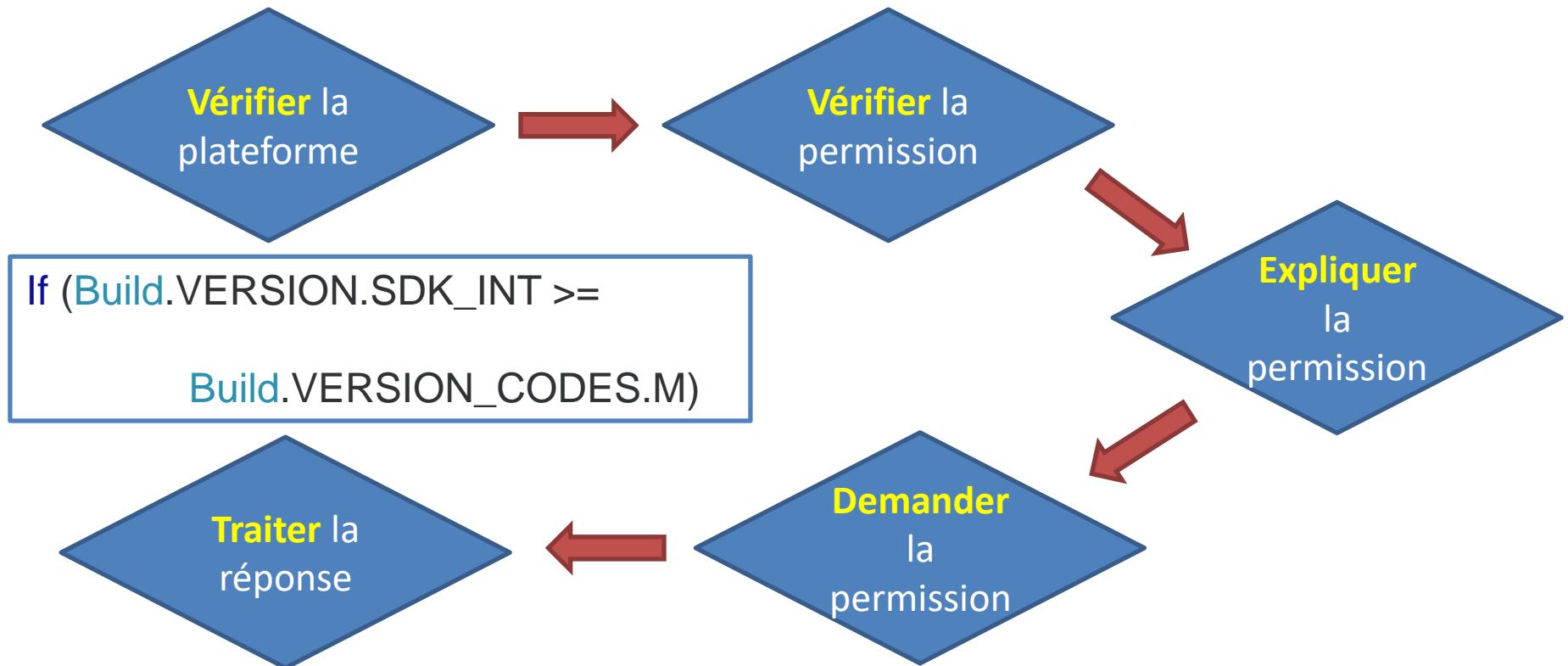
2. Demander la permission

- Le système affichera une boite de dialogue (non personnalisable) à l'utilisateur
- Possibilité de donner des explications à l'utilisateur
- Récupération du résultat de l'utilisateur et traitement

Android Permissions

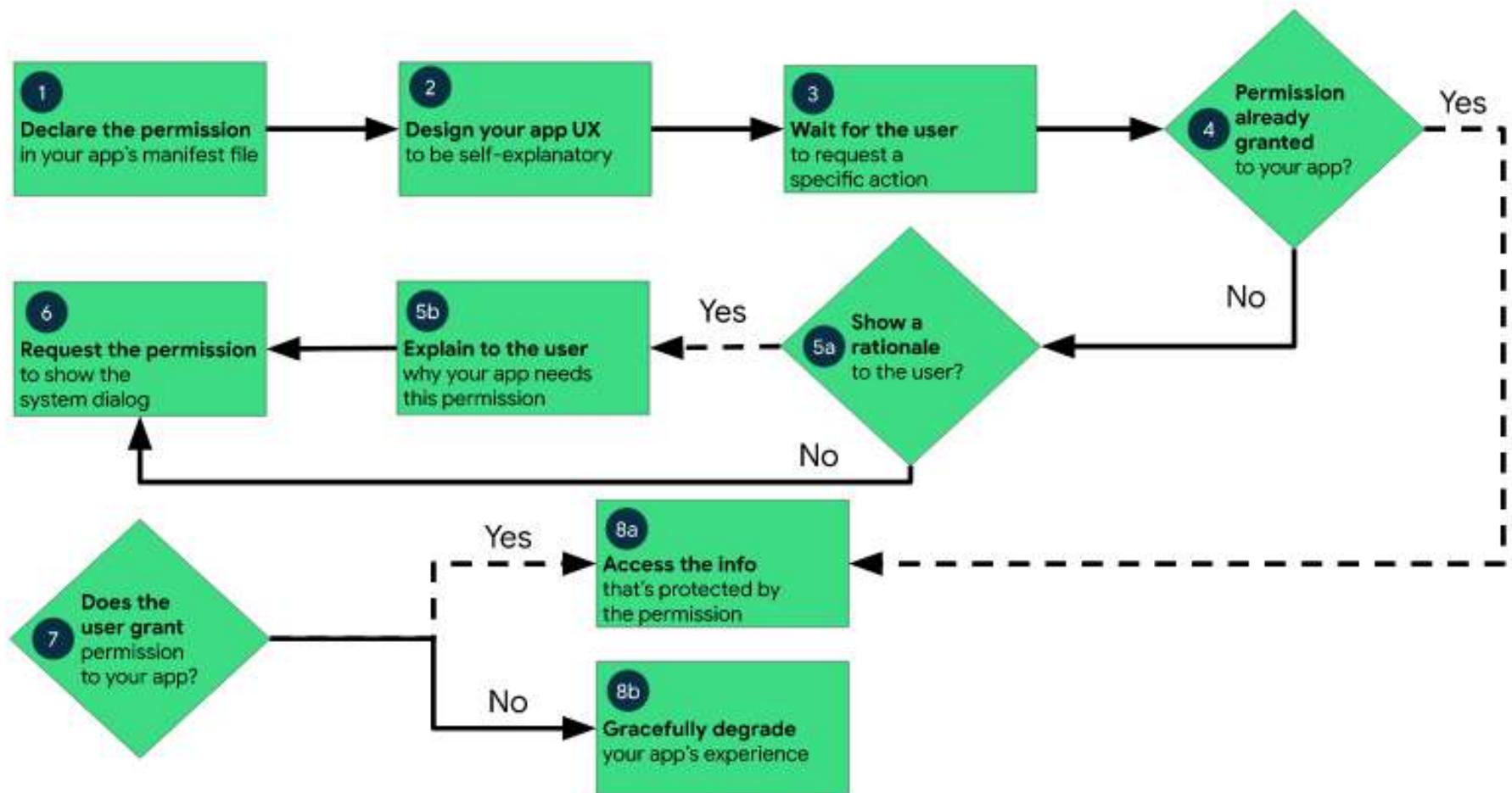
■ Les permissions

- Demander la permission à l'exécution



Android Permissions

■ Les permissions



Android Permissions

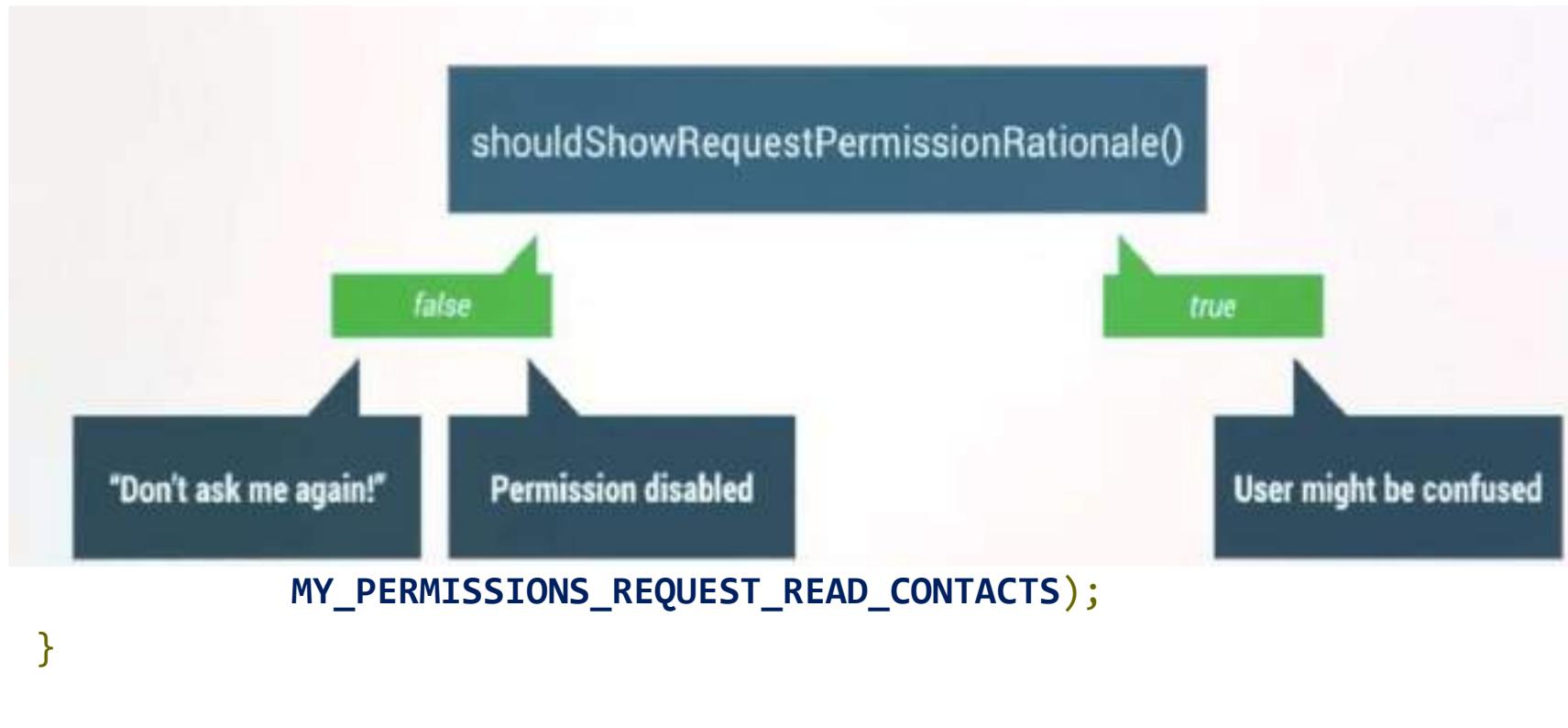
■ Les permissions

```
if //verifier la permission  
(ContextCompat.checkSelfPermission(this,Manifest.permission.READ_CONTACTS)  
!= PackageManager.PERMISSION_GRANTED) {  
  
    // Expliquer la permission  
    if (ActivityCompat.shouldShowRequestPermissionRationale(this,  
        Manifest.permission.READ_CONTACTS)) {  
        Toast.makeText(this,  
            "La permission contacts est nécessaire pour lire le contact",  
            Toast.LENGTH_SHORT).show();  
    }  
  
    ActivityCompat.requestPermissions(this,           //demander la permission  
        new String[]{Manifest.permission.READ_CONTACTS},  
        MY_PERMISSIONS_REQUEST_READ_CONTACTS);  
}  
}
```

Android Permissions

■ Les permissions

```
if //vérifier la permission  
(ContextCompat.checkSelfPermission(this, Manifest.permission.READ_CONTACTS)  
!= PackageManager.PERMISSION_GRANTED) {
```



Android Permissions

■ Les permissions

```
@Override  
public void onRequestPermissionsResult(int requestCode,  
        String permissions[], int[] grantResults) {  
    switch (requestCode) {  
        case MY_PERMISSIONS_REQUEST_READ_CONTACTS: {  
            // If request is cancelled, the result arrays are empty.  
            if (grantResults.length > 0 && grantResults[0] ==  
                PackageManager.PERMISSION_GRANTED) {  
                // Youhou ! permission was granted  
            } else {  
                // permission denied, boo! Disable the  
                // functionality that depends on this permission.  
            }  
            return;  
        }  
        // other 'case' lines to check for other  
    }  }
```

Android Permissions

■ Les permissions

- En demandant certaines permissions le système (Google Play) suppose que votre App. nécessite la fonctionnalité correspondante (ex. caméra)
- L'App. Ne pourra pas s'installer dans un tél sans cette fonctionnalité
- Pour éviter ce filtrage utiliser **<uses-feature>**:

```
<uses-feature android:name="android.hardware.camera" android:required="false" />
```

Et vérifier si la fonctionnalité est disponible :

PackageManager.hasSystemFeature("android.hardware.camera.any")

Android Permissions

Les permissions personnalisées

(section décalée à plus tard dans le cours)

Cycle de vie d'une Activité

Activité et Intent

■ Android Task

- En général, une **application** est composée de plusieurs **Activités**
- L'utilisateur **navigue** à travers ces différentes activités
- Pour gérer cette navigation, **Android** offre le principe de:
 - **Tâche**
 - **Task BackStack** et
 - **Un cycle de vie** des activités
 - cycle de vie du service, cycle de vie du fragment

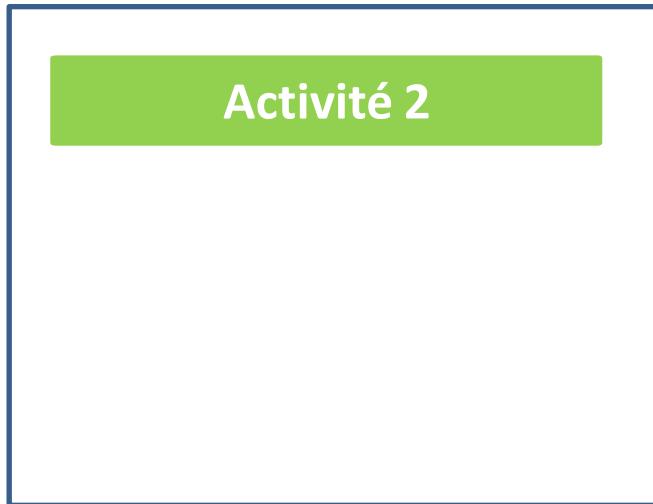
Activité et Intent

■ Android Task

- Une tâche est un ensemble d'activités liées
- Les activités d'une tâche **n'appartiennent pas** forcément à une même application
- Le BackStack sert à organiser la séquence d'exécution des activités :
 - Empilée lorsqu'elle est démarrée
 - Dépilée lorsqu'elle est terminée
- Les tâches démarrent à partir de l'écran d'accueil

Activité et Intent

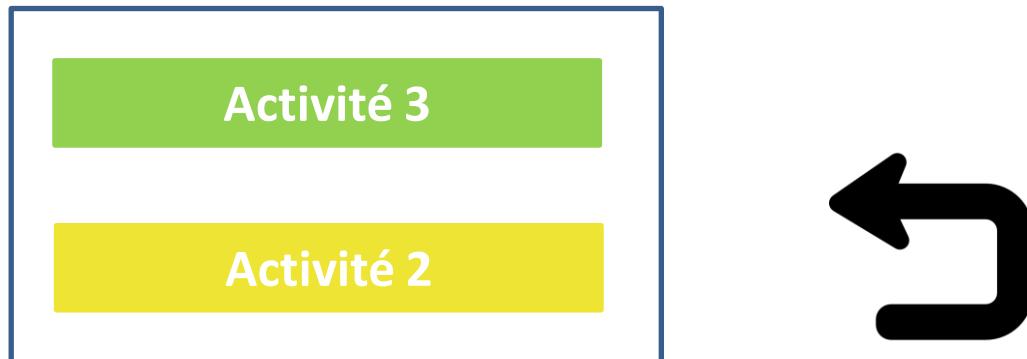
■ BackStack



Back Stack

Activité et Intent

■ BackStack



Back Stack

Activité et Intent

■ BackStack



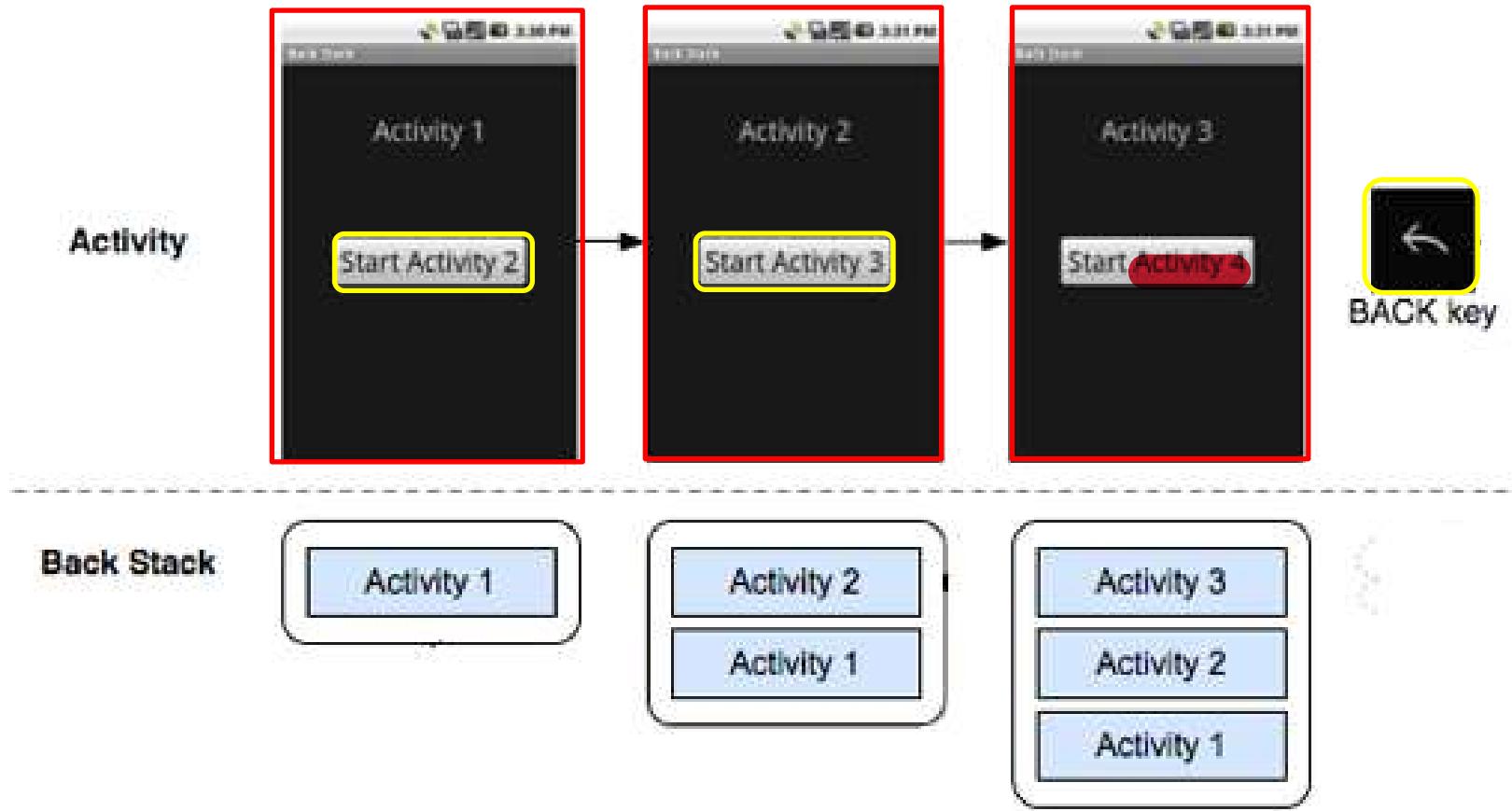
Activité et Intent

■ BackStack



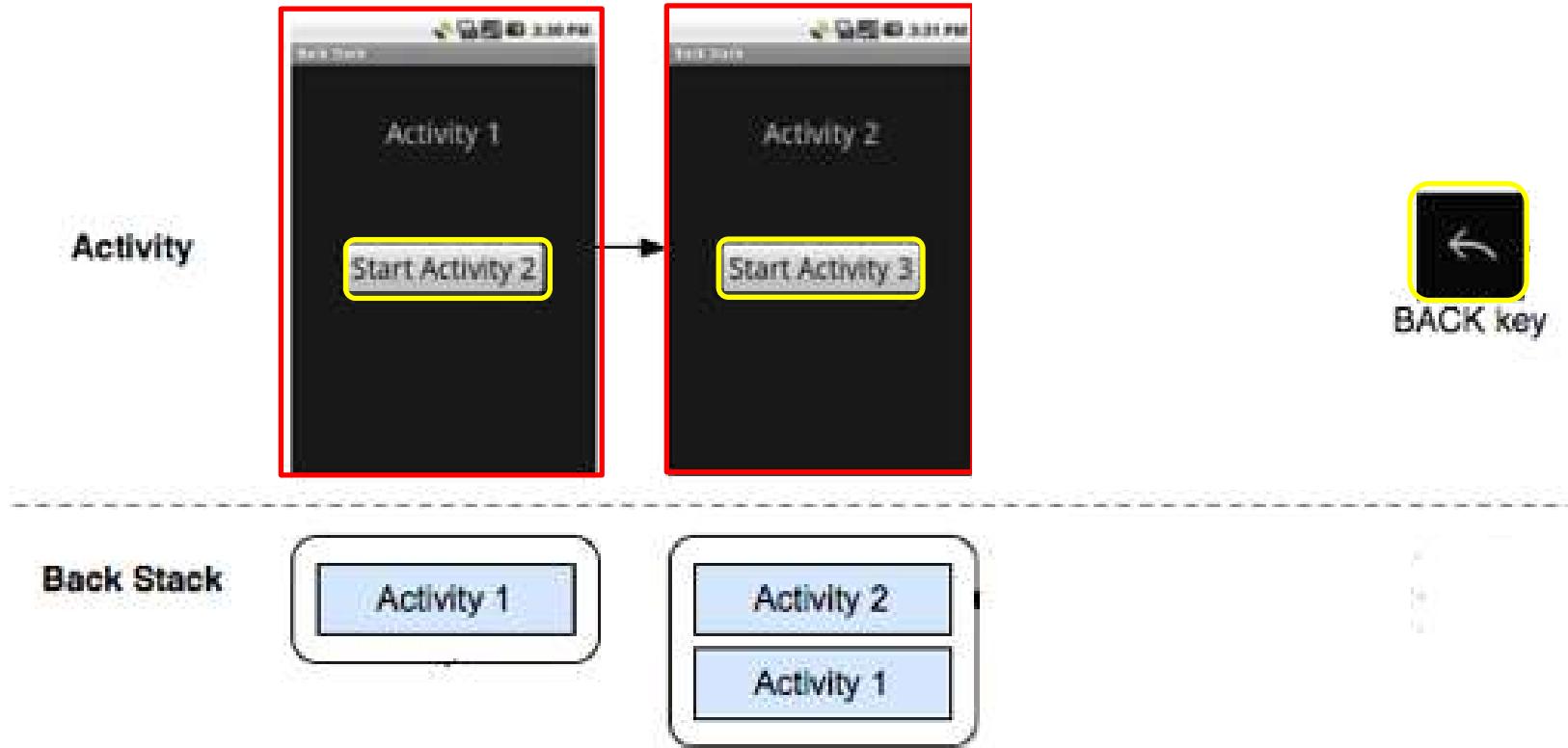
Activité et Intent

■ BackStack



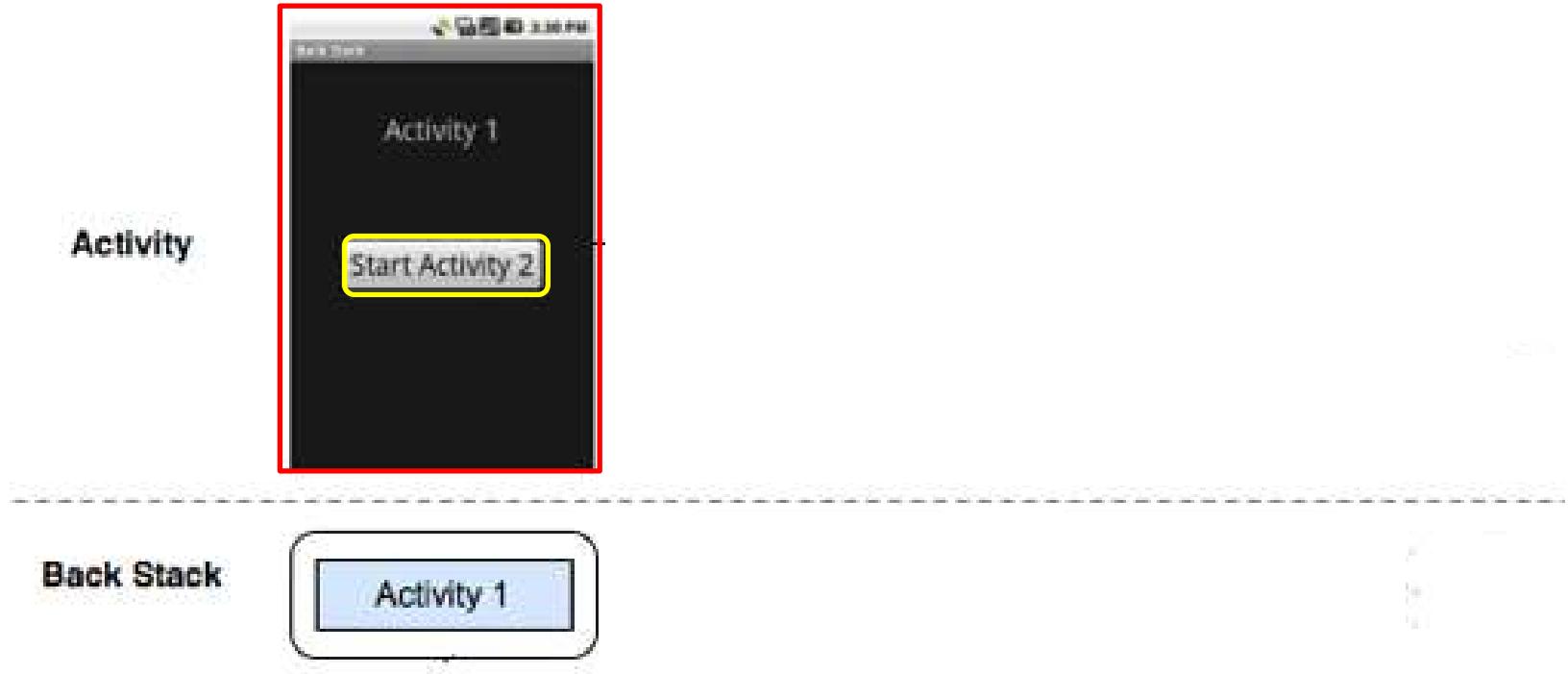
Activité et Intent

■ BackStack



Activité et Intent

■ BackStack

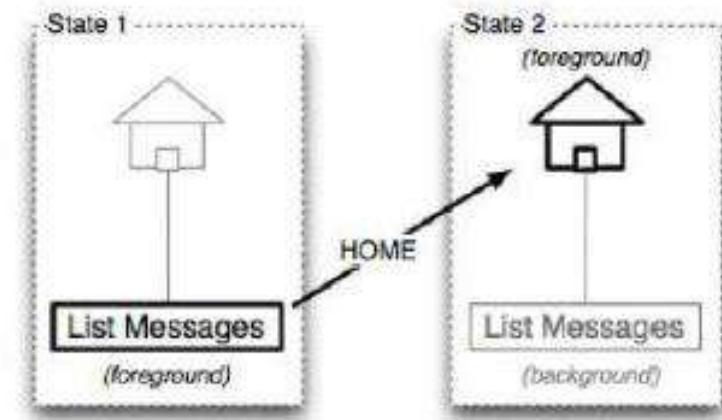
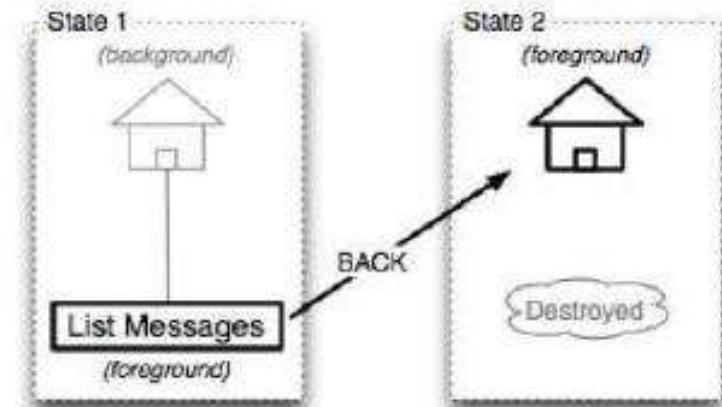
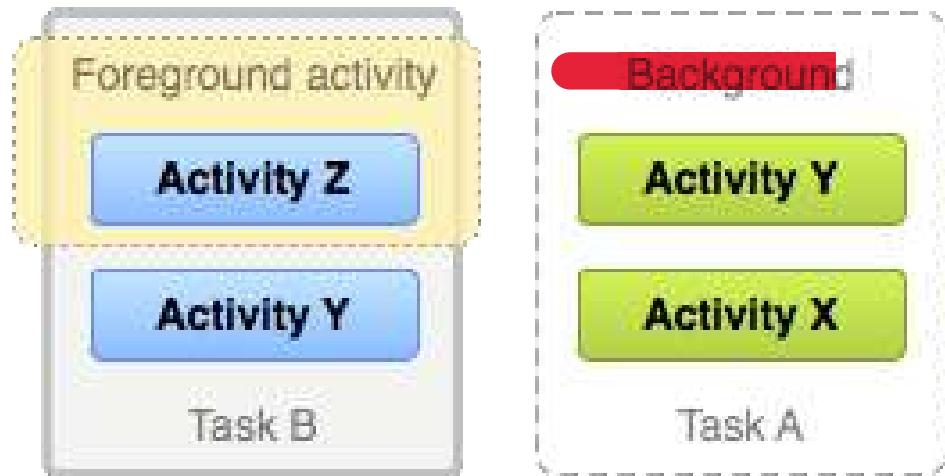


Que se passe-t-il si on clique sur le bouton **Home** ?

Activité et Intent

■ Android Task

- La tâche passe en arrière plan



Activité et Intent

■ Android Task

- Comportement par défaut :

- 1- Si **A** démarre **B**. L'activité **A** s'arrête et Android garde son état. Si le bouton **Back** est cliqué **A** devient active avec son état restauré.
- 2- Le bouton Back dépile et **détruit l'activité courant** (au sommet du backstak)

Activité et Intent

■ Android Task

3- Bouton Home = Arrêt de l'activité en cours et sa tâche (Task) passe en arrière plan. Android garde l'état de toutes les activités de la tâche.

En cliquant sur l'icone de l'activité qui a démarré la tâche, celle-ci revient au premier plan.

4- Une activité peut être instanciée plusieurs fois dans la même tâche ou dans d'autres tâches

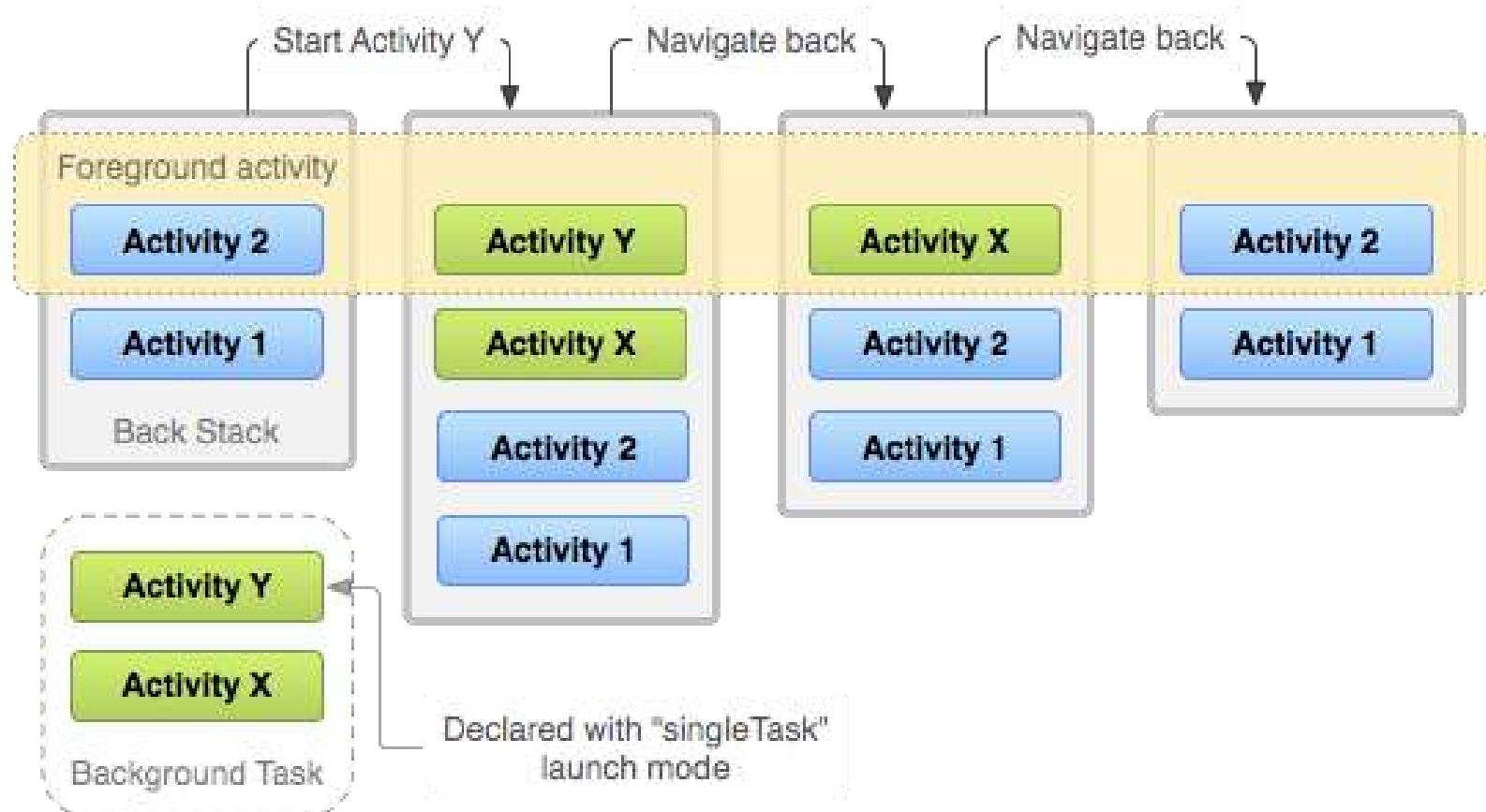
Activité et Intent

■ Gestion des tâches Android

- Le comportement par défaut est suffisant mais peut être modifié
- La modification peut se faire soit :
 - Par les attributs de l'activité dans le Manifest
 - A travers les paramètres de l'Intent lors de l'appel de l'activité
- Exemple : Attribut **LaunchMode**
 - standard : instances multiples
 - singleTop : réutilisation de la même instance si au top de la pile
 - singleTask : (nouvelle tâche + instancie l'activité) sauf si existe déjà
 - singleInstance : l'activité ne partage la tâche avec aucune autre activité

Activité et Intent

■ Gestion des tâches Android



Technologies et développement Mobile



Pr. Slimane Bah, ing. PhD

Génie Informatique option TI & IQL

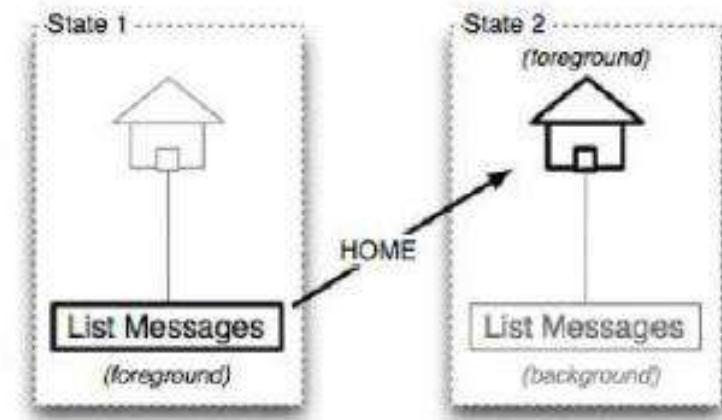
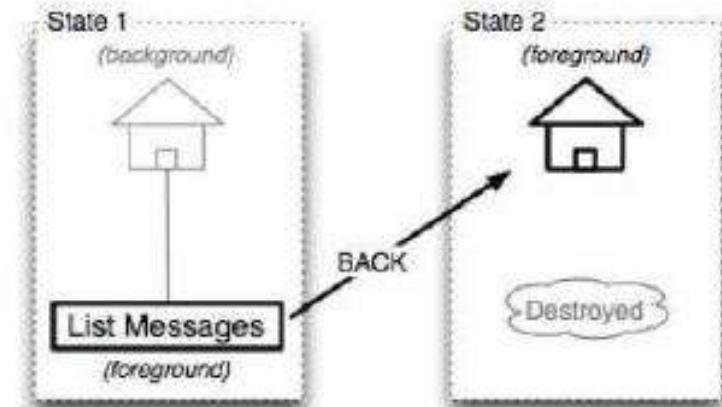
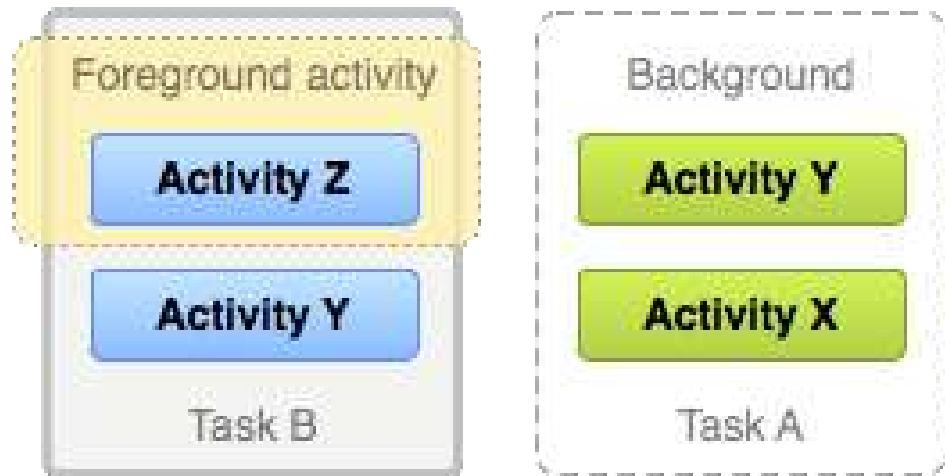
Semaine 12.1

Séance 6

Activité et Intent

■ Android Task

- La tâche passe en arrière plan



Activité et Intent

■ Gestion des tâches Android

- Le comportement par défaut est suffisant mais peut être modifié
- La modification peut se faire soit :
 - Par les attributs de l'activité dans le Manifest
 - A travers les paramètres de l'Intent lors de l'appel de l'activité
- Exemple : Attribut **LaunchMode**
 - standard : instances multiples
 - singleTop : réutilisation de la même instance si au top de la pile
 - singleTask : (nouvelle tâche + instancie l'activité) sauf si existe déjà
 - singleInstance : l'activité ne partage la tâche avec aucune autre activité

Activité et Intent

■ Cycle de vie

- Android gère les activités et leurs états à travers un cycle de vie imposé
- Les changements d'états dépendent soit :

1- Utilisateur

2- Android

Activité et Intent

■ Cycle de vie

- Les états d'une activité :

1. Active / en exécution

Visible, interaction avec utilisateur

2. En pause

semi-visible, perdu le focus, peut-être tuée

3. Stoppée

non visible, peut-être tuée

4. Morte

n'existe plus/pas dans le backstack

Activité et Intent

■ Cycle de vie

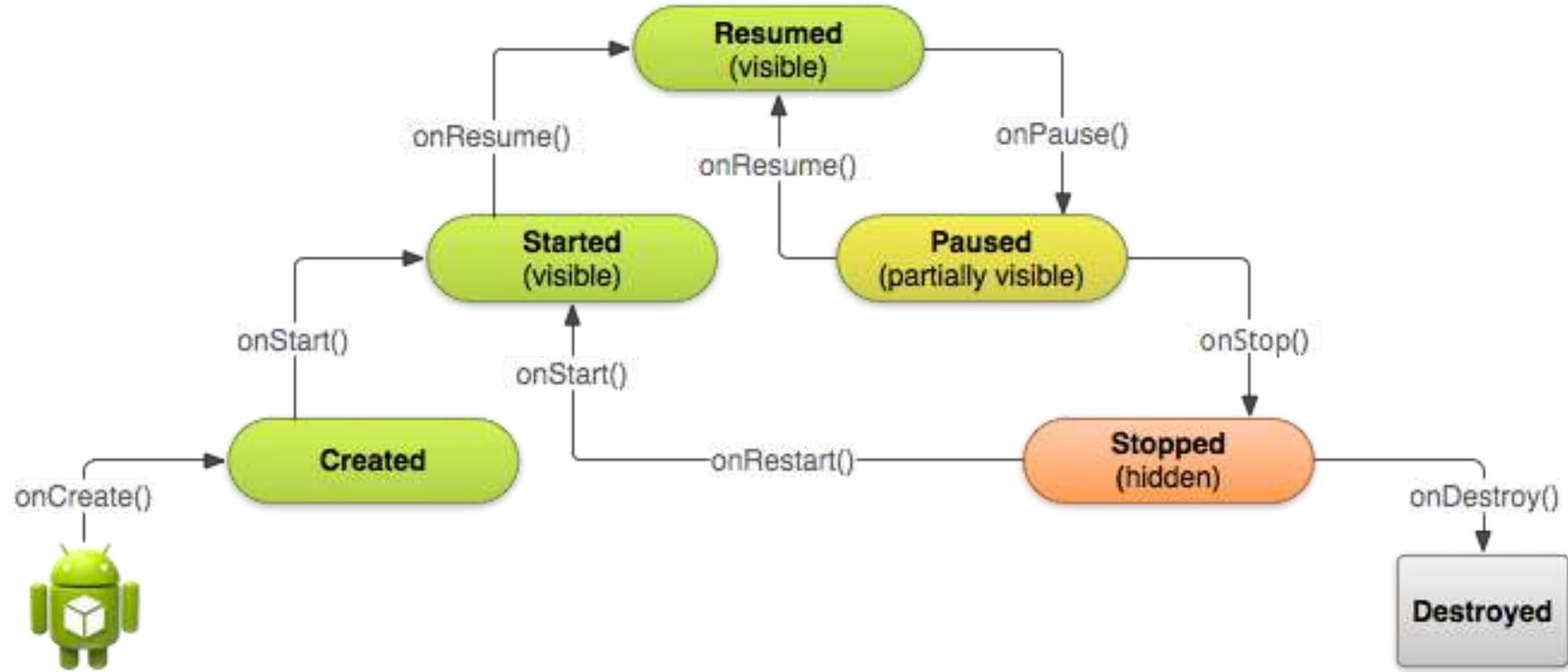
- Android informe une activité des changements d'état grâce à des appels de méthodes spécifiques

```
void onCreate(Bundle savedInstanceState)  
void onStart()  
void onRestart()  
void onResume()  
void onPause()  
void onStop()  
void onDestroy()
```

- Pour faire des actions lors de changements d'états il faut redéfinir ces méthodes

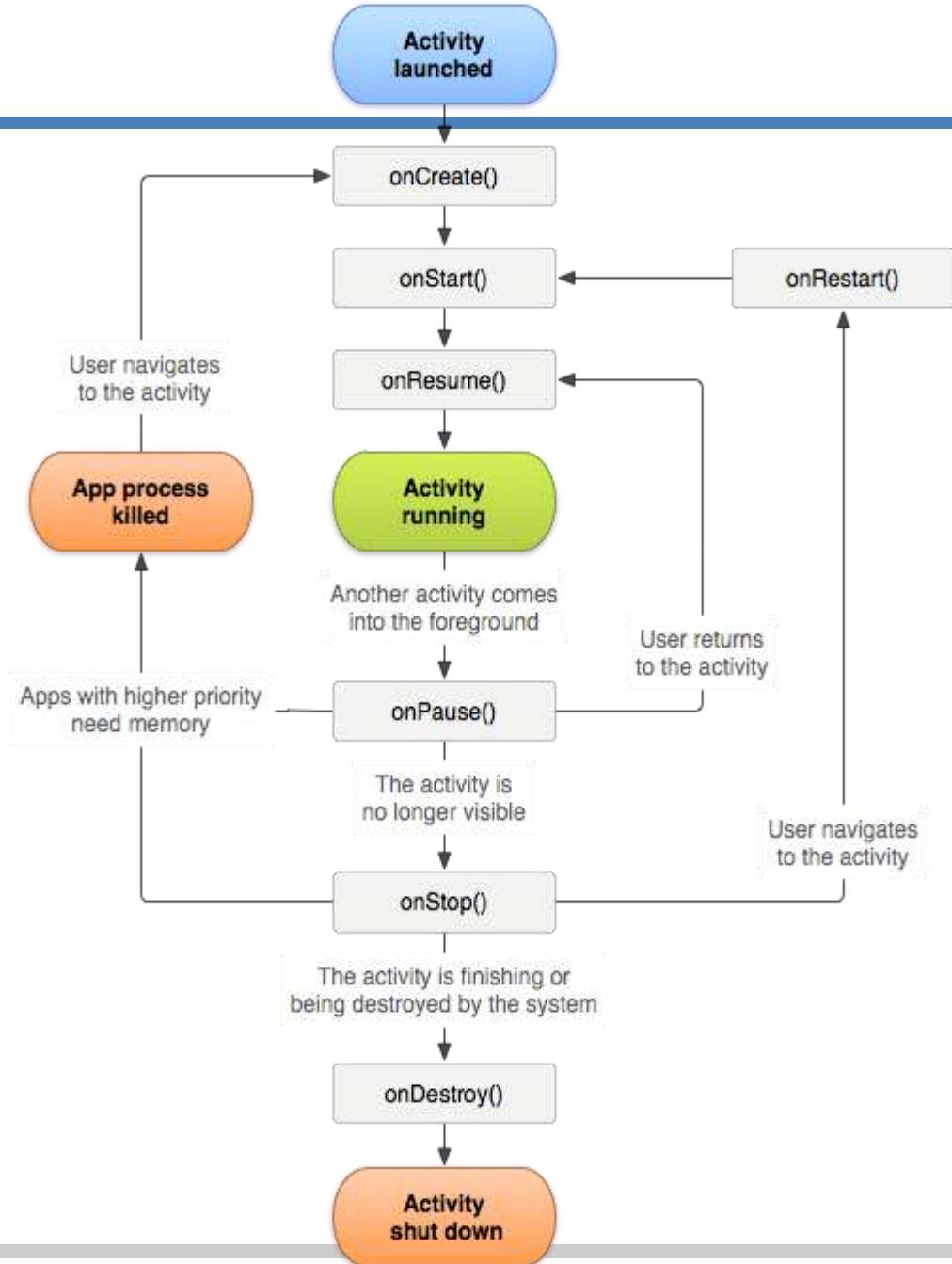
Activité et Intent

■ Cycle de vie



Activité et Intent

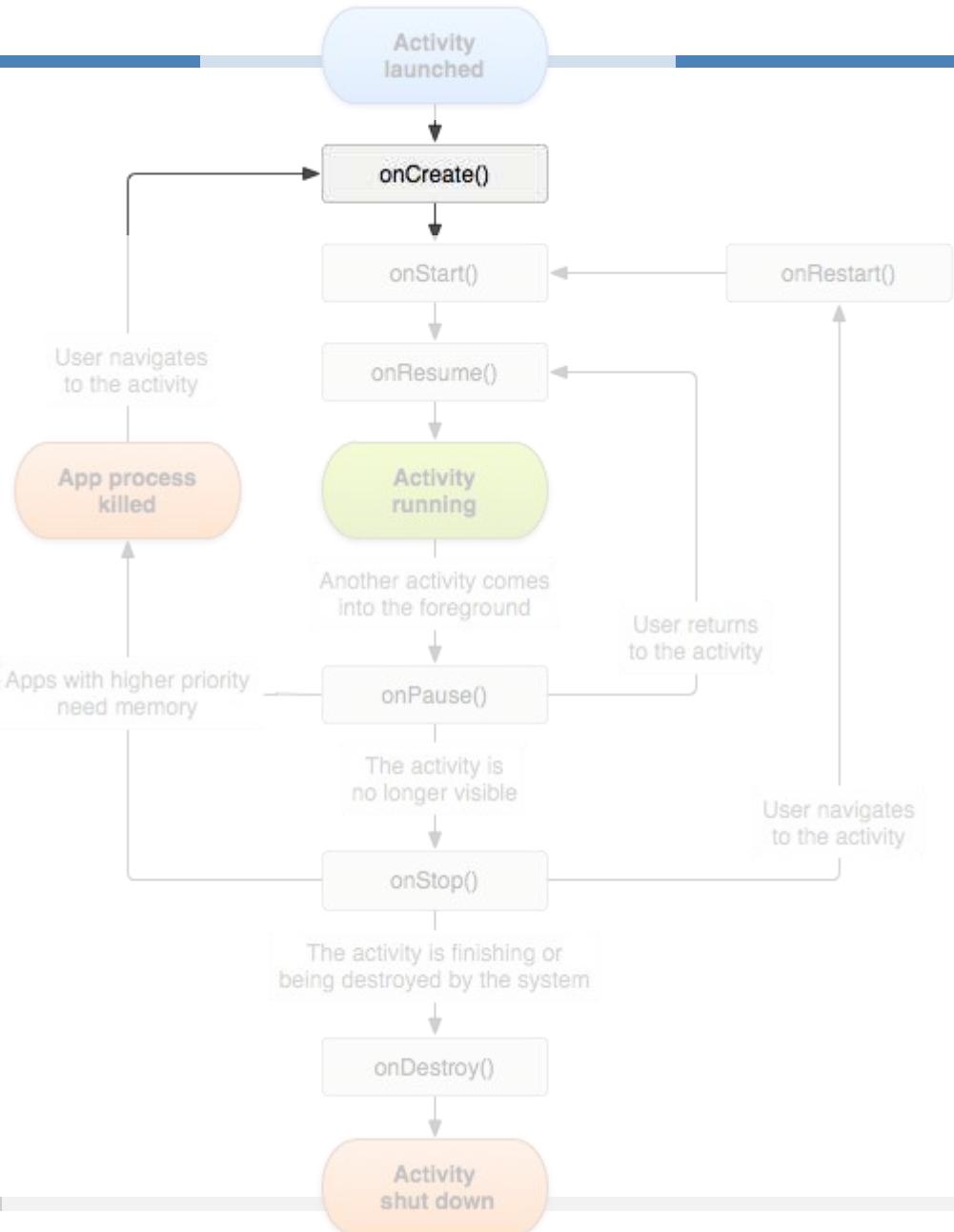
■ Cycle de vie



Activité et Intent

- **onCreate**

- Est appelée à la création de l'activité
- Contient les opérations d'initialisation
- Un *Bundle(Map)* est en paramètre
- Si *onCreate* se termine avec succès, alors *onStart* est appelée



Utilisation :

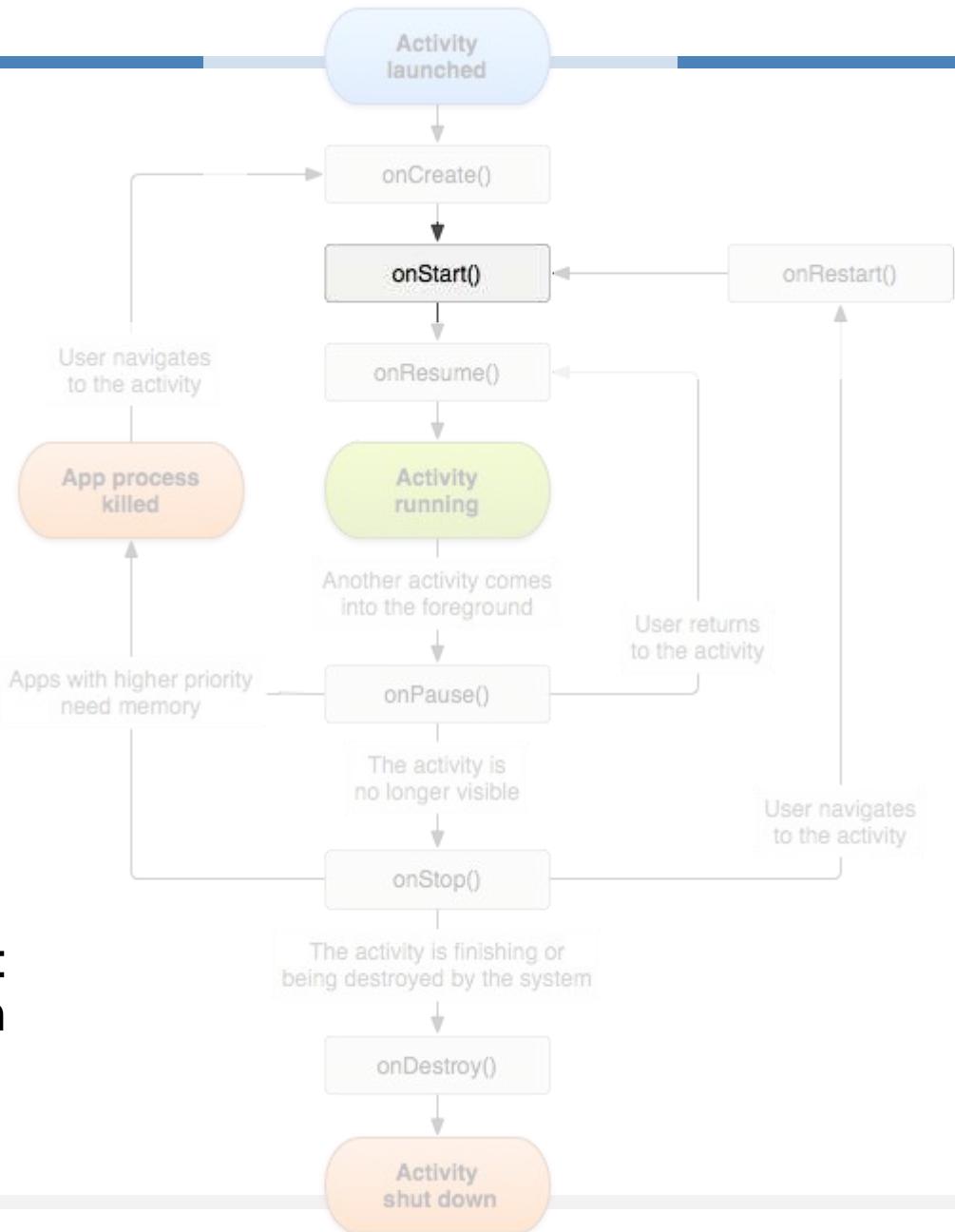
- setup de la vue (UI)
- obtenir les références sur les objets UI (éventuellement)
- Configuration de la vue (listeners...)

Activité et Intent

- **onStart**
 - Est exécutée à la fin de *onCreate*
 - Est appelée avant que l'activité soit visible à l'utilisateur

Utilisation :

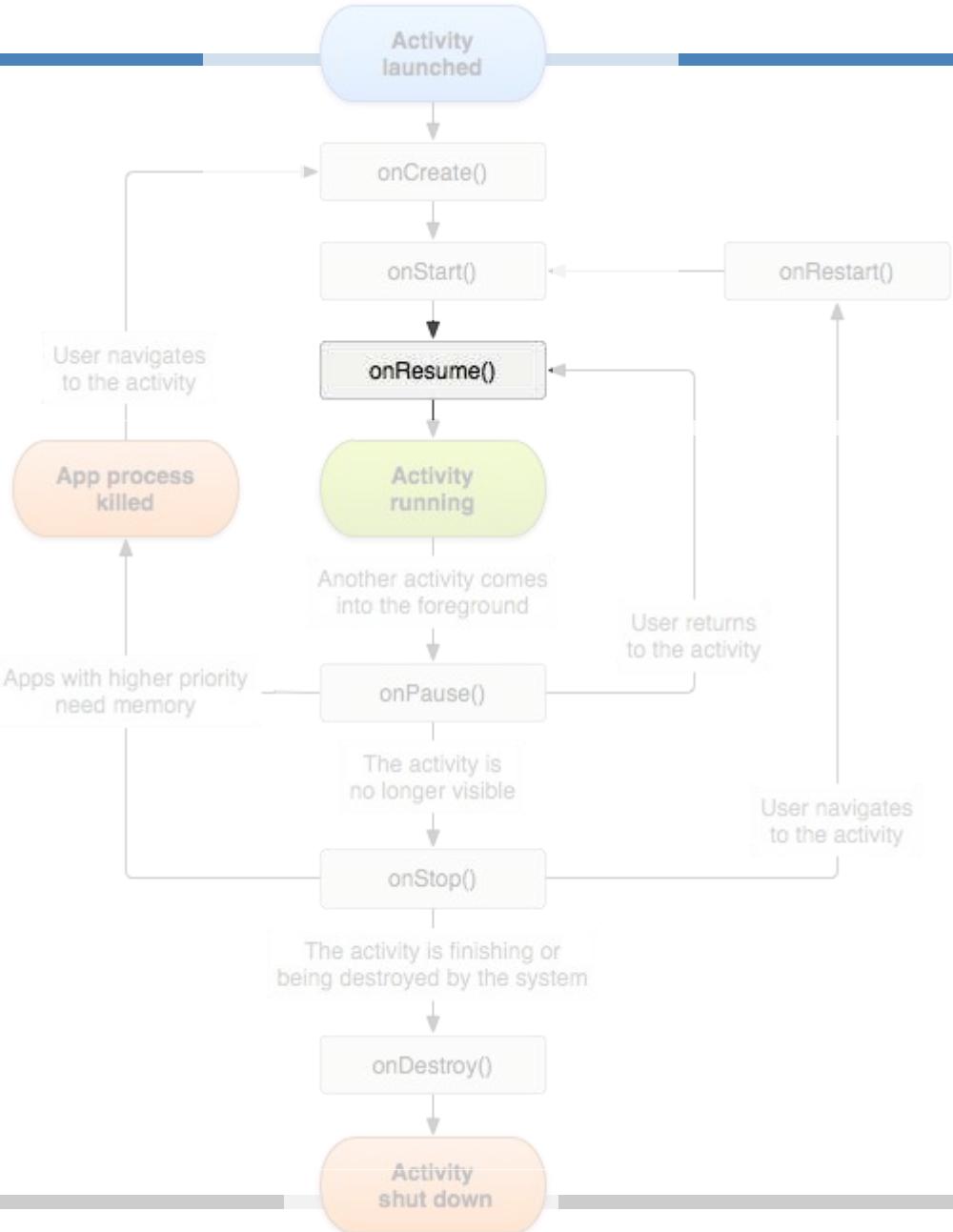
- Traitement lorsque visible : mettre à jour la position géo
- Chargement d'état persistant : télécharger les messages non lus



Activité et Intent

- **onResume**

- Est appelée quand l'activité est prête à interagir avec l'utilisateur
- Est également appelée quand l'activité reprend son exécution
- Si *onResume* se termine avec succès alors l'activité est en **phase d'exécution**



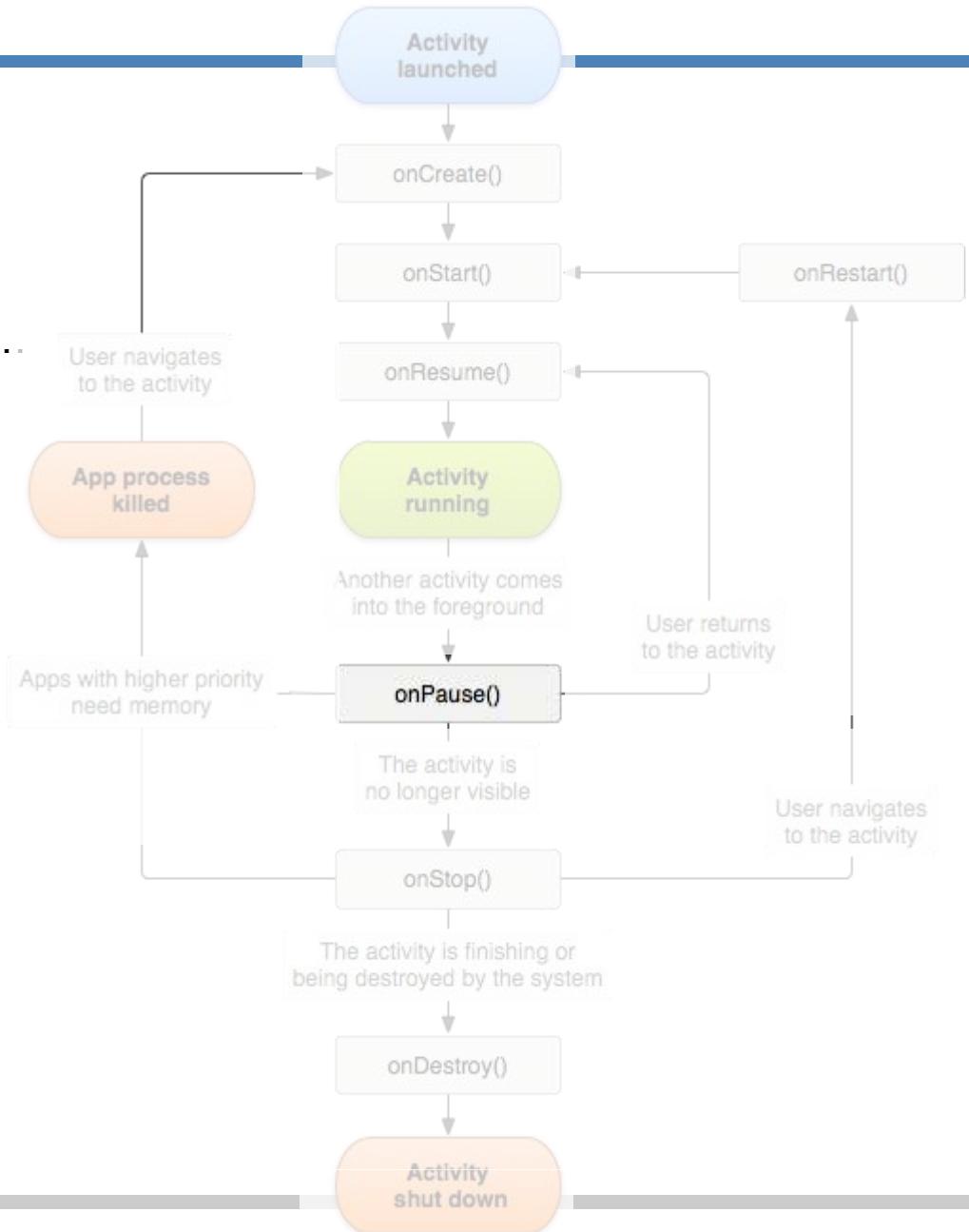
Utilisation :

- Traitements premier plan:
jouer une animation, jouer
musique en arrière plan

Activité et Intent

- **onPause**

- Est appelée quand une autre activité passe au premier plan ou quand la touche retour est appuyée
- Doit libérer les ressources capteurs, ...
- Est arrêtée en fin de onPause(pas de temps CPU),
- *Android peut décider de l'arrêt par destruction de cette activité à tout moment*



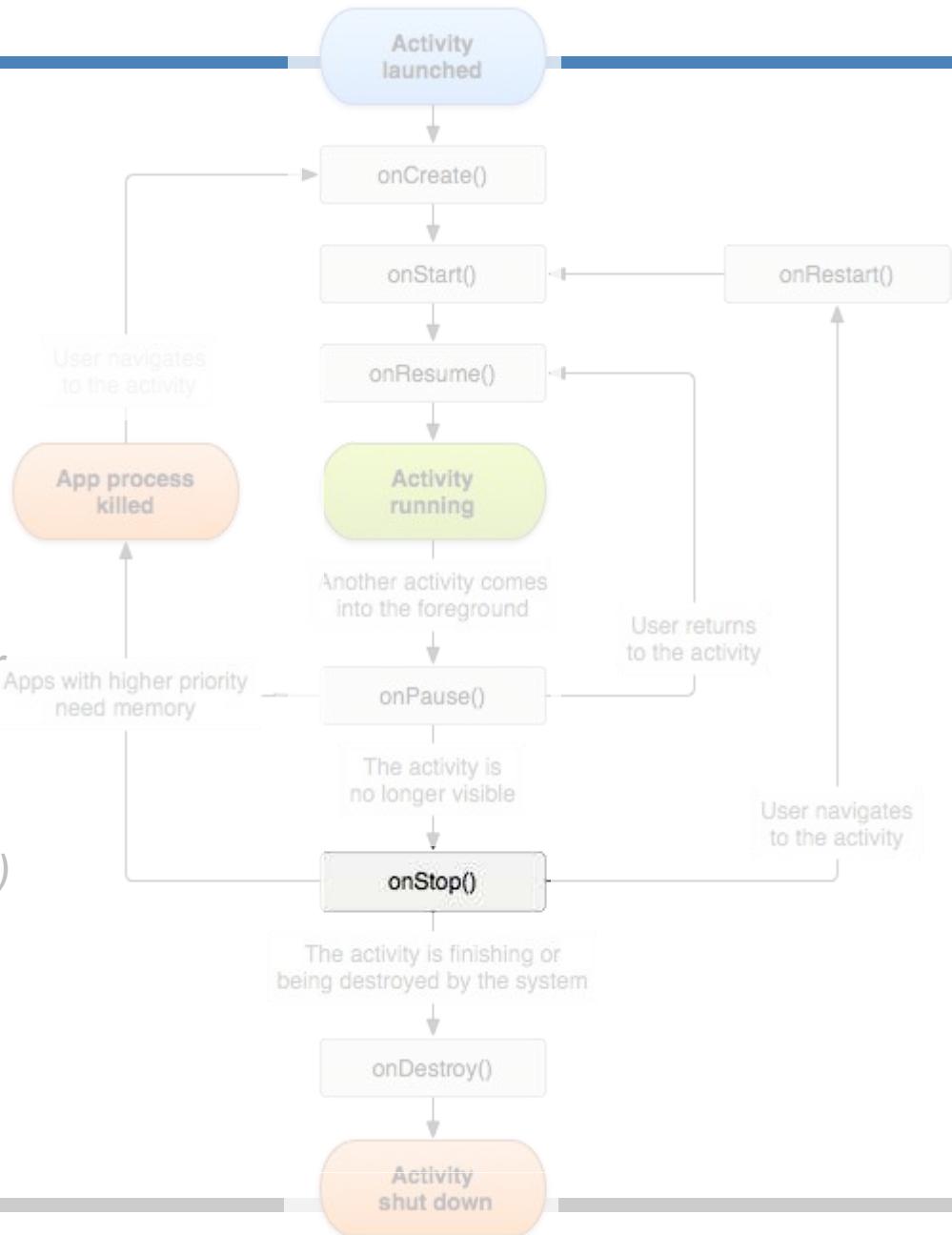
Utilisation :

- Terminer les traitements de premier plan
- **Sauvegarder les états** : Persistance
`onSaveInstanceState()`
`onRestoreInstanceState()`

Activité et Intent

- **onStop**

- L'activité n'est plus visible à l'utilisateur
- Est appelée quand
 - L'activité s'apprête à être détruite,
 - Une autre activité est passée au premier plan.
- *Android peut décider de l'arrêt par destruction de cette activité à tout moment*
- *N'est pas appelée si l'OS décide de tuer l'appli. (Low memory killer)*



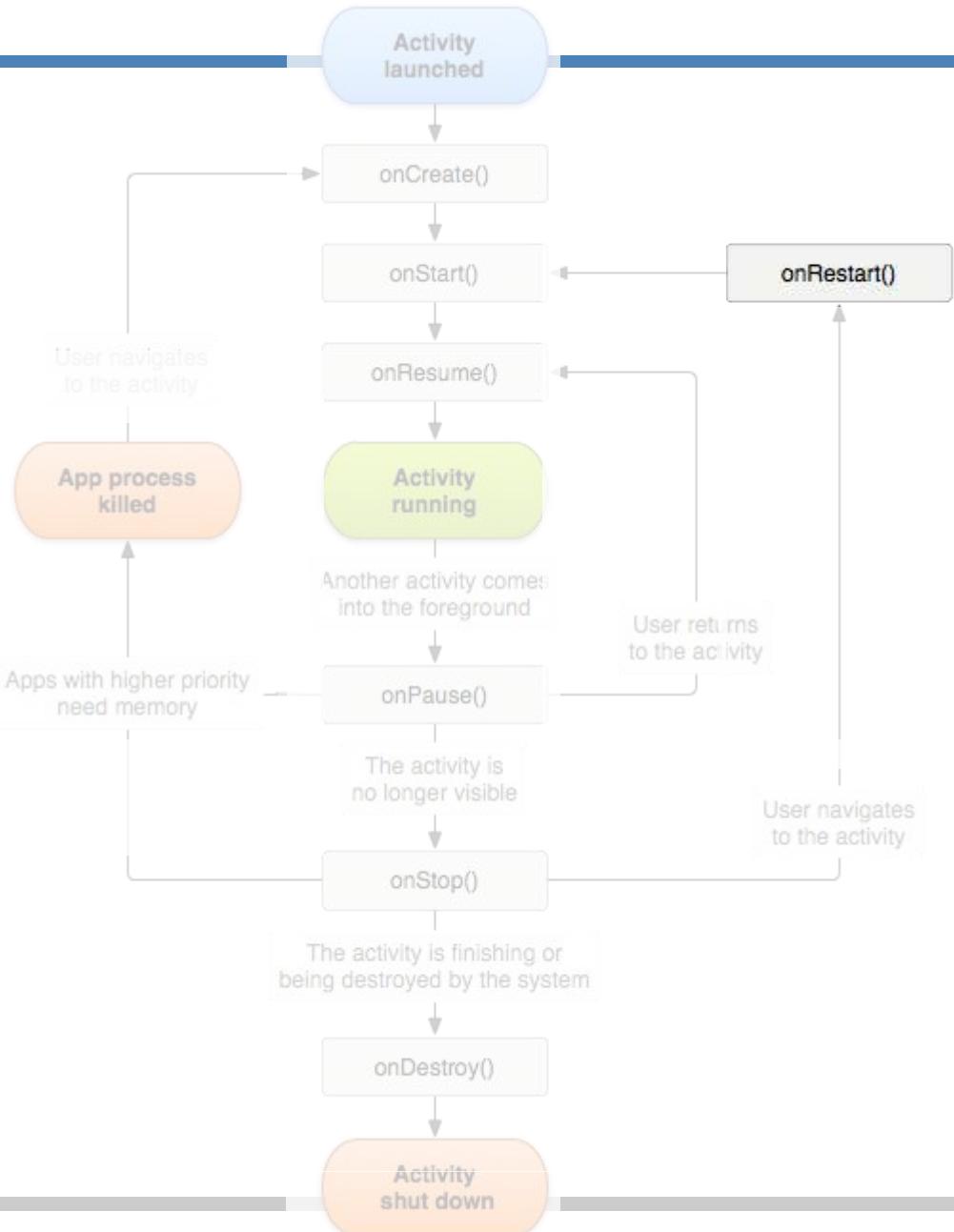
Utilisation :

- mettre en cache l'état de l'appli.

Activité et Intent

- **onRestart**

- Est appelée quand l'activité a été préalablement stoppée
- L'utilisateur a de nouveau sélectionné cette activité
- Réactiver un curseur BD



Activité et Intent

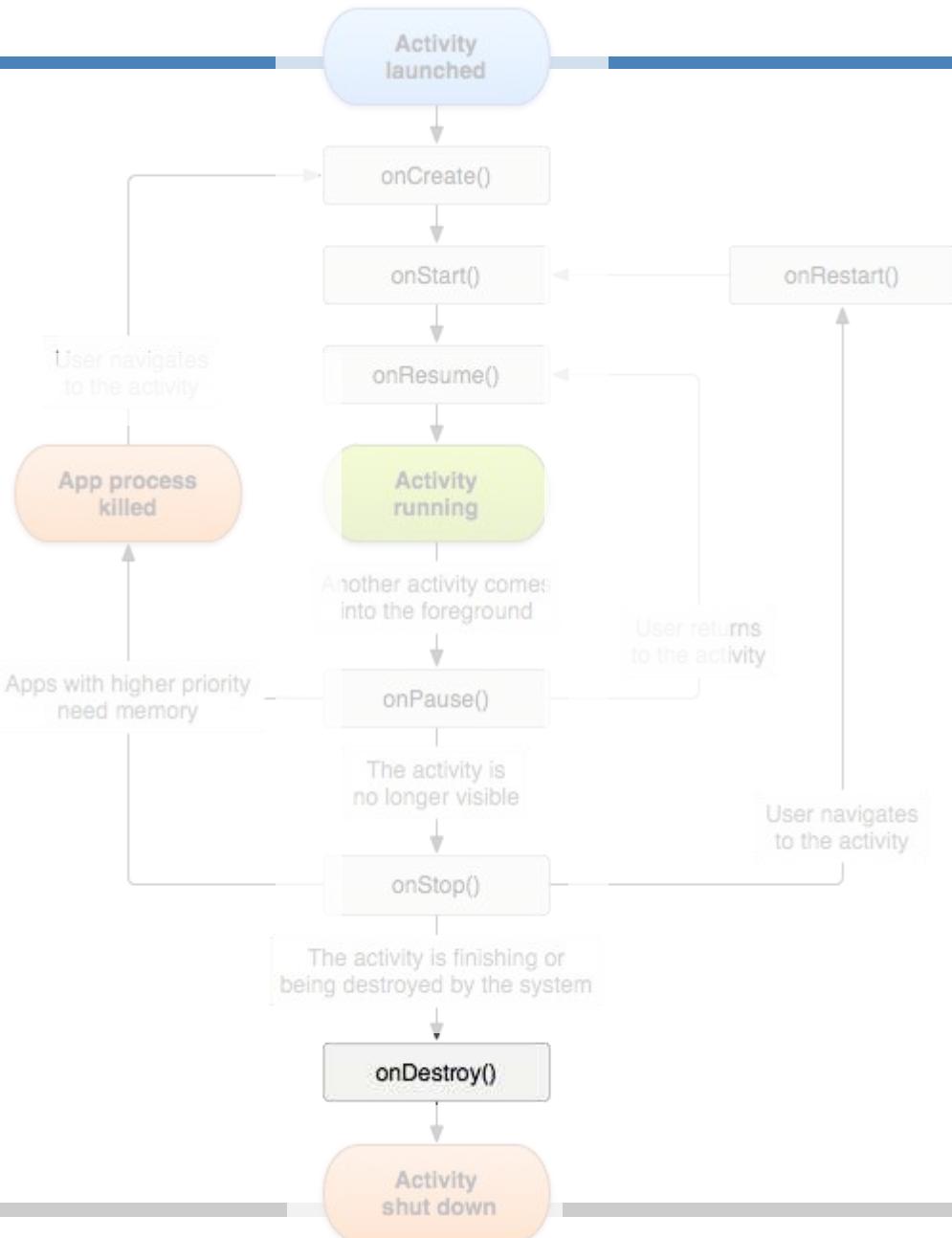
- **onDestroy**
 - L'activité va être détruite

Le système a besoin de place ou bien la méthode ***finish()*** a été appelée

- Peut ne pas être appelée dans le cas du Low memory killer

Utilisation :

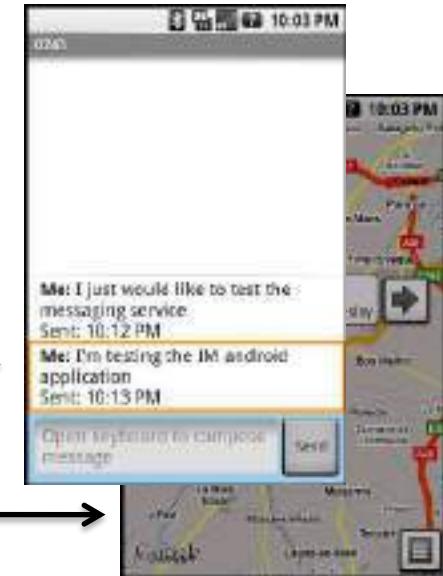
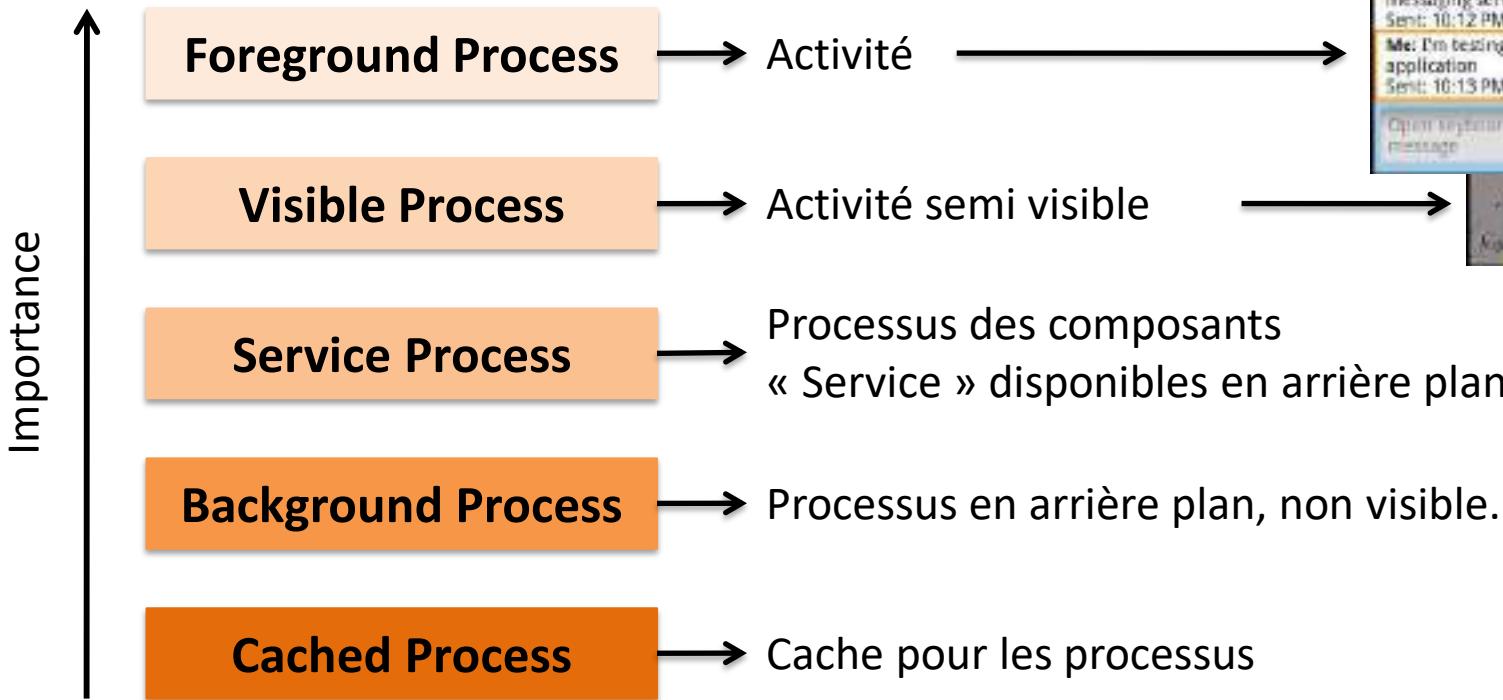
- Libérer les ressources: les threads.



Activité et Intent

■ Priorités des processus

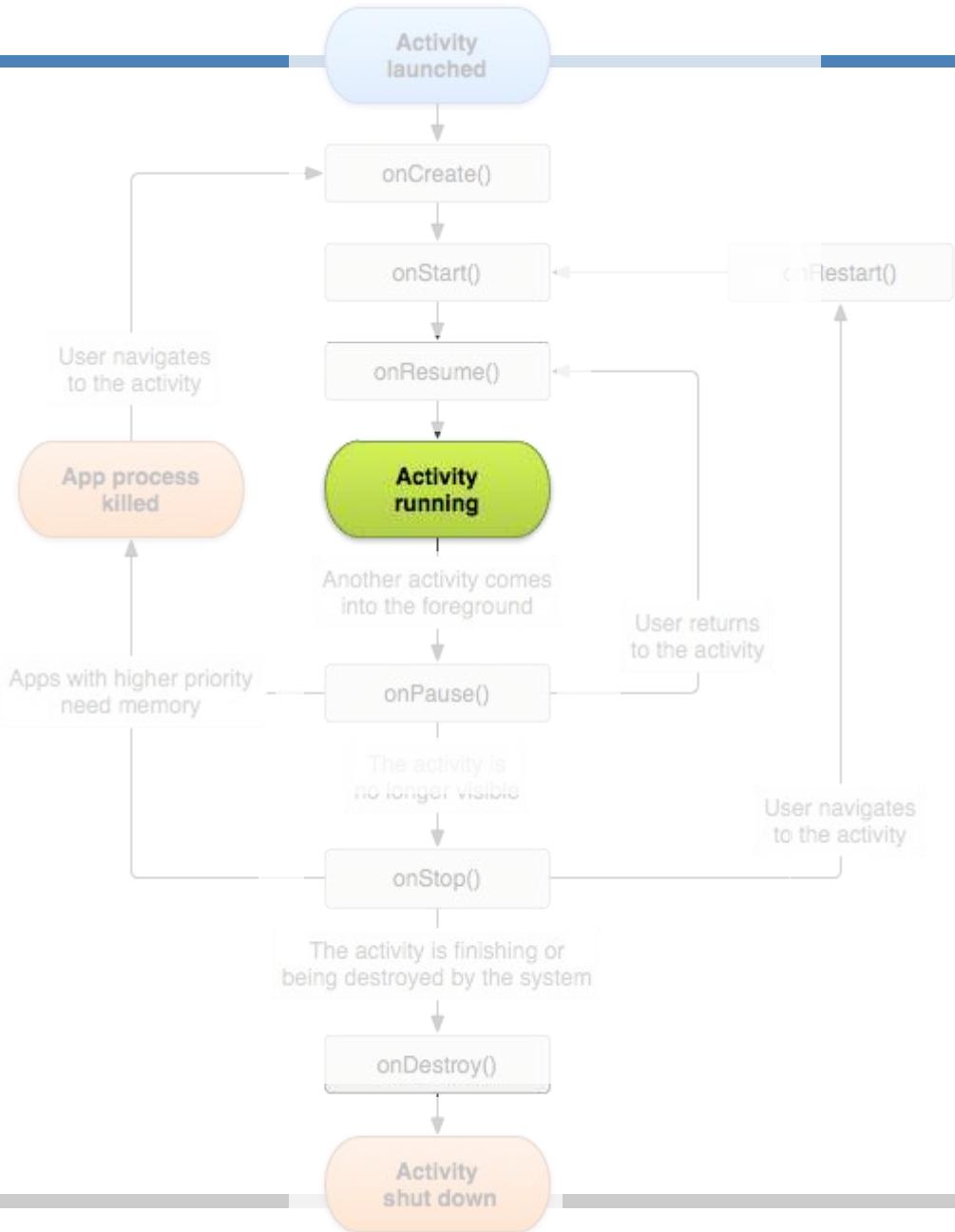
Utilisées pour tuer des processus



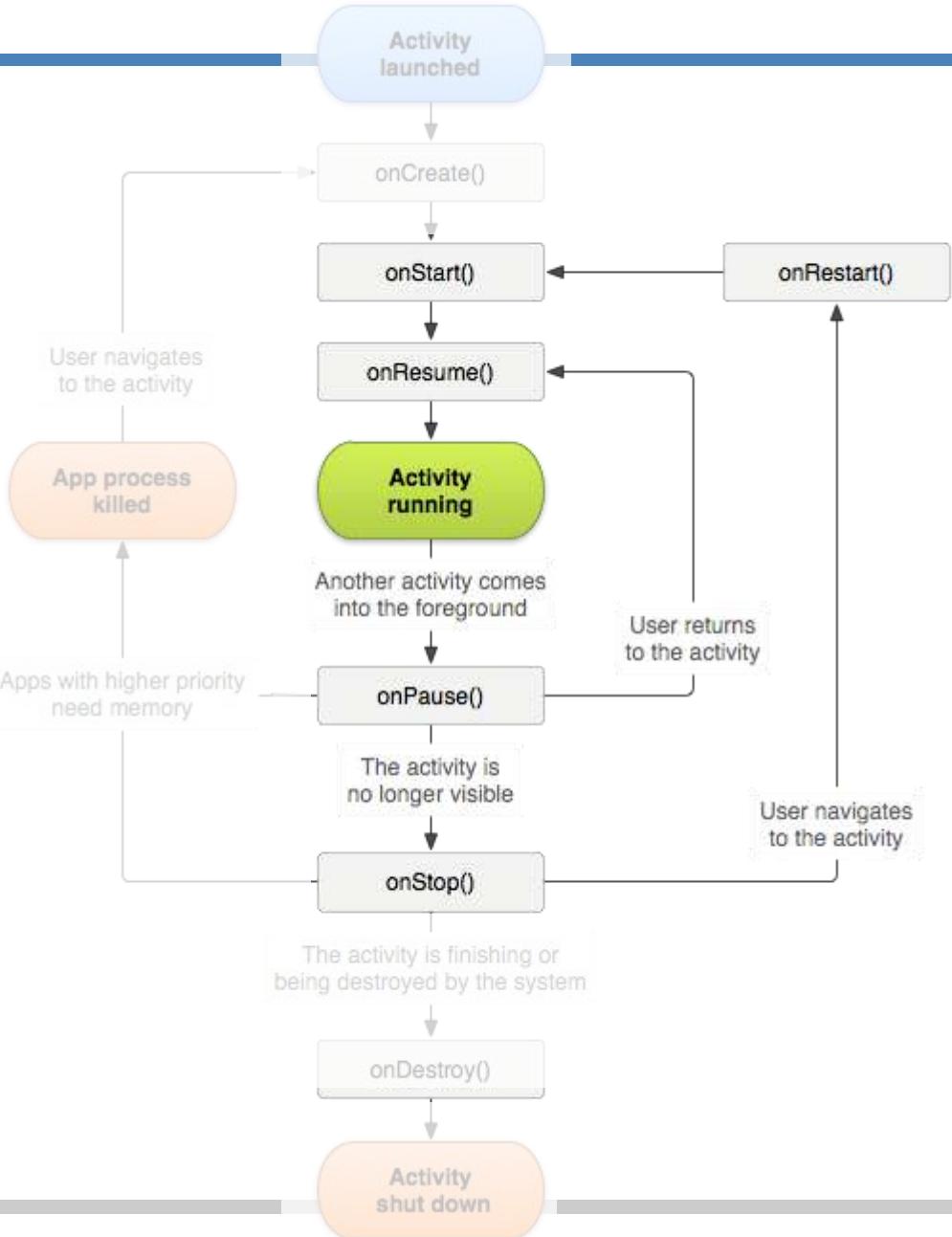
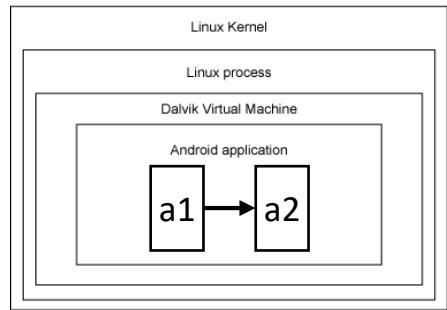
Activité et Intent

- Scénario :

- L'activité est en exécution
- Une fenêtre de dialogue est affichée
 - L'activité est au second plan,
 - semi –visible
 - La fenêtre de dialogue est au 1er plan
 - **onPause**
 - La fenêtre de dialogue disparaît
 - **onResume**
 - **Running**



Activité et Intent



- Un autre scénario :

- Une autre **activité est déclenchée**
- a1 déclenche a2
- L’activité a1 se met en pause
 - **onPause a1**
- L’activité a2 démarre
 1. **onCreate a2**
 2. **onStart a2**
 3. **onResume a2**
 4. **onStop a1**,
(a1 n'est plus visible)
 5. **a2 reprend son exécution**
Running

Activité et Intent

First Activity

1. **onCreate**

- **startActivity ...**

2. **onStart**

3. **onResume**

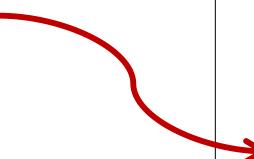
4. **onPause**

8. **onStop**

10. **onRestart**

11. **onStart**

12. **onResume**



Second Activity

5. **onCreate**

6. **onStart**

7. **onResume**

(Touche retour)

9. **onPause**

13. **onStop**

14. **onDestroy**

Activité et Intent

■ Méthodes

La plupart de
votre code va ici

Sauvegarder vos
données ici

```
public class ExampleActivity extends Activity {  
    @Override  
    public void onCreate (Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        // The activity is being created.  
    }  
    @Override  
    protected void onStart() {  
        super.onStart();  
        // The activity is about to become visible.  
    }  
    @Override  
    protected void onResume() {  
        super.onResume();  
        // The activity has become visible (it is now "resumed").  
    }  
    @Override  
    protected void onPause() {  
        super.onPause();  
        // Another activity is taking focus (this activity is about to be "paused").  
    }  
    @Override  
    protected void onStop() {  
        super.onStop();  
        // The activity is no longer visible (it is now "stopped")  
    }  
    @Override  
    protected void onDestroy() {  
        super.onDestroy();  
        // The activity is about to be destroyed.  
    }  
}
```

Activité et Intent

■ Cycle de vie : Récap

Une activité :

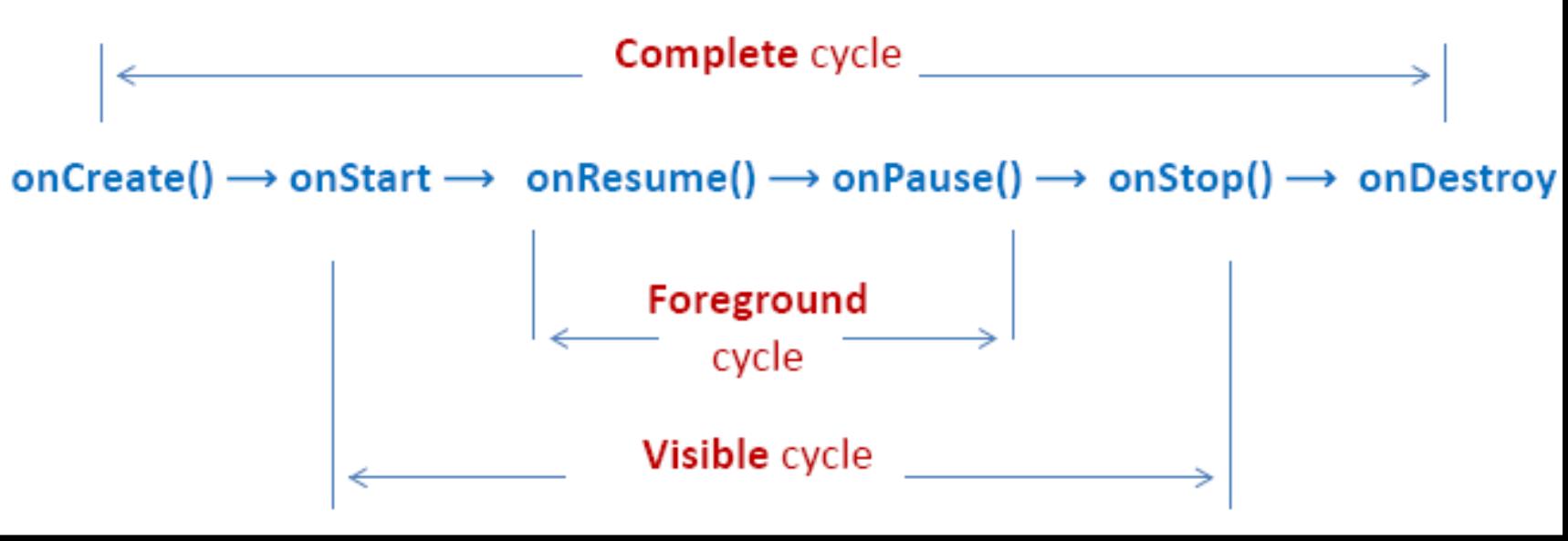
- **4 états** : Active, en pause, stoppée ou morte
- **3 attributs de visibilité** : visible, semi-visible et non-visible
- **Plusieurs méthodes** : OnCreate(), OnStart(), ...OnDestroy()
- **États « Tuables »** : OnPause(), OnStop() et OnDestroy()
- **Obligatoire** : **OnCreate()**
- **Très recommandé** : **OnPause()**

Activité et Intent

■ Cycle de vie : Récap

Une activité :

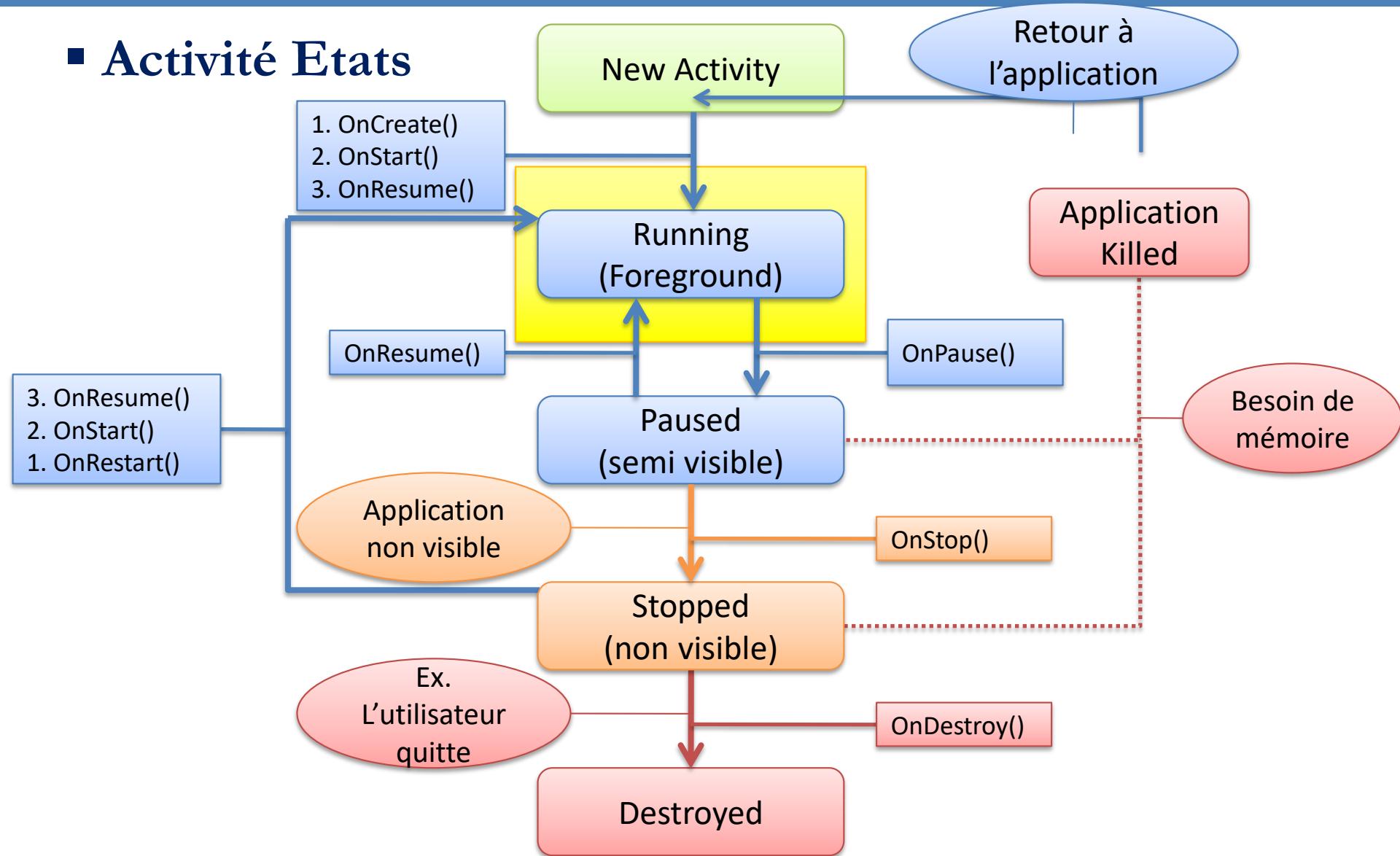
- **4 états** : Active, en pause, stoppée ou morte



- **Très recommandé** : `OnPause()`

Activité et Intent

■ Activité Etats



Instant state

■ Sauvegarde de l'état d'une activité

- L'activité change d'état suite à une action de l'utilisateur ou du système
 - ex. rotation de l'écran, changement de configuration (ex. langue), low mem. killer
- **Objectif :** Retrouver le même état de l'activité
- Une partie de l'état est automatiquement sauvegardée
- Il est possible de sauvegarder les données simple ou complexes

Instant State

■ Sauvegarde de l'état d'une activité

- Trois façons possibles :

1. onPause() et onResume()

2. onSaveInstanceState() et onRestaureInstanceState()

3. SharedPreferences

Instant State

■ Sauvegarde de l'état d'une activité

- Obtenir une référence sur SharedPreferences :

```
SharedPreferences sharedPref = getActivity().getPreferences(Context.MODE_PRIVATE);
```

- Écrire dans Shared preference :

```
SharedPreferences.Editor editor = sharedPref.edit();  
editor.putInt("hScore", newHighScore);  
editor.apply();
```

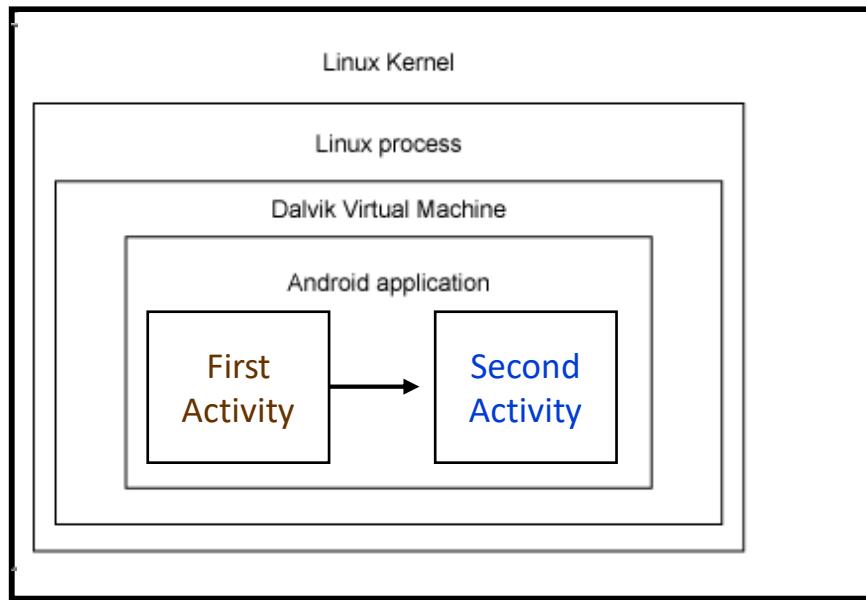
- Lire dans Shared preference :

```
int highScore = sharedPref.getInt("hScore", defaultValue);
```

Activité et Intent

■ Intent

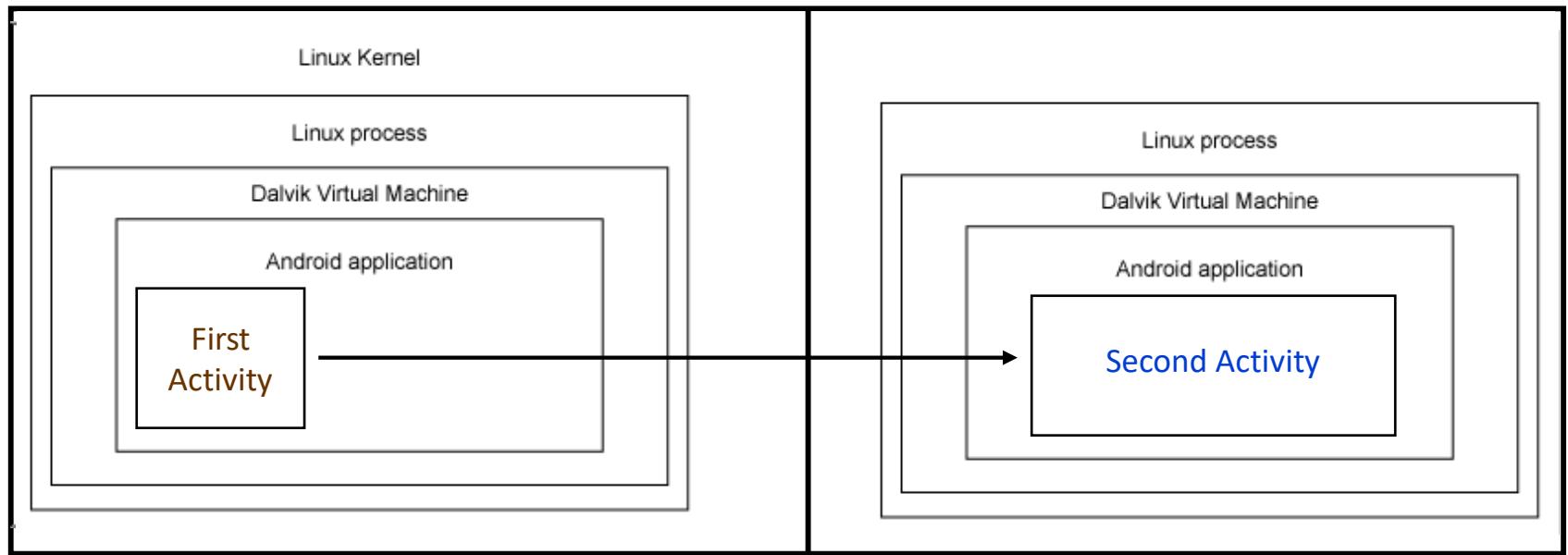
- Une application est composée de plusieurs composants (ex. plusieurs activités)
- Les activités sont indépendantes les unes des autres **mais coopèrent**
Elles utilisent des ***Intent*** pour communiquer



Activité et Intent

■ Intent

- Une application est composée de plusieurs composants (ex. plusieurs activités)
- Les activités sont indépendantes les unes des autres mais coopèrent
Elles utilisent des ***Intent*** pour communiquer



Activité et Intent

■ Intent

- Les Intent offrent un moyen simple et standard pour la communication entre les composants
- Il est construit par un composant qui veut exécuter une action **A**
- Il est reçu par un composant qui peut exécuter l'action **A**
- Il est aussi utilisé pour transmettre toute information nécessaire à la réalisation de l'action **A**

Ex. lister un contact, prendre une photo, composer un numéro

Activité et Intent

■ Intent

- Un Intent est une structure de données qui représente :
 1. Une action qu'on veut exécuter
 - ou
 2. Un événement qui s'est produit et on aimerait en informer d'autres composants
- Exemple d'action à exécuter : Démarrer une autre activité
- Les événements sont destinés au composant BroadcastReceiver
 - + d'autres paramètres : données, catégorie...

Activité et Intent

■ Intent

- Deux types d'**Intent** :

1. **Explicite** : en précisant exactement la classe du composant à qui est destiné le message
2. **Implicite**: La classe n'est pas précisée. Une action est déclarée et Android fait une résolution automatique en se basant sur les attributs de l'**Intent** et les **Intent Filters** des composants

Activité et Intent

■ Intent

- Les activités peuvent réutiliser les fonctionnalités des autres composants en les appelant sous forme d'**Intent**
- Une activité peut être remplacée par une autre offrant le même **IntentFilter**

Activité et Intent

■ Intent Filter

- Il décrit les capacités de l'application (les Intent qu'elle accepte)
- Il est publié par les activités, services et broadcast receiver
- Utilisé par le système pour sélectionner le composant qui satisfait un Intent

```
<intent-filter>
```

```
  <action android:value="android.intent.action.VIEW" />
  <action android:value="android.intent.action.EDIT" />
  <action android:value="android.intent.action.PICK" />
  <category android:value="android.intent.category.DEFAULT" />
  <type
    android:value="vnd.android.cursor.dir/vnd.google.contact" />
</intent-filter>
```

Activité et Intent

■ Intent Filter

- Deux filtres pour la même activité

```
<activity android:name="ShareActivity">
    <!-- This activity handles "SEND" actions with text data -->
    <intent-filter>
        <action android:name="android.intent.action.SEND"/>
        <category android:name="android.intent.category.DEFAULT"/>
        <data android:mimeType="text/plain"/>
    </intent-filter>
    <!-- This activity also handles "SEND" and "SEND_MULTIPLE" with media data -->
    <intent-filter>
        <action android:name="android.intent.action.SEND"/>
        <action android:name="android.intent.action.SEND_MULTIPLE"/>
        <category android:name="android.intent.category.DEFAULT"/>
        <data android:mimeType="application/vnd.google.panorama360+jpg"/>
        <data android:mimeType="image/*"/>
        <data android:mimeType="video/*"/>
    </intent-filter>
</activity>
```

Activité et Intent

■ Intent

- **Démarrer une activité au sein de l'application courante**

Démarrage explicite d'une activité : spécifier l'activité cible

Utilisation :

Navigation entre écrans d'une application

- **Solliciter d'autres applications**

Le système se charge de trouver l'application ou le composant le plus approprié et lui transmet l'Intent

- **Envoyer/recevoir des évènements**

Exemple : batterie défaillante

Activité et Intent

■ Intent

- Les **Intent** sont utilisés comme suit

startActivity (intent) ou startActivityForResult(intent,...) onActivityResult(...)	Démarre une activité
sendBroadcast (intent)	Envoi un <i>Intent</i> à n'importe quel composant <i>BroadcastReceiver</i> qui serait intéressé
startService(intent) ou bindService(intent, ...)	Démarre/Communique avec un Service

Activité et Intent

■ Intent Explicite

- Même application

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

- Autre application se trouvant dans le package : emi.TP.appli2

```
Intent intent = new Intent(Intent.ACTION_MAIN);
intent.addCategory(Intent.CATEGORY_LAUNCHER);
intent.setClassName("emi.TP.appli2", emi.TP.appli2.MainActivity");
intent.setFlags(Intent.FLAG_ACTIVITY_NEW_TASK);
startActivity(intent);
```

Activité et Intent

■ Intent implicite

Données principales (URI)
tel://
http://
sendto://

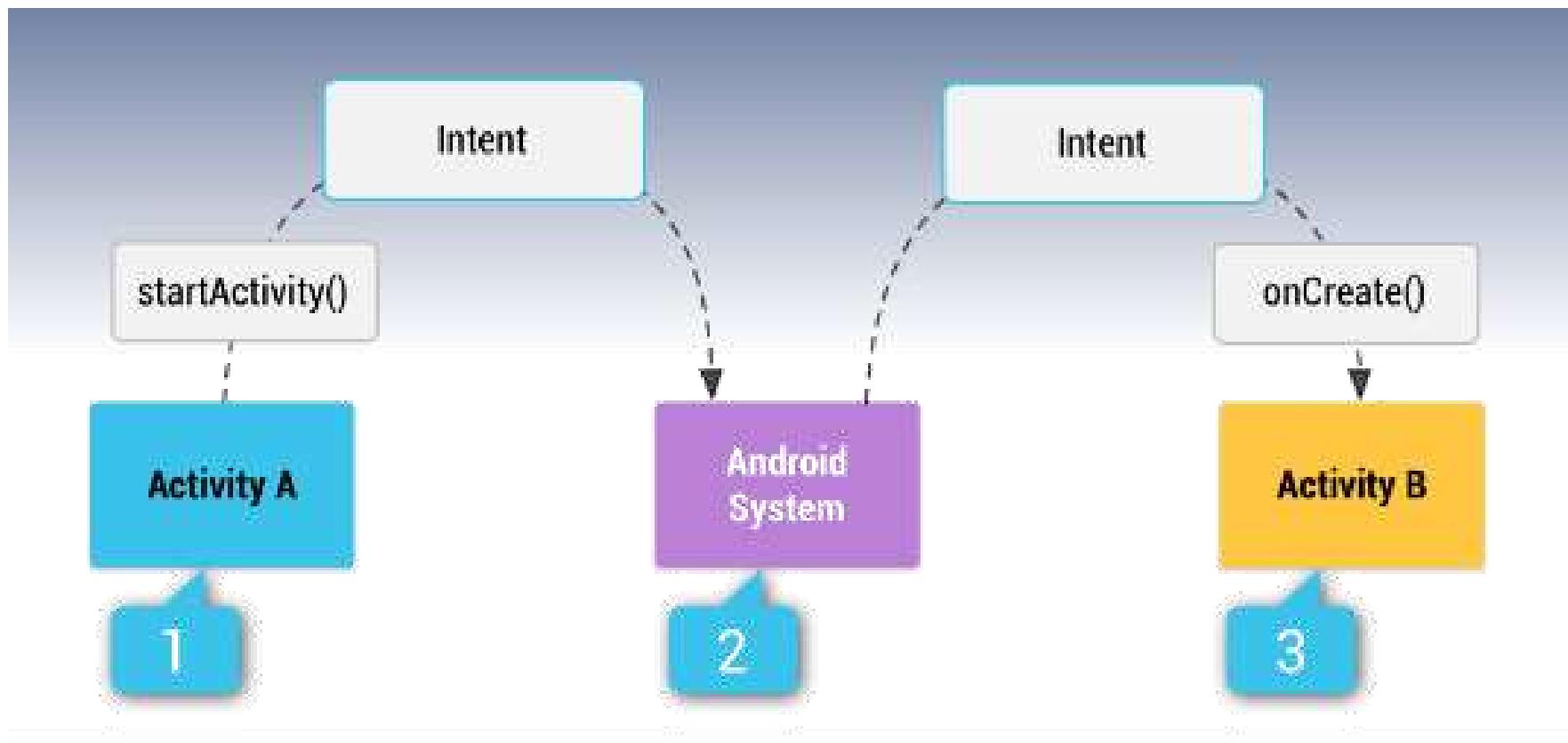
```
Intent monIntent = new Intent (action, data);  
startActivity (monIntent);
```

Prédefinies ou Par l'utilisateur

Activité et Intent

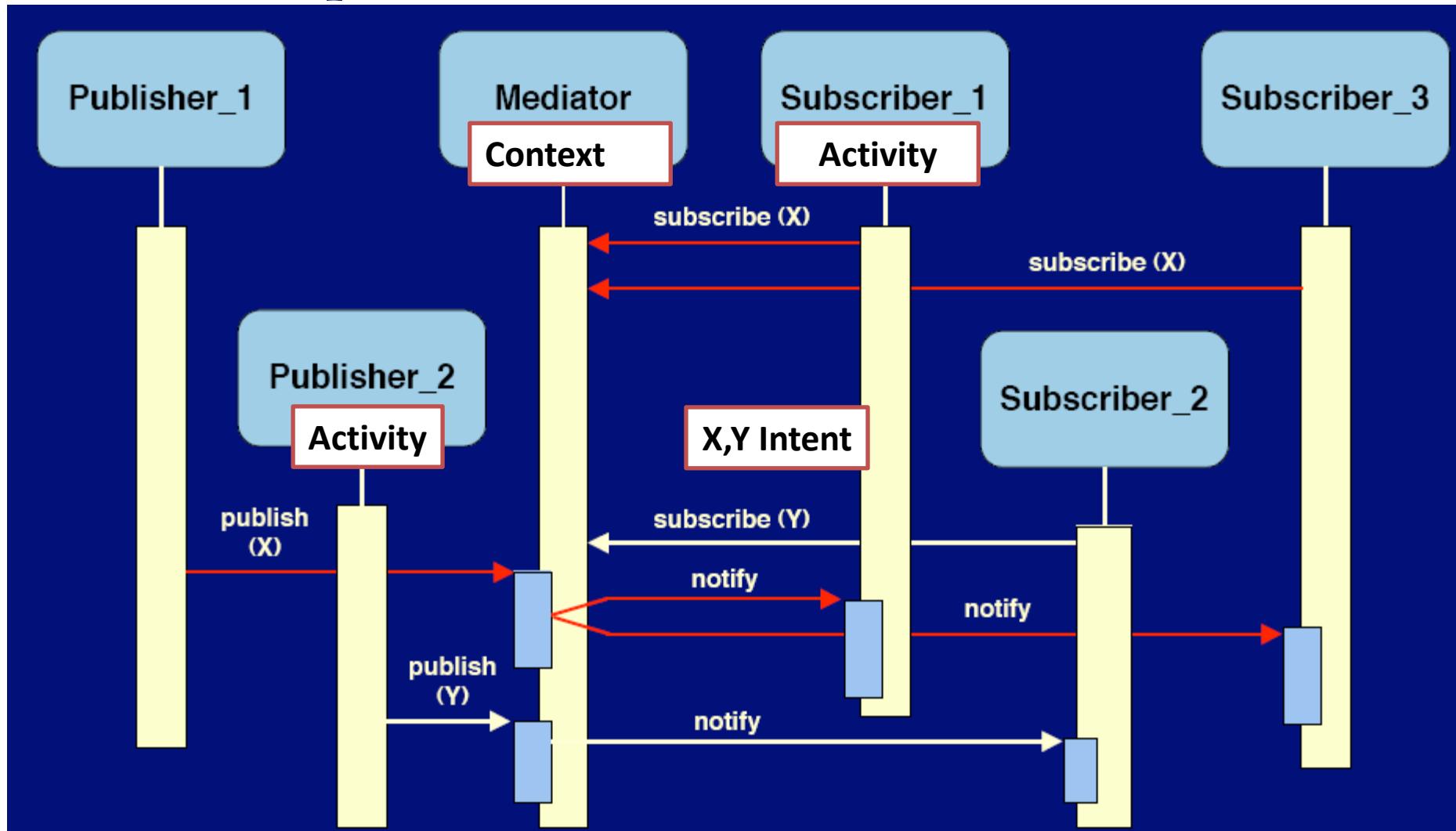
■ Intent Implicit

Fonctionnement



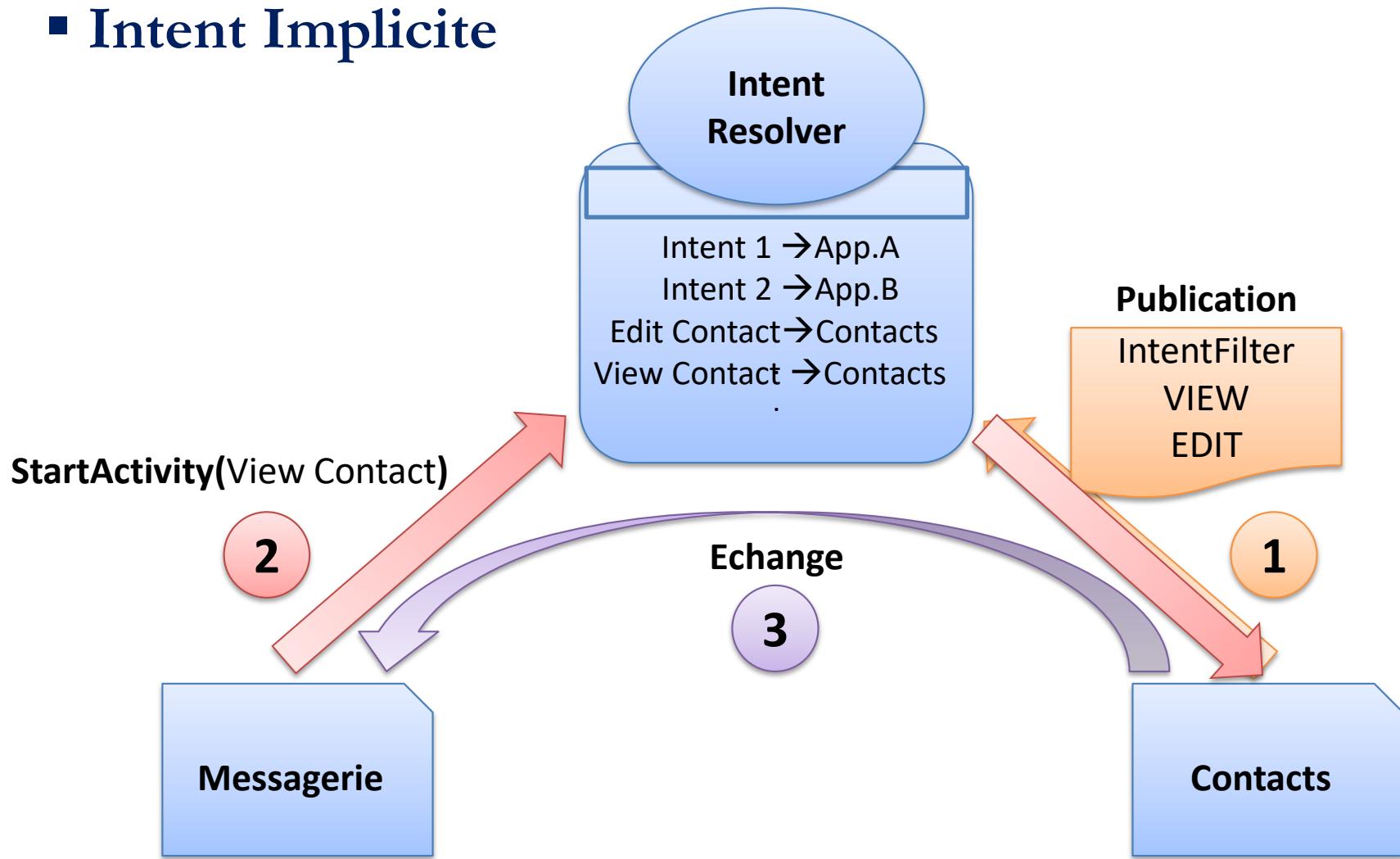
Activité et Intent

■ Intent Implicit



Activité et Intent

■ Intent Implicite



Activité et Intent

■ Intent Implicit

```
// Create the text message with a string
Intent sendIntent = new Intent();
sendIntent.setAction(Intent.ACTION_SEND);
sendIntent.putExtra(Intent.EXTRA_TEXT, textMessage);
sendIntent.setType(HTTP.PLAIN_TEXT_TYPE); // "text/plain" MIME type

startActivity(sendIntent);
}
```

- Si aucune application n'est disponible pour répondre à l'Intent alors l'application plante

⇒ Faire la vérification avant l'appel

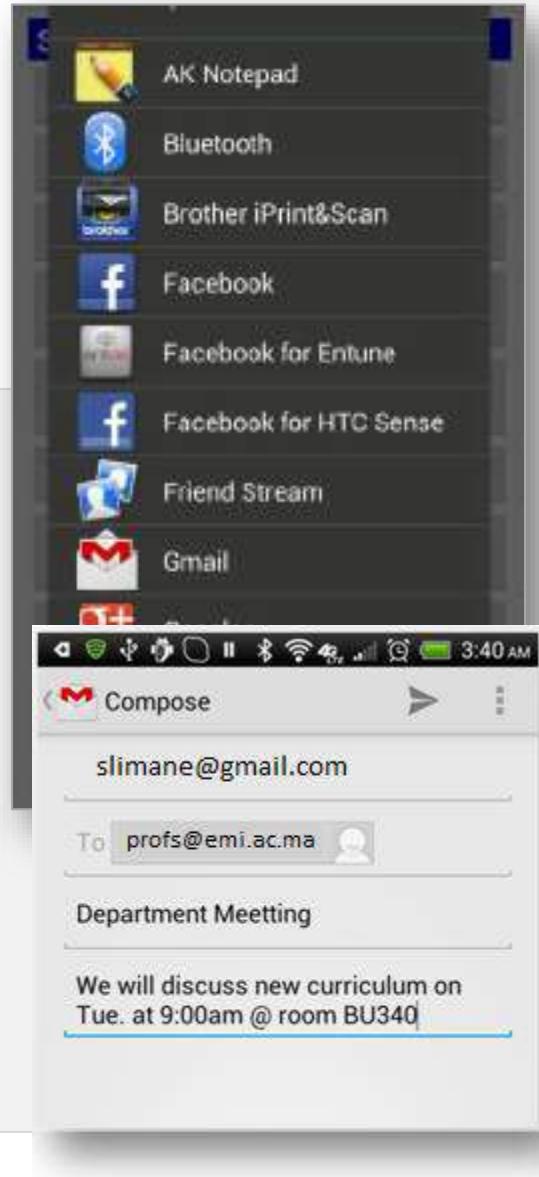
Activité et Intent

■ Intent Implicit

- Lorsque plusieurs applications peuvent répondre à l'Intent

```
// send email
String emailSender = "slimane@gmail.com";
String emailSubject = "Department Meetting";
String emailText = "We will discuss new curriculum "
                  + "on Tue. at 9:00am @ room BU340";
String emailReceiverList[] = {"profs@emi.ac.ma"};

Intent intent = new Intent(Intent.ACTION_SEND);
intent.setType("text/plain");
intent.putExtra(Intent.EXTRA_EMAIL, emailReceiverList);
intent.putExtra(Intent.EXTRA_SUBJECT, emailSubject);
intent.putExtra(Intent.EXTRA_TEXT, emailText);
startActivity(intent);
```



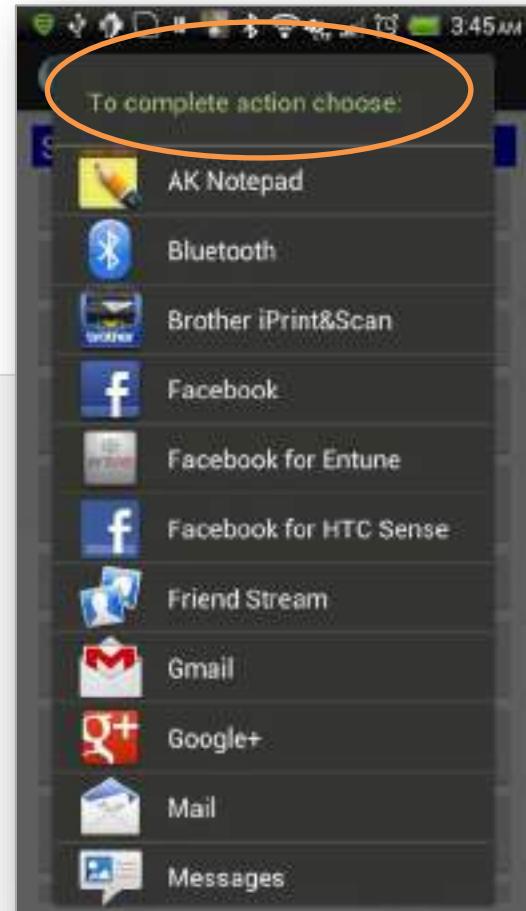
Activité et Intent

■ Intent Implicit

- Forcer le choisir

```
// send email
String emailSender = "slimane@gmail.com";
String emailSubject = "Department Meeting";
String emailText = "We will discuss new curriculum "
                  + "on Tue. at 9:00am @ room BU340";
String emailReceiverList[] = {"profs@emi.ac.ma"};

Intent intent = new Intent(Intent.ACTION_SEND);
intent.setType("text/plain");
intent.putExtra(Intent.EXTRA_EMAIL, emailReceiverList);
intent.putExtra(Intent.EXTRA_SUBJECT, emailSubject);
intent.putExtra(Intent.EXTRA_TEXT, emailText);
startActivity(Intent.createChooser(intent, "To complete action choose:"));
```



Activité et Intent

■ Résolution d'Intent

La recherche d'un composant compatible avec un Intent se fait selon les caractéristiques fondamentales d'un Intent :

- (Composant)
- L'action
- Data : La description des données décrit dans l'Intent (Uri)
- La catégorie de l'Intent

A comparer avec les Intent-filter des autres composants.

Activité et Intent

■ Intent

- Les **attributs** d'un Intent sont :
 - **Action** : chaîne de caractères représentant l'action à exécuter
 - prédéfinies** : Ex. ACTION_DIAL, ACTION_VIEW...
 - définies par l'utilisateur**
 - **Data** : données associées à l'Intent sous forme de URI
 - Ex.** Tel:1123444, http://www.emi.ac.ma...
 - **Category** : information permettant d'identifier le composant qui peut traiter l'action
 - Ex.** Category_BROWSABLE, Category_LAUNCHER...

Activité et Intent

■ Intent : data URI

content: //contacts/ people/ 125

geo: 25,32

tel: 123

google.streetview: cbll=25,32 &cbp=1,yaw,,pitch,zoom

```
<intent-filter>
    <data android:mimeType="video/mpeg" android:scheme="http" ... />
    <data android:mimeType="audio/mpeg" android:scheme="http" ... />
    ...
</intent-filter>
```

Activité et Intent

■ Intent

- **Type** : Le type MIME des données associées à l'intent

Ex. contacts/people, images/pictures, images/jpg

Si non spécifié, Android infère le type qui convient

- **Component** : Le composant cible qui recevra l'intent

Lorsqu'il n'y a qu'un seul composant qui doit recevoir votre intent

Spécifier le **Context** et le **.Class** du composant

- **Flags** : Informations de configuration de l'intent

Ex. FLAG_ACTIVITY_NO_ANIMATION, ...

Activité et Intent

■ Intent : Flags

A, B, C, D sont des activités

- A démarre B ($A \rightarrow B$, par startActivity)
- $A \rightarrow B \rightarrow C \rightarrow D$:

Si $D \rightarrow A$ démarrer l'instance existante de A

Dans D :

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

`Intent.addFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP);`
`startActivity(intent);`

Activité et Intent

■ Intent : Flags

A, B, C, D sont des activités

- A → B → A → B → A ...

Par défaut : création d'une nouvelle activité à chaque startActivity

La même instance est conservée

Dans A

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

```
Intent.addFlags(Intent.FLAG_ACTIVITY_REORDER_TO_FRONT);  
startActivity(intent);
```

Dans B idem

Activité et Intent

■ Intent : Flags

B → A → A → A → A ...

Exemple : activité A = « résultats de la recherche » + recherche encore

Par défaut : plusieurs fois back pour arriver à B

Utiliser une seule instance

Dans B

```
Intent intent = new Intent(this, A.class);
Intent.addFlags(Intent. FLAG_ACTIVITY_SINGLE_TOP);
startActivity(intent);
```

Dans A idem

Technologies et développement Mobile



Pr. Slimane Bah, ing. PhD

Génie Informatique option TI & IQL

Semaine 12.2

Séance 7

Activité et Intent

■ Intent

- Les **attributs** d'un Intent sont :
 - **Action** : chaîne de caractères représentant l'action à exécuter
 - prédéfinies** : Ex. ACTION_DIAL, ACTION_VIEW...
 - définies par l'utilisateur**
 - **Data** : données associées à l'Intent sous forme de URI
 - Ex.** Tel:1123444, http://www.emi.ac.ma...
 - **Category** : information permettant d'identifier le composant qui peut traiter l'action
 - Ex.** Category_BROWSABLE, Category_LAUNCHER...

Activité et Intent

■ Intent

- **Type** : Le type MIME des données associées à l'intent

Ex. contacts/people, images/pictures, images/jpg

Si non spécifié, Android infère le type qui convient

- **Component** : Le composant cible qui recevra l'intent

Lorsqu'il n'y a qu'un seul composant qui doit recevoir votre intent

Spécifier le **Context** et le **.Class** du composant

- **Flags** : Informations de configuration de l'intent

Ex. FLAG_ACTIVITY_NO_ANIMATION, ...

Activité et Intent

■ Intent

- **Extras** : informations additionnelles attachées à l'Intent

Sous forme de (clé-valeur)

Intent.putExtra(clé, valeur)

Intent.getExtras.getXXX(clé) avec XXX = String, Int, ...

```
Intent i = new Intent(this, android.content.Intent.ACTION_SEND);
Bundle b = new Bundle();
b.putString ("Objet", "Rdv demain");
b.putString ("Corps", "Comme d'habitude au même endroit");
i.putExtras(b);
```

```
Bundle MonBdl = getIntent().getExtras();
if (MonBdl != null) {
    String s = MonBdl.getString("Objet");
    if (s != null) { // faire quelque chose }
}...
```

Activité et Intent

■ Intent

- Extras :

Intent.putExtra(clé, valeur)

Intent.getXXXExtra(clé) avec XXX = String, Int, ...

Ex.

```
Intent Myi = new Intent(this, android.content.Intent.ACTION_SEND);
Myi.putExtra("Objet", "Rdv demain");
Myi.putExtra("Corps", "Comme d'habitude au même endroit");
```

```
Intent intent = getIntent();
if (intent != null) {
    String s = intent.getStringExtra("Objet");
    if (s != null) { // faire quelque chose }
}
```

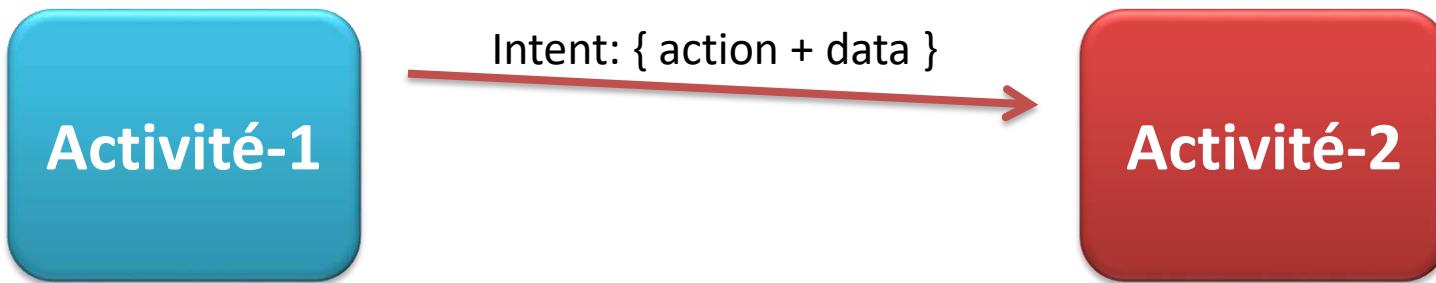
Activité et Intent

■ Intent : exemples

Les arguments principaux d'un Intent sont :

1. Action

2. Data



Activité et Intent

■ Intent Explicite

- Soit l'activité HelloWorld
- On aimeraient appeler une autre activité HelloInfo à partir de HelloWorld

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

```
Intent HelloInf= new Intent(Context.this, HelloInfo.class);  
startActivity(HelloInf);  
}
```

Activité et Intent

■ Intent Implicit

- Résolution en utilisant sa description dans le manifest.xml

Ex. Ouvrir un fichier : *hello_world.mp3*

```
Intent intent = new Intent(Intent.ACTION_VIEW);
```

```
File hello = new File("/sdcard/hello.mp3");
```

```
intent.setDataAndType(Uri.fromFile(hello), "audio/mp3");
```

```
startActivity(intent);
```

⇒ L'activité qui permet de jouer de la musique spécifie une ACTION_VIEW dans son manifest.xml (sous intent-filter) + le type de données.

Activité et Intent

■ Intent Implicit

- Personnalisé :

```
<activity android:name=".HelloWorldActivity"
          android:label="@string/app_name">
    <intent-filter>
        <action android:name="android.intent.action.MAIN" />
        <category android:name="android.intent.category.LAUNCHER" />
    </intent-filter>
    <intent-filter>
        <action android:name="bonjour.ACTION" />
        <category android:name="android.intent.category.DEFAULT" />
    </intent-filter>
</activity>
```

```
public class HelloWorldMasterActivity extends Activity {
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        Intent intent = new Intent();
        intent.setAction("bonjour.ACTION");
        startActivity(intent);
```

Activités et Intent

Exemples du couple **action/data**:

ACTION_DIAL	<i>tel:5551234</i>
	afficher l'activité qui permet de composer un N.tel avec le Num 5551234 déjà rempli.
ACTION_VIEW	<i>http://www.google.com</i>
	afficher la page google dans un browser.
ACTION_VIEW	<i>content://contacts/people/</i>
	afficher la liste des personnes dans les contacts
ACTION_EDIT	<i>content://contacts/people/2</i>
	Editer l'information du contact dont l'identifiant est "2".

Activités et Intent

■ Intent : Exemple

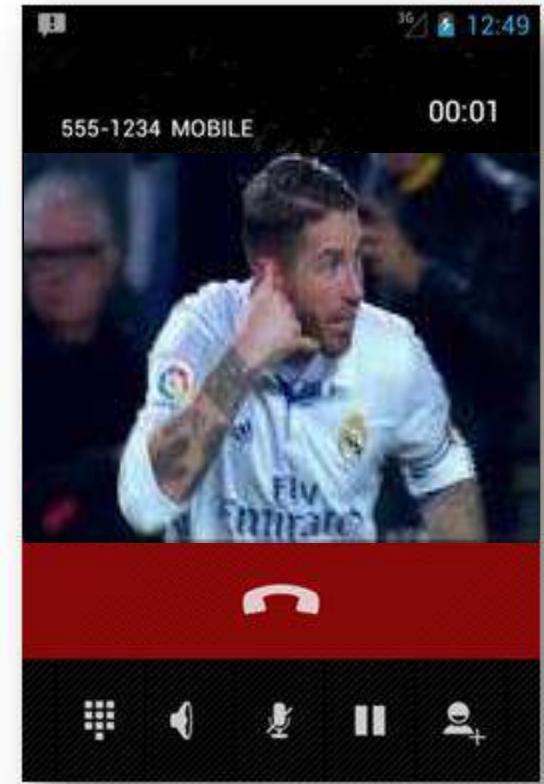
```
String myPhoneNumberUri = "tel:555-1234";
Intent myActivity2 = new Intent(Intent.ACTION_DIAL,
                                Uri.parse(myPhoneNumberUri));
startActivity(myActivity2);
```



Activités et Intent

■ Intent : Exemple

```
String myData = "tel:555-1234";  
  
Intent myActivity2 = new Intent(  
        Intent.ACTION_CALL,  
        Uri.parse(myData));  
  
startActivity(myActivity2);
```



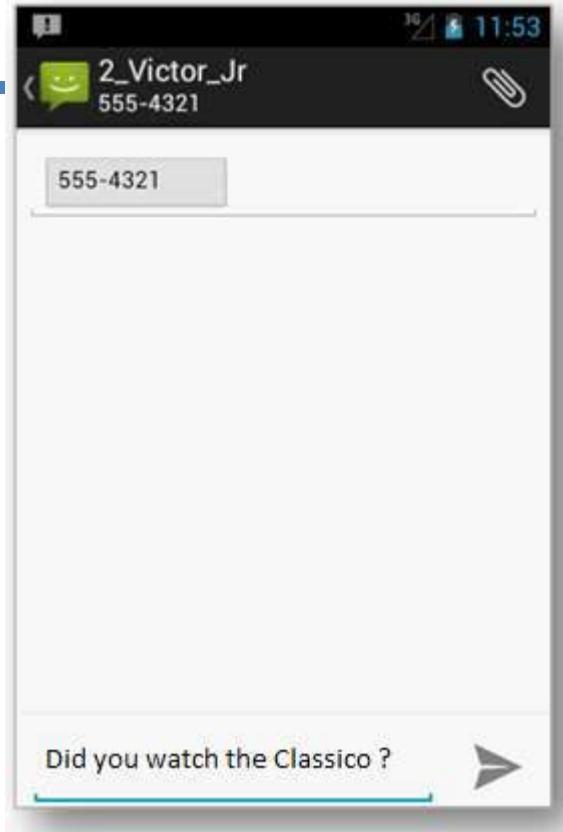
Permission require:

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

Activité et Intent

■ Intent

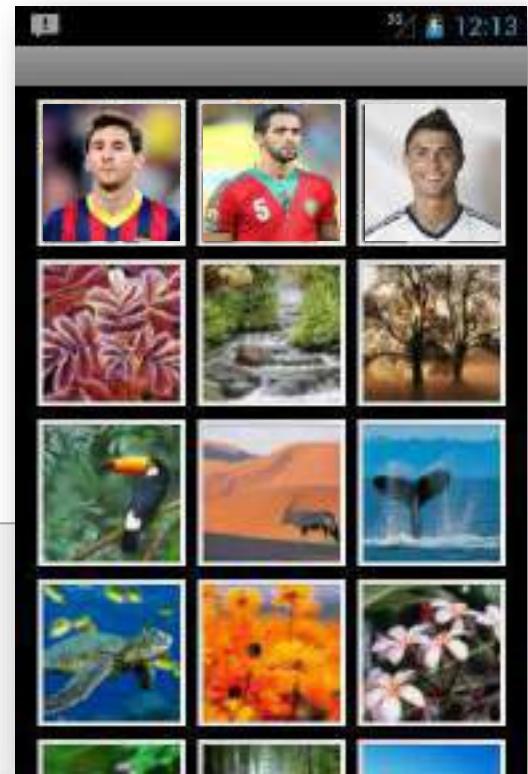
```
Intent intent = new Intent(  
        Intent.ACTION_SENDTO,  
        Uri.parse("smsto:555-4321"));  
  
intent.putExtra("sms_body",  
        "Did you watch the Classico ?");  
  
startActivity(intent);
```



Activité et Intent

■ Intent

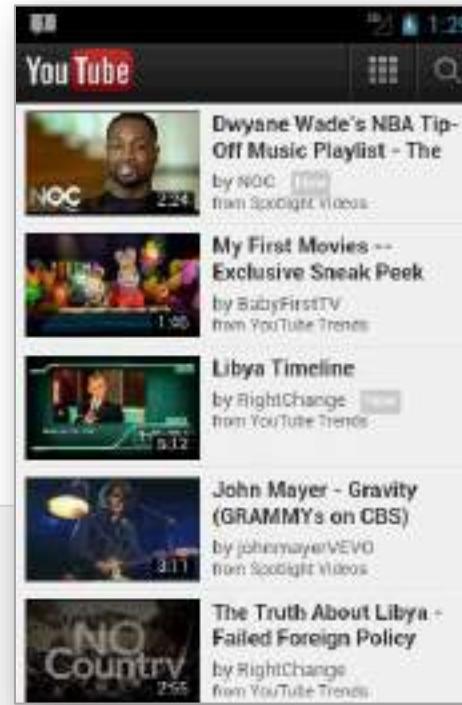
```
Intent intent = new Intent();  
  
intent.setType("image/pictures/*");  
intent.setAction(Intent.ACTION_GET_CONTENT);  
  
startActivity(intent);
```



Activité et Intent

■ Intent

```
String myUriString = "http://www.youtube.com";  
  
Intent myActivity2 = new Intent(Intent.ACTION_VIEW,  
                                Uri.parse(myUriString));  
startActivity(myActivity2);
```



Permission du Manifest à ajouter:

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

Activité et Intent

■ Intent

GeoCode : contenant une Latitude et une Longitude

et un zoom ($?z=xx$) avec **xx** entre 1 et 23

```
// carte centrée autour d'une Lat, Long
String geoCode = "geo:41.5020952,-81.6789717?z=16";
Intent intent = new Intent(Intent.ACTION_VIEW,
                            Uri.parse(geoCode));
startActivity(intent);
```



Permission du Manifest à ajouter:

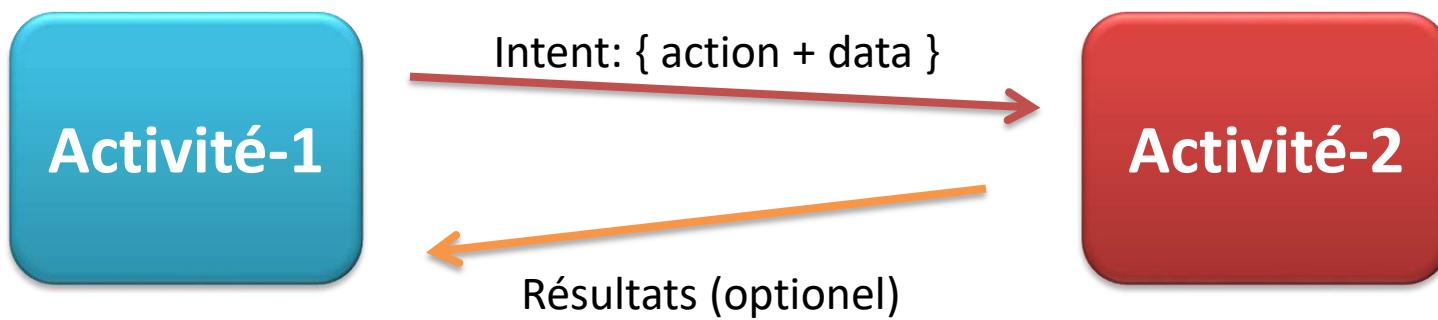
```
<uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION" />
<uses-permission android:name="android.permission.INTERNET" />
```

Activité et Intent

■ Intent : Appel avec résultat

Il est souvent utile d'appeler une activité et attendre un résultat lorsqu'elle termine

Exemple : ouvrir la liste des contacts et retourner un contact sélectionné



Activité et Intent

■ Démarrer une activité pour un résultat

- Pour obtenir un résultat de la part de l'activité appelée il faut utiliser

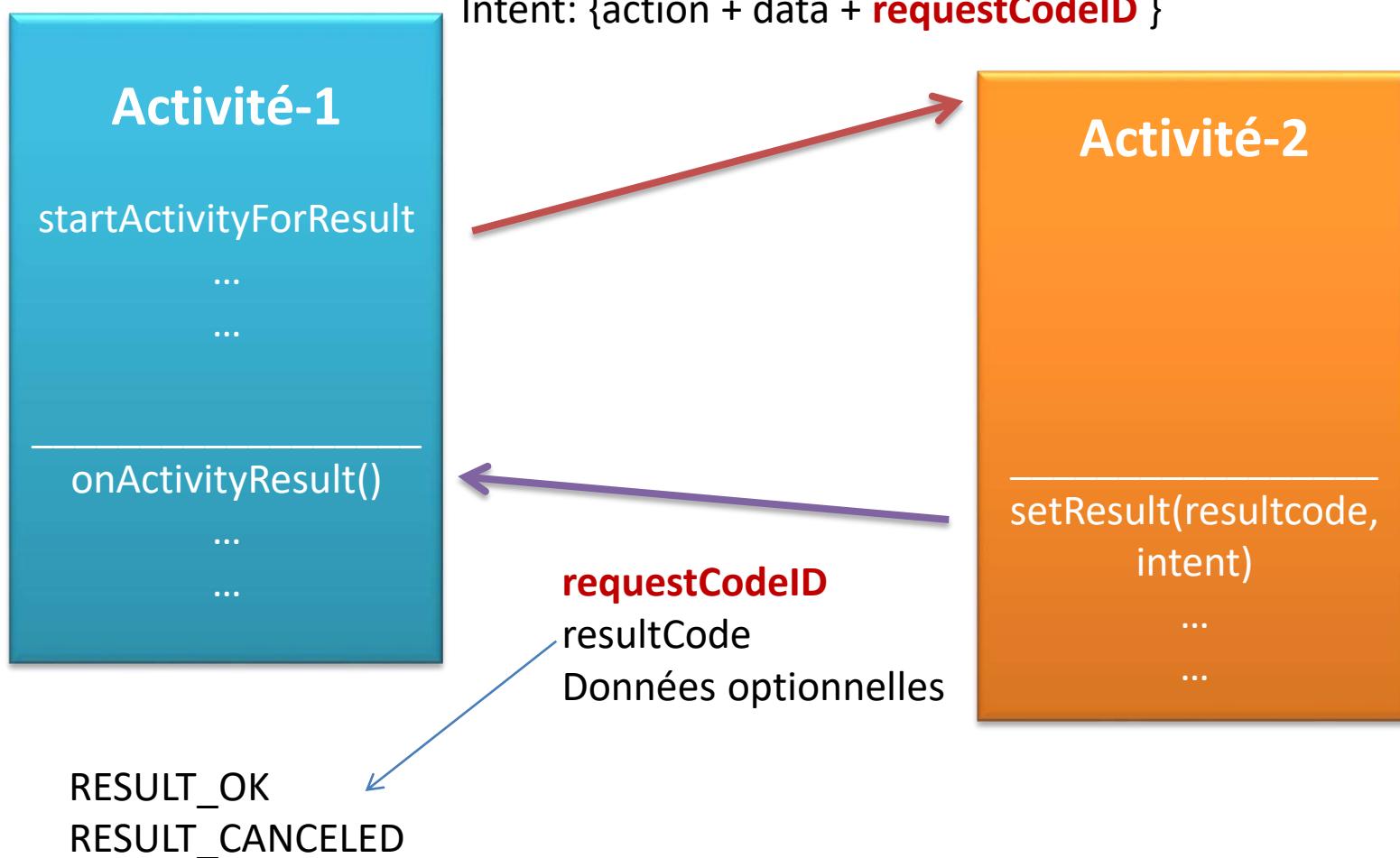
startActivityForResult (Intent, requestCodeID)

- Le résultat est récupéré avec :

onActivityResult (requestCodeID, resultCode, Intent)

Activité et Intent

■ Démarrer une activité pour un résultat

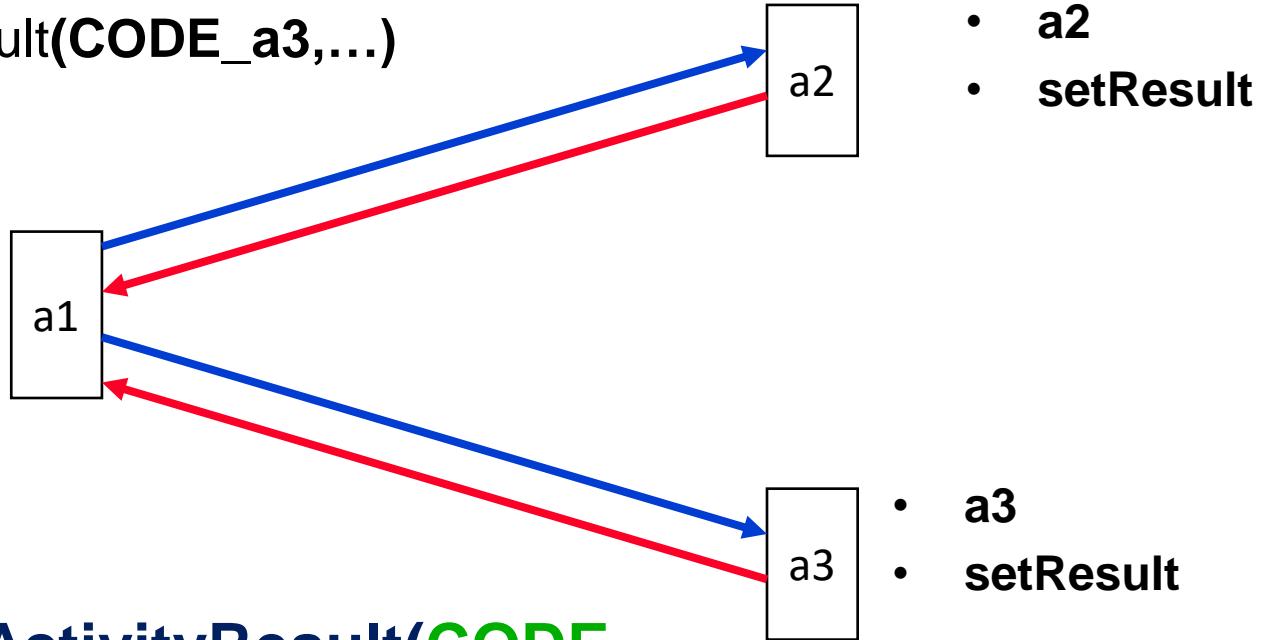


Activité et Intent

■ Démarrer une activité pour un résultat

`startActivityForResult(CODE_a2,...)`

`startActivityForResult(CODE_a3,...)`



public void onActivityResult(CODE...

if (CODE==CODE_a2)

...

else if (CODE==CODE_a3)

...

Activité et Intent

■ Démarrer une activité pour un résultat

Exemple en étapes

Activité appelante

Utilisation de la méthode startActivityForResult()

```
...
private static final int CODE_MON_ACTIVITE = 1;
...
Intent intent = new Intent(this, ClassSousActivite.class);
//représente l'identifiant de la requête qui sera utilisé pour
//identifier l'activité renvoyant la valeur de retour
startActivityForResult(intent, CODE_MON_ACTIVITE);
```

Activité et Intent

■ Démarrer une activité pour un résultat

Exemple en étapes

Activité appelée

Renvoyer une valeur de retour. Utilisation de **setResult()**



```
@Override  
public void onClick(View v) {  
    switch(v.getId()){  
        case R.id.button1:  
            setResult(RESULT_OK);  
            finish();  
            break;  
        case R.id.button2:  
            setResult(RESULT_CANCELED);  
            finish();  
            break;  
    }  
}
```

Activité et Intent

■ Démarrer une activité pour un résultat

Exemple en étapes

Activité appelante

Récupérer la valeur de retour avec **OnActivityResult()** dont les paramètres sont :

- **int requestCode:** permet de distinguer quelle activité a appelé cette méthode
- **int resultCode:** signale l'état de l'activité appelé après la fin de l'exécution
- **Intent Data:** pour l'échange de données

Activité et Intent

■ Démarrer une activité pour un résultat

```
@Override
protected void onActivityResult(int requestCode, int resultCode, Intent data)
    switch(requestCode){
        case LOGIN_REQUEST_CODE :
            switch(resultCode){
                case RESULT_OK :
                    String result1 = data.getStringExtra("key1");
                    String result2 = data.getStringExtra("key2");
                    Toast.makeText(this, "<" + result1 + ", " + result2 + ">", Toast.LENGTH_LONG).show();
                    break;
                case RESULT_CANCELED :
                    Toast.makeText(this,"canceled", Toast.LENGTH_LONG).show();
                    break;
            }
        case ANOTHER_REQUEST_CODE :
            switch(resultCode){
                case RESULT_OK :
                    // ...
                    break;
                case RESULT_CANCELED :
                    // ...
                    break;
            }
        default: super.onActivityResult(requestCode, resultCode, data);
    }
}
```

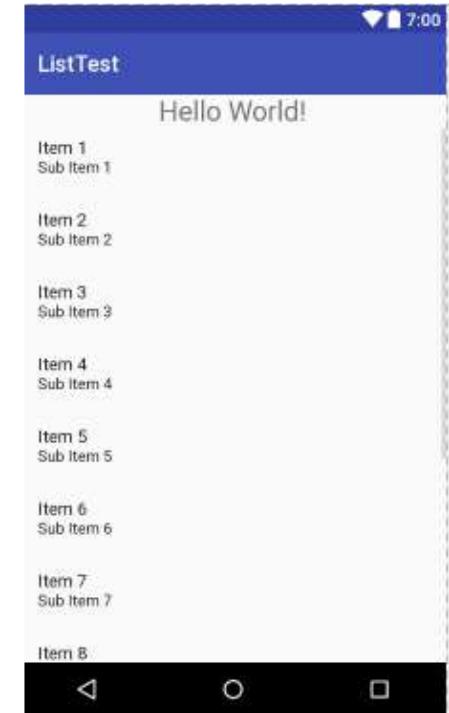
UI : Liste

■ ListView

Un widget très utilisé en développement mobile

Peut être simple ou complexe

Statique Vs Dynamique



UI : Liste

■ ListView : Statique

- Les éléments de la liste sont spécifiés avant l'exécution
- Déf. Des éléments dans **res/values/strings.xml**

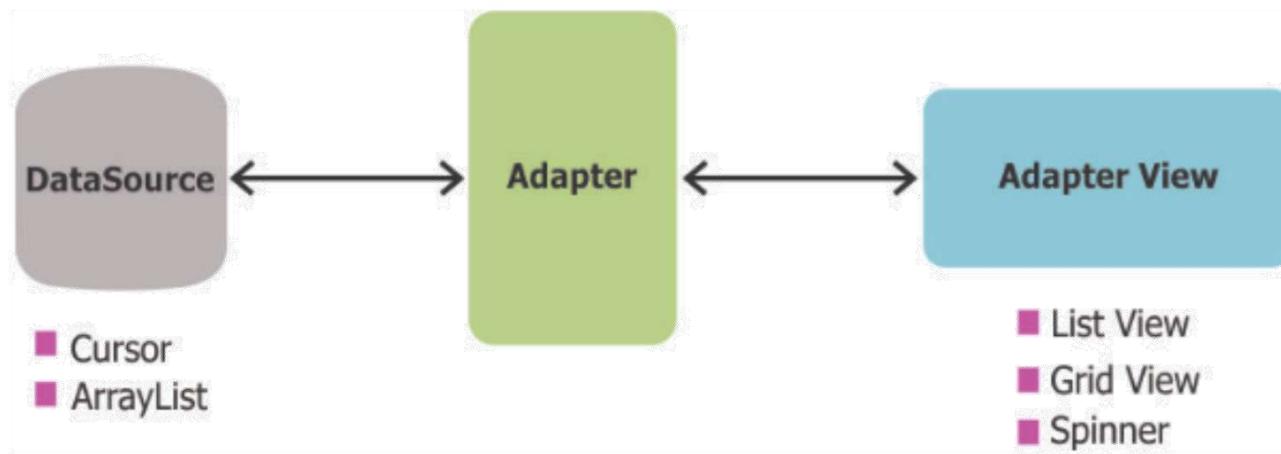
```
<resources>
    <string-array name="Genies">
        <item>Info</item>
        <item>RT</item>
        ...
        <item>Indus</item>
    </string-array>
</resources>
```

- Dans le **layout** :

```
<ListView ... android:id="@+id/mylist"
            android:entries="@array/Genies" />
```

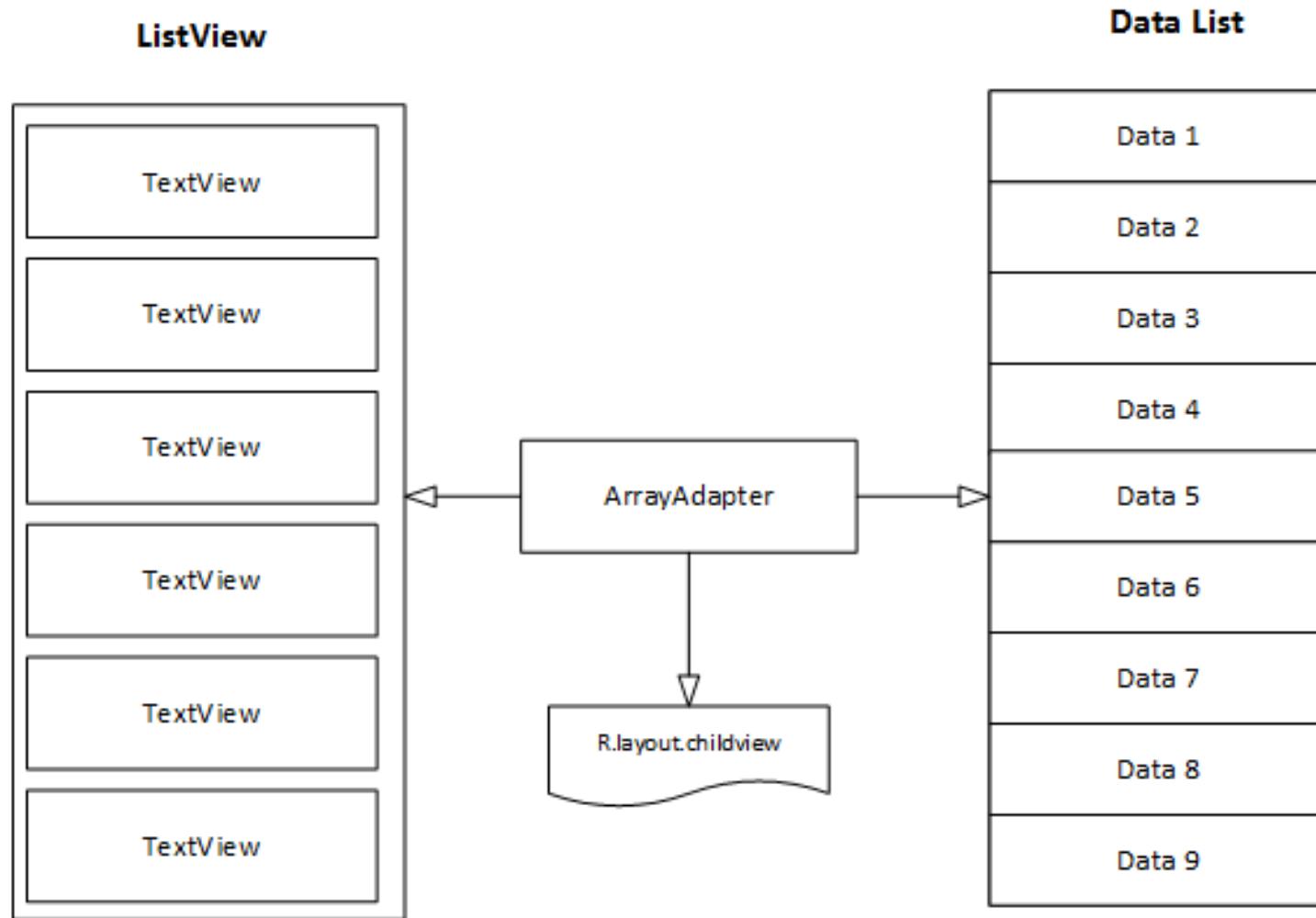
■ ListView : Dynamique

- Les éléments de la liste sont lus et générés à l'exécution
- La source peut être : BD, fichier, internet, User...
- Doit être fait via le code java
- Besoin d'un adaptateur : Adapter



UI : Liste

■ ListView : Dynamique



■ ListView : Dynamique

- Adapter permet d'adapter l'affichage des éléments à une ListView
- ArrayAdapter :

```
ArrayAdapter<String> name =  
    new ArrayAdapter<String>(activity, layout, array);
```

activity : en général = **this**

layout : perso ou prédéfinie ex : **android.R.layout.simple_list_item_1**

array: tableau des données qui vont peupler la ListView

UI : Liste

■ ListView : Dynamique

Exemple

```
String pays[ ] = {  
    "Maroc",  
    "France",  
    "Chine",  
    "Canada"  
};
```

```
ArrayAdapter<String> myAdapter = new ArrayAdapter<String>  
(this,  
    android.R.layout.simple_list_item_1,  
    pays);
```

```
ListView mylist = (ListView) findViewById(R.id.listpays);  
mylist.setAdapter(myAdapter);
```

■ Gestion d'événement

- Malheureusement pas de onClick
- Il faut attacher un **listener** via le code java
- Utilisation d'une classe anonyme

■ Gestion d'événement

- Evénements principaux de ListView :

- **setOnItemClickListener(AdapterView.OnItemClickListener)**
- **setOnItemLongClickListener(AdapterView.OnItemLongClickListener)**
- **setOnItemSelectedListener(AdapterView.OnItemSelectedListener)**

UI : Liste

■ Gestion d'événement

- Exemple

```
mylist.setOnItemClickListener(new  
    AdapterView.OnItemClickListener() {  
        @Override  
        public void onItemClick(AdapterView<?>  
            parent, View view, int pos, long id) {  
            String p =  
                parent.getItemAtPosition(pos).toString();  
        }  
    } );
```

■ Gestion d'événement

- **Parent:** indique l'adapterView ou le clic s'est fait.
 - Il peut être : ListView, GridView, Spinner...
 - Pour implémenter un comportement sur toute la ListView :
 - Ex : parent.setVisibility(View.GONE); //cacher la liste
- **view** : Fait référence à un élément particulier du AdapterView.

Ex : un TextView dans une ligne :

```
TextView tv = (TextView) view.findViewById(R.id.txtV1);
String text = tv.getText().toString();
```

■ Gestion d'événement

- **Position** : La position de la **view** dans **Parent**.

Dans le cas de ListView : le numéro de la ligne

- **id**: l'id de la ligne de l'élément cliqué.

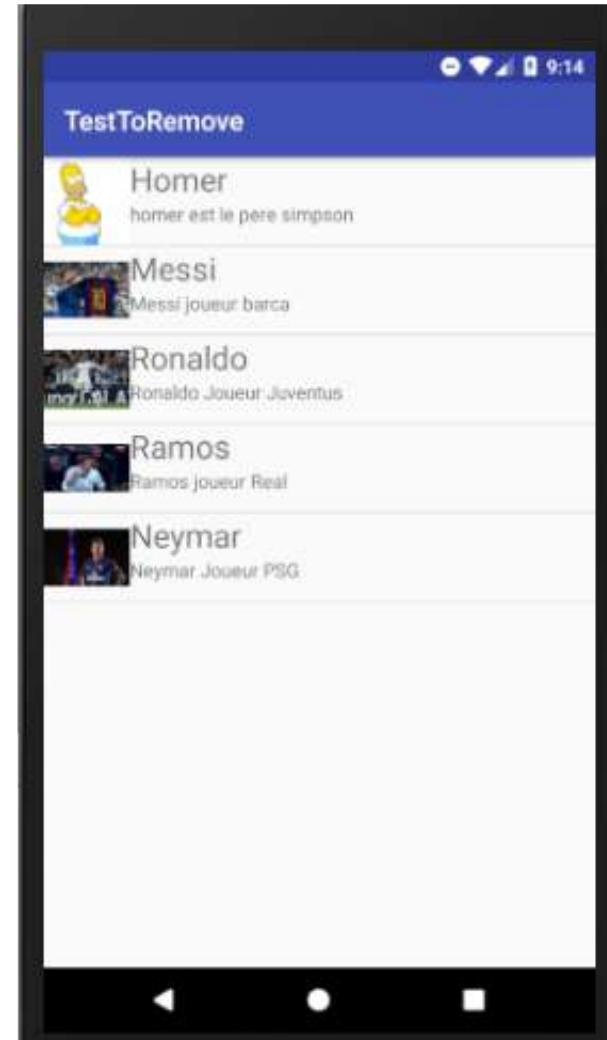
Dans le cas de ArrayAdapter = **Position**

Dans le cas de CursorAdapter = id de la ligne dans la table

UI : Liste

■ Personnaliser la liste

1. Définir le layout correspondant à chaque ligne
2. Écrire une classe Personnalisée héritant de ArrayAdapter
Redéfinir sa méthode : **getView()**



UI : Liste

■ Personnaliser la liste

```
<!-- res/layout/list_single.xml -->
```

```
<LinearLayout ... android:orientation="horizontal">
```

```
    <ImageView ...
```

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

```
        android:layout_width="100dp"
```

```
        android:layout_height="100dp"/>
```

```
    <LinearLayout ... android:orientation="vertical">
```

```
        <TextView ...
```

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

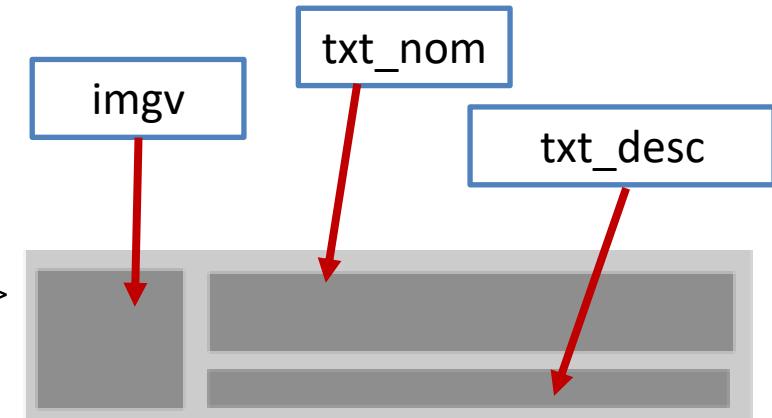
```
            android:textSize="22dp"/>
```

```
        <TextView ...
```

```
            android:id="@+id/txt_desc" />
```

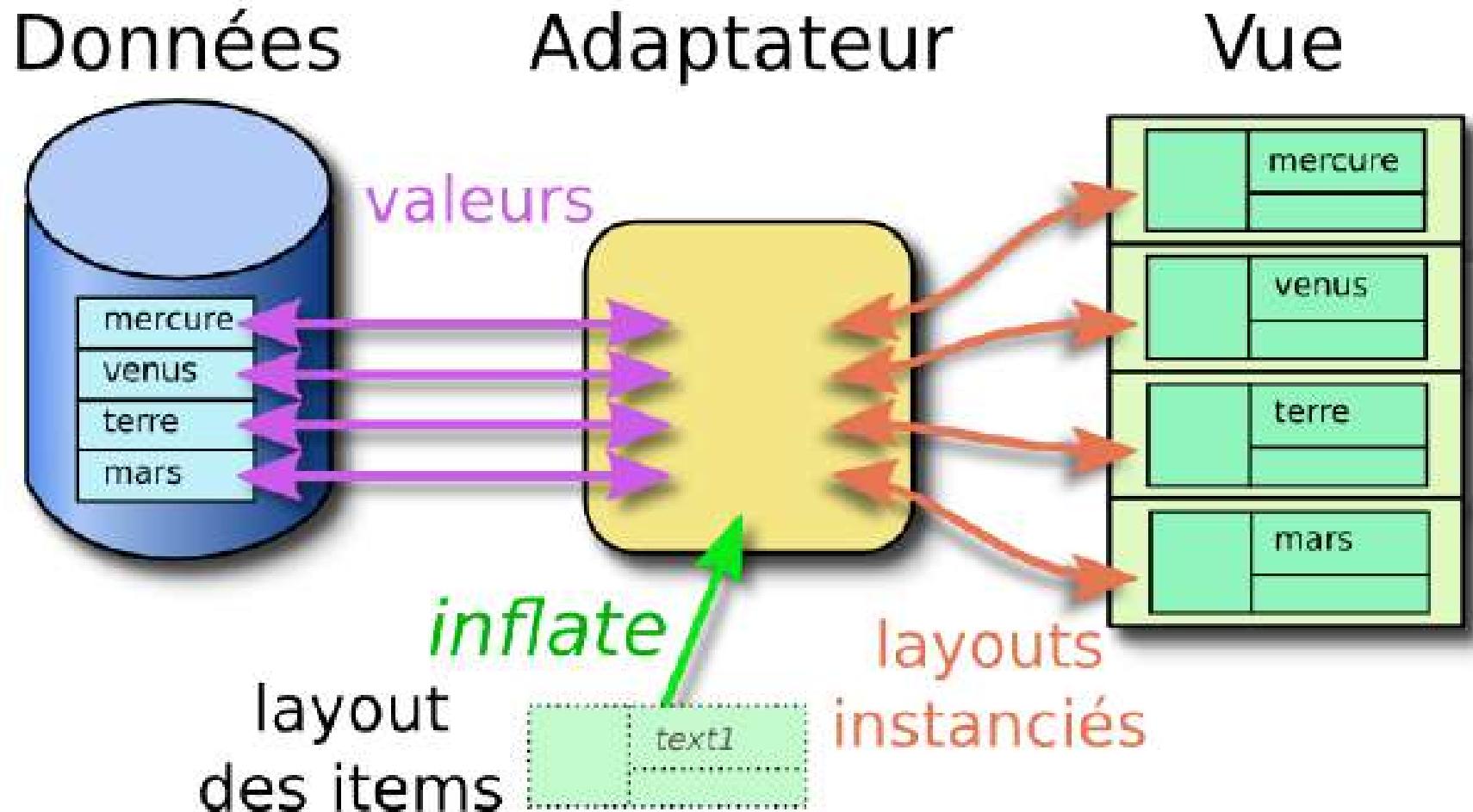
```
    </LinearLayout>
```

```
</LinearLayout>
```



UI : Liste

■ Personnaliser la liste



UI : Liste

■ Personnaliser la liste

```
public class CustomList extends ArrayAdapter<String>{
```

```
    private final Activity context;
    private final String[ ] noms, desc;
    private final Integer[ ] imagId;
```

```
    public CustomList(Activity context, String[ ] noms, String[ ] desc, Integer[ ]
                      imagId) {
```

```
        super(context, R.layout.list_single, noms);
        this.context = context;
        this.noms = noms;
        this.desc = desc;
        this.imagId = imagId;
```

```
}
```

//la suite next slide

UI : Liste

...
@Override

```
public View getView(int position, View view, ViewGroup parent) {
```

```
    LayoutInflater inflater = context.getLayoutInflater();
```

```
    View rowView= inflater.inflate(R.layout.list_single, null, true);
```

```
    ImageView imageView = (ImageView) rowView.findViewById(R.id.imgv);
```

```
    TextView tvNom = (TextView) rowView.findViewById(R.id.txt_nom);
```

```
    TextView tvDesc = (TextView) rowView.findViewById(R.id.txt_desc);
```

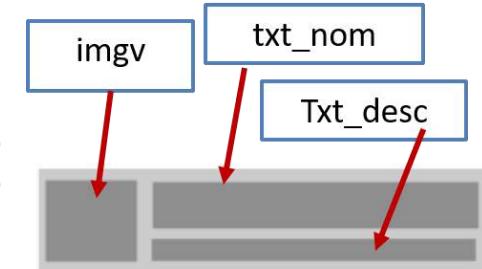
```
    tvNom.setText(noms[position]);
```

```
    tvDesc.setText(descs[position]);
```

```
    imageView.setImageResource(imageId[position]);
```

```
    return rowView;
```

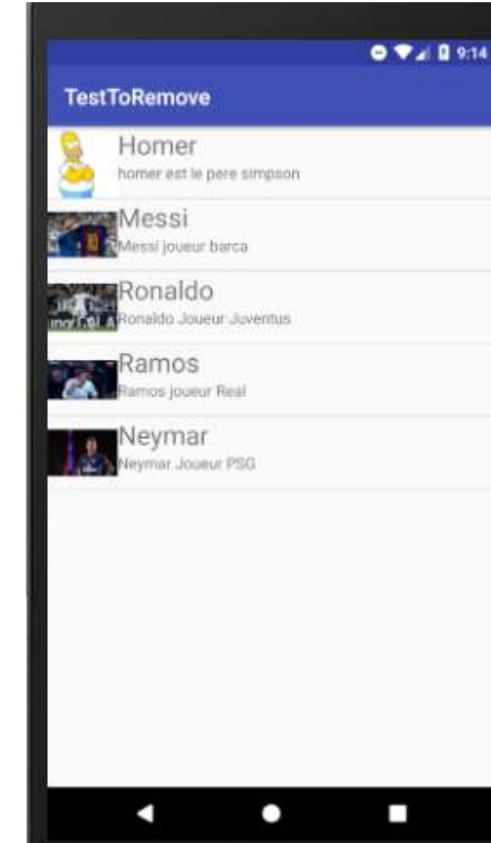
```
}
```



UI : Liste

■ Personnaliser la liste

```
public class MainActivity extends Activity {  
    ListView list;  
    String[ ] noms = {"Homer", "Messi", "Ronaldo", "Ramos", "Neymar"} ;  
    Integer[ ] imgId = {  
        R.drawable.hom, R.drawable.mess, R.drawable.rnld,  
        R.drawable.ram, R.drawable.ney};  
    String[ ] descS = {"Homer est le pere simpson", "Messi",.....} ;  
  
    @Override  
    protected void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        setContentView(R.layout.activity_main);  
  
        CustomList listAdapter = new  
            CustomList(MainActivity.this, noms, descS, imgId);  
  
        list=(ListView) findViewById(R.id.list);  
        list.setAdapter(listAdapter);  
    }  
}
```



Technologies et développement Mobile



Pr. Slimane Bah, ing. PhD

Génie Informatique option TI & IQL

Semaine 13.1

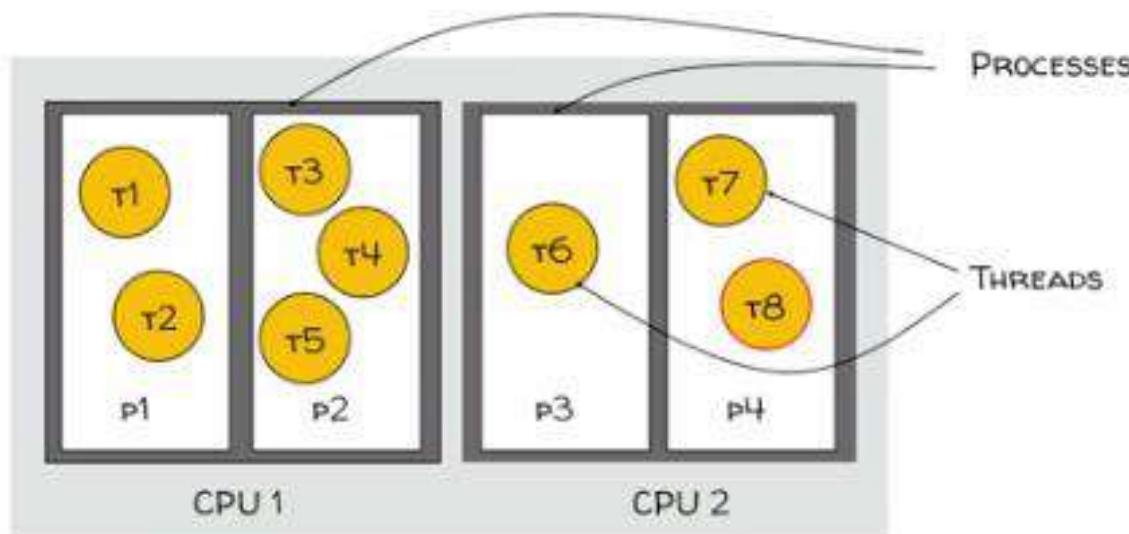
Séance 8

Android Threads

Les Threads sous Android

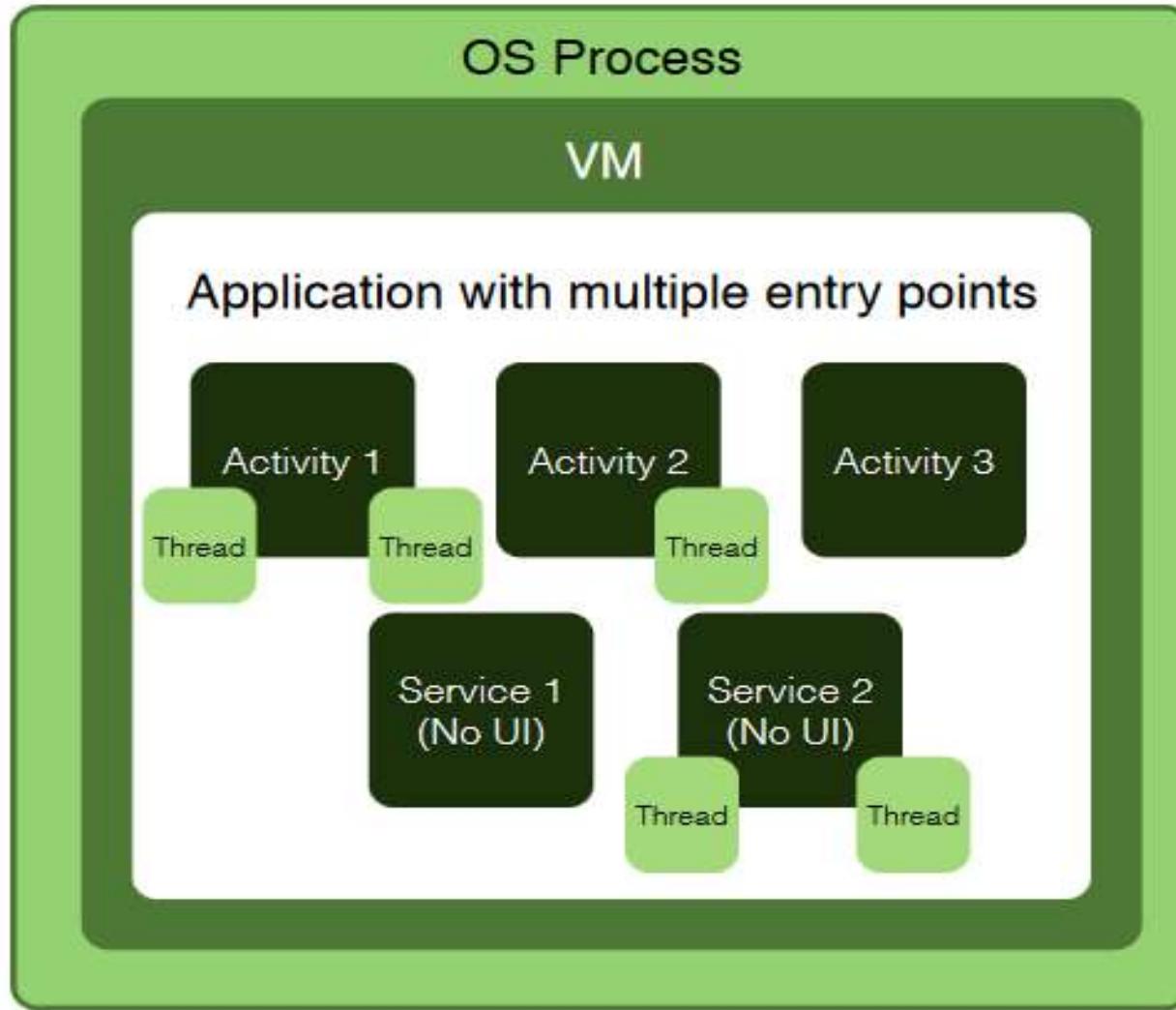
■ Thread

- Ensemble d'opérations de calculs exécutées dans un processus
- Possède son compteur de programme et sa pile
- Partage le heap et les parties statiques avec les autres Threads



Les Threads sous Android

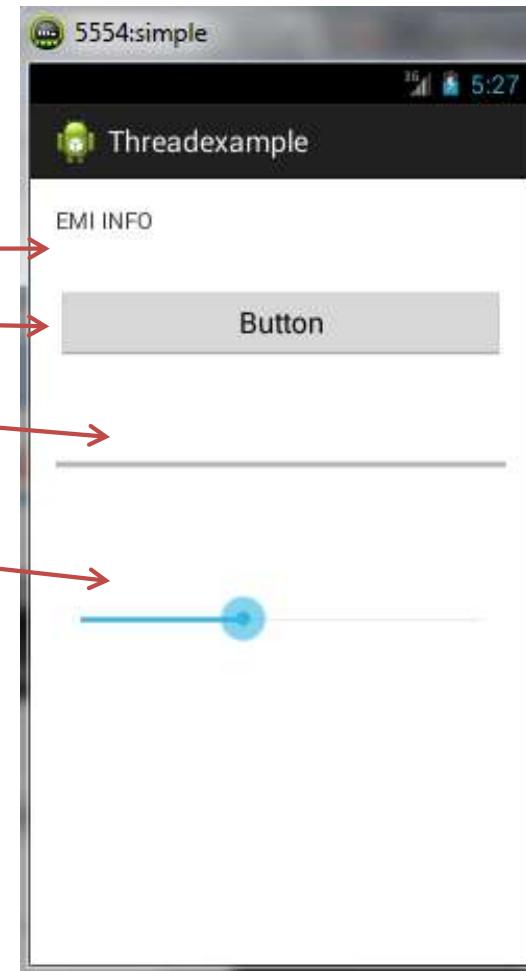
■ Threads vue Android



Les Threads sous Android

■ Exemple NoThread

- Une zone de texte
- Un bouton
- Une barre de progression
- Une Jauge



Comportement souhaité :

- Sleep 1 sec x 7
- Affichage d'un compteur toute les secondes
- Refléter l'avancement dans la barre de progression

Les Threads sous Android

■ Solution No Thread

La fonction **heavywork** est associée au bouton

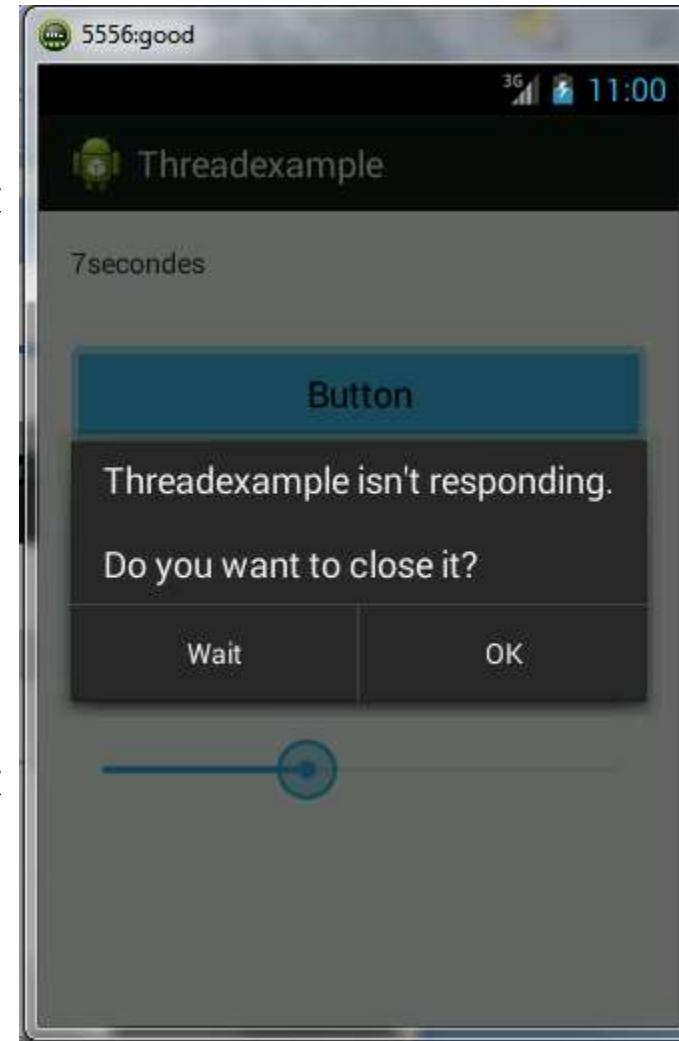
tv : TextView progress : barre de progression

```
public void heavywork(View view) {  
    try{  
        for (int i=1;i<=7;i++) {  
            Thread.sleep(1000L);  
            tv.setText(i + " : secondes");  
            progress.setProgress(i*10);  
        }  
    }  
    catch (Exception e) {  
        Log.i("errory","Erreur " + e);  
    }  
}
```

Les Threads sous Android

■ Exemple NoThread

- Le bouton est la jauge ne sont accessible qu'après les 10 secondes
- L'affichage n'est fait qu'au bout de 10 secondes (un seul affichage)
- Message ANR : Application Not Responding



Les Threads sous Android

■ Tâches en arrière plan

- Android offre plusieurs moyens pour exécuter des jobs en arrière plan :
 - Les services
 - Les threads Java

Les Threads sous Android

■ Service Vs Thread

- Un service **ne** s'exécute **pas** dans un Thread à part
- Il faut utiliser **IntentService** au lieu de **Service**
- Limite de IntentService :
 - Ne peut accéder à l'UI : doit passer par l'activité
 - L'exécution est séquentielle : l'IntentService termine d'abord une opération avant de traiter une 2^{ème}
 - Impossible d'annuler une opération démarrée

Les Threads sous Android

■ IntentService

1. Créer l'IntentService (extends IntentService)
2. L'ajouter au manifest
3. Créer et envoyer une opération à l'IntentService
4. IntentService informe l'activité du résultat d'exécution (via un BR)

Les Threads sous Android

■ Tâches en arrière plan

- Android offre plusieurs moyens pour exécuter des jobs en arrière plan :

- ~~Les services~~

- Les threads Java

Les Threads sous Android

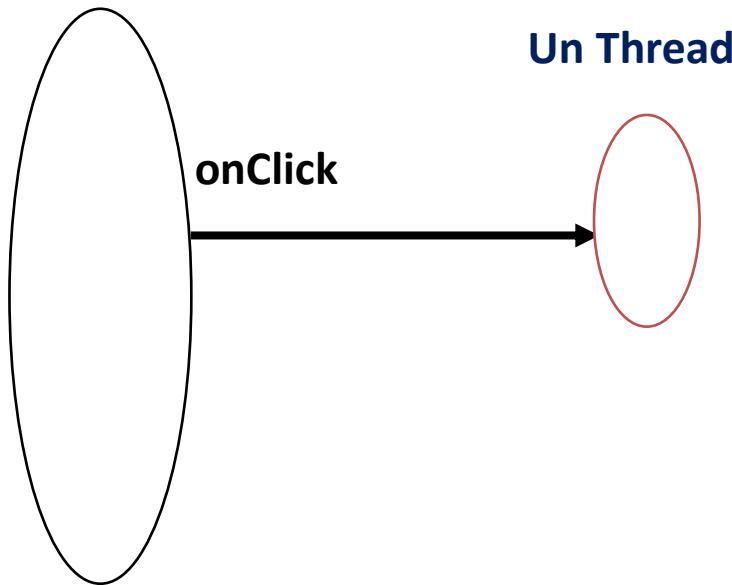
- Thread : Java
- Une classe
 - java.lang.Thread
- Une interface
 - java.lang.Runnable
- 2 méthodes obligatoires
 - start()
 - run()

Les Threads sous Android

■ Solution NaiveThread

- Créer un Thread au clic qui va dormir 1 sec x 7

Thread principal



```
public void heavywork(View view) {
    Thread t = new Thread(new Runnable() {
        public void run() {
            try{
                for (int i=1;i<=7;i++) {
                    Thread.sleep( millis: 1000L);
                    tv.setText(i + " : secondes");
                    progress.setProgress(i*10);
                }
            } catch (InterruptedException e) {
                Log.i( tag: "errory",e.toString());
            }
        }
    });
    t.start();
}
```

Les Threads sous Android

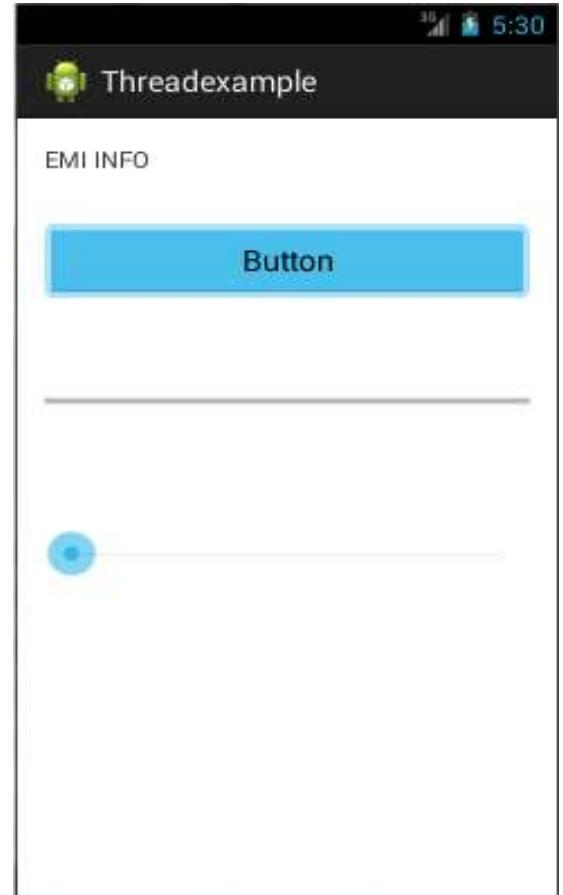
■ Solution NaiveThread

- C'est mieux !

- La jauge répond
- Le bouton est accessible en tout temps

Mais

Pas d'affichage ou incertain

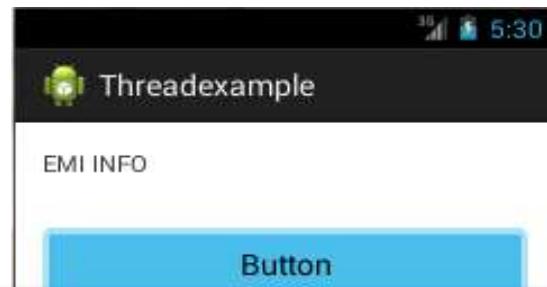


Les Threads sous Android

■ Solution NaiveThread

- C'est mieux !

- La jauge répond



	PID	TID	Application	Tag	Text
18:17:12.771	1346	1...	mine.threadex...	errory	android.view.ViewRootImpl\$CalledFromWrongThreadException: Only the original thread that created a view hierarchy can touch its views.
18:17:14.381	1346	1...	mine.threadex...	Chor...	Skipped 44 frames! The application may be doing too much work on its main thread.
18:17:14.431	1346	1...	mine.threadex...	Chor...	Skipped 30 frames! The application may be doing too much work on its main thread.
18:17:28.992	1346	1...	mine.threadex...	Chor...	Skipped 37 frames! The application may be doing too much work on its main thread.

Les Threads sous Android

■ Solution Android

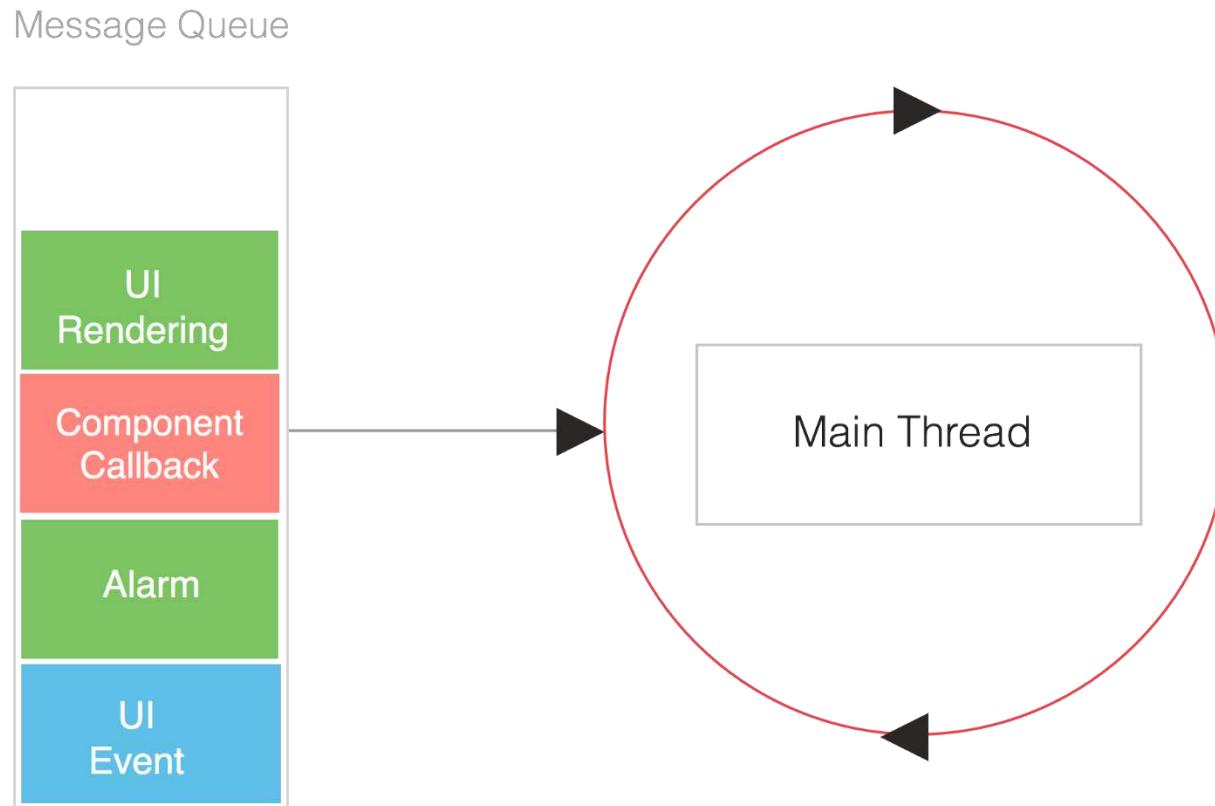
- Une application = un processus linux
- Ce processus démarre un Thread principal
- Il est nommé ***main*** alias ***UI Thread***

Son rôle :

- + Crée, initialise l'application
- + Déclenche les méthodes selon le cycle de vie
- + Est chargé de l'affichage
- + Prend en charge des actions de l'utilisateur

Les Threads sous Android

■ UI Thread (main)

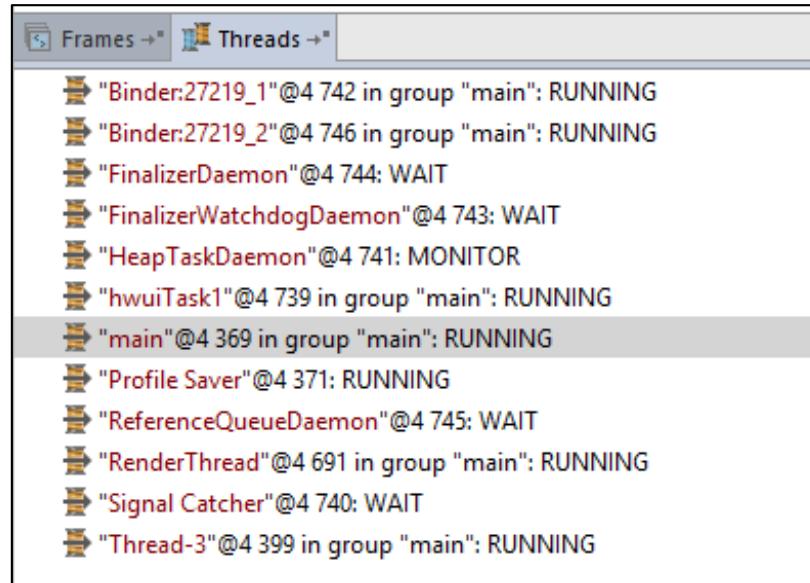


Les Threads sous Android

■ UI Thread (main)

2 règles :

- Il ne faut pas le bloquer plus de 5 secondes
- Il est le seul autorisé à accéder à l'UI toolkit
⇒ toute modification ou MAJ de l'UI **doit** passer par lui



Les Threads sous Android

■ Solution Android

Implications :

- Faire les longs calculs dans un Thread d'arrière plan
- Faire les MAJ de l'UI via l'UI Thread

Android offre plusieurs façons de faire

Les Threads sous Android

■ Solution Android

- Boolean View.post(Runnable action)
- Void Activity.runOnUiThread(Runnable action)
- Handler class
- AsyncTask class

Les Threads sous Android

■ Solution Android

Boolean View.post(Runnable action)

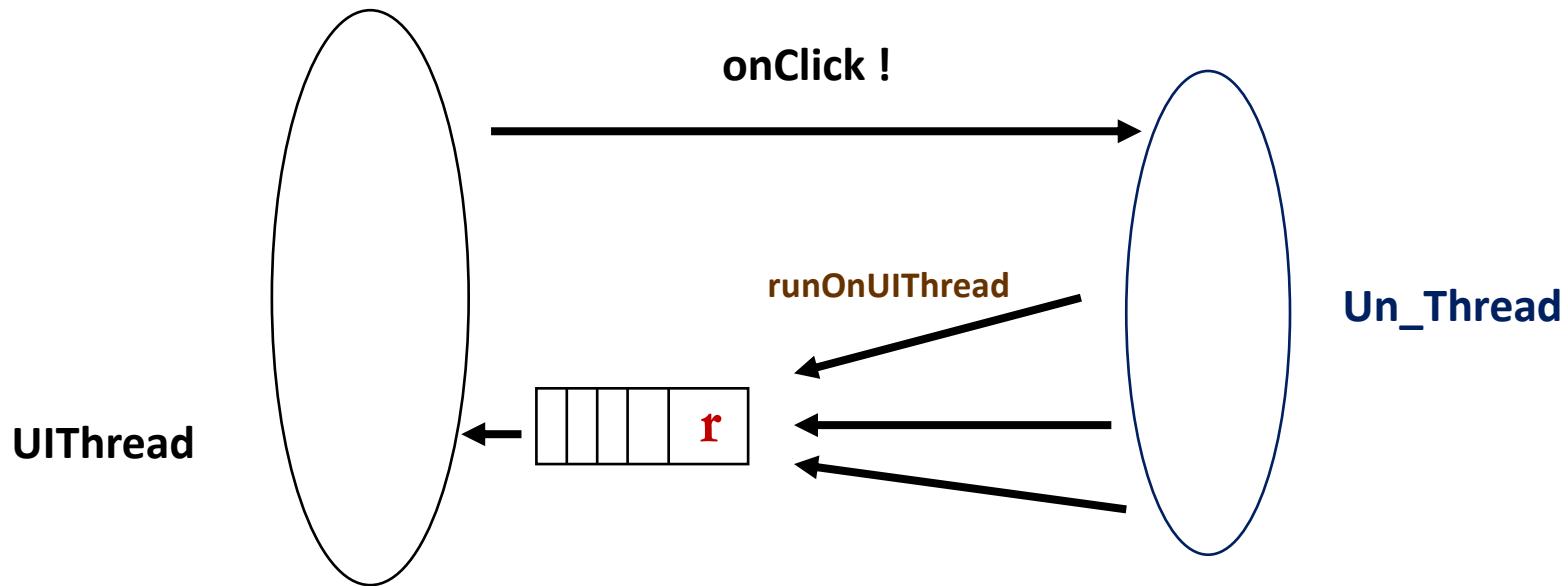
```
public void heavywork(View v) {  
  
    Thread t = new Thread(new Runnable() {  
        int i;  
        public void run() {  
            try{  
                for (i=1;i<=7;i++) {  
                    Thread.sleep(1000L);  
                    → tv.post(new Runnable() {  
                        public void run(){  
                            tv.setText(i + "secondes");  
                            progress.setProgress(i*10);  
                        }  
                    });  
  
                }  
            } catch (Exception e) {}  
        }  
    };  
    t.start();  
}
```

Les Threads sous Android

■ Solution Android

Void Activity.runOnUiThread(Runnable r)

- Méthode de la classe Activity
- Dépôt dans la file d'attente de l'UIThread



Les Threads sous Android

■ Solution Android

Void Activity.runOnUiThread(Runnable r)

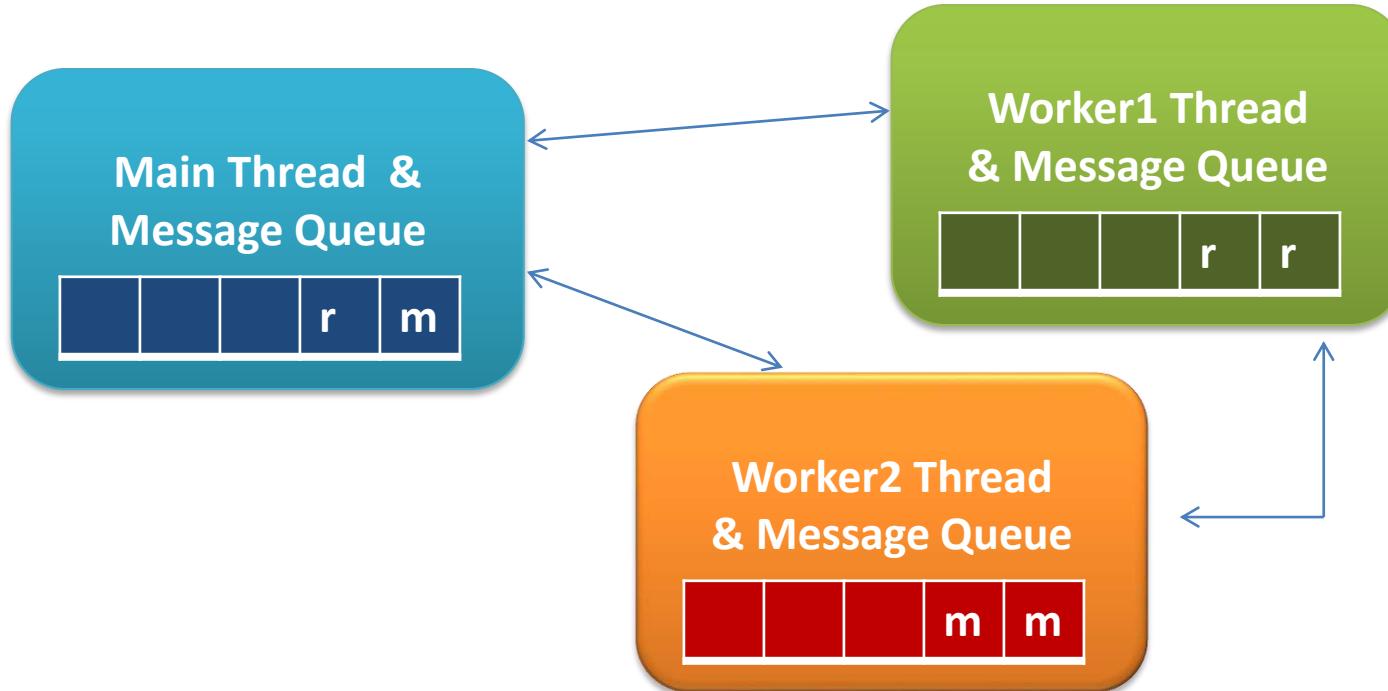
```
public void heavywork(View v) {  
  
    Thread t = new Thread(new Runnable() {  
        int i;  
        public void run() {  
            try{  
                for (i=1;i<=10;i++) {  
                    Thread.sleep(1000L);  
                    → runOnUiThread(new Runnable() {  
                        public void run(){  
                            tv.setText(i + "secondes");  
                            progress.setProgress(i*10);  
                        } });  
                }  
            } catch (Exception e) {}  
        }  
    );  
    t.start();  
}
```

Les Threads sous Android

■ Solution Android

Handler Class

Permet la communication entre 2 Threads quelconque



Les Threads sous Android

■ Solution Android

Handler Class

- Le Handler traite des objets **Message** ou des objects **Runnable**
- Les méthodes du Handler :
 - post(Runnable)
 - postAtTime(Runnable, long)
 - postDelayed(Runnable, long)
 - sendMessage(Message)
 - sendMessageAtFrontOfQueue(Message)
 - sendMessageAtTime(Message, long)
 - sendMessageDelayed(Message, long)

Les Threads sous Android

■ Solution Android Handler Class

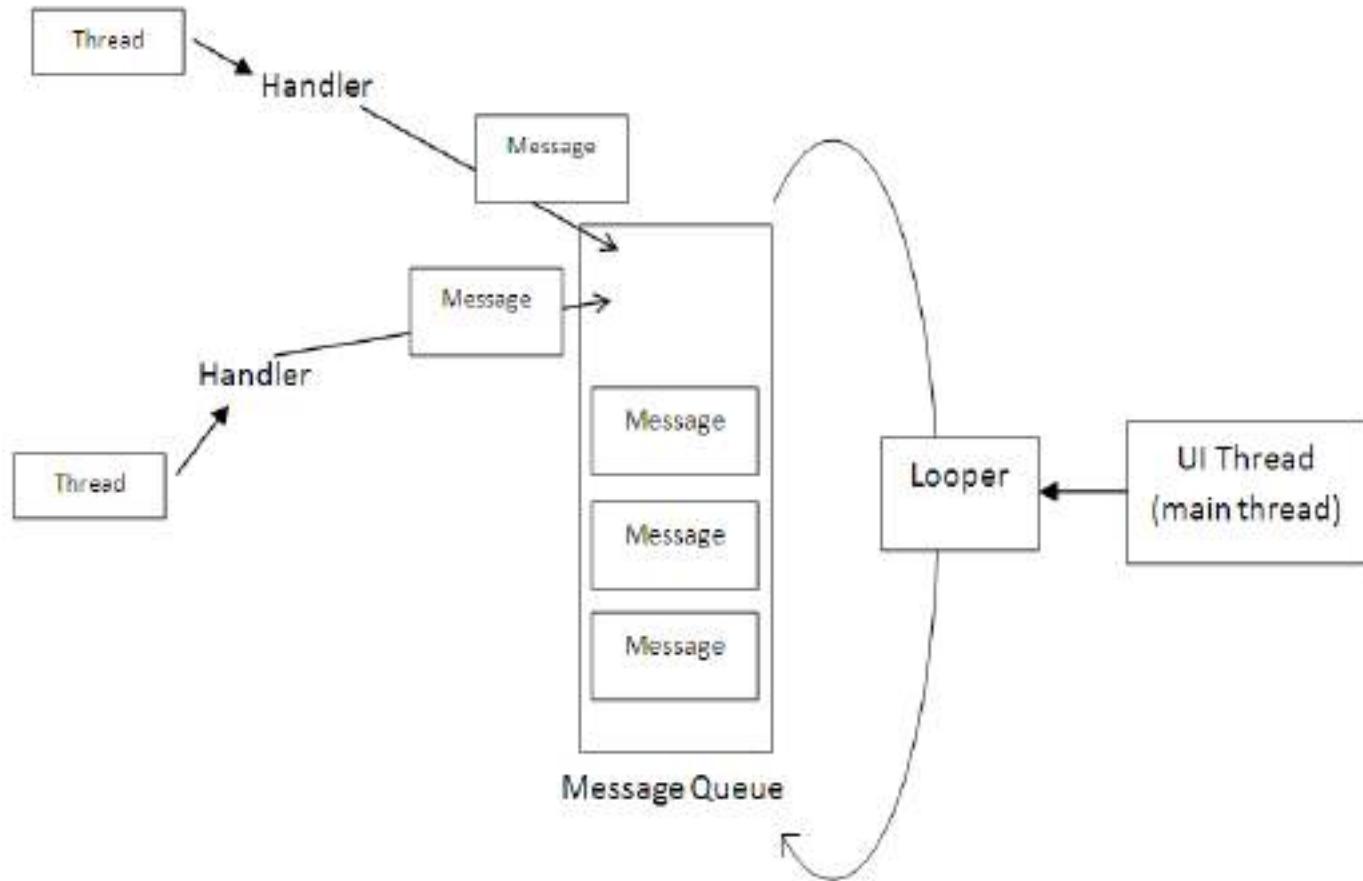
- Main Thread gère une file d'attente
- Le Handler peut être utilisé pour communiquer avec cette file d'attente
- Le Handler est attaché à la file d'attente du thread dans lequel il a été créé



Les Threads sous Android

■ Solution Android Handler Class

Envoyer un message à l'UI Thread

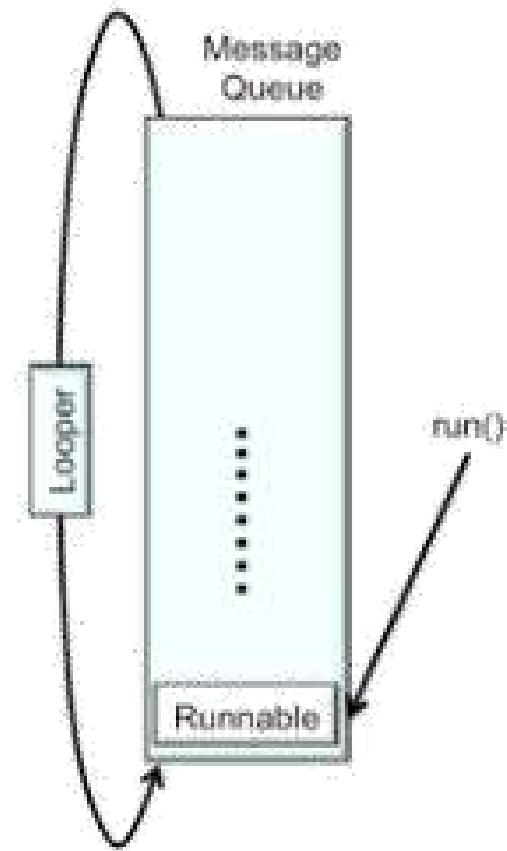
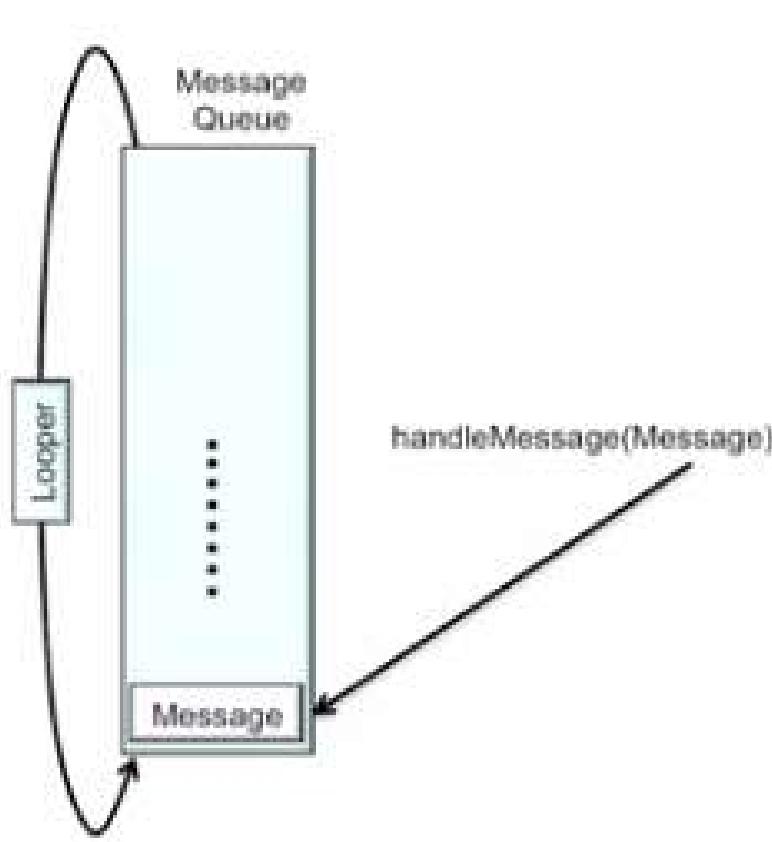


Les Threads sous Android

■ Solution Android

Handler Class

Comportement :



Les Threads sous Android

■ Solution Android

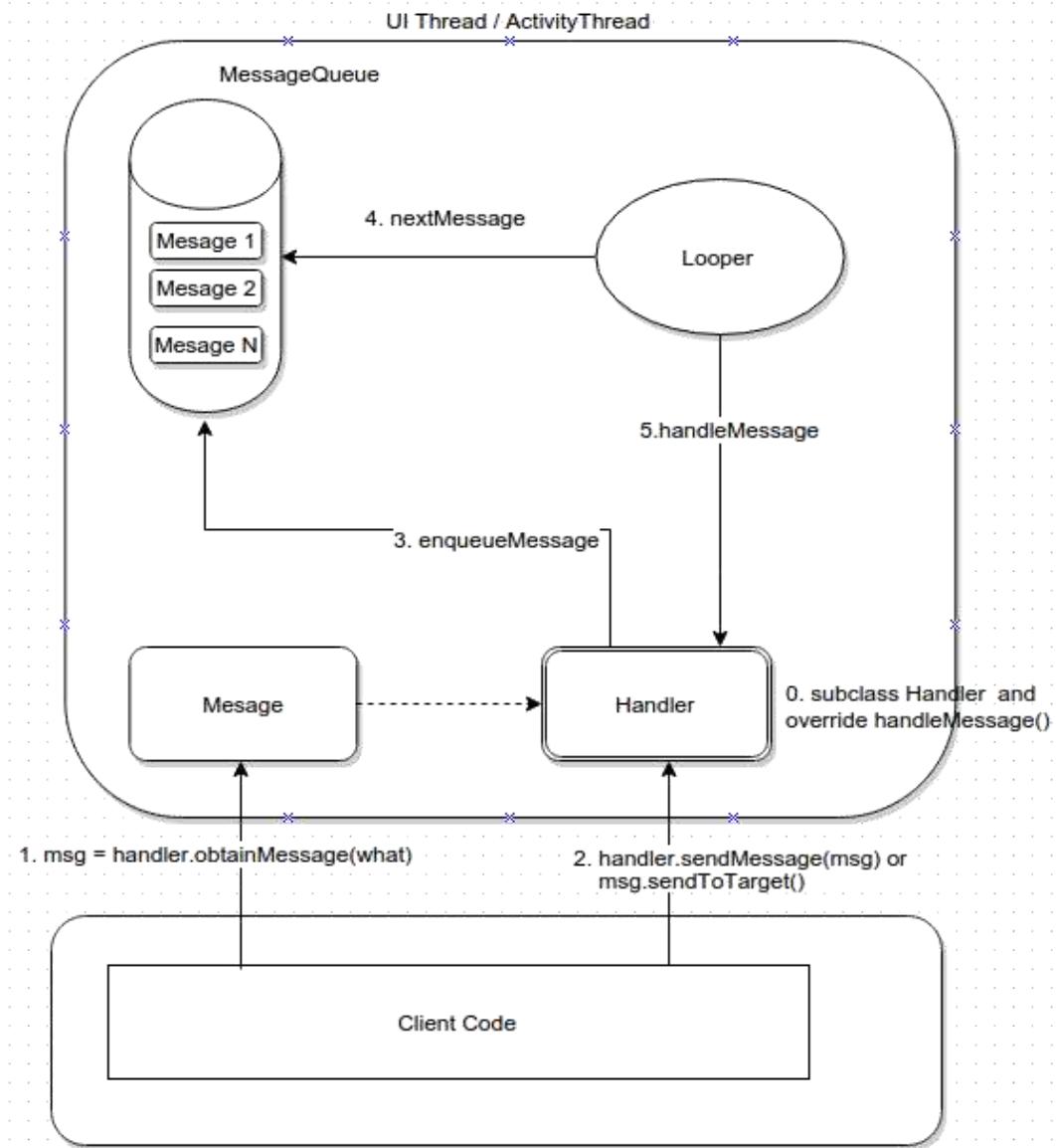
Handler Class

- Un handler comme variable d’instance de l’activité
redéfinition de la méthode **handleMessage()**
- Dans le Thread :
 - Créer le message (obtenir un token)
 - Envoyer le message

Les Threads sous Android

■ Solution Android Handler Class

message



Les Threads sous Android

■ Solution Android

Main Thread	Background Thread
<pre>... Handler myHandler = new Handler() { @Override public void handleMessage(Message msg) { // do something with the message... // update GUI if needed! ... }//handleMessage };//myHandler ...</pre>	<pre>... Thread backgJob = new Thread (new Runnable () { @Override public void run() { //...do some busy work here ... //get a token to be added to //the main's message queue Message msg = myHandler.obtainMessage(); ... //deliver message to the //main's message-queue myHandler.sendMessage(msg); }//run });//Thread //this call executes the parallel thread backgroundJob.start(); ...</pre>

Les Threads sous Android

■ Solution Android

Handler Class

```
public void heavywork(View v) {
    Thread t = new Thread(new Runnable() {
        public void run() {

            for (int i=1;i<=5;i++) {
                try{
                    Thread.sleep(1000);
                }
                catch (InterruptedException e) {}
                Message msg = handler.obtainMessage(i);
                handler.sendMessage(msg);
            }
        }
    });
    t.start();
}
```



Les Threads sous Android

■ Solution Android

Handler Class

```
public void heavyTask() {
    Thread t = new Thread(new Runnable() {
        public void run() {
            for (int i = 0; i < 1000000; i++) {
                Message msg = handler.obtainMessage();
                msg.what = i;
                handler.sendMessage(msg);
            }
        }
    });
    t.start();
}

public class MainActivity extends Activity {
    private TextView tv;
    private ProgressBar progress;

    // with handler
    private Handler handler = new Handler() {
        @Override
        public void handleMessage(Message msg) {
            tv.setText(msg.what + " sec");
            progress.setProgress(msg.what * 3);
        }
    };
}
```



Les Threads sous Android

■ AsyncTask

- Avec la Classe `AsyncTask<Params, Progress, Result>` on a un Thread et un Handler créés en interne :
 - Un thread : traitement lourd + informer de l'avancement
 - Un handler : Mise en place + MAJ de l'UI (résultats intermédiaires et finaux)

Avec :

Params type des paramètres transmis au Thread

Progress type des paramètres en cours de traitement transmis au Handler

Result type du résultat pour l'appelant

Les Threads sous Android

■ Solution Android : AsyncTask

- Depuis l'UiThread

création d'une instance et appel de la méthode execute

Exemple : `new MonAppAsync().execute(url1, url2, url3);`

- **AsyncTask<Params, Progress, Result>**

Réalise une encapsulation d'un Thread et d'un Handler

Les Threads sous Android

■ Solution Android : AsyncTask

AsyncTask<Params, Progress, Result>

Méthodes

- **onPreExecute()**

Préambule, l'UI exécute cette méthode

- **Result doInBackground(Params...p)**

Le contenu de cette méthode s'exécute dans un autre Thread

- **onProgressUpdate(Progress...p)**

Mise à jour de l'UI à la suite de l'appel de *publishProgress*

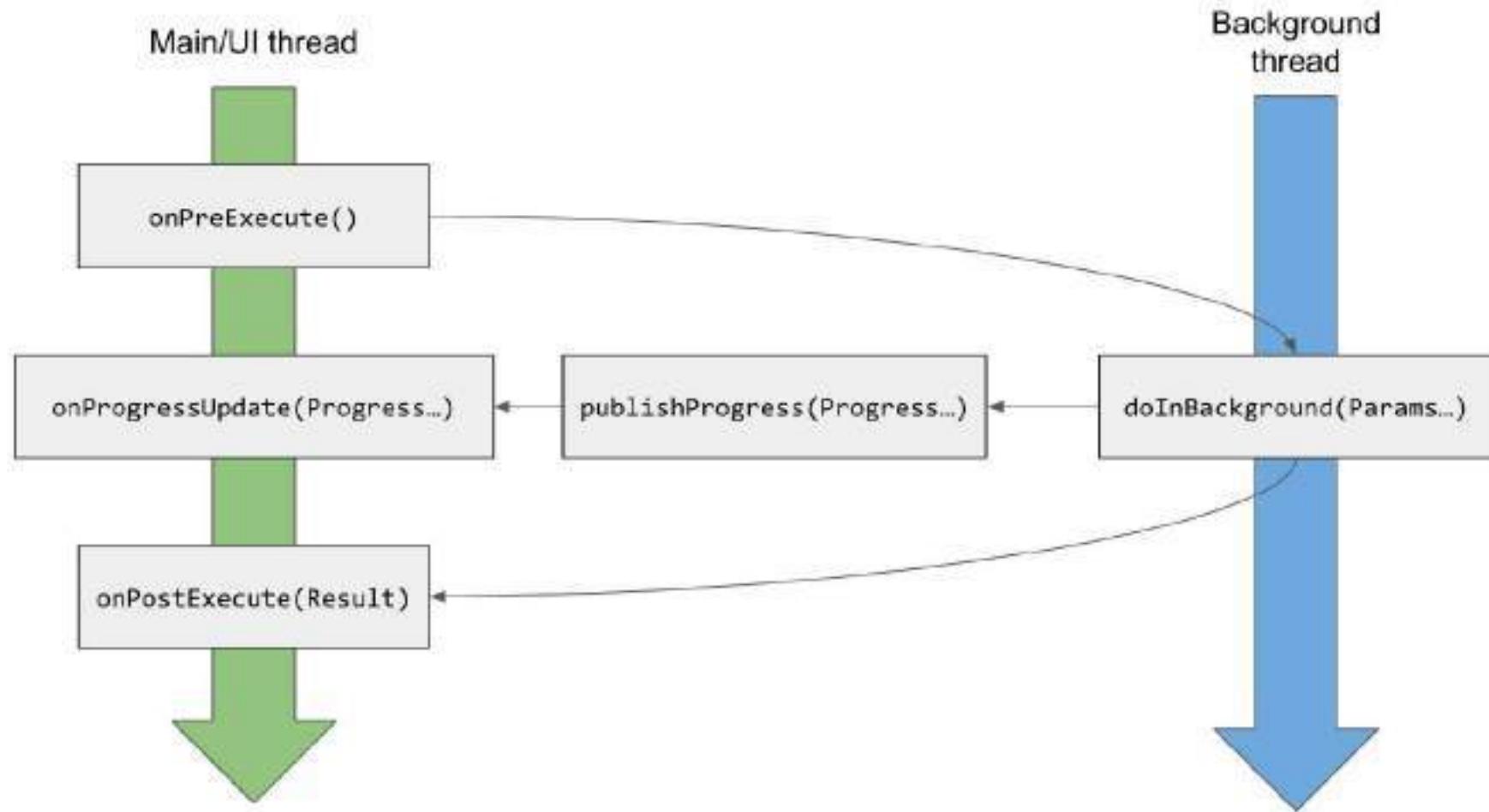


- **onPostExecute(Result)**

– Mise à jour de l'UI à la fin de la méthode *doinBackground*

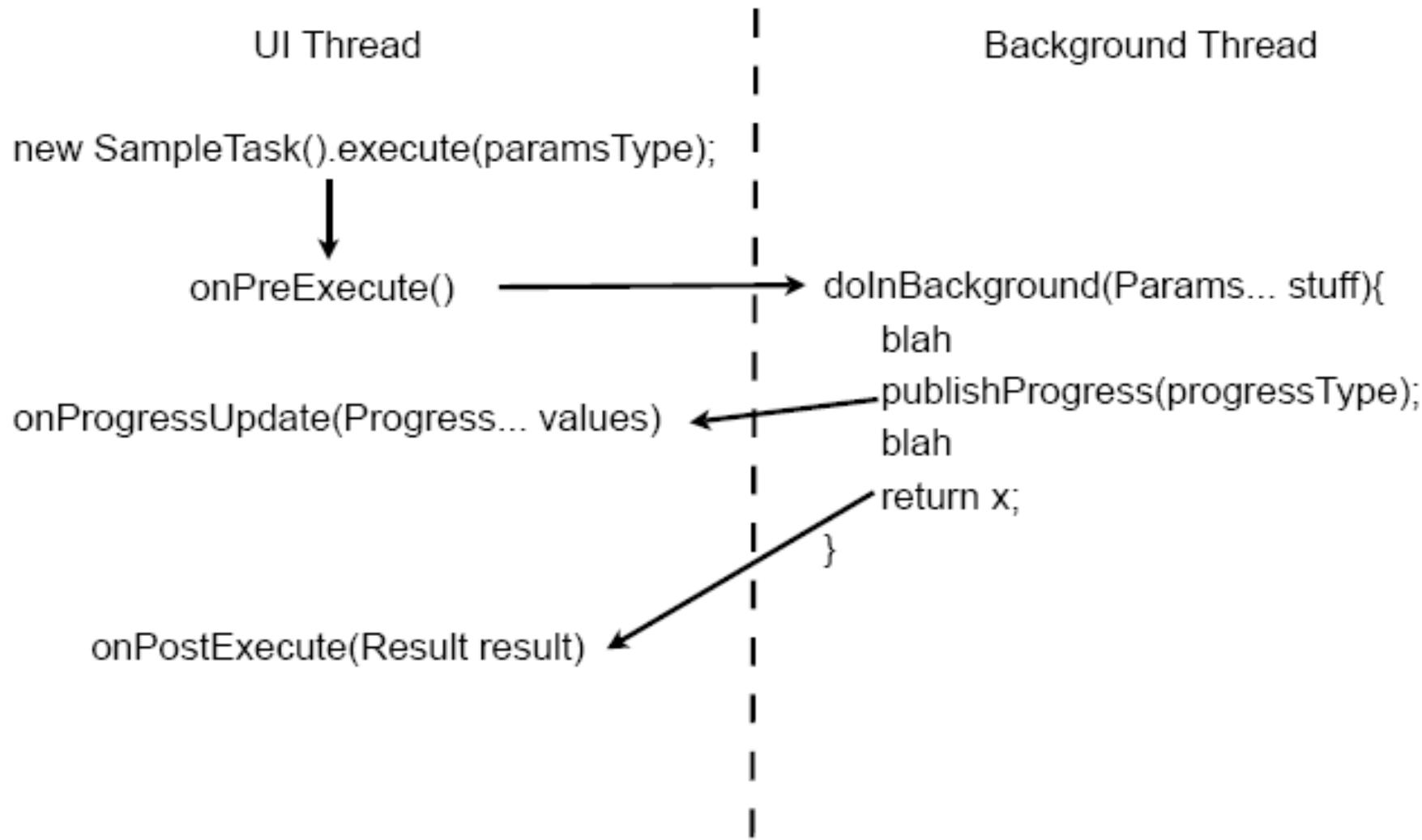
Les Threads sous Android

■ Solution Android : AsyncTask



Les Threads sous Android

```
private class SampleTask extends AsyncTask<Params, Progress, Result>
```



Les Threads sous Android

■ Solution Android

AsyncTask

- Méthode public pour l'utilisateur : **execute(param1, parm2,...)**

```
public void onStart() {  
    ...  
    MonAsyncTask mt = new MonAsyncTask();  
    mt.execute(string1, string2, string3);  
    ...  
}
```

Les Threads sous Android

■ Solution Android AsyncTask

```
private class MonAsyncTask extends AsyncTask<String, Long, Boolean>{
```

```
    void onPreExecute() { // faire patienter l'utilisateur, affichage d'un sablier...
```

```
    Boolean doInBackground(String... t){ // effectuer la tâche coûteuse en temps //  
        t[0]/string1, t[1]/string2,...
```

```
    void onProgressUpdate(Long... v) { // informer l'utilisateur que le traitement est en cours.  
        Déclenchée par publishProgress(Long... v)
```

```
    void onPostExecute(Boolean b) { // le sablier disparaît, une éventuelle erreur est affichée  
    }
```

Les Threads sous Android

■ Solution Android

AsyncTask

A chaque Clic :

```
public void heavywork(View v) {  
    new progTask().execute();  
}
```

La classe :

```
private class progTask extends AsyncTask<Void, Integer, Integer> {  
    private Context thiss = MainActivity.this;  
    private ProgressDialog p;  
  
    @Override  
    protected void onPreExecute() {  
        // TODO Auto-generated method stub  
        super.onPreExecute();  
        p = ProgressDialog.show(thiss, "starting...", "ca commence", true);  
    }
```

Les Threads sous Android

■ Solution Android

AsyncTask

La classe :

```
    @Override
    protected Integer doInBackground(Void... params) {
        int val = 0;
        try{
            Thread.sleep(500);
        }catch (Exception ee) {}
        p.dismiss();
        for (int i=1;i<=5;i++) {
            try{
                Thread.sleep(1000);
                publishProgress(i);
            }
            catch (Exception e) {}
        val = i;
        }

        return val;
    }
```

Les Threads sous Android

■ Solution Android

AsyncTask

La classe :

```
@Override  
protected void onProgressUpdate(Integer... values) {  
    // TODO Auto-generated method stub  
    super.onProgressUpdate(values);  
  
    tv.setText(values[0] + " sec");  
    progress.setProgress(values[0]);  
}
```

Les Threads sous Android

■ Solution Android

AsyncTask

La classe :

```
@Override  
protected void onPostExecute(Integer result) {  
    // TODO Auto-generated method stub  
    super.onPostExecute(result);  
    Toast.makeText(getApplicationContext(), "Info: " + result,Toast.LENGTH_LONG).show();  
}  
  
} // AsychEND
```

Technologies et développement mobile

■ Solution Android

AsyncTask : récap.

3 Types génériques	4 états principaux (méthodes)	1 Méthode auxilliaire
Params, Progress, Result	onPreExecute, doInBackground, onProgressUpdate onPostExecute.	publishProgress

Technologies et développement mobile

■ Solution Android

AsyncTask : récap.

Thread principal

Constructeur

onPreExecute

onPostExecute

onProgressUpdate

Thread privé

```
dolnBackground {  
    publishProgress()  
}
```

Broadcast Receivers

Broadcast receiver

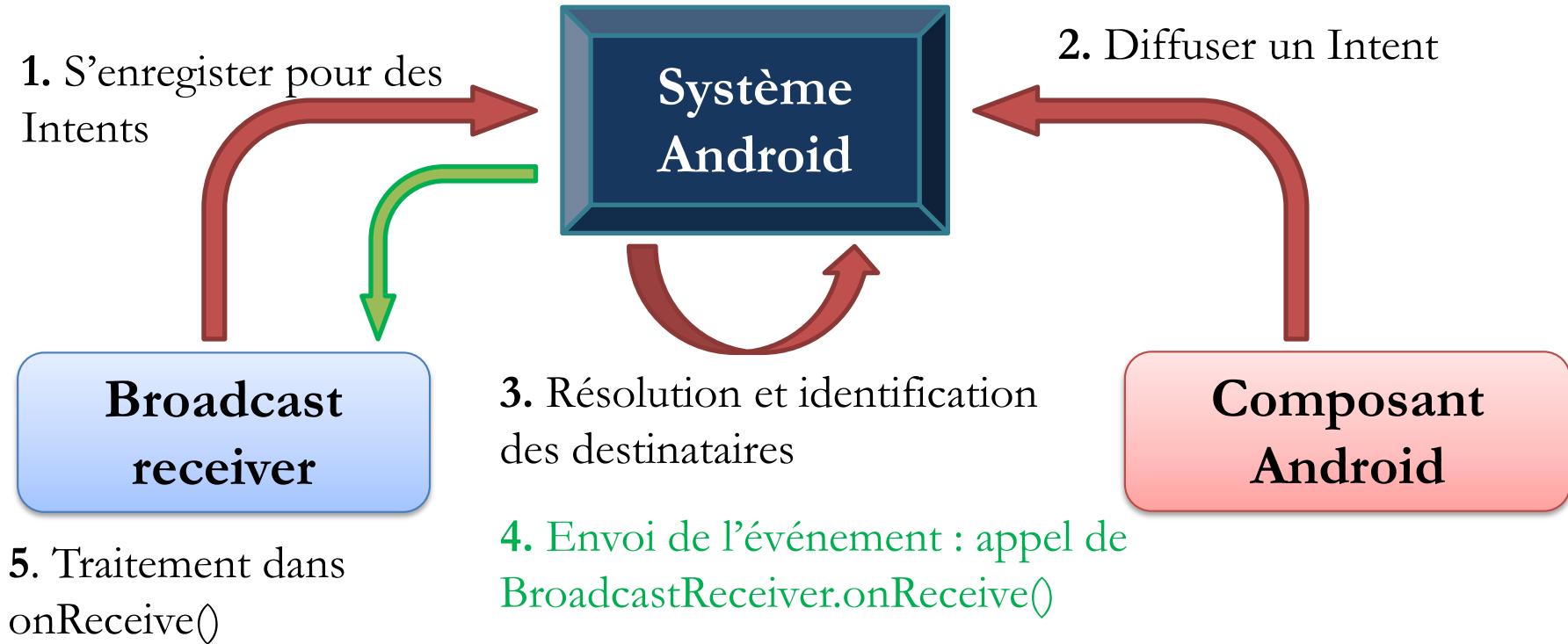
■ Broadcast Receiver (BR)

- Sans interface graphique
- Classe de base pour recevoir des événements et y réagir
 - Les événements sont envoyés sous forme d'Intent
 - Diffusés dans tout le système
 - Les BR intéressés reçoivent l'Intent via la méthode **onReceive()**

Broadcast receiver

■ Principe

- Se base sur le modèle : publish / subscribe



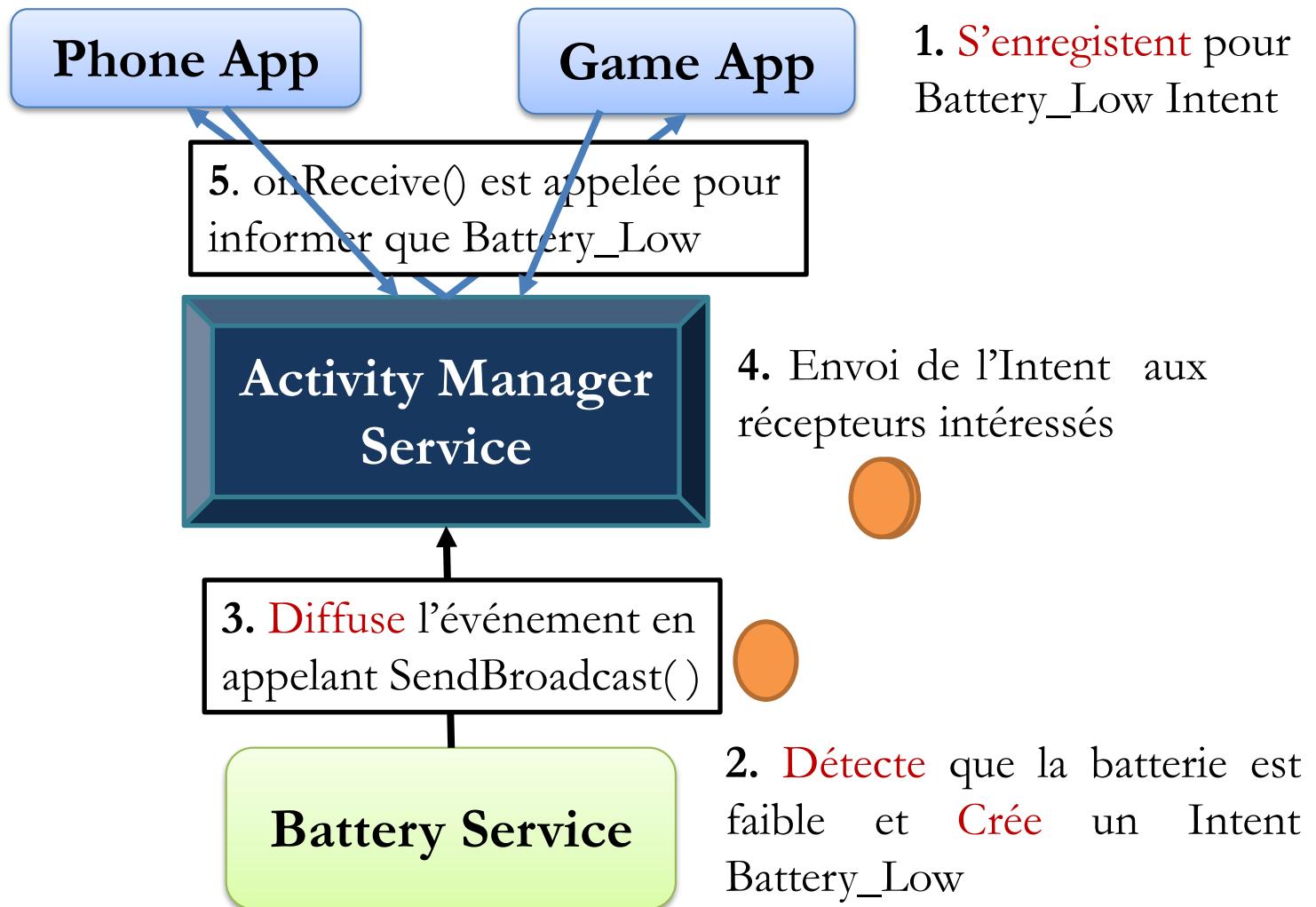
Broadcast receiver

■ Évènements système exemple

Event	Description
Intent.ACTION_BOOT_COMPLETED	Boot completed. Requires the android.permission.RECEIVE_BOOT_COMPLETED permission.
Intent.ACTION_POWER_CONNECTED	Power got connected to the device.
Intent.ACTION_POWER_DISCONNECTED	Power got disconnected to the device.
Intent.ACTION_BATTERY_LOW	Triggered on low battery. Typically used to reduce activities in your app which consume power.
Intent.ACTION_BATTERY_OKAY	Battery status good again.

Broadcast receiver

■ Exemple



Broadcast receiver

■ Types

- Statique Vs Dynamique
- Avec permission Vs sans permission
- Normal Vs Ordered
- Local ([LocalBroadcastManager.sendBroadcast](#))

Broadcast receiver

■ Principe

- L'enregistrement peut se faire d'une manière :

1. Statische :

```
<application ...>
    <activity android:name=".SimpleBroadcast" ...> ... </activity>
    <receiver android:name=".myReceiver">
        <intent-filter android:priority="5">
            <action android:name = "Emi.BroadcastReceiver.action.TEST">
            <action android:name = "android.intent.action.BOOT_COMPLETED" >
                </action>
            </intent-filter>
        </receiver>
    </application>
```

Se fait au démarrage du système ou quand le package est ajouté

Broadcast receiver

■ Principe

2. Dynamique :

- Créer un Intent filter
- Créer un Broadcast Receiver
- Enregister le BR via **Context.registerReceiver()** pour recevoir des Intent correspondant à l'Intent filter (dans onResume())
- Libérer le BR **Context.unregisterReceiver()** (dans onPause())

Broadcast receiver

■ Principe

2. Dynamique :

```
IntentFilter filter = new IntentFilter(Intent.ACTION_BATTERY_LOW);
MyReceiver myReceiver = new MyReceiver();
registerReceiver(myReceiver, filter);
```

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

Broadcast receiver

- Types : Normal avec/sans Permission

```
// send Intent to interested BroadcastReceivers  
void sendBroadcast (Intent intent)
```

```
// send Intent to interested BroadcastReceivers  
// if they have the specified permissions  
void sendBroadcast (Intent intent, String receiverPermission)
```

Broadcast receiver

■ Types : Ordered

```
// send Intent to interested BroadcastReceivers in priority order
void sendOrderedBroadcast (Intent intent,
                           String receiverPermission)

// send Intent to interested BroadcastReceivers in priority order
// sender can provide various parameters for greater control
void sendOrderedBroadcast (Intent intent,
                           String receiverPermission,
                           BroadcastReceiver resultReceiver,
                           Handler scheduler,
                           int initialCode,
                           String initialData,
                           Bundle initialExtras)
```

Broadcast receiver

■ Gestion des événements (réception)

- Normale

```
public class Receiver1 extends BroadcastReceiver{
```

```
    public void onReceive(Context context, Intent intent) {  
        System.out.println(this+ ": INTENT RECU");  
        Vibrator v = (Vibrator)  
        context.getSystemService(Context.VIBRATOR_SERVICE);  
        v.vibrate(500);  
    }  
}
```

Broadcast receiver

■ Gestion des événements (réception)

- Ordered

```
public class Receiver1 extends BroadcastReceiver{  
    public void onReceive(Context context, Intent intent) {  
        String tmp = getResultData() != null ? getResultData() : "";  
        setResultData(tmp+ ": Receiver 1:");  
    }  
}
```

Broadcast receiver

- Gestion des événements (réception)
- Ordered

```
public class Receiver1 extends BroadcastReceiver{  
    public void onReceive(Context context, Intent intent) {  
        if (isOrderedBroadcast()) {abortBroadcast();}  
        System.out.println(this+ ": INTENT RECU");  
    }  
}
```

Technologies et développement Mobile



Pr. Slimane Bah, ing. PhD

Génie Informatique option TI & IQL

Semaine 13.2

Séance 9

AsyncTask

- Dépréciée mais non retirée
- Sol préférée : Kotlin – Coroutine
- Sol Java :

```
ExecutorService executor = Executors.newSingleThreadExecutor();
Handler handler = new Handler(Looper.getMainLooper());
executor.execute(() -> {
    //Background
    handler.post(() -> {
        //UI Thread });
});
```

Permissions personnalisées

Les permissions personnalisées

Permissions personnalisées

■ Les permissions personnalisées

- Lorsque votre application fait des opérations dangereuses ou privilégiées elle peut exiger des permissions
 - ⇒ Empêcher n'importe quel composant d'utiliser votre application
- Il faudra **déclarer** et **renforcer** des permissions personnalisées
- Les autres applications devront demander la permission sinon une exception se produira

Permissions personnalisées

■ Les permissions

1. Il faut déclarer la permission

```
<!-- Defines a custom permission -->
<permission
    android:name="course.examples.permissionexample.BOOM_PERM"
    android:description="@string/boom_perm_string"
    android:label="@string/boom_permission_label_string" >
</permission>
```

```
<permission android:name="com.example.perm.READ_INCOMING_MSG"
    android:label="Read incoming messages from the EX service."
    android:description="Allows the app to access any messages received by
        the EX service. Any app granted this permission will be able to read all
        messages processed by the ex1 application."
    android.protectionLevel="dangerous"
    android:permissionGroup="android.permission-group.PERSONAL_INFO"
/>
```

Permissions personnalisées

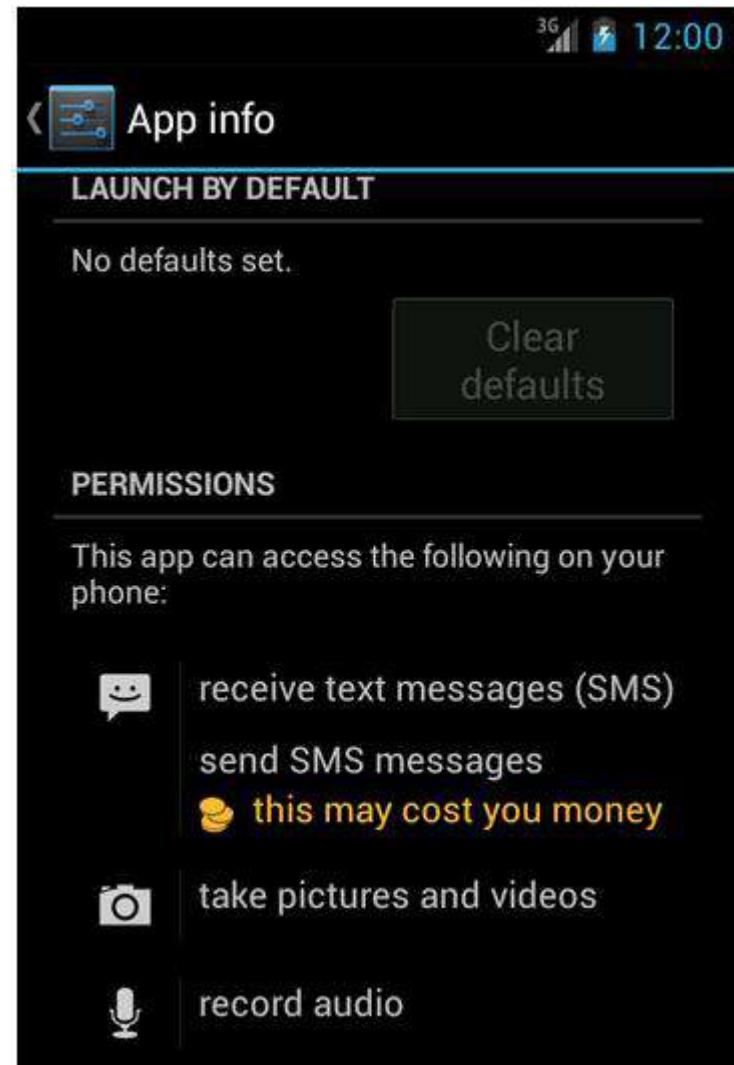
■ Les permissions

La déclaration de la permission doit inclure

- Le nom
- Le label
- La description
- Un niveau

Peut inclure

- Une icone
- Un groupe



Permissions personnalisées

■ Les permissions

2. Il faut renforcer la permission **dans le Manifest**

```
<!-- Enforces the BOOM_PERM permission on users of this application -->
<application
    android:allowBackup="true"
    android:icon="@drawable/ic_launcher"
    android:label="@string/app_name"
    android:permission="course.examples.permissionexample.BOOM_PERM" >
    <activity
        android:name=".BoomActivity"
        android:label="@string/app_name" >
        <intent-filter>
            <action android:name="android.intent.action.MAIN" />

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

Permissions personnalisées

■ Les permissions

- Il est possible d'ajouter des permissions pour **un composant** en particulier et non pour l'application en entier

⇒

Activité : Elle ne peut pas démarrer sans la permission

Service : ne peut pas démarrer ou bind à une activité

Broadcast receivers : ignorent les messages reçus sauf si permission

Content Provider : possède 2 permissions Read / Write

Permissions personnalisées

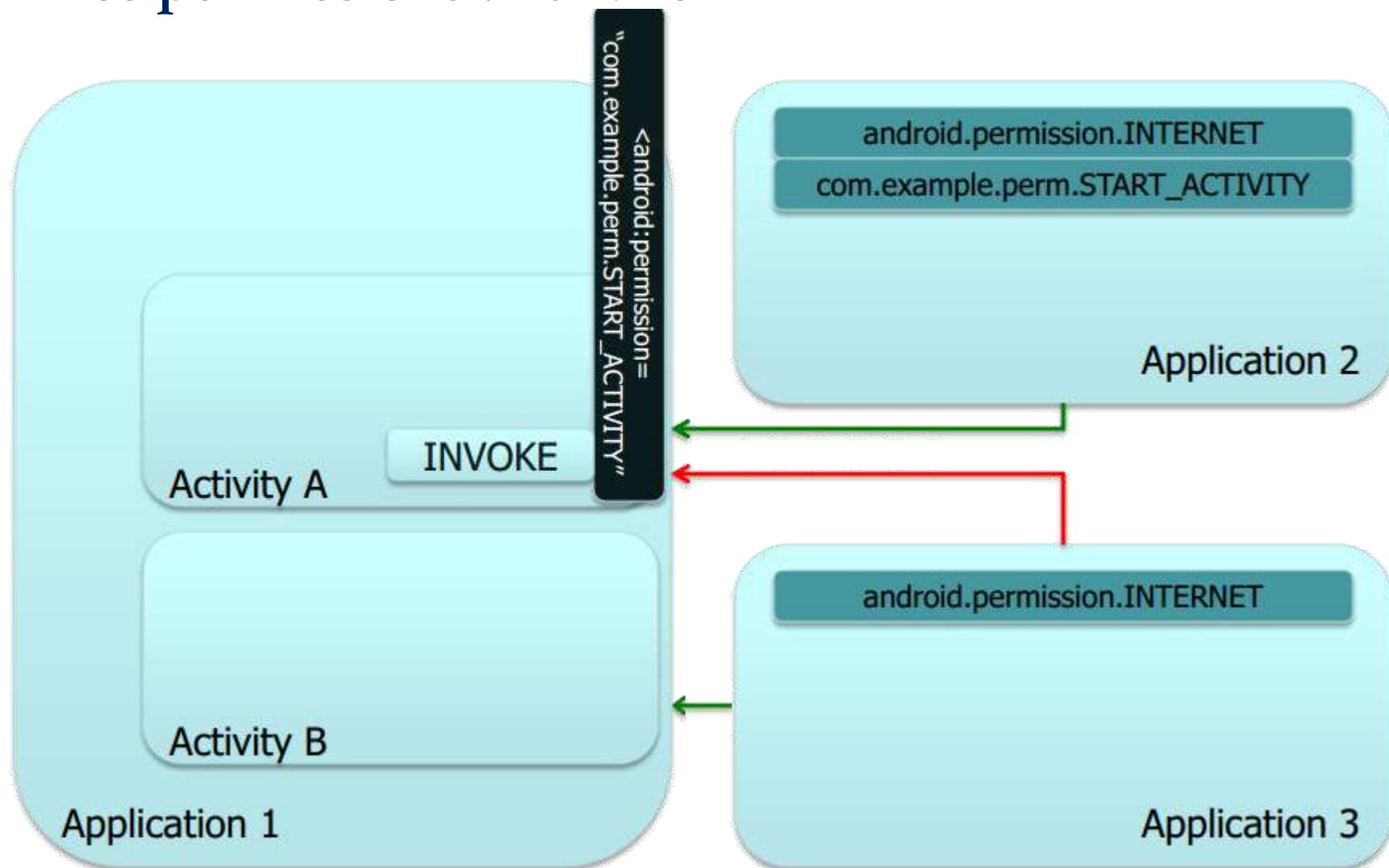
■ Les permissions

Renforcer la permission dans une activité

```
<activity android:name=".ActivityA"
          android.permission="com.example.perm.START_ACTIVITY">
    <intent-filter>
        ...
    </intent-filter>
</activity>
```

Permissions personnalisées

■ Les permissions : activité



Permissions personnalisées

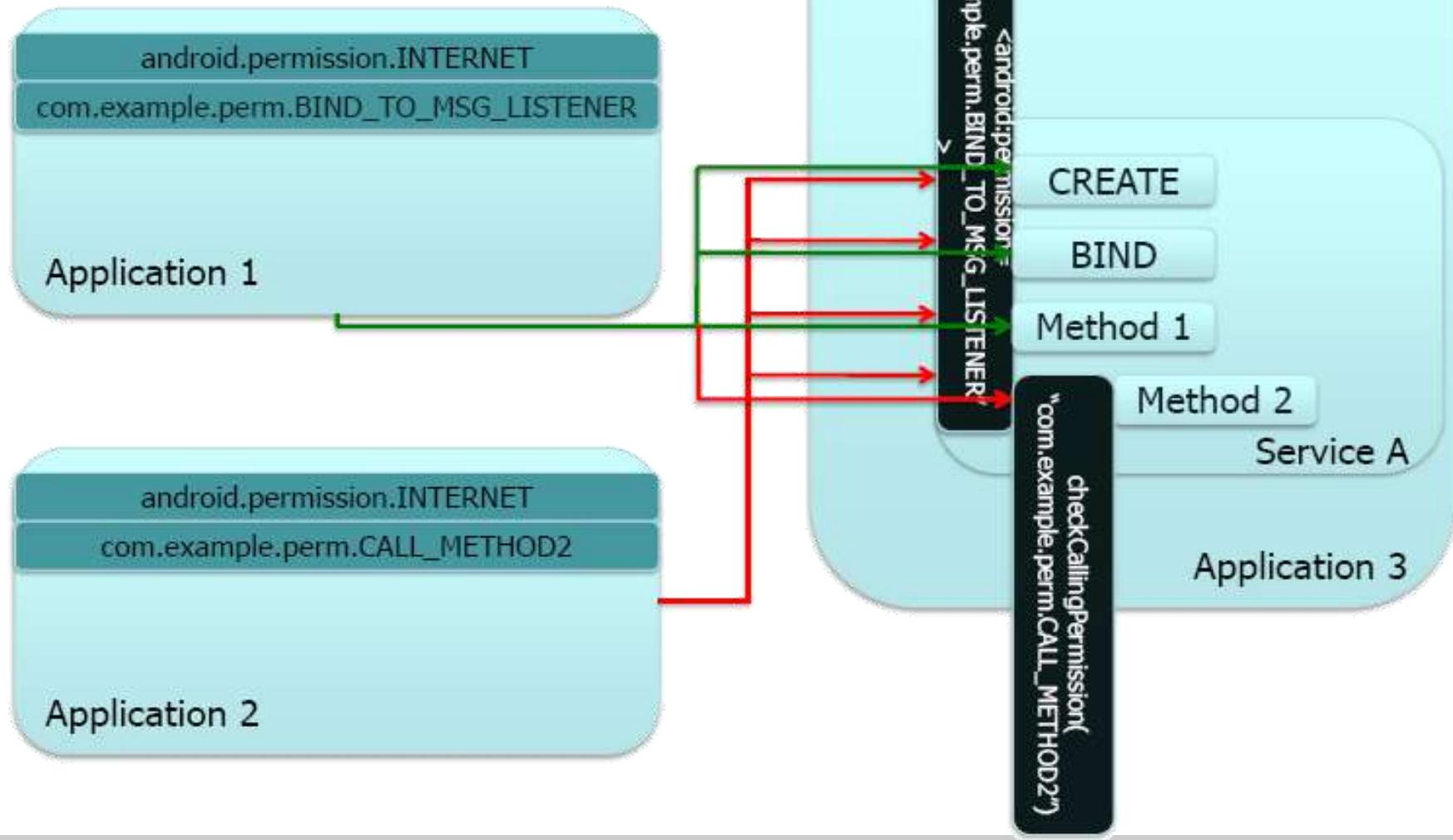
■ Les permissions : service

```
<service android:name=".MailListenerService"
    android:permission="com.example.perm.BIND_TO_MSG_LISTENER"
    android:enabled="true"
    android:exported="true"
    <intent-filter></intent-filter>
</service>
```

- Ajoute des restrictions sur l'interaction avec le service (création, Bind, appel des méthodes sur le service)
 - Ne permet pas de spécifier des permissions particulières pour certaines méthodes du service
- ⇒ Utiliser **checkCallingPermission()** dans chaque méthode en plus du manifest

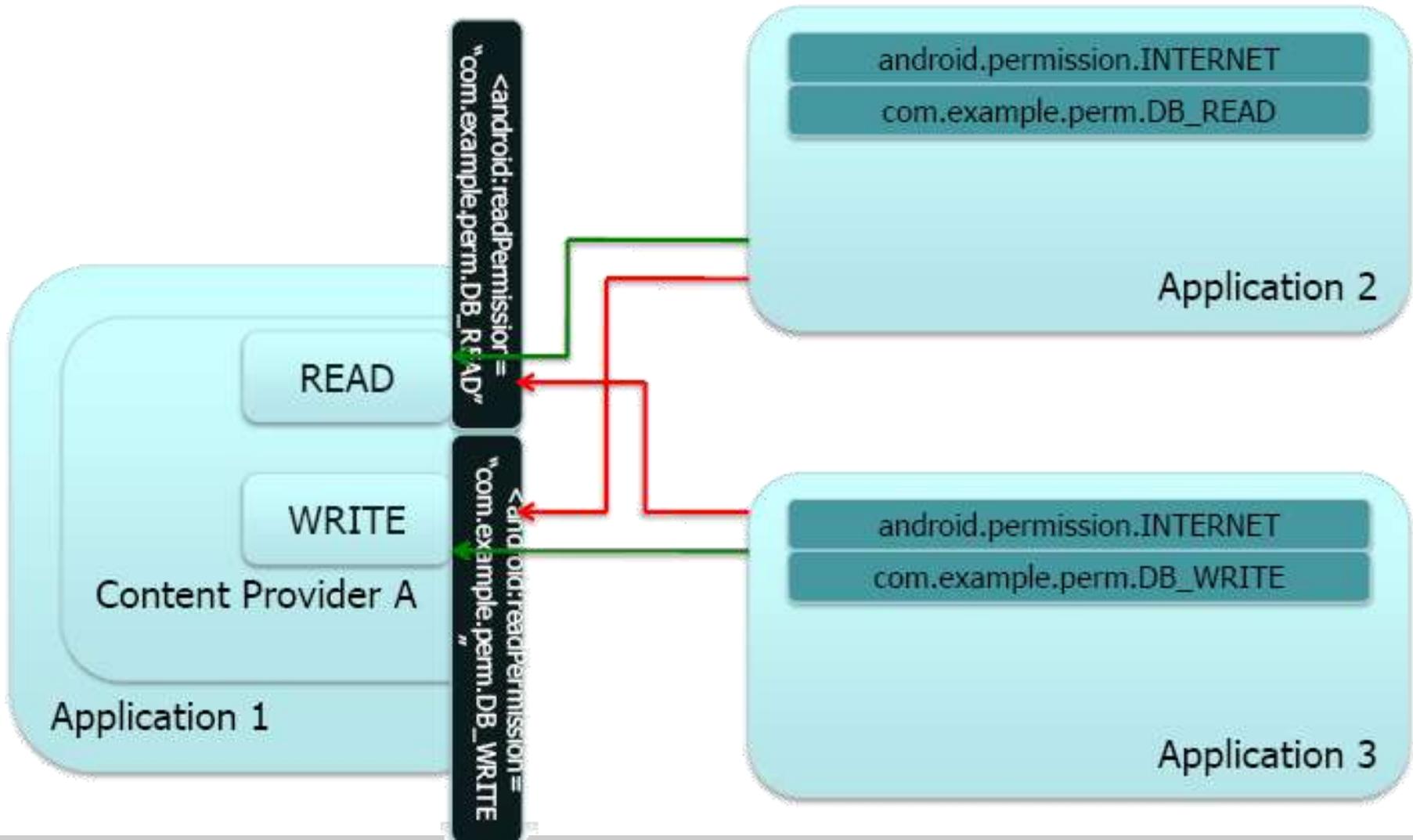
Permissions personnalisées

■ Les permissions : service



Permissions personnalisées

■ Les permissions : content provider



Permissions personnalisées

■ Les permissions : content provider

- Spécifier des permissions pour une branche du content provider
permission temporaire

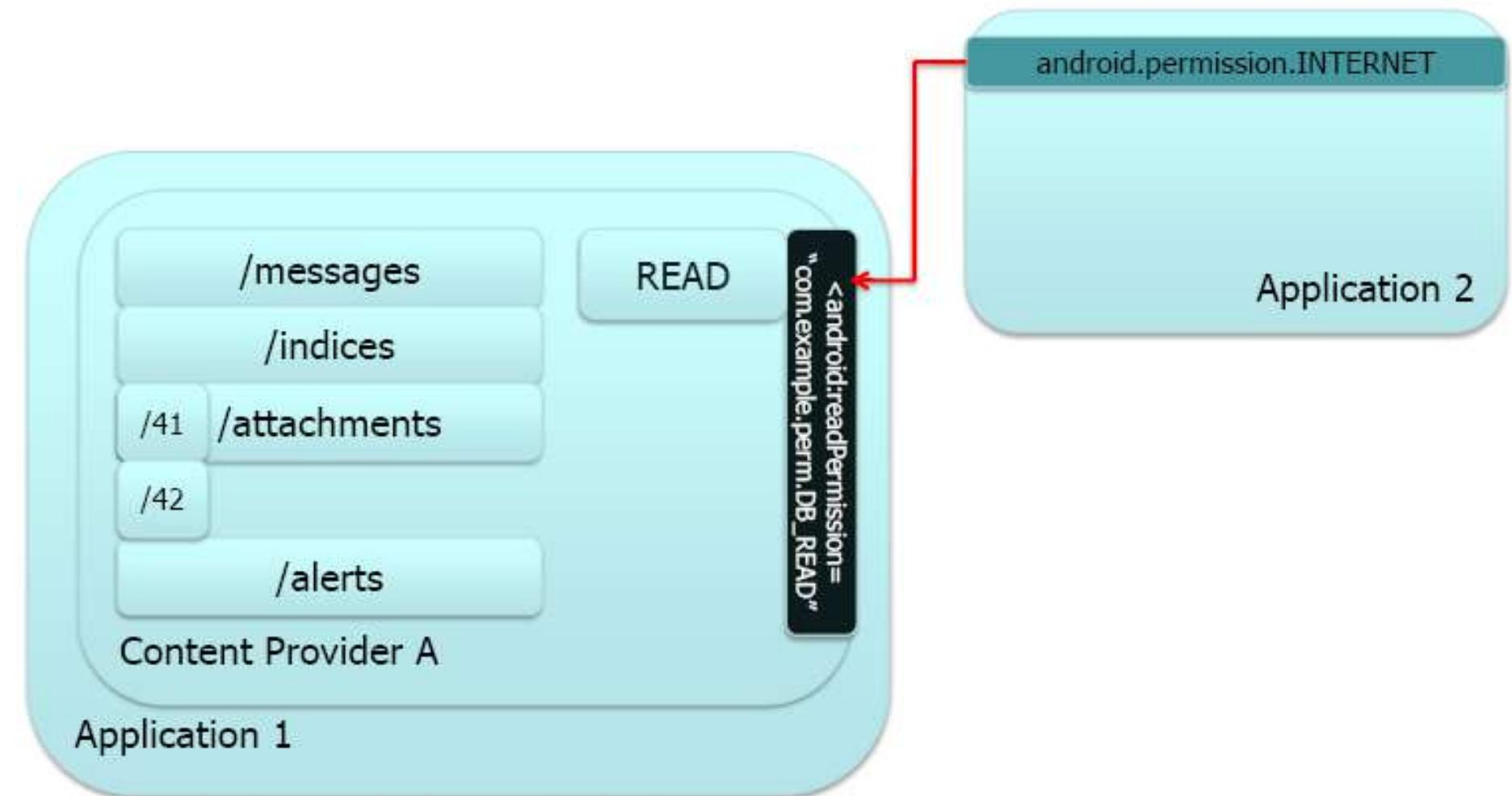
```
<provider android:name="com.example.test.app1.MailProvider"  
        android:authorities="com.example.test.app1.mailprovider"  
        android:readPermission="com.example.perm.DB_READ"  
        android:writePermission="com.example.perm.DB_WRITE">  
    <grant-uri-permission android:path="/attachments/">  
</provider>
```

- Accorder la permission à une application :

```
uri = "content://com.example.test.provider1/attachments/42";  
Context.grantUriPermission("com.example.test.app2", uri,  
    intent.FLAG_GRANT_READ_URI_PERMISSION);
```

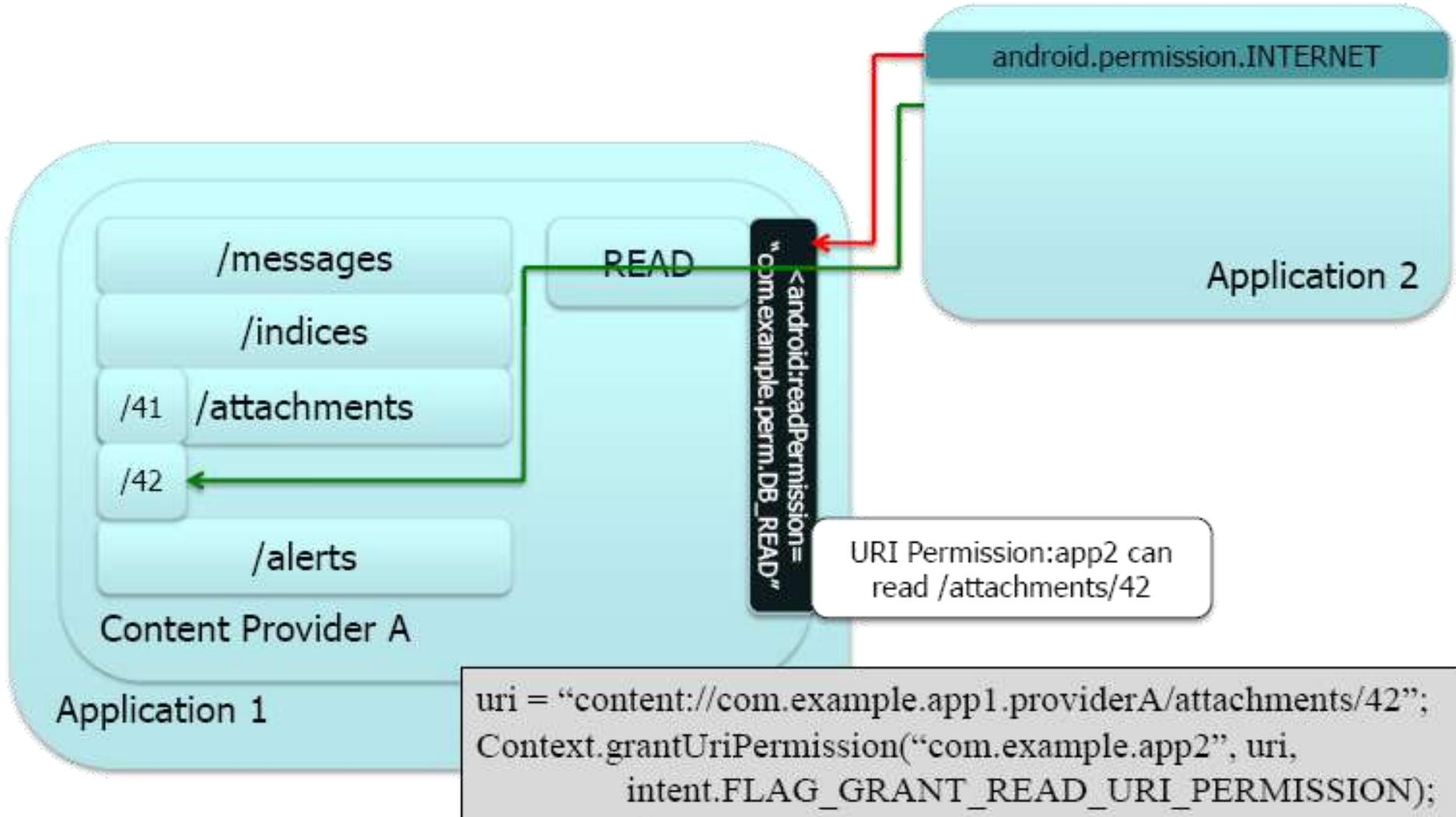
Permissions personnalisées

■ Les permissions : content provider



Permissions personnalisées

■ Les permissions : content provider



Permissions personnalisées

■ Les permissions : Broadcast Receiver

- Le BR envoie un message via le système et qui sera reçu par toutes les application qui veulent le recevoir. Les permissions permettent :

1. Exiger une permission pour recevoir le message (décider qui recevra le message)

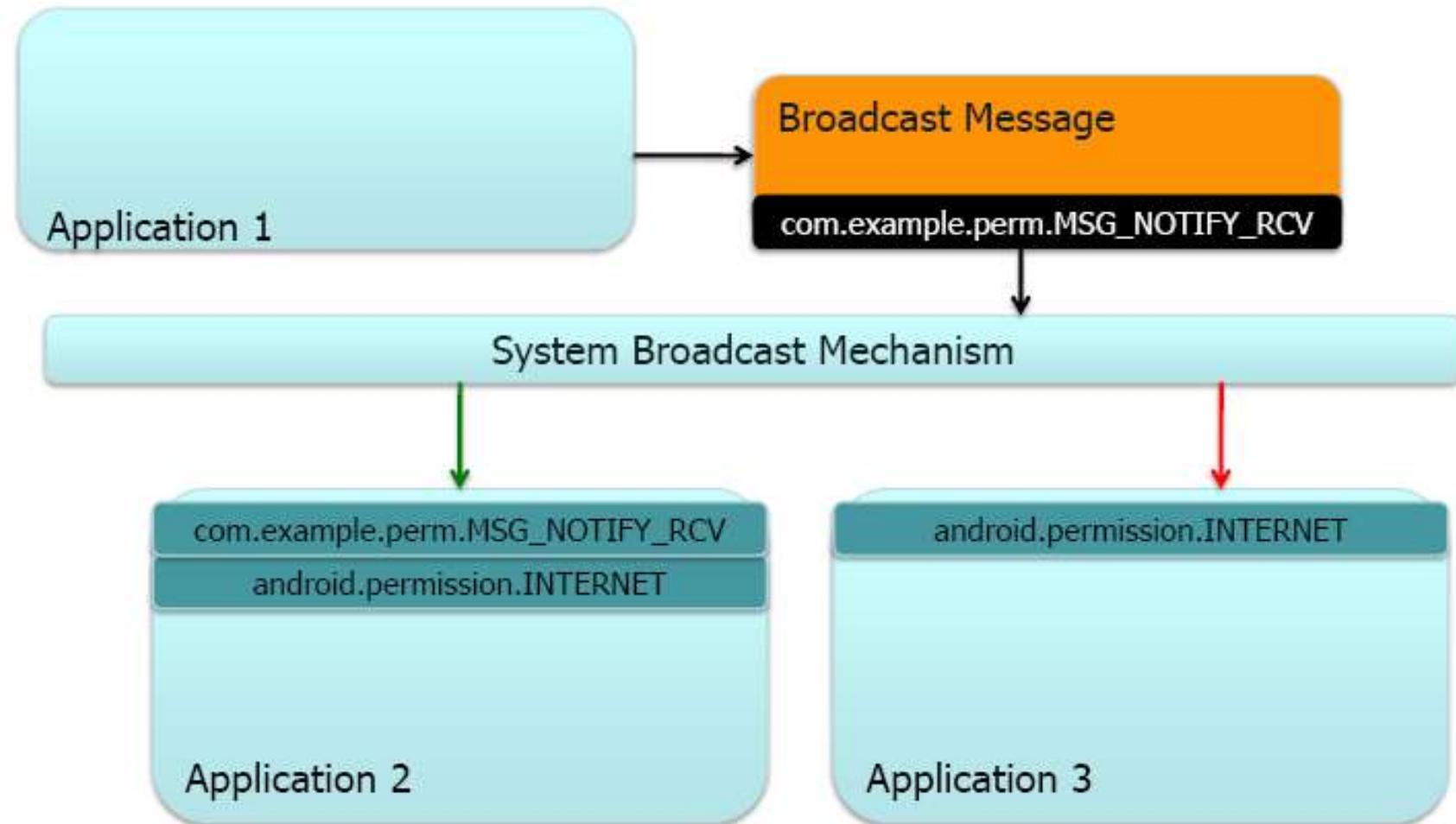
```
context.sendBroadcast(bcastIntent, "com.example.perm.MSG_NOTIFY_RCV");
```

2. Exiger une permission de la part des émetteurs de messages (une Application peut choisir qui peut lui envoyer un message)

```
context.registerReceiver(rcv, intentFilter,
    "com.example.perm.MSG_NOTIFY_SEND", null);
```

Permissions personnalisées

■ Les permissions : Broadcast Receiver



Permissions personnalisées

■ Les permissions : Broadcast Receiver

