



www.zeddini.com

Développement Mobile - Android



CH1- Introduction Générale

Walid ZEDDINI

Directeur IT – Expert Java EE / Mobile

« Enseignant Expert »
ISI - Tunis



SOMMAIRE

- Introduction
- Le marché Android et les enjeux - positionnement et dernières avancées :
La plateforme Android
- Anatomie de la plateforme Android et environnement de développement:
L'architecture Android
Les Outils de développement
- Travaux Pratiques suite à l'introduction générale:
Ateliers





SOMMAIRE

■ Introduction

- Le marché Android et les enjeux - positionnement et dernières avancées :
 ## La plateforme Android ##

- Anatomie de la plateforme Android et environnement de développement:
 ## L'architecture Android ##
 ## Les Outils de développement ##

- Question /Réponse avec la salle :
 ## Partage d'expériences ##





INTRODUCTION



3 milliards



1,5 milliard



ANDROID



ChromeBook
(Chrome OS)





INTRODUCTION



Qu'est-ce que c'est ?

- Souvent présenté comme l'alternative de **Google** à l'iPhone
- Système d'exploitation pour terminaux mobiles
- Basé sur Linux
- Open Source (licence Apache)



Google

- Acteur majeur d'internet
- 1er moteur de recherche
- 1er publicité en ligne
- Solutions d'entreprises: Google Apps, Google Enterprise Appliance...
- Services gratuits: gmail, apps, photos, vidéos...



Quelle est son histoire ?



INTRODUCTION

Historique

2005 : Rachat de la startup **Android Inc.**

- Développement d'applications mobiles :Richard Miner* + Andy Rubin

Objectif du rachat : créer une plateforme mobile

- Flexible
- Accessible à tous les intégrateurs et développeurs
- Profiter de la convergence web / mobile

Android : un système d'exploitation **open source** pour Smartphones, PDA, tablettes : systèmes légers

* Richard Miner a quitté Google fin 2008 et Andy Rubin en 2014



L'Alliance dans tout ça ?

Quelle Alliance ?





INTRODUCTION

Historique

Open Handset Alliance (OHA)

- 5 novembre **2007**, création de l'OHA (Open Handset Alliance)
- Un consortium créé à l'initiative de Google réunissant des entreprises, opérateurs mobiles, constructeurs et éditeurs logiciels
- 47 sociétés en 2008, plus de 84 aujourd'hui !

But de l'OHA

- Favoriser l'innovation sur les appareils mobiles en fournissant une plate-forme véritablement **ouverte** et **complète**
- Gratuit



des exemples d'entreprise membres ?





INTRODUCTION

Historique

Membres OHA



Ok, Très bien, et la Plateforme Monsieur ?





SOMMAIRE

- Introduction
- Le marché Android et les enjeux - positionnement et dernières avancées :
La plateforme Android
- Anatomie de la plateforme Android et environnement de développement:
L'architecture Android
Les Outils de développement
- Question /Réponse avec la salle :
Partage d'expériences





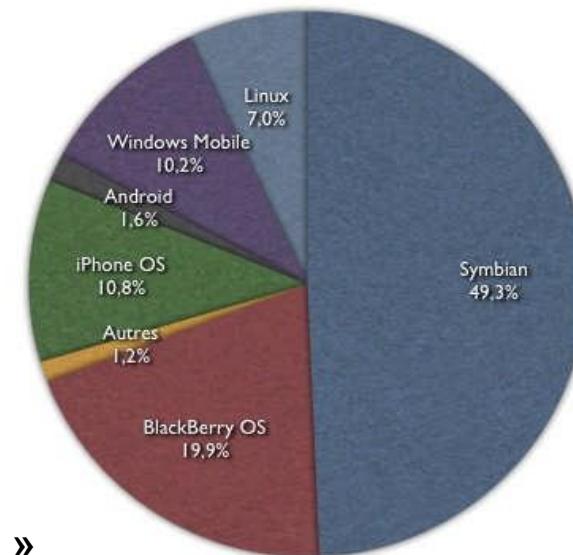
Android

Le marché, le matériel, le potentiel

Acteurs système d'exploitation mobile 2009, 2 ans après OHA

- Symbian
- Windows Mobile
- RIM (BlackBerry)
- Palm Source (Palm Os -PDA)
- Apple
- Linux

Part de marché des OS mobiles vendus au premier trimestre 2009



Baisse des parts de marché

- OS: Palm Source, Windows
- Décroissance du marché téléphone

Marché du smartphones en hausse

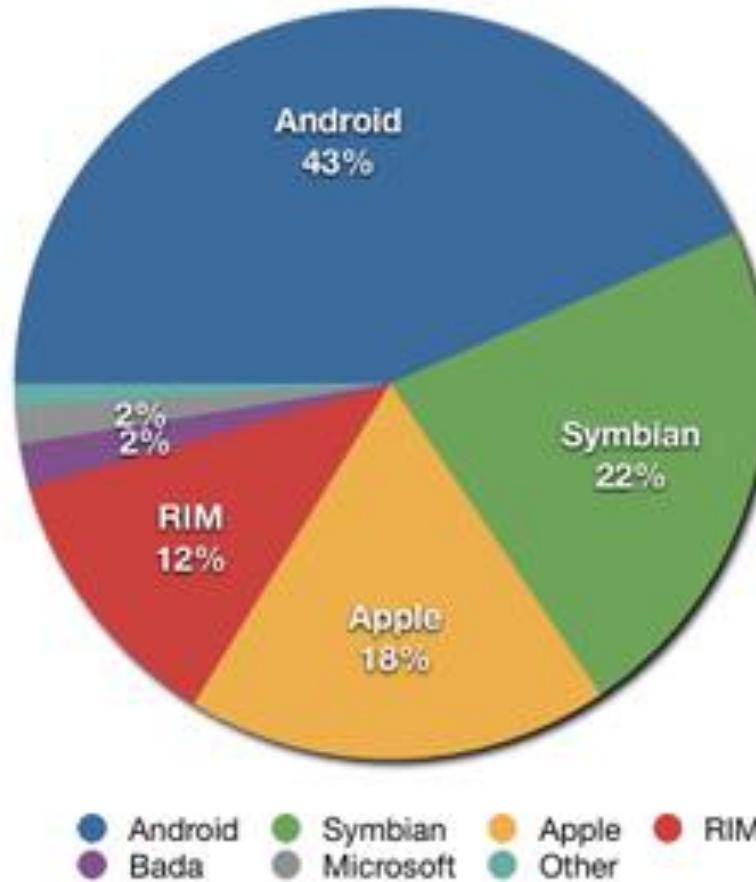
- Décroissance du marché « téléphone »
- Croissance smartphone: +37% en 2009 (GFK)
- Croissance smartphone: +12% en 2009 (Gartner)
- Services mobiles: nouveaux usages, multimédia





Android & ses Concurrents

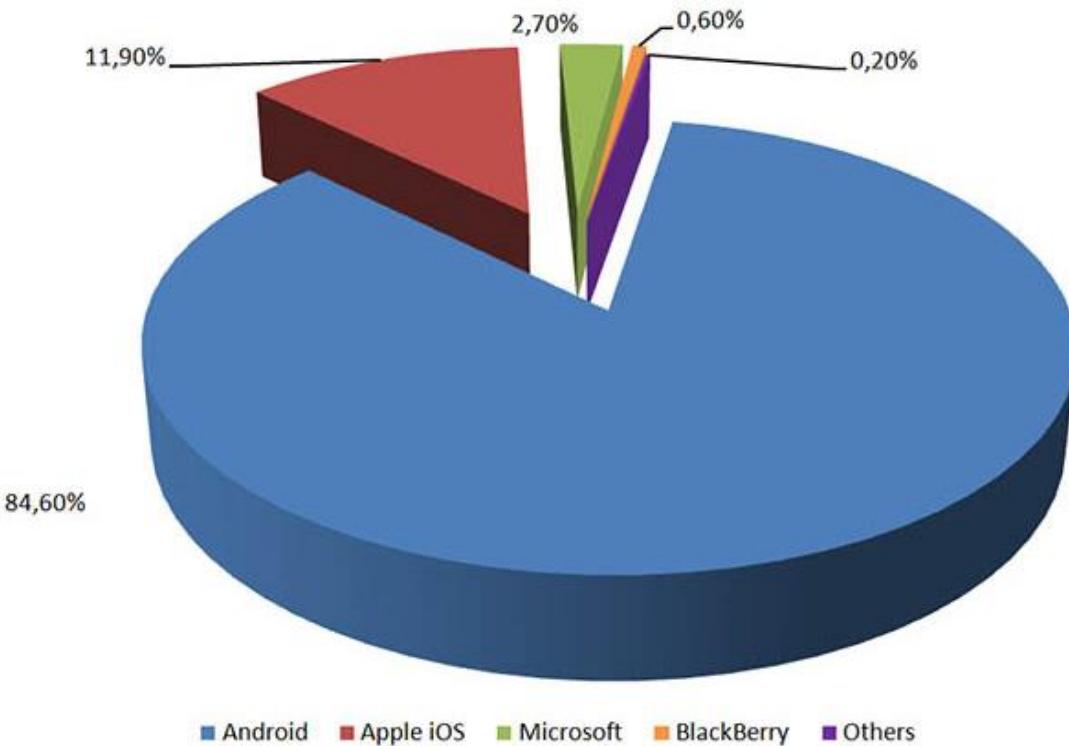
Acteurs système d'exploitation mobile 2011, 4 ans après OHA





Android & ses Concurrents

Acteurs système d'exploitation mobile 2012-2017



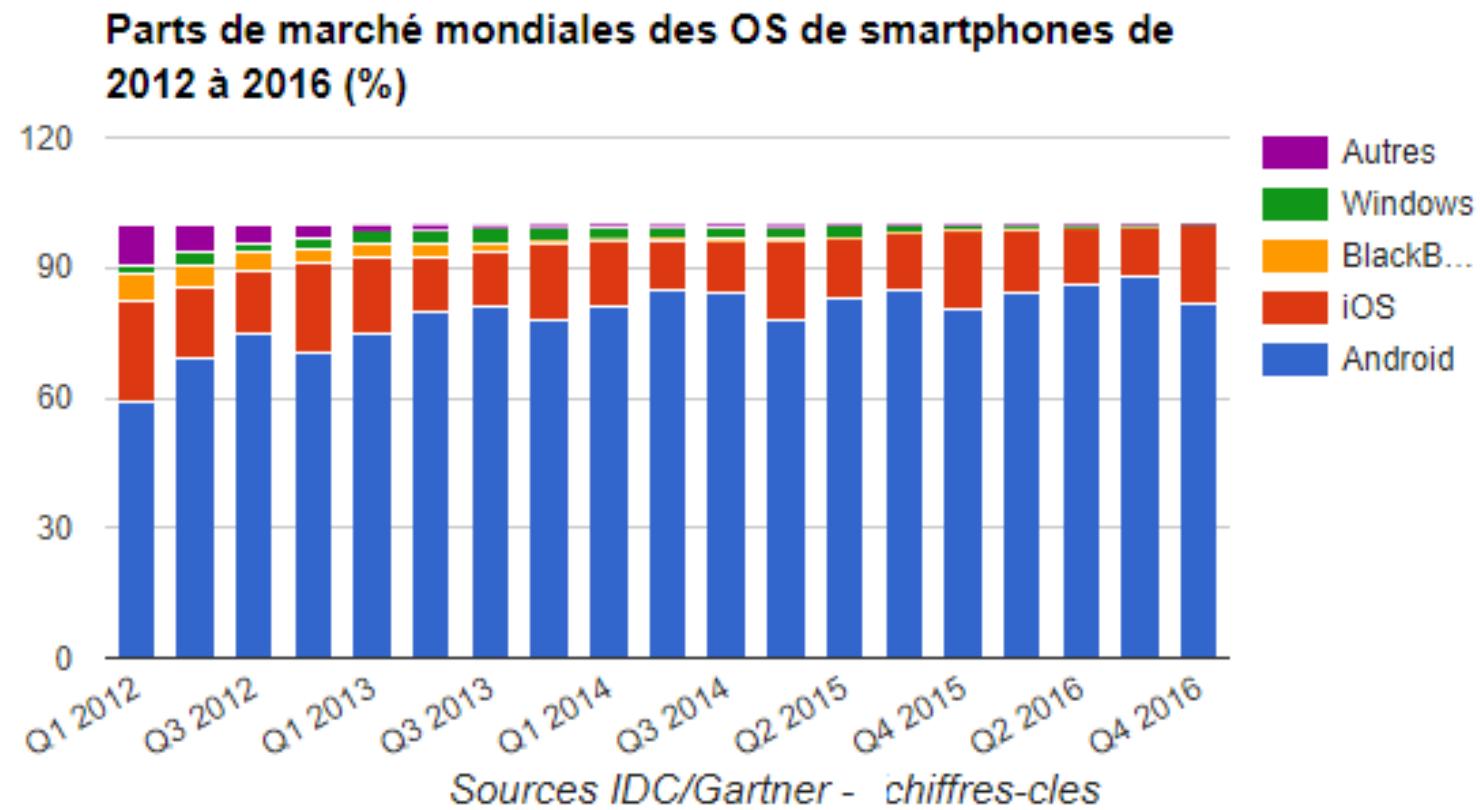
Source: Strategy Analytics, Android

295 millions mobiles vendus en
2014, 250 millions avec **Android**.





Android & ses Concurrents



Mais, comment Google a-t-il fait pour récolter tout cela ?

Près de 9 smartphones sur 10: Android !!!

Bonne Stratégie !



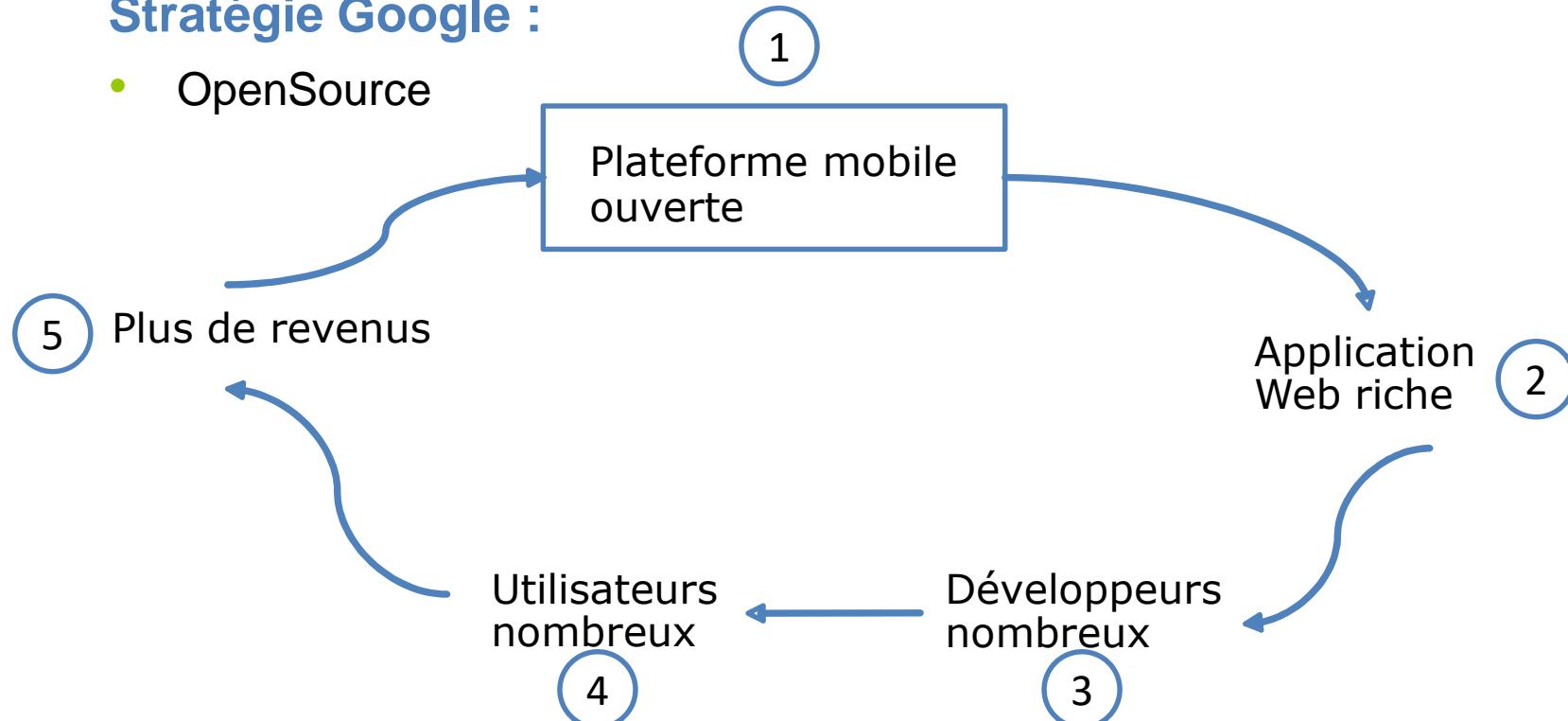


Android

Le marché, le matériel, le potentiel

Stratégie Google :

- OpenSource



- Android Market, puis Google Play / Google Checkout
- Publicité mobile

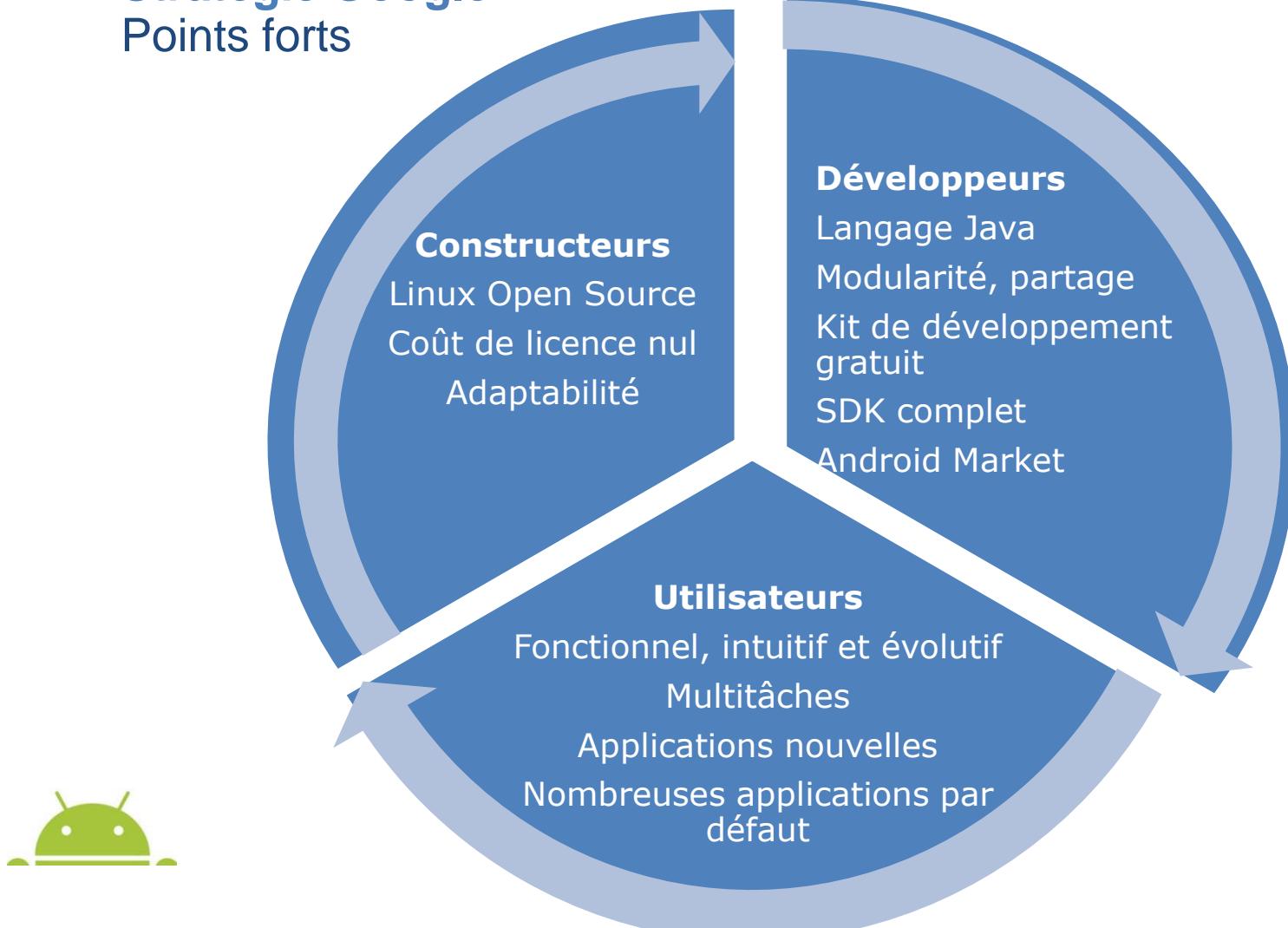




Android

Le marché, le matériel, le potentiel

Stratégie Google Points forts





Android

Le marché, le matériel, le potentiel

Android pour qui ?

- Constructeurs de matériels (téléphones, GPS, Netbook ,bornes internet...)
- Opérateurs
- Éditeurs de solutions logicielles, SSII...

Terminaux visés?

- Téléphones portables (HTC, Samsung, Motorola, LG, **Condor(Algérie), Evertek (Tunisie)**...)
- Netbook /Smartbook (HP Airlife 100, Acer Aspire D250...)
- Tablette Multimedia (Samsung Galaxy Tab, Archos, ...)
- Automobile (Audi, Volkswagen,Ford...)
- Mais aussi : GPS, Réfrigerateur, Machine à laver...





Android

Le marché, le matériel, le potentiel

Evolutions des SmartPhones



En 2008 : HTC Dream / G1



En 2009 : Une quinzaine (HTC, LG, Samsung, Motorola...)



20 ans
!!!



En 2010 .. 2017 : De très nombreux mobiles, Wear, TV, Watch, Car





Smartphone != Ordinateur

Android tire partie des particularités des smartphones :

- interface homme machine adapté (tactile, widget)
- divers modes : vibrer, sonnerie, silencieux, alarme
- notifications (d'applications, d'emails, de SMS, d'appels en instance)
- de boussole, accéléromètre, GPS
- divers capteurs (accélération linéaire, baromètre)
- NFC, RFID
- téléphonie (GSM) et réseau EDGE, 3G et 4G

En plus de ce qu'on peut avoir sur un **ordinateur** : navigateur, bibliothèques graphiques 2D, 3D (Open GL), base de données (SQLite), applications de rendu multimedia (audio, video, image) de divers formats, réseau Bluetooth et Wi-Fi, camera



Android

Evolution de l'IHM



- Android M3 – Nov 2007



- Android M5 – Mars 2008





Android

Evolution de l'IHM



- Android
Novembre 2008



- Android
Fevrier 2009



- Android
Juin 2010

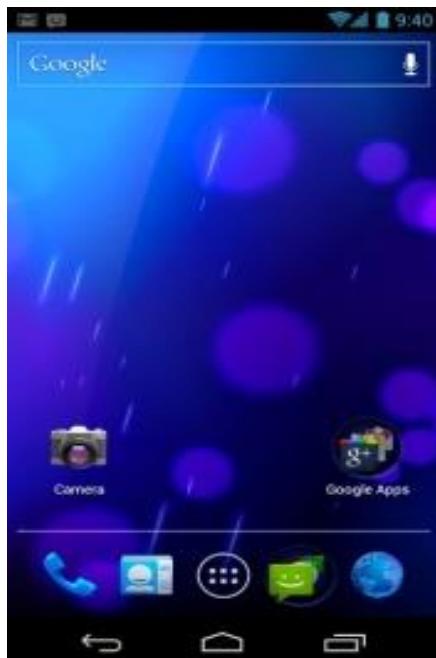




www.zeddingi.com



Android Evolution de l'IHM



Android
2013



Android
2014



Android
2015



Android
2017 – Pixel (Google)





Android

Conclusions (provisoire...)

Les facteurs clés de succès :

- Le nom Google et les services associés
- Une plateforme de développement banalisée
- L'Open Source, facteur de création d'un communauté productive et d'une adoption par la majorité des acteurs

→ Tous les docs , tutoriaux, compétences autour d'Android existent sur Internet en accès libre, une opportunité à saisir pour gagner de l'argent....



La Plateforme Monsieur ?

Plus tard, plus tard



SOMMAIRE

- Introduction
- Le marché Android et les enjeux - positionnement et dernières avancées :
 - ## La plateforme Android ##
- Anatomie de la plateforme Android et environnement de développement:
 - ## L'architecture Android ##
 - ## Les Outils de développement ##
- Question /Réponse avec la salle :
 - ## Partage d'expériences ##





Android et Google

- A suscité l'engouement des développeurs grâce à deux **Android Developer Challenge** en 2008 et 2009 financé par Google
- Conçu pour intégrer les applications Google : Gmail, Google Maps, Google Agenda, YouTube et la géolocalisation
- Les différentes versions ont des noms de **dessert** (qui suivent l'ordre alphabétique, de A à Z) qui sont sculptés et affichés devant le siège social de Google (Mountain View)



source : <http://fr.wikipedia.org/wiki/Android>



Android, les différentes versions

- 1.0 : sortie avant le premier téléphone Android (fin 2007)
- 1.1 : Version incluse dans le premier téléphone , le **HTC Dream**
- 1.5 : **Cupcake** (Petit Gâteau), sortie en avril 2009
- 1.6 : **Donut** (Beignet), sortie en septembre 2009
- 2.0 (2.0.1) : A causé de nombreux bugs, remplacée par la 2.1
- 2.1 : **Eclair** sortie en janvier 2010
- 2.2 : **FroYo** (Frozen Yogourt : Yaourt glace), sortie en mai 2010
- 2.3 : **Gingerbread** (Pain d'épice), sortie le 6 decembre 2010
- **3.0** : Honeycomb10 (Rayon de miel), pour Tablette , 26 janvier 2011, version unifiée :Smartphone, Tablette
- **4.0** : Ice Cream Sandwich (Sandwich à la crème glacée), version unifiée : Smartphone, Tablette et GoogleTV



Android, les différentes versions

- 4.1 : API 16 (Jelly Bean) sortie le 9 juillet 2012
- 4.2.2 : API 17 (Jelly Bean) sortie le 11 fevrier 2013
- 4.3 : API 18 (Jelly Bean) sortie le 24 juillet 2013
- 4.4 : API 19 (KitKat) sortie le 31 octobre 2013
- 5.0 : API 21 (Lollipop), ART (Dalvik) , Material Desig, sortie le 3 novembre 2014
- 5.1 -5.1.1: API 22 (Lollipop) sortie le 19 Février 2015
- 6.0 -6.0.1: API 23 (Marshmallow) sortie Fin mai 2015
- 7.0 -7.1.2: API 24 - API 25 (Nougat) sortie 21 Septembre 2016
- 8.0 : API 26 (Oreo) - 21 Aout 2017
- 8.1 : API 27 (Oreo) - 5 Décembre 2017 (Go Edition: variante d'Android, version allégée capable de fonctionner de manière rapide sur du matériel d'entrée de gamme)



Android, les différentes versions

- 9.0 : API 28 (Pie) – Rapidité, économie en énergie, Applications IA, 9 Aout 2018
- 10 : Android 10 - Septembre 2019
- 11 : Android 11 - Septembre 2020
- **12** : Android **12** - 18 février 2021 **(Vie privée)** :Les fonctions d'apprentissage automatique au niveau du système d'exploitation seront mises en bac à sable dans le « Android Private Compute Core », auquel il est expressément interdit d'accéder aux réseaux, Les applications qui demandent des données de localisation peuvent désormais être limitées à l'accès à des données de localisation « approximatives » plutôt que « précises ». Des **contrôles** visant à empêcher les applications d'utiliser l'appareil photo et le microphone ont été ajoutés aux boutons de réglage rapide. Un indicateur s'affichera également à l'écran s'ils sont actifs)

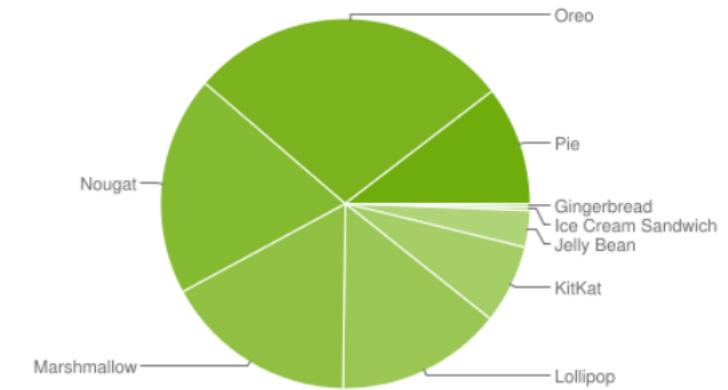
http://fr.wikipedia.org/wiki/Historique_des_versions_d'Android

<https://www.cnetfrance.fr/news/android-retour-sur-toutes-les-versions-de-l-os-google-39872199.htm>



Android, les différentes versions

Version	Codename	API	Distribution
2.3.3 - 2.3.7	Gingerbread	10	0.3%
4.0.3 - 4.0.4	Ice Cream Sandwich	15	0.3%
4.1.x	Jelly Bean	16	1.2%
4.2.x		17	1.5%
4.3		18	0.5%
4.4	KitKat	19	6.9%
5.0	Lollipop	21	3.0%
5.1		22	11.5%
6.0	Marshmallow	23	16.9%
7.0	Nougat	24	11.4%
7.1		25	7.8%
8.0	Oreo	26	12.9%
8.1		27	15.4%
9	Pie	28	10.4%



*Data collected during a 7-day period ending on May 7, 2019
Any versions with less than 0.1% distribution are not shown.*

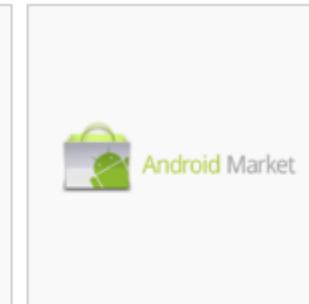


Google Play, l'ex Android Market

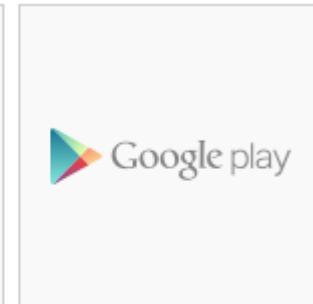
- Google Play, est une boutique en ligne créée par Google (le 6 mars 2012) par fusion des services Android Market et d'autres services Google (location de films, achat de musique, etc.). Elle permet de télécharger et d'installer de nouvelles applications ("apps") dans le smartphone.
- Android market est "né" le 22 octobre 2008



Premier logo d'Android Market



Logo d'Android Market utilisé à partir de 2011



Logo de Google Play de 2012 à 2015

- "Au 1 janvier 2013, Google Play est fort de 800 000 applications devant " iOS" & WinPhone voir historique à http://fr.wikipedia.org/wiki/Android_Market
- Les développeurs d'applications payantes reçoivent **70 %** du prix du logiciel, **30 %** allant à Google (redistribués aux développeurs via Google Checkout)
- Chaque nouveau développeur paie **\$25** comme frais de dossier (une seule fois)



Anatomie d'Android

Android: 2 parties

- Système d'exploitation (noyau Linux)
- Environnement d'exécution Dalvik (Java)
(Environnement d'exécution ART, Depuis 2015 avec Lollipop SDK 5)

Kit de développement Java disponible depuis 2007

- Concours Android Developer Challenge (Google) – Lancé 2008
- Août 2017 SDK 8.0

OS et SDK sont disponibles en Open Source

Développer une application ne requiert pas l'obtention de l'OS et réciproquement



Enfin, la Plateforme.

Ce n'est pas tout ! A green Android robot icon on the right side of the slide, holding a red apple in its hand.



Le SDK Android

■ l'Android **SDK** (Software Development Kit) amène des outils :

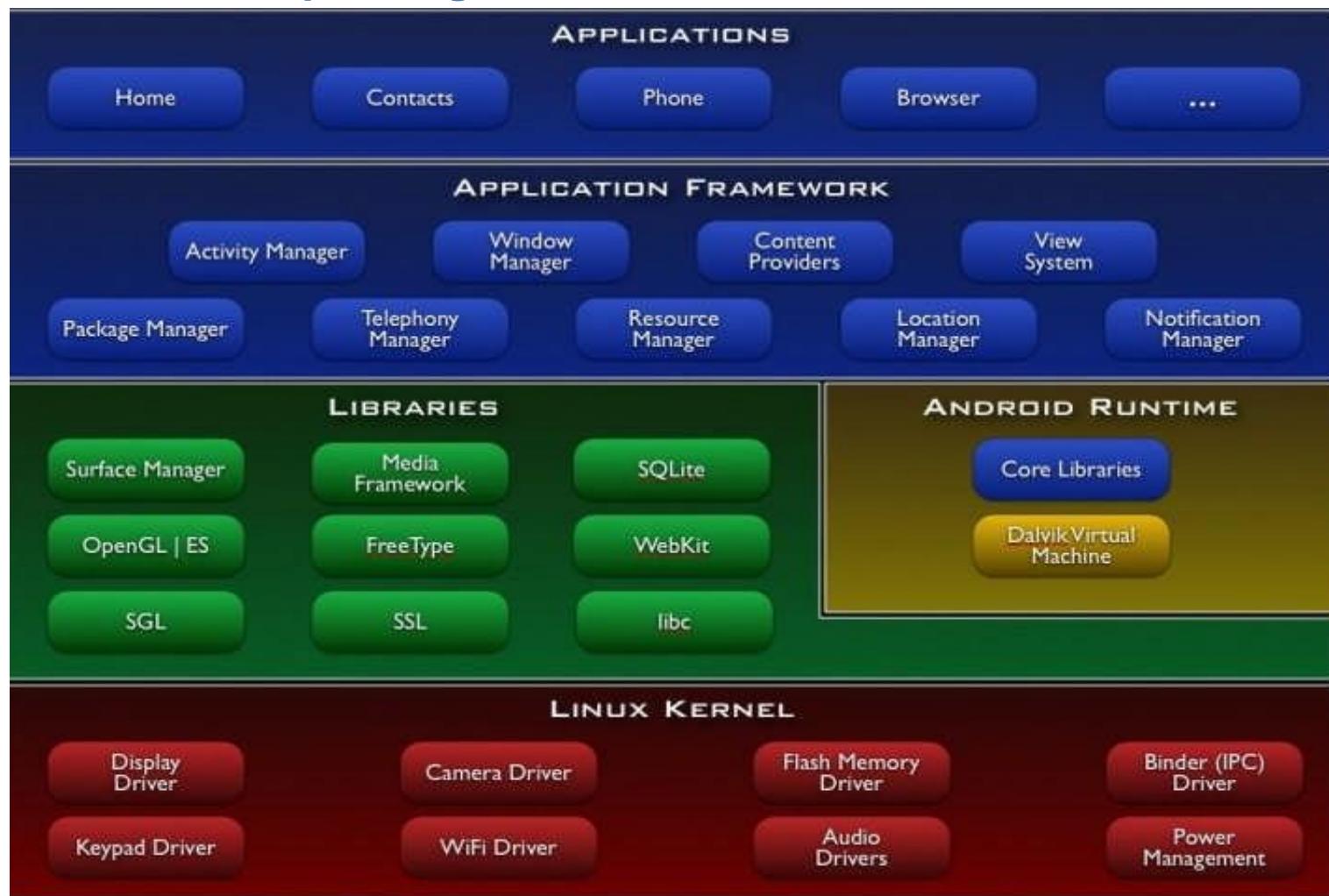
- un environnement de **développement**
- une machine virtuelle Java adaptée : la **Dalvik** virtual machine (ART 2014)
- un environnement debugueur **DDMS** (Dalvik Debug Monitor Service) utilisant **adb** (Android Debug Bridge)
- un environnement de construction d'application Android **aapt** (Android Asset Packaging Tool)
- des émulateurs de téléphones ou de tablettes **AVD** (Android Virtual Device)

■ et une énorme API (voir <http://developer.android.com/reference/packages.html>)



Anatomie d'Android

Architecture en « pile Logicielle »





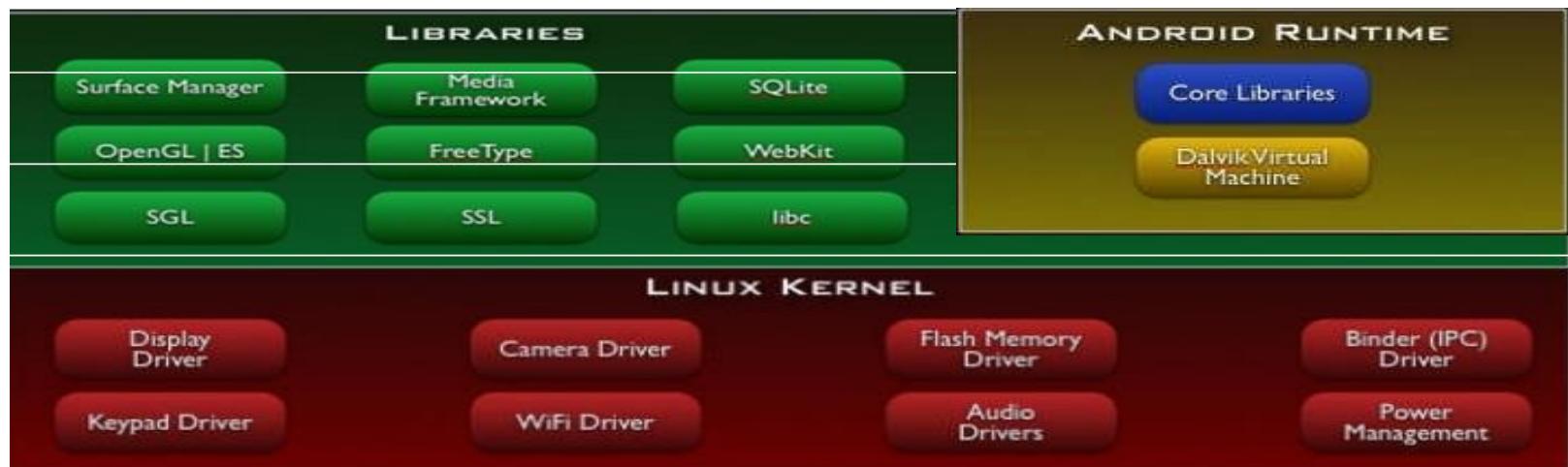
OS Android: Les 2 couches inférieures:

Linux

- Linux kernel 2.6.24 ARM
- Pas de système natif de fenêtrage
- Pas de support Glibc
- Optimisation mémoire, processus et alimentation
- Gestion utilisateurs

Dalvik

- VM Android
- Optimisée embarqué
- Multi-instance
- Optimisation mémoire, sécurité
- Optimisation bytecode
- Fichier .dex
- JIT





VM Android

de Dalvik à Android RunTime (2014 SDK 5.0)



- Etait la machine virtuelle Java pour les applications Android
- Conçu pour exécuter du code Java pour des systèmes ayant des contraintes de **place mémoire** et rapidité d'exécution
- Exécute du code **.dex** (Dalvik executable) = des **.class** adaptés à l'environnement Android
- Ecrit par **Dan Bornstein** d'où le nom est le village islandais de ses ancêtres
- Le code de la DVM est **open source**
- **JIT** : compilateur Just-In-Time

Android RunTime

- Environnement d'exécution plus **performant**
- **AOT** : compilateur Ahead-Of-Time
- Gains en performance et en autonomie des batteries (+20 à 30 %)

Dalvik vs. ART Battery Benchmark Results (Android 4.4.2) : Test 3b - Animation Rundown with updated images						
	Dalvik			ART		
	Time (hours)	Final Level	Fall Rate (%/hour)	Time (hours)	Final Level	Fall Rate (%/hour)
Nexus 4	3	63%	12.33%	3	64%	12.00%
Nexus 5	3	63%	12.33%	3	67%	11.00%

ART wins on both devices.



Comment plus performant ?



VM Android



- ART est un moteur d'exécution **multi-plate-forme** qui prend en charge les architectures **x86, ARM, MIPS** et les environnements **32 bits et 64 bits**.
- Contrairement à Dalvik, qui utilise la compilation juste à temps (**JIT**), ART compile les applications lors de l'installation (compilation anticipée) ou AOT (Ahead of Time), elles sont ensuite exécutées exclusivement à partir de la version compilée.
- Cette technique élimine la charge de traitement associée à la compilation JIT, améliorant les **performances du système** ainsi que la **consommation d'énergie**
- La machine virtuelle **Dalvik** a été officiellement remplacée par **ART** qui est maintenant activé par défaut.



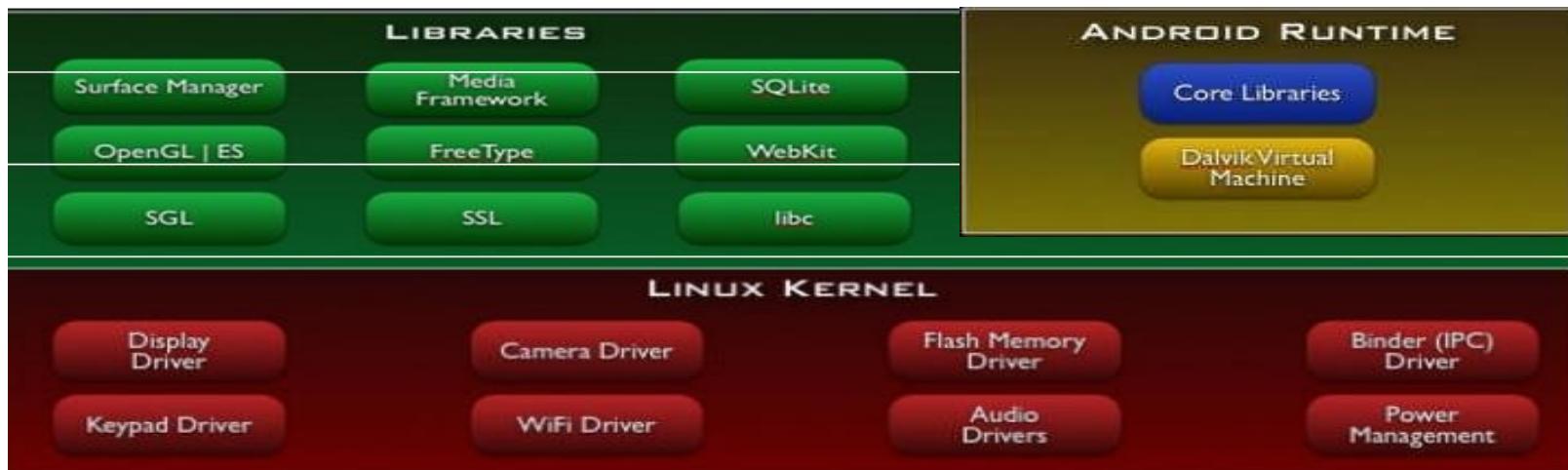
Anatomie d'Android

La couche "Libraries" (bibliothèques) = couche logicielle basse pour utiliser:

- les formats multimedia : images, audio et video enregistrement comme rendu
- les dessins 2D et 3D, bitmap et vectoriel
- une base de données SQLite (SQLite)

■ L'environnement d'exécution (Android Runtime). Toute application est exécutée dans son propre processus, dans sa propre Dalvik virtual machine ou ART

■ Le noyau Linux sur lequel la Dalvik virtual machine s'appuie pour gérer le multithreading, la mémoire. Le nouveau Linux apporte les services de sécurité, la gestion des processus, etc.





Anatomie d'Android

La couche "Applications" : Android est utilisé dans un ensemble contenant déjà des applications natives comme, un client de mail, des programmes pour envoyer des SMS, d'agenda, de navigateur web, de contacts personnels



Base de l'API

- Point d'entrée pour les applications
- Accès à toutes les ressources inférieures via librairie
- Accès possible aux ressources C via JNI (bypass de cette couche)





Anatomie d'Android

La couche "Application Framework" : permet au programmeur de construire de nouvelles applications. Cette couche fournit la gestion :

- des Views (= IHM)
- des ContentProviders = l'accessibilité aux données des autres applications (ex : les contacts) et donc les partages de données
- des ressources = les fichiers non codes comme les images, les écrans (Resource Manager)
- des Notifications (affichage d'alerte dans la barre de titre)
- des Activities = l'enchaînement des écrans





Anatomie d'Android

Développement

Environnement

- Windows XP / Vista / 7 / 8.1/ 10
- Mac OS
- Linux

JDK (J2SE)

- 1.5
- 1.6
- 1.7 / 1.8 / 1.9 / 1.10 / 1.11

SDK Android

- SDK Min : 2.3 / 3.0
- SDK Target : 4.2 / 4.4/ 5.1/ 6.0 /7.0 /8.0 /8.1 / 9.0

IDE :

- Eclipse + plugin ADT
- Android Studio (2015)

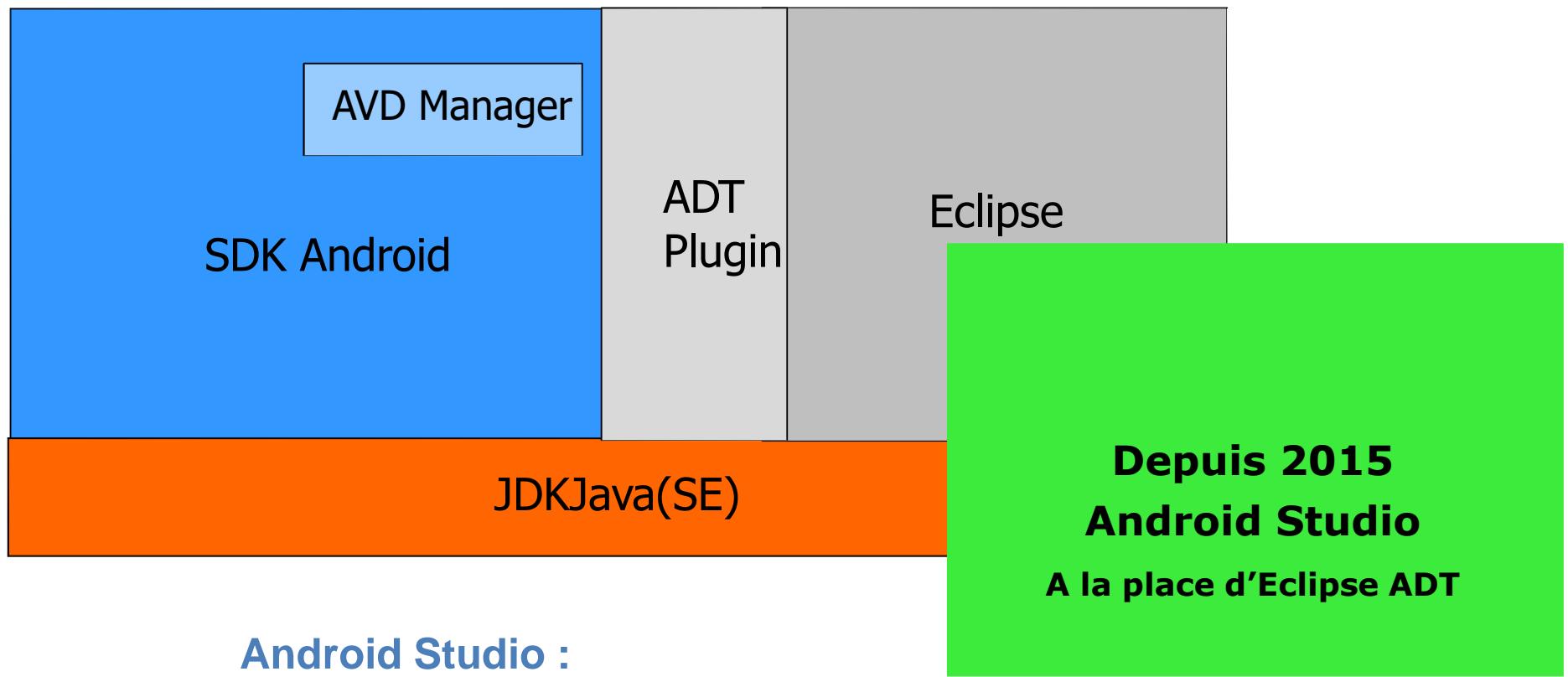


Optionnel: Outil dev C: Intégration code C/C++ en tant que librairie (JNI)





La pile des outils de développement pour Android





Fin de l'installation l'environnement

- Android Studio - Installé
 - SDK - Installée
 - JDK - Installée
-
- Et c'est tout : on peut commencer à faire du développement pour Android
-
- Le site officiel d'Android : <http://developer.android.com/>
Il y'en a tout tout tout !
 - Pour installer l'environnement de développement Android:
<http://developer.android.com/sdk>



Résumé de la première partie

- Le terme Android dénote à la fois,
 - une société initiatrice pour le développement d'applications pour smartphones rachetée par Google,
 - un environnement de **développement**,
 - un environnement **d'exécution**.
- L'environnement d'execution est une pile logicielle avec, à la base un système d'exploitation **Linux**
- Pour développer des applications Android, utiliser l'outils de développement **Android Studio**



Anatomie d'Android

Concept de base

- Le code est écrit en Java (eh oui)
- L'Android SDK compile l'ensemble du développement (code Java, données, fichier de ressources, fichier XML) dans un paquetage Android : un .apk
- Toute une application tient dans un .apk et un .apk = une application
- Les composants fondamentaux d'une application Android sont : Activity, Service, ContentProvider, BroadcastReceiver
- Certains de ces composants communiquent entre eux à l'aide d'Intent



Anatomie d'Android

Développement

Une application est une succession d'écrans

Elle inclut un ensemble de descripteurs pour chaque écran

Un écran peut ouvrir un autre écran d'une même application ou d'une autre application

5 composantes majeures

- Intent
- Activity
- Broadcast Receiver
- Content Provider
- Service

L'ensemble de ces composantes est décrit dans le fichier `AndroidManifest.xml`

Les applications ont pour extension APK (Android Package)





Anatomie d'Android

Développement

Activity

- Une classe par Activity
- Généralement le point d'entrée d'une application
- Généralement une interface graphique, un écran
- Cycle de vie

Intent

- Type d'action possible déclaré auprès du système
- Indique le besoin ou le service offert

Content Provider

- Accès aux données partagées
- Données internes ou externes à l'application
- Encapsulation du schéma de la base
- Pas de requête
 - URI
 - Services de manipulation des données fournis par le package





Anatomie d'Android

Développement

Services

- Composants sans interface graphique
- Activité de longue durée
- Démarrer par context.startService: interne à l'application
- Démarrer par context.bindService: externe à l'application

Broadcast Receiver

- Réagit à des évènements externes
 - Logiciel: Alarme, notifications...
 - Matériel: Activation puce GPS, exting
- Réveiller une application, afficher un message, lancer une activity

Classe R.java

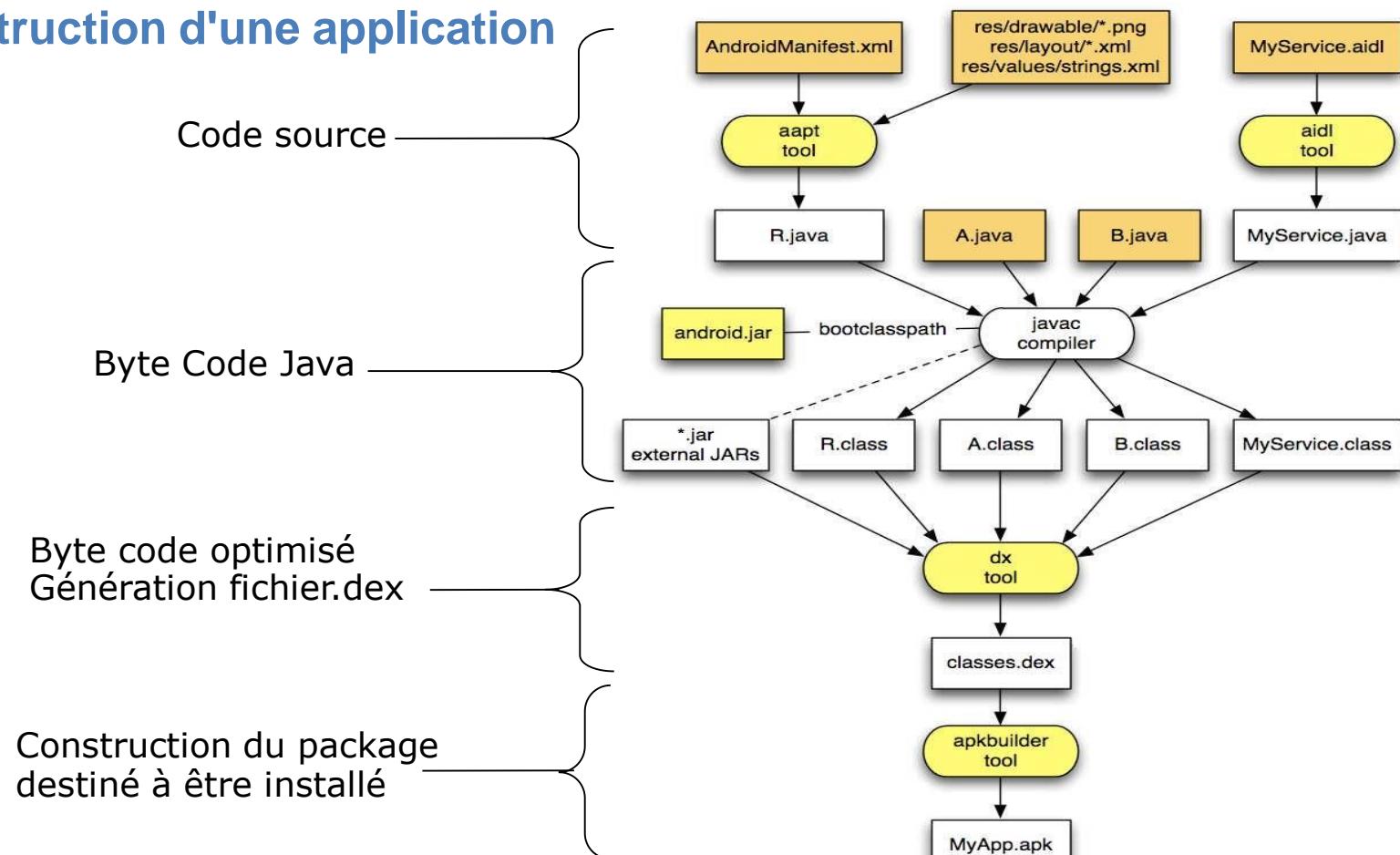




Anatomie d'Android

Développement

Construction d'une application





Anatomie d'Android

Hello Android

Création d'un nouveau projet

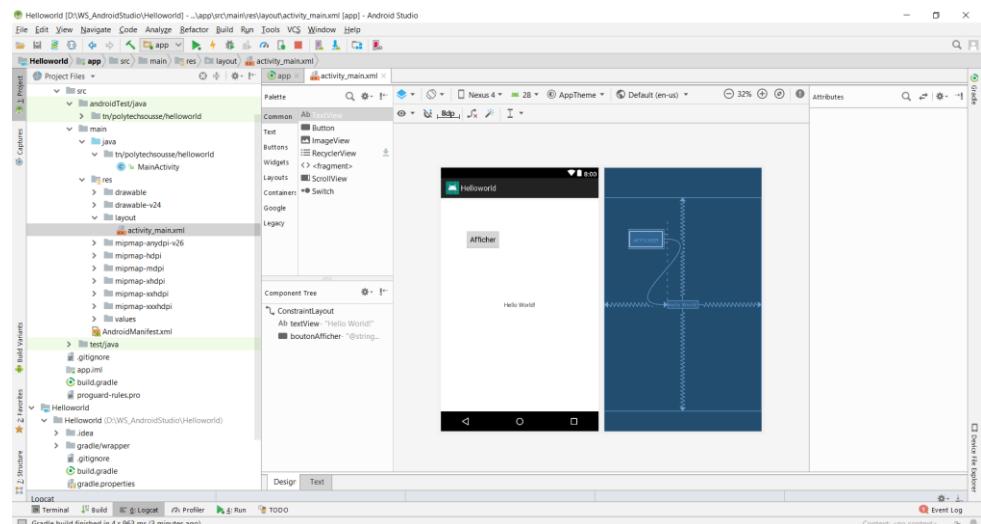


Ecoutez, passez aux Ateliers, il y a de tout !

Ce PC > Data (D:) > COURS_ISI > ANDROID > ATELIERS_TP

Nom

- A0_Android_Installation
- A1 - Initiation Android
- A2 - Compsant Android
- A3 - 1 Android - Boite de Dialogue
- A3 - 2 Android - Menus
- A3 - Compsant Android-1
- A4 - Android - Intent
- A5 - 1 - Services
- A5 - 2 - BroadcastReceiver
- A5 - 3 - StartedServices
- A5 - 4 - LocalBoundServices
- A6 - 1 - Persistance.DAO
- A6 - 2 - Persistance_ContentProvider
- A6 - 3 - Persistance_GeneratorDAO
- A7 - 1 - RESTWebServices



Juste Hello Word SVP !



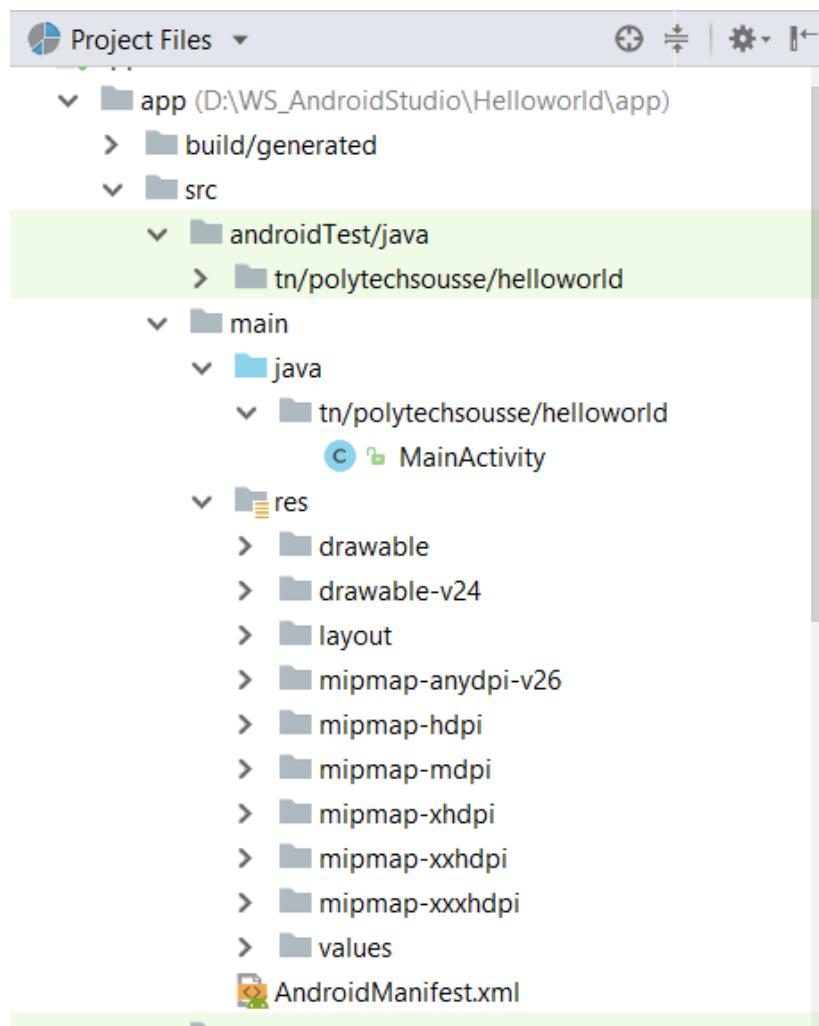


www.zeddi.com

Anatomie d'Android

Hello Android

Arborescence Project





Les Activity et leurs interactions

Activity

Une classe étendant de Activity

Point d'entrée d'une application

Généralement un écran d'une application (plein écran ou non)

Comportement défini dans le fichier **AndroidManifest.xml**

Une Activity peut

- Être sans interface
- Être une fenêtre flottante
- Retourner des valeurs
- Lancer d'autres Activity

Possède un cycle de vie

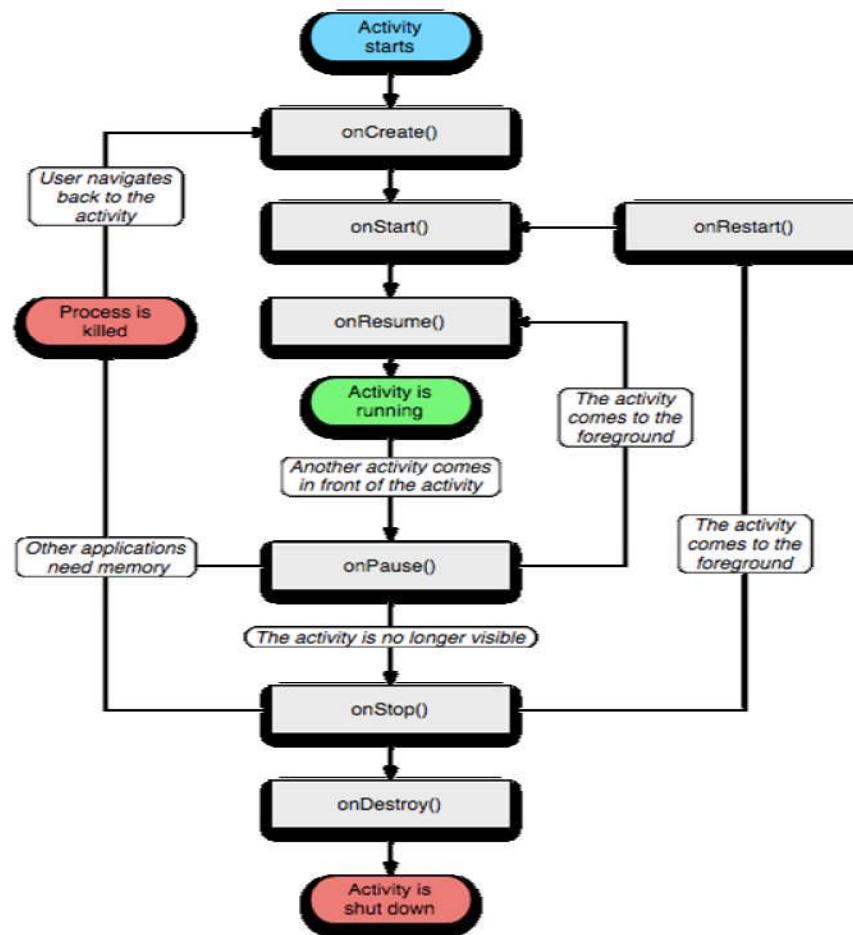




Les Activity et leurs interactions

Activity

Cycle de vie d'une Activity





Les Activity et leurs interactions

Activity

Différentes Activity disponibles - android.app.*

- ActivityGroup
- ListActivity
- AliasActivity
- ExpandableListActivity
- PreferenceActivity
- LauncherActivity
- TabActivity

Une Activity est capable de lancer une autre Activity – que nous nommerons SubActivity

- Interne à l'application
- Externe à application
- Récupérer le résultat d'une SubActivity
- Notion importante d'Intent





Les Activity et leurs interactions

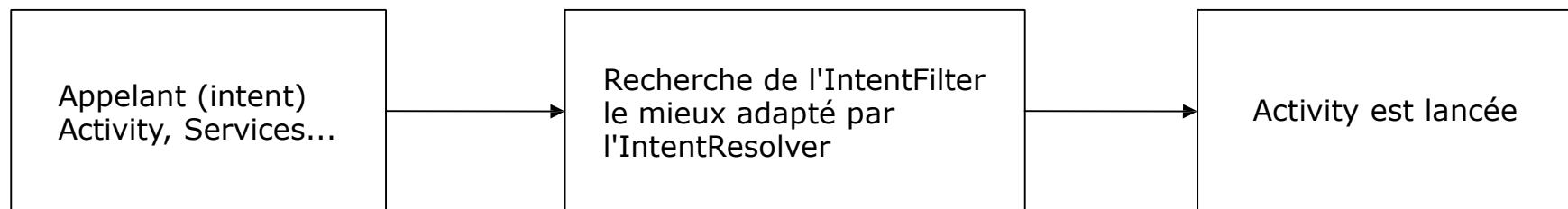
Intent

Intent

- Définir une action à effectuer
- Créer des liens entre application
- Décrit par une action (VIEW, EDIT, MAIN,...)
- URI pour renseigner les données à traiter

IntentFilter

- Décrit les capacités d'une application
- Attaché à une Activity
- Écoute les demandes d'Intent (handler)
- Enregistré au niveau système





Les Activity et leurs interactions

Activity

Lancer une autre Activity

- Définition du type d'Intent
- Définition du chemin de l'Activity
- startActivityForResult

```
startActivity(new Intent(this, ScoreActivity.class));
```

- StartActivityForResult
- Lancer l'activity
- Récupérer le résultat





Les Activity et leurs interactions

Descripteur d'application

1 fichier de description de l'application: **AndroidManifest.xml**

Descripteur de l'application

- Renseignements sur l'application (nom, version de code, SDK supporté...)
- Définit le comportement des Activity et leurs actions
- Déclare auprès du système les Intent accessibles
- Définit les services, broascat receiver et content provider

Déclaration des ressources nécessaires (applications, Internet, matériels, données...)

- L'utilisateur est toujours prévenu à l'installation des ressources requises pour le bon fonctionnement de l'application. Il peut accepter ou refuser.





Interface graphique (IHM)

IHM : Présentation

Différents de J2ME, AWT ou Swing

Deux méthodes de création

- XML (recommandé)
- Code Java

Avantages XML

- Lisibilité du code Java
- Lisibilité de la construction de l'IHM
- Allègement du code Java
- Rapidité de développement (Similaire à un page HTML)
- Moins de risque d'erreur

Large éventail de composants déjà disponibles

Personnalisation possible de chaque élément

Système de « thèmes » disponible





Interface graphique (IHM)

IHM : les layouts, les vues

Chaque composant correspond à un Layout

- Présent dans /res/layout
- Représente un écran
- Représente un composant spécifique d'un écran

Chaque composant

- Dispose d'un identifiant unique (de préférence)
- Peut être accédé, ajouté, modifié et supprimé depuis le code Java
- android:id=@+id/monidentifiant pour votre composant
- android:id=@android:id/empty pour référencer un composant spécifique d'Android

Identification par le fichier R.java (généré par aapt)

Compiler sous forme de vue embarquée dans le fichier dex





Interface graphique (IHM)

IHM : les layouts, les vues

Ecrire son fichier XML

- Contient obligatoire un élément racine (Root)
- Peut contenir autant de layout ou vue que désiré
- Sauvegardé sous /res/layout/monlayout.xml
- Ne pas contenir de majuscule ou de chiffre

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





Interface graphique (IHM)

IHM : les layouts, les vues

Charger le layout désiré

- Référencés sur le nom de fichier (sans extension) dans **R.java**
- **R.layout.monlayout**
- **setContentView(layout)**

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

Un layout peut être chargé au sein d'un autre composant en utilisant l'objet **LayoutInflater**

Chaque composant dispose d'un certain nombre de paramètres

- android:layout_width et android:layout_height (fill_parent, wrap_content...)
- android:id, android:text, android:background...

Chaque composant dispose de ses arguments propres:

- android:orientation pour LinearLayout (vertical, horizontal)





Interface graphique (IHM)

IHM : string et multilinguisme

Les chaines de caractères et l'IHM

- Fichier XML
- /res/values
- Par défaut: **string.xml**
- <string name="cle">Valeur à afficher</string>

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string name="hello">Hello World, HelloActivity!</string>
    <string name="app_name">Mon Hello Android</string>
</resources>
```

Multilingues

- Basée sur la locale du système
- /res/values-fr, /res/values-en contenant un fichier string.xml
- Utiliser les mêmes clés pour afficher la valeur dans la langue correspondante





Interface graphique (IHM)

IHM : Les menus



Sous menu



Menu contextuel





Interface graphique (IHM)

IHM : évènements

Gérés comme en Java / AWT / Swing / J2ME

Mise en place de Listener (écouteurs)

Peuvent être appliqués sur chaque composant View

Évènements supplémentaires en fonction du type de la vue

Processus

- Récupérer l'objet vue concerné
- Ajout du listener
- Implémentation des actions à réaliser par le listener

```
Button monBouton = (Button)findViewById(R.id.monBouton);
b.setOnClickListener(evtClickSurMonBouton);
...
...
OnClickListener evtClickSurMonBouton = new OnClickListener(){
    public void onClick(View v){
        // Implémentation
    }
};
```





Interface graphique (IHM)

IHM : Boîte de dialogue

Une vue affichée sur la couche supérieure du contenu courant

Composition

- Titre, Texte
- Icône
- Boutons
- Animation, personnalisable

Bloquante

AlertDialog.Builder

Personnalisable à volonté



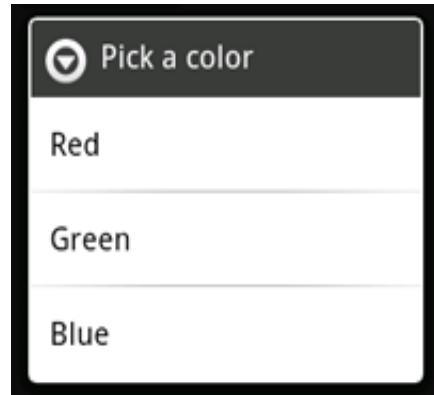


www.zedding.com

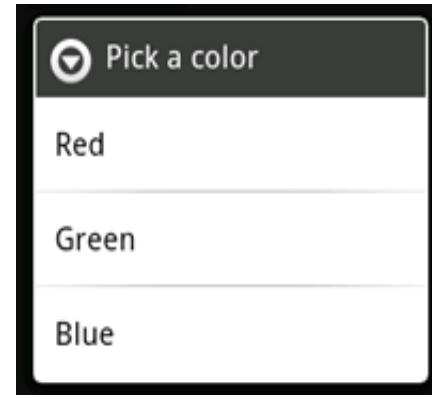
Interface graphique (IHM)

IHM : Boîte de dialogue

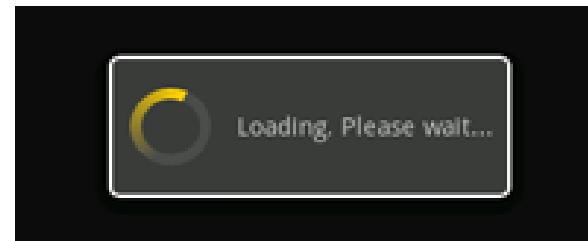
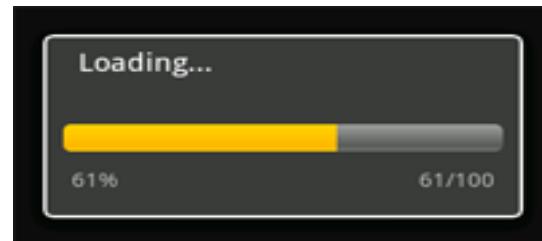
Des listes dans les boîtes de dialogues



Des listes et boutons radios



Barre de progression (fin connue ou indéterminée)





Données persistantes

3 types de données persistantes:

- Préférences utilisateurs
- Base de données (**Sqlite3**)
- Content Provider

Gestion de vos données dans des fichiers





Données persistantes

Les préférences utilisateurs

Composants fournis

- IHM
- Données

Simple

Ensemble de clés (String)/valeurs(primitive)

Propre à l'Activity ou à l'application entière

Déclarations dans AndroidManifest

Données sauvegardées dans

/data/data/monpackage/shared_prefs/monpackage.xml





Données persistantes

Bases de données

SQLITE

- Projet Open Source
- Base de données transactionnelles
- Pas de partie serveur
- Toutes les tables sont contenues dans un seul fichier
- Crossplatform
- Données typées
- Langage SQL
- Clé primaire
- Non disponible: FOREIGN_KEY, RIGHT/LEFT_OUTER_JOIN, certaines options ALTER_TABLE

SQLITE3

- Librairie de manipulation
- Accessible par adb shell

5 Types de données supportées: NULL, INTEGER, REAL, TEXT, BLOB





Données persistantes

Bases de données

SQLITE & Android

- Aucune base par défaut: tout doit être construit
- Manipulation SQLiteDatabase
- SQLiteOpenHelper
 - onCreate
 - onUpdate

Le fichier est sauvegardé par défaut dans
`/data/data/monpackage/databases`

- L'enregistrement de fichier de base de données sur une carte mémoire ne pourra être réalisé par SQLiteOpenHelper, recours à l'objet SQLiteDatabase





Données persistantes

Bases de données

Manipulation des données

- execSQL: exécution de commande SQL passée en paramètre sous forme de chaîne de caractères (dans la classe de l'adapter)

```
db.execSQL("insert into " + DATABASE_TABLE_USER  
+ "(" + COL_TAB_HELLO_USER_NOM  
+ ") values('" + name + "');");
```

- Insert(), delete() et update() de SQLiteOpenHelper
 - Nom de la table concernée
 - ContentValues().put(nom de colonne, valeur)

```
ContentValues cv = new ContentValues();  
cv.put(COL_TAB_HELLO_USER_NOM, name);  
db.insert(DATABASE_TABLE_USER, COL_TAB_HELLO_USER_ID, cv);
```

- Permet de retourner des valeurs (ex.: identifiant d'un nouveau élément ajouté)





Données persistantes

Bases de données

Query

```
public Cursor query (String table,// nom de la table  
String[] Columns, // nom des colonnes a retourner ou null pour toute  
String Selection, // clause where, sans le mot 'where' ex.: nom=?  
String[] SelectionArgs, // tableau ordonné des valeurs  
// utilisées dans la clause where  
String GroupBy, // argument groupBy, ex: nom, prenom – null sinon  
String Having, // argument clause having – null sinon  
String OrderBy) // argument d'ordonnancement ex: nom - null sinon
```

```
public void getInfo(String nomRecherche){  
    String[] columns=new String[]{"nom", "prenom"};  
    String[] params={nomRecherche};  
    Cursor result=db.query(this.DATABASE_TABLE_USER,  
                           columns, "nom=?",params,  
                           null, null, null);  
    ...  
}
```





Données persistantes

Content provider

Toutes les URI commençant par URI

Encapsulation de la structure des données

Base de données, fichiers plats, accès distant

A partir d'une URI

- Create
- Read
- Update
- Delete

Utilisation de ContentProvider existant – d'autres applications

Créer, utiliser et partager vos données

Définition d'une URI

PREFIXE://IDENTIFIANT DU TYPE DE DONNEES/DEFINITION DE LA DONNEES/ENREGISTREMENT

content://contact/people/123

content://com.mycompany.hello/user/53





Données persistantes

Content provider

Réaliser une requête

- managedQuery() depuis un objet Activity
 - URI
 - Un tableau des propriétés du ContentProvider à obtenir dans le résultat (nommé projection)
 - Les contraintes (clause Where)
 - Ensemble des paramètres permettant de compléter les contraintes (? dans la clause where)
 - Clause d'ordonnancement
- Retourne un Cursor

Propriété d'un ContentProvider = Colonnes d'une base de données

```
private static final String[] PROJECTION = new String[] {  
    Provider.Constants._ID, Provider.Constants.TITLE,  
    Provider.Constants.VALUE};  
....  
constantsCursor=managedQuery(Provider.Constants.CONTENT_URI,  
PROJECTION, null, null, null)
```





Données persistantes

Content provider

Insérer des enregistrements

- `insert()`
 - Uri + contentValues
 - Retourne l'identifiant de l'élément inséré
- `bulkInsert()`
 - Tableau d'Uri et tableau de ContentValues pour ajouter plusieurs enregistrements en une seule fois
 - Retourne le nombre d'éléments insérés

Suppression d'un enregistrement

- `Delete`
 - Uri, clause where, arguments complétant la clause where
 - Les informations dépendantes à la table visée seront également supprimées (cascading)
 - Retourne le nombre d'éléments supprimés

Mise à jour

- `Update`
 - Uri, valeurs à insérer, clause where, arguments complétant la clause where
 - Retourne le nombre d'éléments mis à jour





Données persistantes

IHM, Cursor et Adapter

Composant dédié à l'affichage de liste

- Gestion du Cursor ou de listes
- Mise à jour automatique lors d'un ajout, modification, suppression
- Composants graphiques déjà disponibles
- Personnalisation de l'affichage possible
- Adapter: mapping entre la vue et les données

Processus

- Utilisation d'un composant gérant un adapter (spinner, listview...)
- Récupération d'un Cursor
- Création de l'adapter
- Affichage

En cas de personnalisation de l'affichage, nécessité de créer son propre Adapter





Données persistantes

IHM, Cursor et Adapter

Exemple simple: ListActivity

- Une Activity devient une ListActivity
- Obligation de présence d'un objet ListView dans le layout
- Id spécifique: android:id="@android:id/list"
- SimpleAdapterCursor
- Application de l'adapter sur la vue courante

```
Cursor c = managedQuery(uri, PROJECTION,
    null, null, null);
startManagingCursor(c);
ListAdapter adapter = new SimpleCursorAdapter(
    this, android.R.layout.two_line_list_item,
    c,
    PROJECTION,
    new int[]{android.R.id.empty,
        android.R.id.text1,
        android.R.id.text2}
);
setListAdapter(adapter);
```





Services et Multithreading

Services

- Process actif tant que la mémoire n'est pas limitée
- Attention à la surcharge!
- Implémentation simple
 - Étend de Services
 - OnCreate
 - OnStart
 - OnDestroy
 - Déclarations dans l'Android Manifest





Services et Multithreading

Services

AIDL

- Android Interface Description Language
- Partage du service avec d'autres applications (Inter Process Communication)
- Développement d'interface
- N'accepte que
 - Des primitives
 - Des String et CharSequence
 - List ou Map
 - Autre AIDL
 - Autres classes Java implémentant Parcelable
- **void retrieveInfo(in String nom, out String Info)**
 - in / out: définit le sens de l'information
 - in: paramètre d'entrée est utilisée pour lecture uniquement par le service
 - out: paramètre pouvant être modifié par le service et propagé





Services et Multithreading

Alarmes, notifications

Notifications

- Toutes les alertes émanant du système ou d'autres applications
- Affichage d'une icône et/ou texte dans la barre de notifications
- Coloration de la led du téléphone

NotificationManager

- Hardware (son, led, vibreur)
- Icons / Texte

Obtenir une instance du NotificationManager

Création de l'objet Notification (icon, message...)

Création d'un PendingIntent pour la finalité de la notification (ouverture d'une activity)

Attribuer le pendingIntent à la notification

Envoyer la notification





Services et Multithreading

Accès distants

Librairies Apache HttpComponents

HTTP

Ajouter des librairies supplémentaires pour réaliser d'autres protocoles: XMPP, SMTP...

HTTP POST/GET

- HttpClient
- Renseigner les informations sur la requête
- execute()

Requête GET

```
DefaultHttpClient httpclient = new DefaultHttpClient();
HttpGet httpget = new HttpGet(url);
try{
    HttpResponse response = httpclient.execute(httpget);
}
catch (ClientProtocolException cpe) {
    Log.e(TAG, "ClientProtocolException retrieveInfo:" + cpe);
}
```





Services et Multithreading

Accès distants

Requête POST

```
DefaultHttpClient httpclient = new DefaultHttpClient();
HttpPost httpost = new HttpPost(http://www.monserveur.com);
List<NameValuePair> nvps = new ArrayList<NameValuePair>();
nvps.add(new BasicNameValuePair("cle1", valeur1));
nvps.add(new BasicNameValuePair("cle2", valeur2));
httpost.setEntity(new UrlEncodedFormEntity(nvps, null));
HttpResponse response = httpclient.execute(httpost);
```

Envoi de fichier par POST

- Utilisation de HttpURLConnection
- Définition des paramètres de la connexion (entrée, sortie, boundary, méthodes...)
- Ouverture d'un DataOutputStream sur la connexion
- Pour chaque élément à envoyer:
 - Ouverture du fichier en lecture
 - Ecriture du type de données et du flux dans le DataOutputStream





Services et Multithreading

Accès distants

Réponse d'une requête

- `HttpResponse`
- Sur retour de la commande `execute`
- Création d'un `HttpEntity` par `httpResponse.getEntity()`
- Lecture d'un `InputStream` sur `httpEntity.getContent()`





Services et Multithreading

Utilité des Threads

Blocage de l'application lors des accès distants (Wake Lock)

Solution (recommandation!)

- Utilisation de Thread pour les traitements bloquant: accès distant, calculs...

Les threads n'ont pas accès aux IHM

- Système de message de communication entre un Thread et un Handler
- Classe implémentant l'interface Runnable
- Constructeur prenant en paramètre l'Activity qui appellera le Thread
- Surcharge de la méthode run() dans laquelle
- Les traitements sont exécutés
- Les messages sont envoyés au Handler
- Création d'un objet Handler
- Gestion des codes reçus
- Mise à jour de l'IHM de l'Activity appelante

Une Thread ne peut être tuée, le système s'en charge

- Gestion d'un état permettant ou non d'appeler ou pas le contenu de run()





www.zeddini.com



Autres composants

Gestion des appels entrants et sortants

Connexion GSM/Bluetooth/Wifi/GPS

Envoi/réception SMS

SearchManager

Géolocalisation

- Localisation
- Map/MapView
- Accéléromètre

Multimedia

- Son
- Caméra
- 2D/3D





Livres

Busy coder's guide to Android Development – Mark. L. Murphy – Edition CommonsWare –
2 livres tutoriaux et Advanced Development
<http://www.commonware.com>

Hello Android, Introducing Google's mobile development Platform – Ed Burnette – 220 pages
<http://www.pragprog.com>

Ressources en lignes: site web

<http://developer.android.com> (documentations officielles)
<http://sites.google.com/site/io/> (recherche sur Mobile, présentations vidéos)
<http://www.anddev.org> (tutoriaux, forums)
<http://www.androidcommunity.com> (actualités)
<http://www.devx.com/wireless> (actualités, tutoriaux)
<http://www.helloandroid.com> (actualités)

Ressources en lignes: forums

<http://groups.google.fr/group/android-developers>
<http://groups.google.fr/group/android-beginners>

Ressources francophones: actualités et développement

<http://www.frandroid.com> (actualités, forums)
<http://www.pointgphone.com> (actualités, forums)





Développement Android - Google Maps

- Intégration de Google Maps V2
- Vues plan / satellite
- Gestion des données
- Zoom & défilement
- Surcharge de la carte



Développement Android - Zxing





Conclusions du Chapitre 1





Conclusion & Perspectives

- Des outils de développement
- Une communauté solide
- Une plateforme en évolution
- Des terminaux
- Google Play





Plus de détail dans les prochains Chapitres...

**Bien
Compris ?**

=>

**Passons aux
Ateliers**