Département d'Informatique

Développement ANDROID

- Frédéric MOAL
- Année 2011/2012

1

Révisions du document

26/11/2009	Création	F.Moal

2

Android : références

- Présentation Android : http://www.android.com/
- Développeurs : http://developer.android.com/intl/fr/index.html
- Tutoriaux : http://www.vogella.de/articles/Android/article.html http://florentgarin.developpez.com/tutoriel/android/client-xmpp/ http://www.ibm.com/developerworks/edu/os-dw-os-eclipse-android.html

3

Android ?

- C'est un système d'exploitation basé sur Linux, avec une interface de programmation en Java
- Android est géré par l' Open Handset Alliance qui est dirigée par Google
- De plus en plus de terminaux sous cet OS :
  - Des téléphones : HTC G1, Tatoo, Hero…Samsung Galaxy S,Spica…, Motorola Droid, Dell, LG, Sony..
  - Des « tablettes » : Archos 5, Galaxy Tab, Dell…
  - Des smartbooks : Acer, Quanta, Dell, …
- De plus en plus d'applications/développeurs
- SDK libre et gratuit !

4

Android ?

- Il utilise une Java virtual machine spéciale (Dalvik) qui est basée sur Apache Harmony. Dalvik utilise du Bytecode Java particulier

=> il est obligatoire d'utiliser le compilateur Android (et pas sun) pour créer ce bytecode spécial

- Stockage en BD natif avec SQLLite
- Pour le dev, Android Development Tools (ADT) avec plugin Eclipse fournit par Google

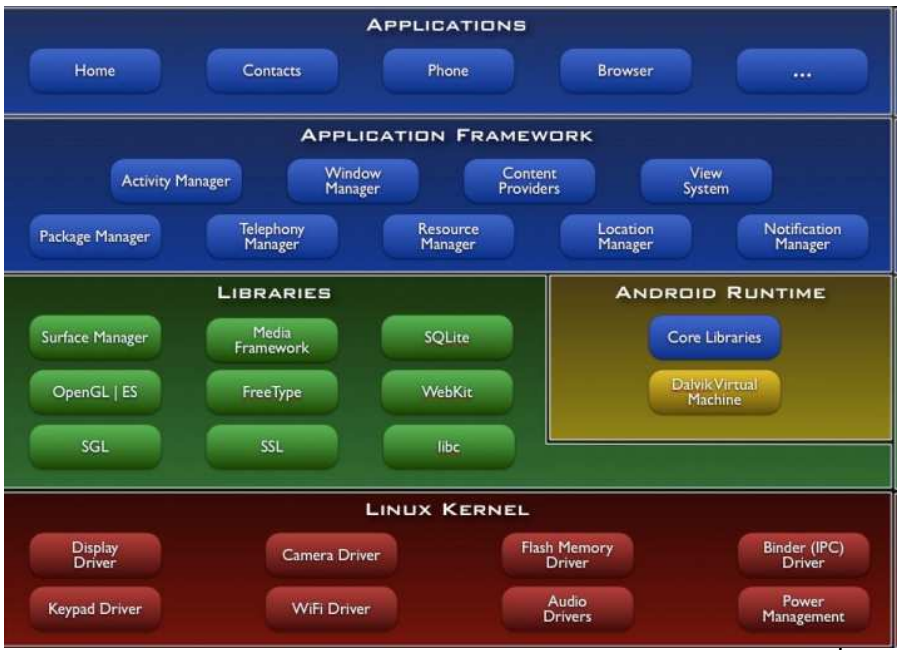
5

Fonctions supportées

- Framework complet de développement, avec une librairie de composants pour l'interface et…
- Machine virtuelle optimisée pour appareils mobiles (processeurs « lents » 500Mhz-1Ghz)
- Navigateur web intégré (basé sur Webkit)
- Optimisations graphiques avec une librairie 2D et 3D (basée sur OpenGL ES 1.0/…)
- Support média audio, image, vidéo (MPEG4, H.264, MP3, AAC, AMR, JPG, PNG, GIF)
- Support GSM indépendant du matériel/opérateur
- Support Bluetooth, EDGE, 3G et Wi-Fi
- Appareil photo, GPS, boussole, et un accéléromètre

6

Architecture



Framework

- Les développeurs ont accès à l'ensemble des fonctions su SDK core d'Android, les mêmes utilisées par les applications de base (email, web, contacts…)
- Approche par composants :
  - toute application peut mettre à disposition des autres applications ses services (avec une gestion des autorisations)
  - Tout composant peut être réutilisé ou remplacé

8

Une application Android

- Chaque application est exécutée dans un contexte isolé :
  - Par défaut, chaque application tourne dans un processus linux différent (noyau 2.6)
  - Chaque process à sa propre machine virtuelle Java
  - Par défaut, chaque appli reçoit un user ID linux différent, avec des droits de lecture/écriture uniquement pour cet uid

⇒ pour éviter les problèmes d'instabilité du système

Il est possible d'avoir deux applications avec le même uid ou la même VM pour partage de fichiers/données

9

Une application Android

- Une application Android est constituée de :
  - Activity
    - Un écran de l'appli
  - Intent / Broadcast Receiver
    - permet à une application de fournir et/ou utiliser des services d'autres applis. Par exemple, l'application de téléphone demande via un intent un contact à l'application de gestion des contacts. Les applications s'enregistrent via un IntentFilter
  - Services
    - activités en tâches de fond sans UI
  - Content Provider
    - fournit des données aux applications, eg une BD SQLLite qui peut être utilisée comme Data Provider

10

Activity

- Une activity = un écran
- Dans une application, plusieurs activity ; une principale (accueil), qui lance les suivantes
- Hérite de la classe Activity
- Chaque activity a une fenêtre par défaut pour s'afficher ; normalement, tout l'écran sauf …
- Peut utiliser d'autres fenêtres, eg boites de dialogue
- Contenu = hiérarchie de View (zone rectangulaires)
- Activity.setContentView() place la racine de la hiérarchie

11

Activity

- L'interface utilisateur (Activities) est définie par des layouts.
- Le layout défini les éléments de l'UI, leurs propriétés et leurs dispositions.
- Un layout peut être défini en XML ou par du code à l'exécution.
- XML plus simple si le layout est fixe, sinon le code est plus souple. Une approche mixte est possible.

12

Activity et cycle de vie

- Le système d'exploitation contrôle le cycle de vie de votre application. A tout moment, le système peut arrêter ou détruire votre application (appel entrant…).
- Le système Android définit un cycle de vie des activités par des méthodes pré-définies, dont :
  - onSaveInstanceState() : appelée si l'activité est arrêtée. Utilisé pour enregistrer les données de telle sorte que l'activité peut restaurer ses Etats si relancée.
  - onPause() : toujours appelé si l'activité se termine, peut être utilisée pour libérer ressource ou enregistrer des données
  - onResume() : appelée si l'activité est relancée

13

Services

- Sans interface
- = démon
- Lancés dans le thread principal
- Si opérations longue : création d'un nouveau thread
- Héritent de la classe Service

14

Broadcast receivers

- Composant qui reçoit et réagit aux annonces broadcast
- Beaucoup d'annonces systèmes : changement time zone, batterie faible, photo prise, …
- Peut lancer des broadcast (data dispos, …)
- Plusieurs receivers par application, pour chaque « event »
- Hérite de BroadcastReceiver
- Peut lancer une activity ou utiliser le NotificationManager (eg vibration ou status bar)

15

Content provider

- Fournit des données aux applications
- Stockage des données dans le file system, dans BD SQLLite, …
- Hérite de la classe ContentProvider
- Pour l'appel, utilisation de ContentResolver

16

Activation des composants

- Les composants (activity, service, broadcast receiver) sont activés par des messages asynchrones : Intents
- Un objet Intent contient le message :
  - Activity, Service : nom de l'action et URI des données (eg activity pour affichage d'une image)
  - BroadcastReceiver : nom de l'action annoncée (eg appui sur le bouton de l'appareil photo)

17

Shutdown des composants

- Content provider : actif uniquement quand il répond à une requête d'un ContentResolver : pas d'arrêt
- BroadcastReceiver : actif uniquement lors de la réception d'un broadcast : pas d'arrêt
- Activity : appel à finish() pour s'arrêter elle-même, ou à finishActivity() pour terminer une activity lancée précédemment par la méthode startActivityForResult()
- Service : appel à stopSelf() ou Context.stopService()
- Le système peut arrêter directement des composants non utilisés ou pour récupérer de la mémoire.

18

Tâches

- Les Activity lancent d'autres Activity, dans la même application ou dans une autre (eg google maps)
- Quand l'utilisateur appuie sur le bouton back, l'activity précédente est affichée
- Une tâche = une PILE d'Activity, avec une racine
- Appui sur Home => la pile complète est mise en background
- Back = dépile une Activity
- Retour en foreground : toute la pile est « réactivée »
- Ce comportement par défaut peut être modifié à l'aide de flags

19

Manifest

- Toute application Android est décrite par un fichier XML "AndroidManifest.xml".
- Il contient :
  - la description de toutes les classes de l'application
  - Les permissions demandées par l'application, eg accès au réseau, aux contacts, …
  - L'utilisateur devra valider (accepter ou refuser) les permissions lors de l'installation de l'appli
- C'est le fichier de déploiement de l'application = WEB.XML
- Dans le fichier apk de l'application

20

Manifest

- Structure :

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

Tous les composants doivent être déclarés ainsi

<service>, <receiver>, <provider>

Exception pour les BroadcastReceiver qui peuvent être enregistrés plus tard

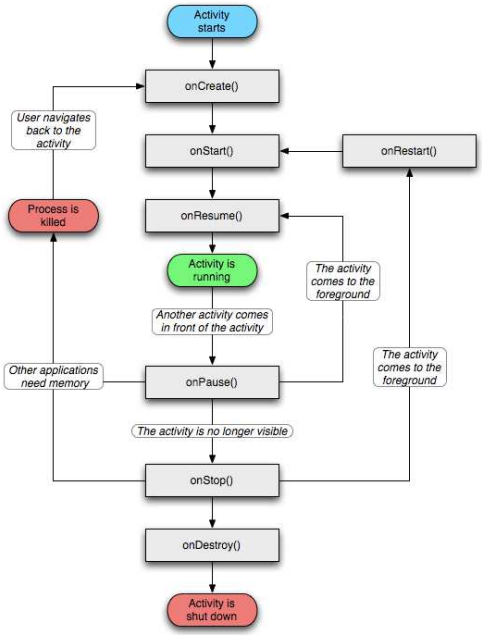
21

Processus/Threads

- Par défaut, 1 seul process et 1 seul thread pour tous les composants, 1 seul UID
- Modifiable par un attribut process (dans chaque composant ou dans <application>)
- !! Pas d'opération longue ou bloquante (eg accès réseau)
- Utilisation de la classe standard Thread de Java pour créer des threads
- Classes spécifiques pour la gestion des threads : Looper, Handler, HandlerThread
- Attention, L'OS peut décider de tuer un process

22

Cycle de vie



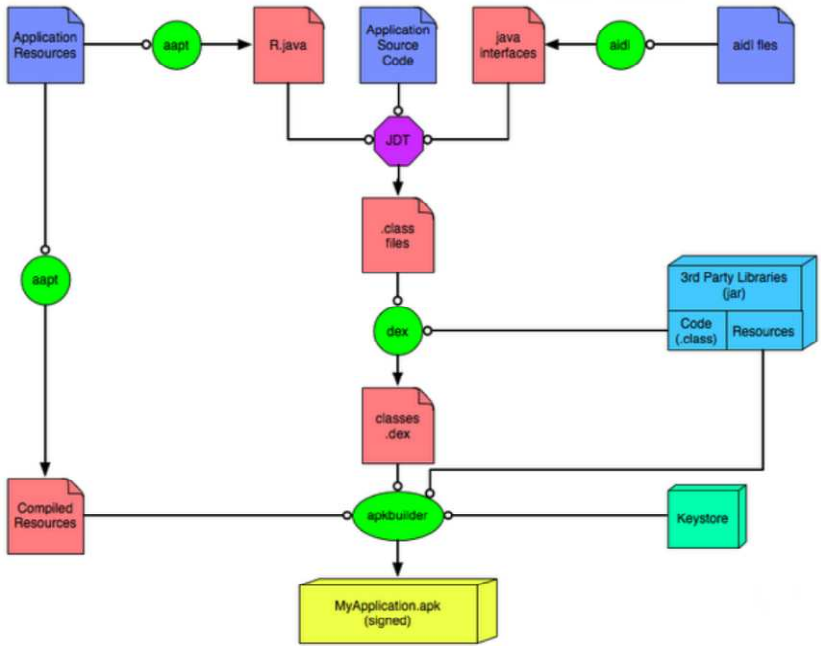
23

Build

- Dans le SDK, plusieurs outils pour la compilation/génération/packaging des Application Android
- Aapt, aidl, dex, adb…
- Un shell adb
- Le téléphone/terminal doit être configuré pour accès [via USB], mode appelé debug ou ADB
- Déploiement direct via Ordinateur ou install standalone via .apk [depuis carte SD ou market]

24

Build



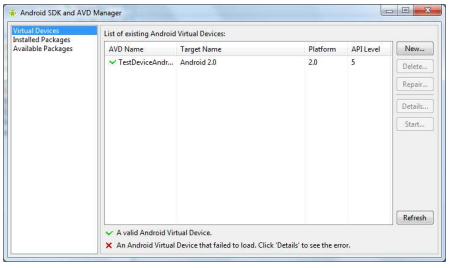
25

Installation

- Un SDK à télécharger http://developer.android.com/intl/fr/sdk/index.html (Windows, MAC/OS, Linux)

Attention : version 2.0, juste downloader

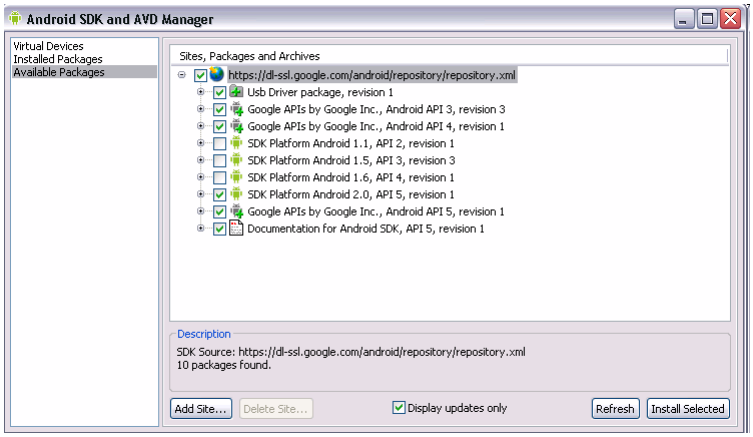
- Une installation d'Eclipse
- Un plugin Eclipse à installer par le update manager à l'URL : https://dl-ssl.google.com/android/eclipse/



26

Installation

- Config plugin eclipse pour trouver le SDK
- Une interface de gestion des librairies (1.5, 1.6 et 2.0) et des devices virtuels



27

Emulateur

- Run as… Android application : un émulateur (lent!)

28