Legond-Aubry Fabrice
fabrice.legond-aubry@u-paris10.fr

PPM(A)

Programmation
sur Plateformes Mobiles (Android)

PLAN

3 parties:

Généralités & Outils et Debug

API

Sécurité & Infrastructure

& Spécificité de la PM

PLAN

Généralités et Architecture

Généralités

- L'OS pour smartphone / tablette le plus déployé.
- Il existe de nombreuses versions
  - Elles se côtoient
- Il existe de nombreux supports physiques
  - Ils se côtoient
- Obligation de qualité de code
  - Utiliser de la manière la plus standard possible les API
  - Respecter les consignes de code de Google et des experts
  - Bien séparer le fonctionnel, de la GUI, du non métier.

Généralités

- Pré-requis
  - Langage JAVA
  - Pattern MVC / Observer
  - Programmation évènementielle
  - XML, Sax, Dom, json, sqlite
- Logiciels
  - Android Developpement Tool développé par « Google »
- IDE
  - NetBeans (module NBandroid)
  - Eclipse IDE - Plugin ADP
  - Android Studio de google (repack de Intellij android)

Généralités

Fonctionnalités

- Packaging automatique (*.apk)
- Construction graphique ou textuelle ou programmatique de la GUI
- Outils de debugage - Dalvik Debug Monitor System (DDMS)
- Gestion des signatures par certificats

Généralités

- Fonctionnalités
  - Packaging et déploiement « on click » ou en ligne de commande
  - Utilisation d'un simulateur
    - Particulièrement inadapté aux applications RT ou haute performance
    - **LENT**
  - Utilisation sur mobile
  - Debugging plus complexe
    - Debug, exécution pas à pas
    - Tests unitaires
  - MVN, Graddle

Historique

Historique du « système » Android

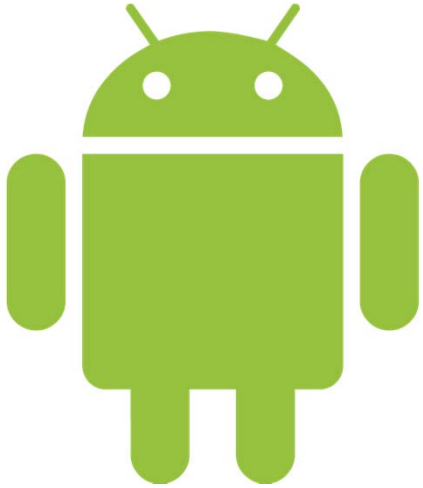
- Août 2005 : Google rachète la Startup Android Inc
- Novembre 2007 : Consortium « Open Handset Alliance »
  - Grand nombre de participants (samsung, ...)
  - « Standard et Norme » appareil mobile avec Android
- Décembre 2008 : Android SDK 1.0 sur un T-Mobile G1
- Octobre 2010 : Android devient rentable pour Google

Historique

Historique du « système » Android

- Mars 2012 : Google Play
  - Fusion du Android Market et de Google Music

- Le logo du système Android
  - Bugdroid
  - Personnage d'un jeu d'Atari des années 1990 «Gauntlet: the third encounter»



Versions

Date	Version	API Level
Septembre 2008	1.0.x	1 et 2
Avril 2009	1.5.x (Cupcake)	3
Septembre 2009	1.6.x (**D**onut)	4
Octobre 2009	2.0 et 2.1 (**E**clair)	5 à 7
Mai 2010	2.2.x (**F**royo)	8

Date	Version	API Level
Décembre 2010	2.3.x (**G**ingerbread)	9 à 10
Février 2011	3.x (**H**oneycomb)	11 à 13
Octobre 2011	4.0.x (**I**cescream sandwich)	14 à 15
Juillet 2011	4.1.x, 4.2.x 4.3.x (**J**ellyBean)	16 à 18
Octobre 2013	4.4.x et 4.4.x**W** (**K**itKat)	19 à 20

Versions

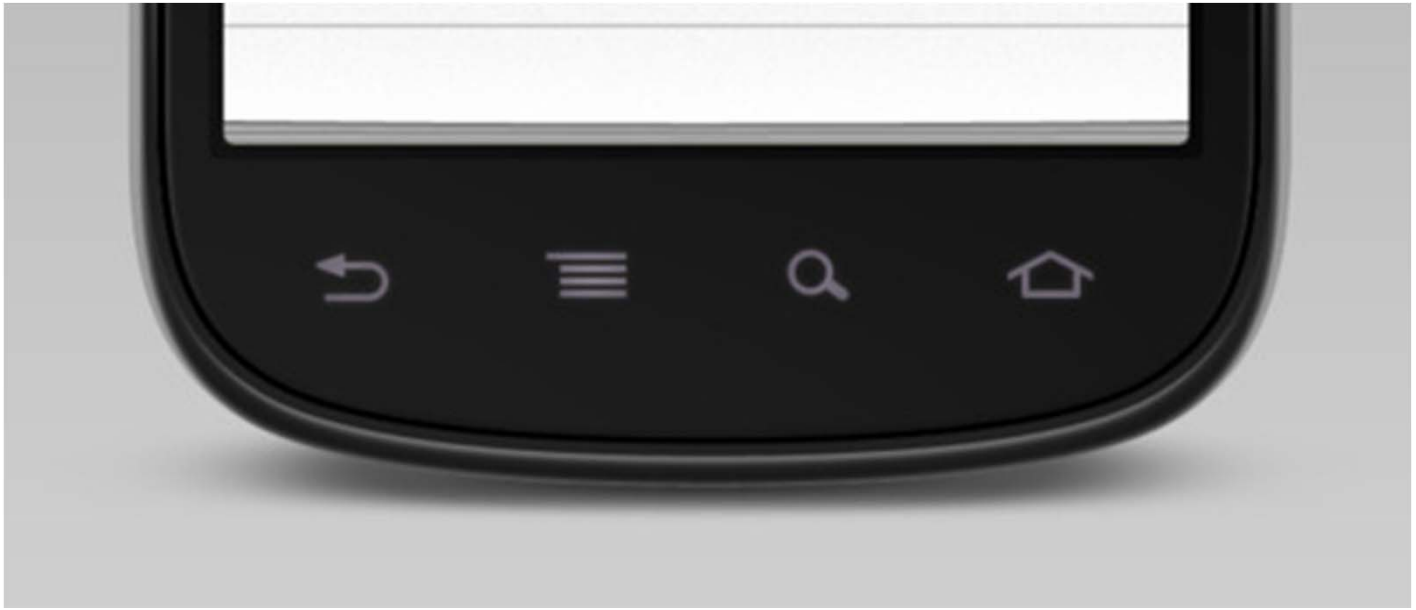
Date	Version	API Level
Novembre 2014	5.0.x (Lollipop)	21

- Pour les statistique de répartition, prière de regarder (statistiques Google Store)
  https://developer.android.com/about/dashboards/index.html

Versions

Pb: La fragmentation du marché
ie les différentes version de l'OS
Android dépend des constructeurs
iOS subit moins ce défaut

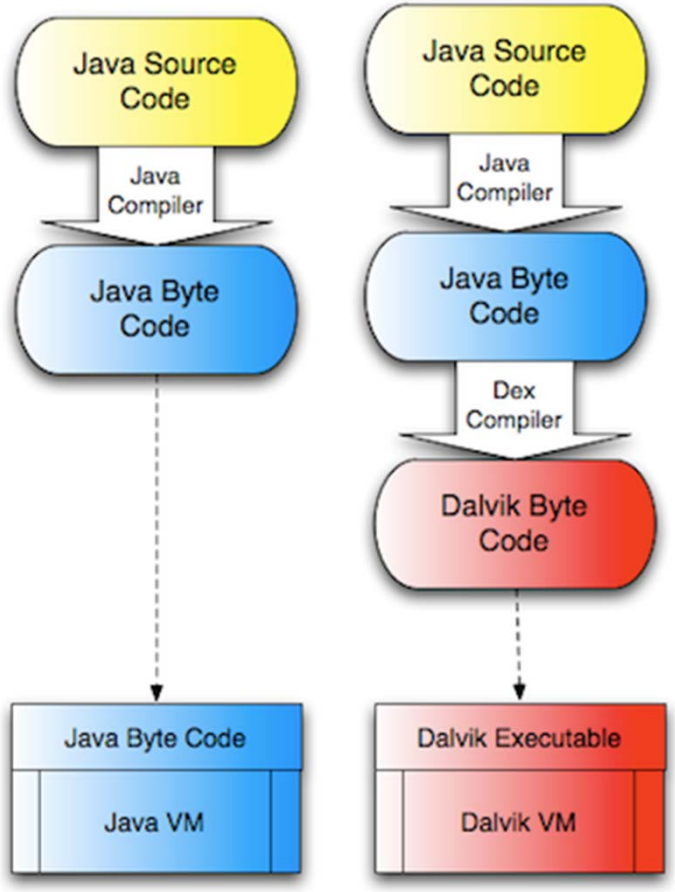
Les boutons du téléphone



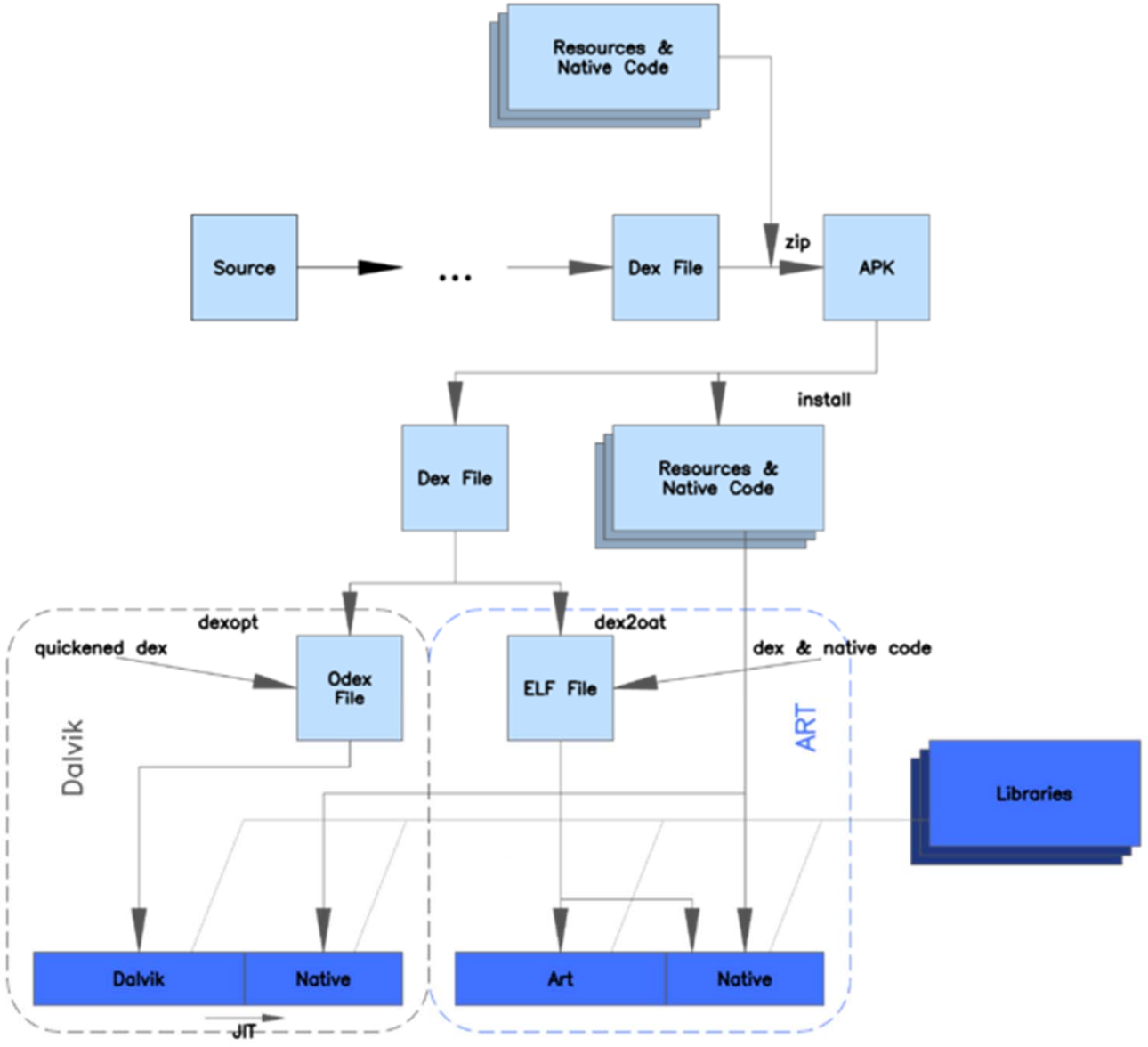
- Annuler / Retour : retour en arrière
- Menu : Afficher le menu de l'application
- Rechercher : activer la recherche Google
- Home : Retour à l'écran d'acceuil

Architecture

- OS basé sur un noyau Linux
  - SDK Android 4.0 basé sur noyau 3.0.1
  - **JAVA 32bits (min 1.6) pour android avant 5.x**
  - **JAVA 64bits (min 1.6) pour Lollipop et futur**
  - VM : Dalvik Machine Virtuelle
  - ART remplace Dalvik dans Lollipop (v5.x) et suivant
  - ART (Compilation à l'installation – fichiers elf) VS Dalvik (JIT – fichiers dex)



Architecture



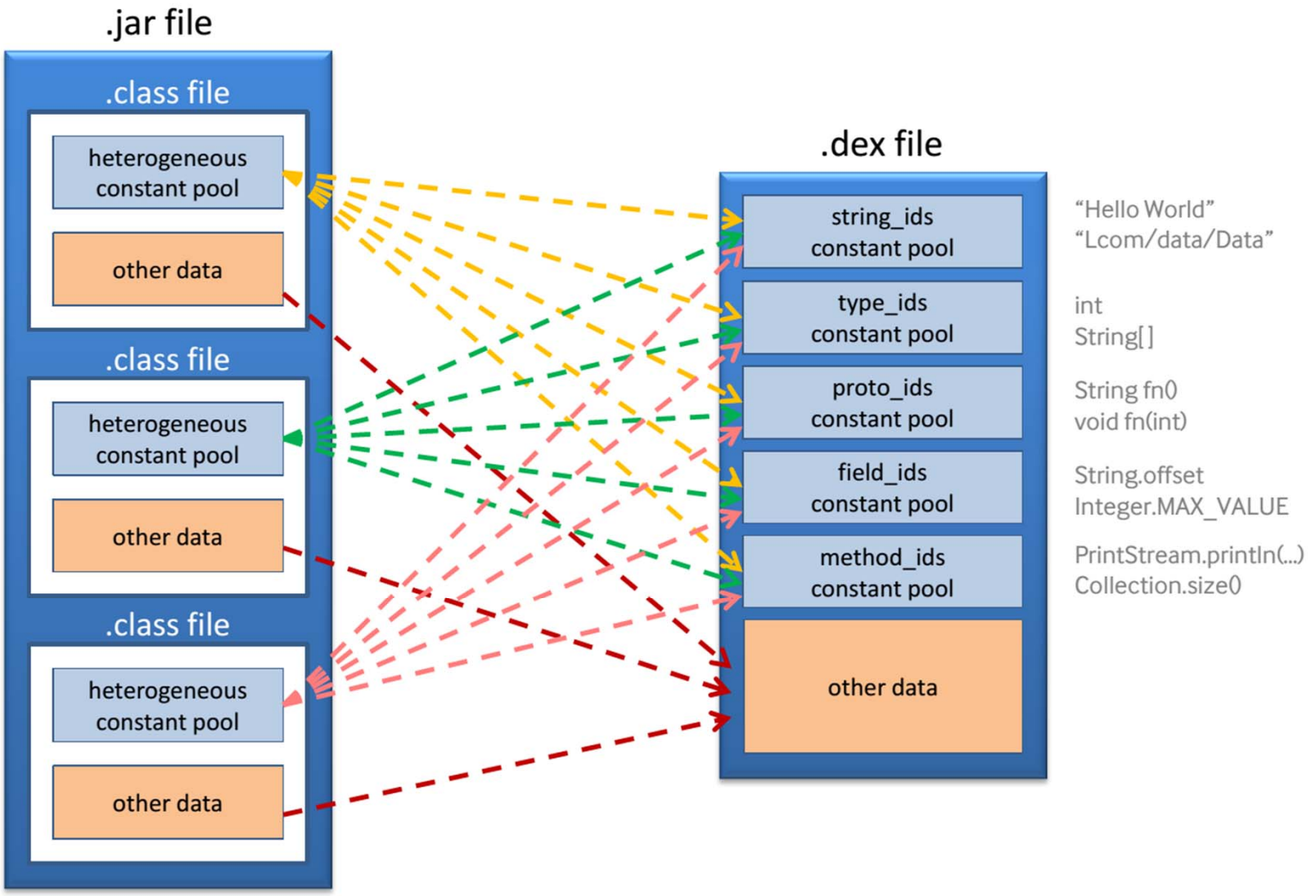
Architecture

- Xml pour les interfaces
- Beaucoup de vendeur vont inclure:
  - Leur propre surcouche graphique
  - Des programmes
  - **Des mouchards**
- Dalvik
  - VM à registres illimités (non à pile comme la JVM)
  - Peu d'instructions, code units, ...
  - Optimisation mémoire
  - Optimisation vitesse d'initialisation
  - Optimisation vitesse d'exécution (2x JVM classique)
  - Chaque processus Android possède sa propre VM

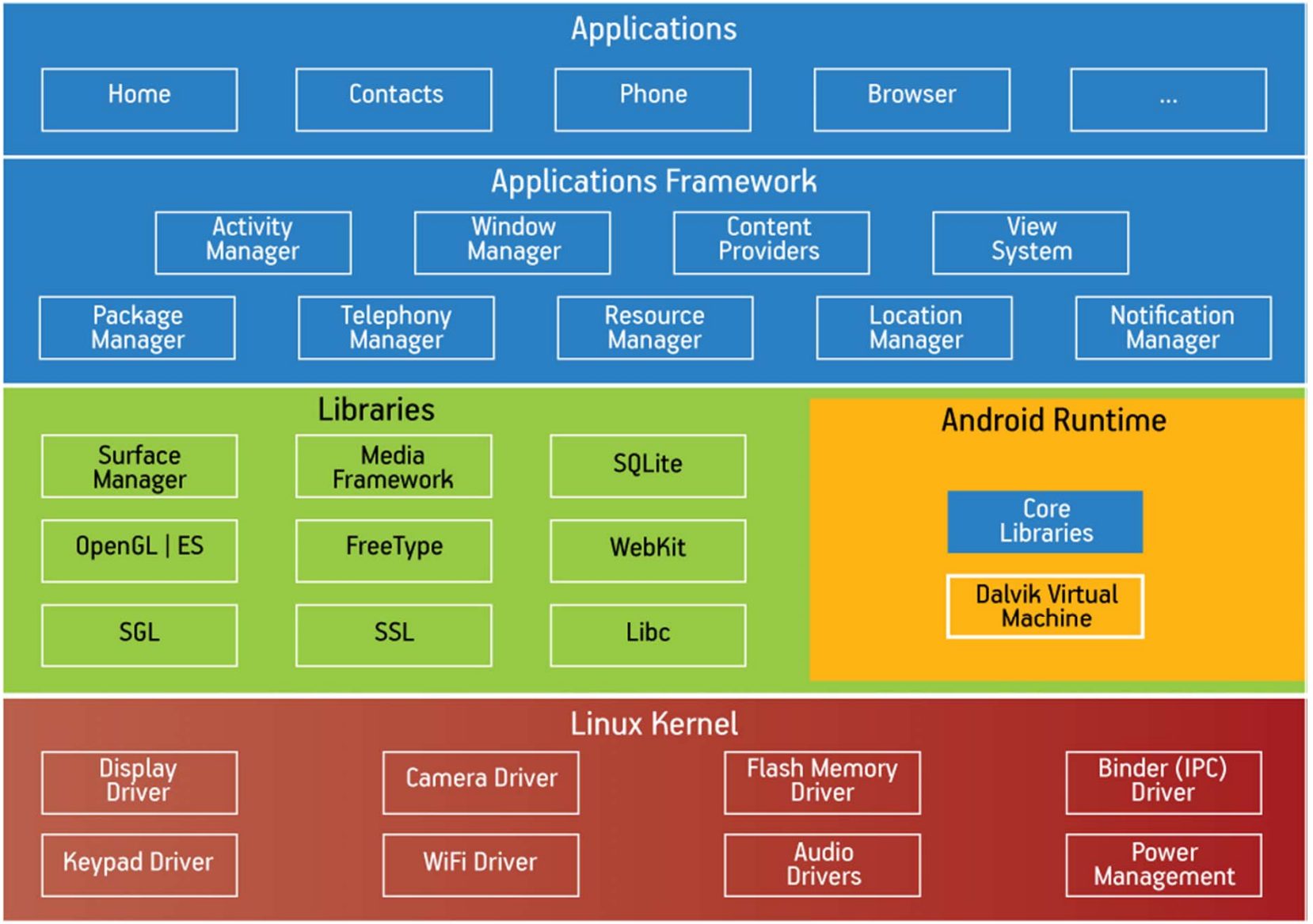
Architecture

- Dalvik (suite)
  - Ecrit Dan Bornstein (pour la petite histoire)
  - Transformation des fichiers “.class” en un seul .DEX
  - Cette VM a pour caractéristiques ...
    - Intégrée au noyaux android Linux
    - Utilise de la mémoire partagée (mmap)
    - Pour un OS sans swap; sur batterie
    - Zygote (duplication efficace de VMs)
  - Vérification & optimisation du code à l’installation Mais JIT jusqu’à la version 5 d’Android

Architecture



Architecture



Architecture

- Une application Android
  - Packaging sous forme de fichier APK
  - Utilisation de jar pour fabriquer les APK
    - classes.dex (*.dex)
    - « res » pour les ressources (icônes, images, layouts XML, …)
    - « AndroidManifest » : PERMISSIONS / configuration
  - L'archive DOIT être signé
    - Par défaut: debug.keystore (certificat d'un an)
    - Le dev doit générer son certificat avant la distribution
    - Il est conseillé d'avoir un certificat dev plutôt que un par application
    - Le certificat est utilisé pour la détection des mises à jours et signer le code !

Architecture

- Déclaration des droits dans AndroidManifest.xml
- Spécifier les ressources utilisées
  - Informer l'utilisateur sur l'application
    - ✔ <application> … </application>
  - Sécuriser le droit d'accès aux ressources
    - ✔ <uses-permission android:name=""/>
  - Utiliser ce qui est nécessaire
    - ✔ <uses-feature android:name="" android:required=""/>
  - Ainsi que d'autres spécifications
    - ✔ <use-library>, <use-sdk>, …

Architecture

- Mettre à disposition l'archive
  - Par un market ....
    - Google, Orange, autres ...
    - Payant (25$ chez google)
    - Gratuit (http://fdroid.org , http://aptoide.com)
  - En téléchargement libre/payant
    - API pub/achat par application
  - Installation « Copy And Install On Click »
    - La sécurité ... (NSA?)
    - API Google License Verification Library (Google play)
- Monter votre serveur de distribution
  - http://aptoide.org
- Installation manuelle

PLAN

Outils

Outils

- Android Studio : IDE
  - Il existe aussi IntelliJ Android, Eclipse Android
- Graddle : outil de compilation
  - Gestion des dépendances via Ivy ou Maven
  - Langage de script: Groovy (pas de xml)
  - http://gradle.org/documentation
- Lint : analyse de code
  - Chasse aux bugs, spécificités Android
  - Dos and Donts pour les ressources, i18n, ...
  - Gestion des erreurs de programmation Java
- AVD : Android Virtual Device (Manager)
  - Machines virtuelles android pour tester le code
- ADB : Android Debug Bridge
  - Outils de debug, logcat, ...
- NDK / SDK : Native (C++) / Software Development Kit
  - Outils de compilation Java, XML, packaging, ...
- Il reste les outils classiques (Git/Svn/Cvs), diff, merge, intégration continue, …

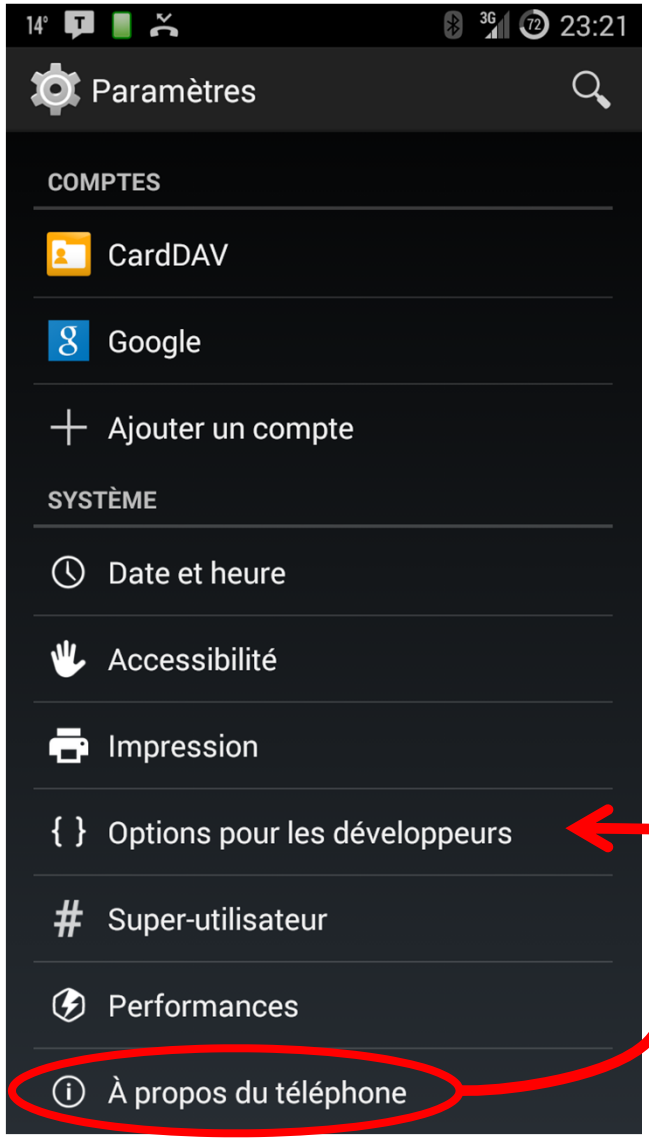
Les premières manipulations

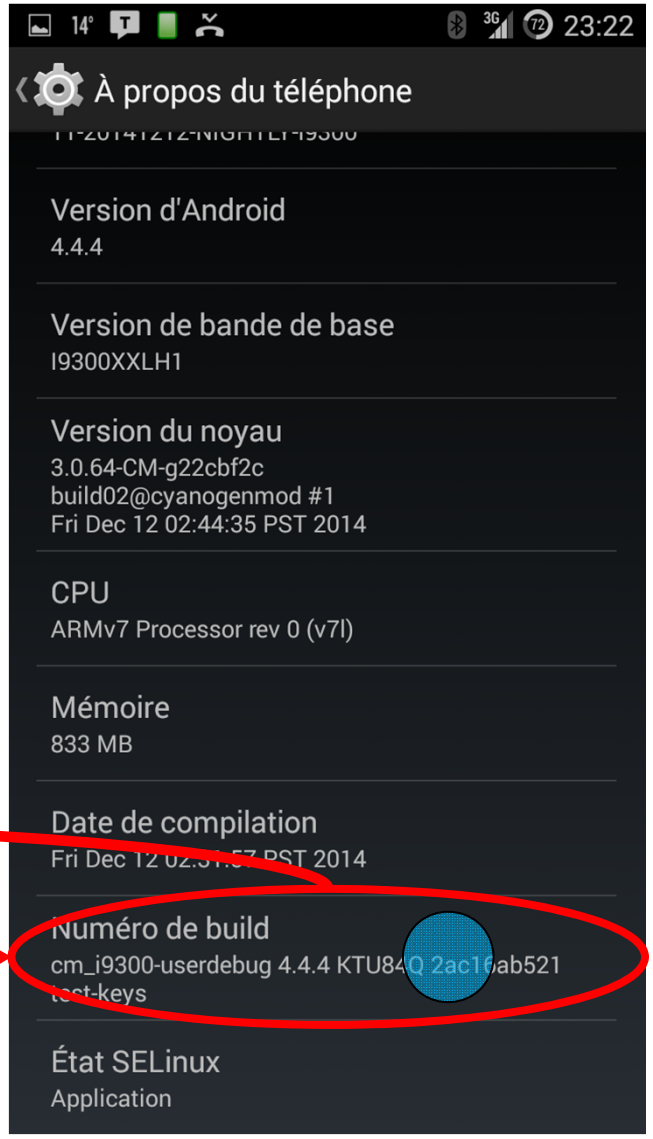
- Créer une instance VD
  - SD Card (512Mb)
  - Choix du matériel (Caméra, GPS, Touch …)
  - Skin : choix du mobile à simuler
    - Galaxy, Nexus, etc.
- Dans le menu configuration / A propos du téléphone
  - **CLIQUER 7 FOIS SUR « Numéro de Build »**
- Paramètre du mobile
  - Activer le Débugage USB
  - Activer « sources inconnues »
  - Type de connexion USB
  - Connecter le mobile

Les premières manipulations

- Android Studio
  - Pas mieux que intellij / eclipse / netbeans
  - Choisissez le votre
  - Existe depuis 2013 avant plugin Eclipse ADP
- Tous utilisent
  - Obligatoirement le SDK Android (Soft. Dev. Kit)
  - Optionnellement le NDK Android (Native Dev. Kit)
- Limiter le code NDK au minimum pour la portabilité
  - Librairies de performances / d'accélération
  - Jeux

Les premières manipulations

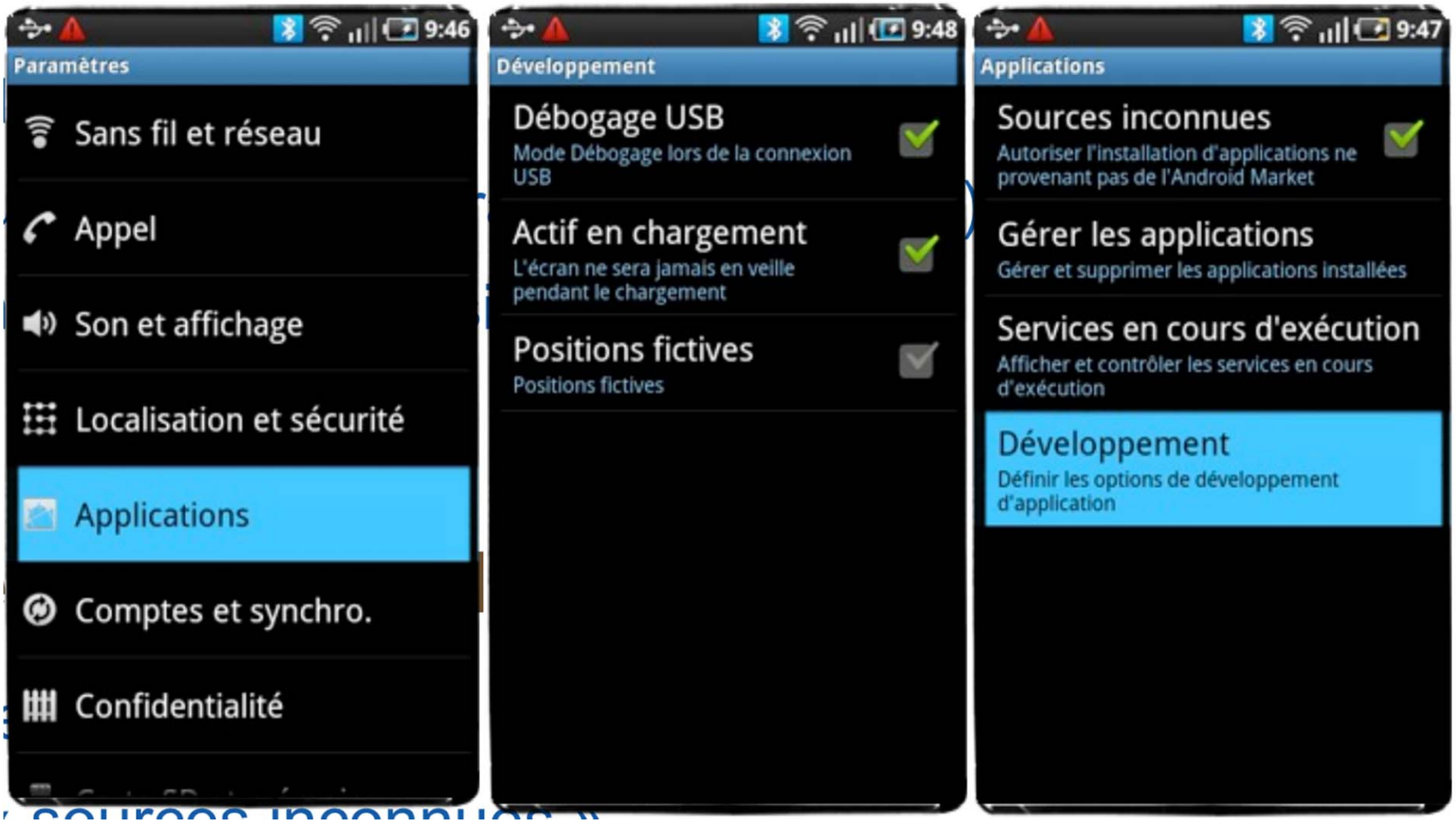




7 fois

Les premières manipulations

Généralités et outils



Les premières manipulations

- SDK android - ANDROID_HOME
  - Mise à jour du SDK: « SDK Manager »
  - GUI création des VMs android « AVD Manager »
  - ./tools
    - ↙ Émulateur qemu pour arm / x86 / mips
  - ./platform-tools
    - ↙ Adb, fastboot, sqlite3
  - ./build-tools
    - ↙ Compilateur java vers dex
    - ↙ Outils apk

Les premières manipulations

- Fixer ANDROID_SERIAL
  - C'est le device par défaut lorsqu'il y en a plusieurs
  - Les téléphones VM (VD) se nomment « emulator-xxxx »
    - Le premier se nomme « emulator-5554 »
- Fixer JAVA_HOME
  - **Soit un JDK 64 avec les options pour 32bits soit JDK 32 bits**
  - **Attention Android >5.x utilise un java 64 bits**
- Fixer ANDROID_HOME, ANDROID_SDK_HOME
  - avec la même valeur
  - Sur les anciens SDK, il faut fixer ANDROID_SDK_ROOT
- **Note: Si %ANDROID_SDK_HOME% est défini, les machines virtuelles seront créées dans %ANDROID_SDK_HOME%\.android**
- Fixer PATH
  - Ajouter %JAVA_HOME%\bin
  - Ajouter %ANDROID_HOME%
  - Ajouter %ANDROID_HOME%\tools
  - Ajouter %ANDROID_HOME%\platform-tools

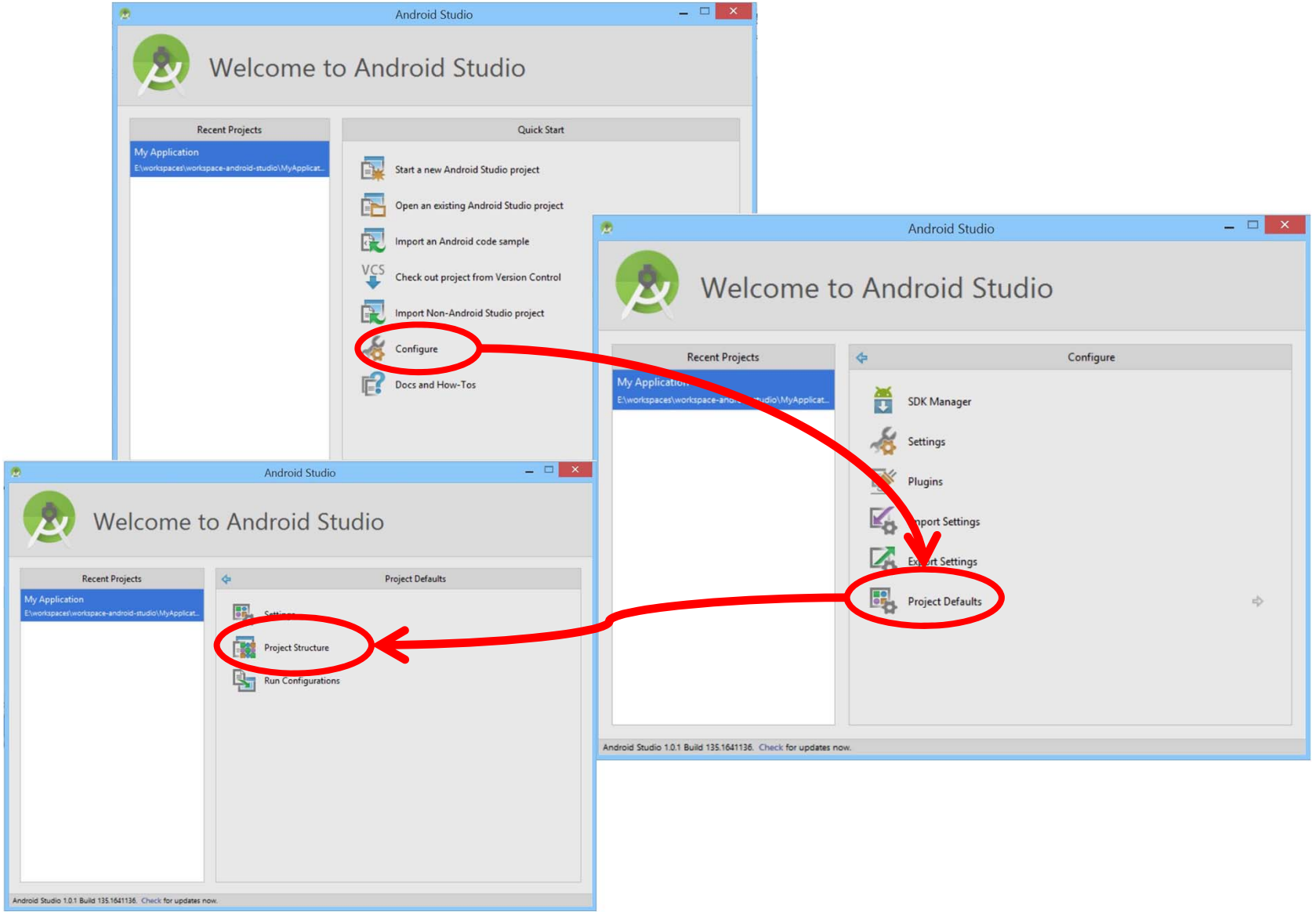
Les premières manipulations

- Télécharger android-studio et android-sdk

  https://developer.android.com/sdk/index.html
- Choisir les versions ZIP
- Décompressé dans
  - D:\login\PPM\android-studio
  - D:\login\PPM\android-sdk
- Lors de la première exécution, il y a installation.
  - Choisir « custom »
  - Et faire pointer sur le SDK et le JDK
- Sortir de l'IDE, et lancer « SDK Manager »
  - Télécharger les API 4.2 ou 4.3 ou 4.4
- Créer les setEnv.bat pour fixer les variables d'environnement et créer un androidstudio.bat pour lancer l'IDE

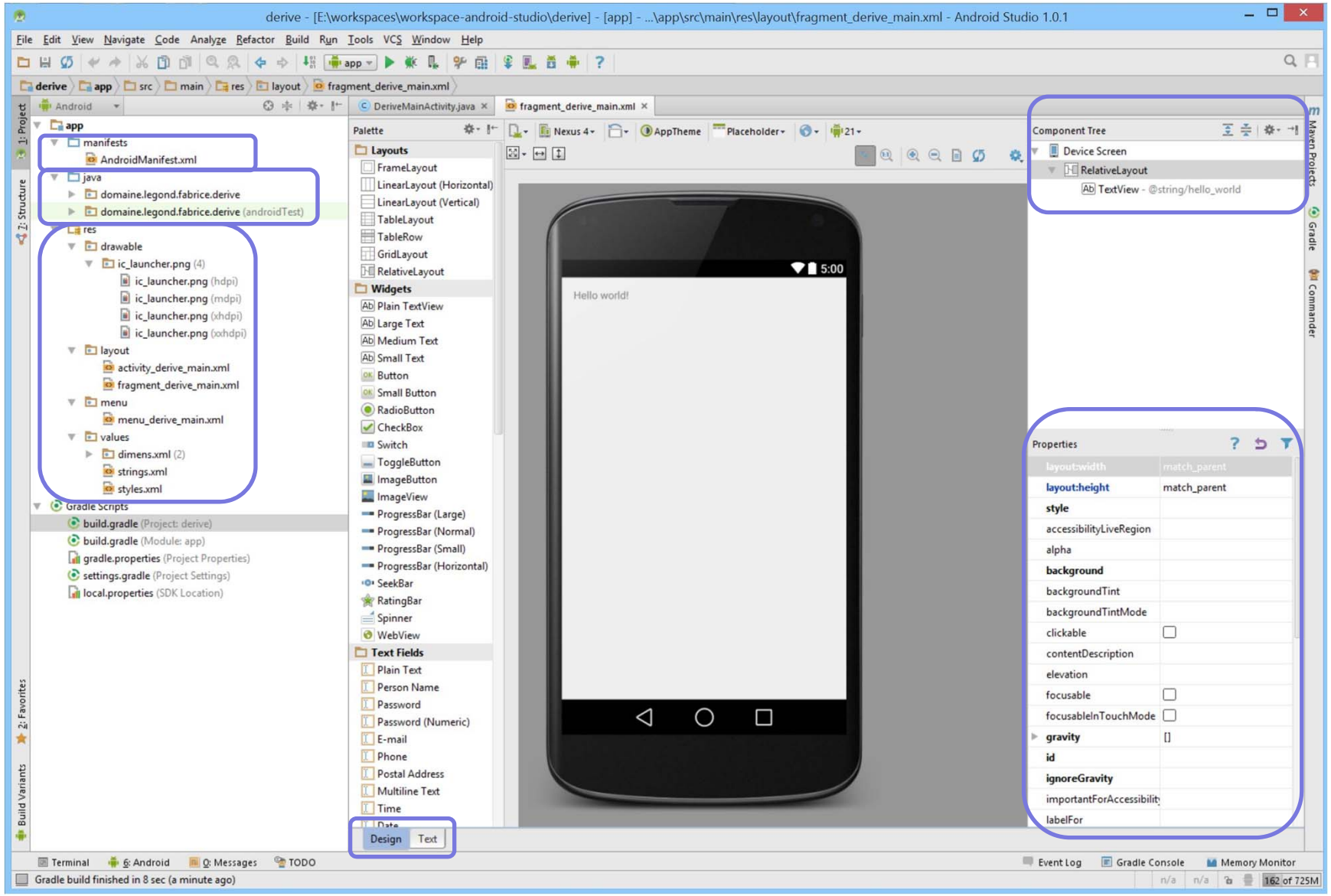
Les premières manipulations

Généralités et outils



Les premières manipulations (android-studio)

Généralités et outils



Les premières manipulations (android-studio)

- Structure des projets
- App/AndroidManifest.xml
  - Fixer les permissions, pré-requis
  - Déclarer les composants des applications (activités, services, widget)
- App/Java
  - Code des classes et T. U. (voir vos autres cours)
- App/res
  - layout : éléments graphiques de l'application
  - layout-land : éléments graphiques en mode paysage
  - menu : menus de vos applications
  - values : chaînes de caractères (i18n)
  - drawable / mipmap : images
  - raw : musiques, son, ...
- 2 façons d'ajouter des éléments au projet
  - Bouton droit → new ... l'IDE va créer l'ensemble des fichiers et générer des squelettes
  - Ajouter à la main → vous devez créer les liens à la main (par ex. entre le xml et la classe)

Les premières manipulations

- Création d'une machine virtuelle
  - Utilisation de « AVD Manager » du sdk
    - Dans ANDROID_HOME, ANDROID_SDK_HOME
  - Android studio / tools / android / avd manager
- Un VD a pour nom « emulator-x »
  - x est le numéro du port telnet (sms, geoloc, ...)
  - x+1 est le numéro du port ADB (shell, install, ...)
  - Premiers ports 5554/5555, puis 5556/5557, ...
- Lancer adb (on peut aussi utiliser android.bat dans tools)
  - https://developer.android.com/tools/help/adb.html

Les premières manipulations

- Un PD (Physical Device) a un nom (pas de standard)
- ATTENTION le port telnet n'existe pas sur un device PHYSIQUE (ie par ex. un vrai téléphone)
- Le seul moyen de générer des faux évènements (fake sms, fake geoloc, …) est d'utiliser adb
- Utilisation de « adb shell », programme « am »
  - « am broadcast » pour générer un intent https://developer.android.com/tools/help/adb.html#IntentSpec
  - « am startservice »
  - « am profiling <PROCESS> » start/stop
  - « am monitor » pour surveiller
- D'autres commandes dans /system/bin (aapt, pm)

Les premières manipulations

- Création d'une machine virtuelle
  - Utilisation de « AVD Manager » du sdk
    - Dans ANDROID_HOME
  - Android studio / tools / android / avd manager
- Lancer adb (on peut aussi utiliser android.bat dans tools)
  - https://developer.android.com/tools/help/adb.html
  - Quelques commandes
    - « adb devices –l » / fixer « ANDROID_SERIAL »
    - « adb –s TARGET_NAME shell »
    - « adb pull /system/app/Email.apk »
    - Extraire (jar) et afficher AndroidManifest.xml et META-INF
    - Votre ami le chat Log : « adb logcat »

Les premières manipulations

Les raccourcis claviers des AVD :

Emulated Device Key	Keyboard Key
Home	HOME
Menu (left softkey)	F2 or Page-up button
Star (right softkey)	Shift-F2 or Page Down
Back	ESC
Call/dial button	F3
Hangup/end call button	F4
Search	F5
Power button	F7
Audio volume up button	KEYPAD_PLUS, Ctrl-F5
Audio volume down button	KEYPAD_MINUS, Ctrl-F6
Camera button	Ctrl-KEYPAD_5, Ctrl-F3

Les premières manipulations

Généralités et outils

Emulated Device Key	Keyboard Key
Switch to previous layout orientation (for example, portrait, landscape)	KEYPAD_7, Ctrl-F11
Switch to next layout orientation (for example, portrait, landscape)	KEYPAD_9, Ctrl-F12
Toggle cell networking on/off	F8
Toggle code profiling	F9 (only with -trace startup option)
Toggle fullscreen mode	Alt-Enter
Toggle trackball mode	F6
Enter trackball mode temporarily (while key is pressed)	Delete
DPad left/up/right/down	KEYPAD_4/8/6/2
DPad center click	KEYPAD_5
Onion alpha increase/decrease	KEYPAD_MULTIPLY(*) / KEYPAD_DIVIDE(/)

Les premières manipulations

- Signature pour Déploiement
  - Voir le cours de sécurité sur les PKI
- Pour vérifier un certificat
  - openssl pkcs7 -in CERT.RSA -inform DER -print_certs -out CERT.CER
  - openssl x509 -in CERT.CER –text
- Pour créer son certificat sous android studio
  - Build / Generate signed apk
- Vérification de la signature en ligne de commande
  - ✔ jarsigner -verify -verbose -certs my_application.apk
- On peut aussi faire en ligne de commande
  - « android » est le mot de passe par défaut de « debug.keystore »

Les premières manipulations

Pour créer son certificat en ligne de command

- Génération de la clef et du certificat
  - Keytool → Voir les commands list, printcert, importcert, exportcert
  - keytool -genkey -v -keystore my.keystore.jks -alias alias_name -keyalg RSA -keysize 2048 -validity 10000
  - keytool -exportcert -alias alias_name -keystore my.keystore.jks -list -v
- Signature de l'archive apk (support multiple sign)
  - zip –d my_applicatio.apk META-INF/*.RSA META-INF/*.SF
  - jarsigner -verbose -sigalg SHA1withRSA -digestalg SHA1 -keystore my.keystore.jks my_application.apk alias_name
- Optimisation du jar pour le déploiment
  - zipalign -v 4 my_application.apk my_application_zipaligned.apk

Les premières manipulations

- Mettre à jour l'application pour y inclure des logs
  - 1ère méthode

```
package packagea.packageb;
public interface Constants
{ String TAG_LOG = "packagea.packageb"; }
…
if (BuildConfig.DEBUG)
     { Log.e (Constants.TAG_LOG, “message"); }
```

  - 2ème méthode (LENTE !!! Temporaire)

```
Log.i (this.getClass().getCanonicalName(),”message”);
Log.i (MainActivity.class.getName(),”message”);
Log.i(getApplication().getBaseContext().getPackageName(), “message”);
```

- Filtrage du log
  - En ligne de commande

```
ANDROID_LOG_TAGS="packagea.packageb:I MyApp:D *:S"
adb logcat
```

  - Sinon par la GUI de l'IDE

Les premières manipulations

- DDMS
  - GUI android-studio (Run / Attach to running process)
  - Ddms/monitor (%ANDROID_HOME%/tools)

  https://developer.android.com/tools/debugging/ddms.html
- Ouvrir la perspective
  - Tools / Android / Android Device Monitor
    - ↙ Ancienne version de Android Studio
  - Run / Debug (version recente)
  - Choisir le processus

Quelques sites de références

- Site Web tutoriaux
  - http://www.vogella.com/tutorials/
  - http://www.androidhive.info/
  - http://www.tutorialspoint.com/android/
  - http://www.xda-developers.com/
- Site web de référence sur Android
  - http://source.android.com
  - https://developer.android.com/index.html
  - http://android-developers.blogspot.com/
  - http://source.android.com/source/building-kernels.html

Quelques sites de références

- Programmation andoid
  - http://www.androiddesignpatterns.com
  - http://www.coreservlets.com/android-tutorial/#PPT
- Livres (ils se périment VITE !)
  - La série des « Android Programming » (Mark L. Murphy)
  - La série des « Professional Android » (Reto Meyer)
  - Android Apps Security (Sheran A. Gunasekera)
  - Android Forensic (Andrew Hoog)
  - Java Concurrency in Practice (Goetz, et al)