Code d'Armor

android

What else ?

Marc Poppleton (Orange Labs)

sommaire

OHA, qu'est-ce?

L'Open Handset Alliance est un groupement d'une trentaine de sociétés dans le domaine de la technologie et des mobiles:

- Google, Nuance, eBay…
- Intel, Texas Instruments, NVIDIA, …
- HTC, Motorola, Samsung, LG Electronics, …
- T-Mobile, Telefónica, NTT DoCoMo, Telecom Italia,…

Son but est d'accélérer l'innovation dans le domaine du mobile en fournissant une plateforme mobile ouverte.

Android, what's it?

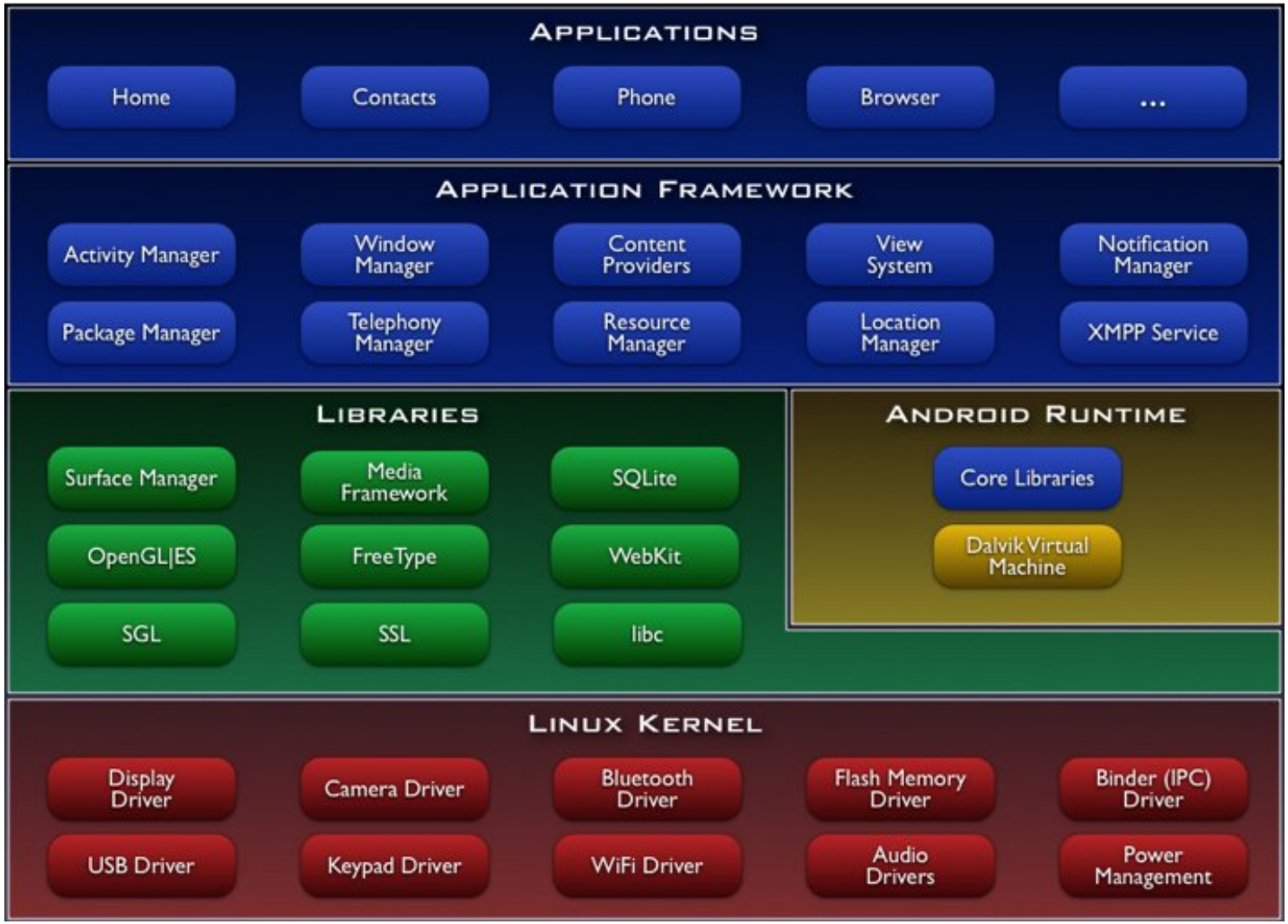
1. C'est un système d'exploitation pour terminaux mobiles
2. C'est un ensemble d'APIs
3. C'est un ensemble d'applications clefs.

Code d'Armor

Architecture du système

Code d'Armor

Architecture du système Android



Architecture du système Android

- Noyau Linux 2.6
- Couche d'abstraction entre le hardware et le software
- Implémentation à la charge des constructeurs de terminaux

Architecture du système Android

- Librairies en C ou C++
- Accessibles à travers le framework
- Support du MPEG-4, H.264, MP3, AAC, …
- Graphismes en 2D et 3D, accélération hardware possible

Architecture du système Android

- Librairies standard Java
- Machine virtuelle Dalvik reposant sur le noyau Linux
- Une machine virtuelle par application
- Exécution de bytecode DEX (code Java compilé puis converti)

Architecture du système Android

- APIs Java
- Communes aux applications natives et aux applications externes
- Applications spécialisées
- Publies leurs fonctionnalités pour être utilisées par les autres

Architecture du système Android

- Applications natives
- Codées en Java
- Utilisent les APIs de l'application framework

Code d'Armor

Environnement de développement

Environnement de développement

Le SDK fourni en plus des APIs un ensemble d'outils pour le développement, le débogage, le packaging et le déploiement.

On y trouve:

- un pluging pour Eclipse
- un émulateur de terminal
- divers outils pour gérer une base de données SQLite, analyser les logs, …

Environnement de développement

- Le classique:

Eclipse (v3.2 min), un JDK (v6 min), SDK + plugin ADT

- L'alternatif:

IntelliJ IDEA (v10 CE min), un JDK (v6 min), SDK.

- Le hardcore:

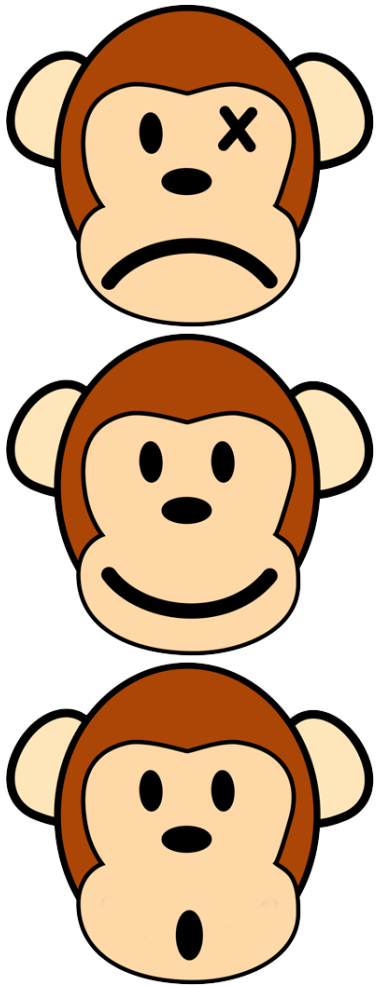
Un JDK (v6 min), SDK, vi, command line.

Android UI Utils

- Librairie Open Source
- Une appli web:
  - Génération d'icônes
  - Génération d'illustrations de rendu
- Calques pour design d'IHM:
  - Stencils pour PenciL GUI
  - Prototypage rapide d'interfaces
- Preview temps réel sur un terminal:
  - Affichage sur un terminal d'une partie de l'écran
  - Idéal pour tester le rendu réel
- http://code.google.com/p/android-ui-utils/

Tester

- Android Junit extension:
  - Dans Eclipse avec ADT
  - Ant pour les autres
- Dalvik Debug Monitor Server:
  - Logs
  - Debug dans l'IDE
  - Dump de stack
  - Allocation monitor
- The Monkey!
  - Générateur d'évènements sur l'UI et le système
  - Stress-test de l'application
  - Prière de jeter votre app aux singes



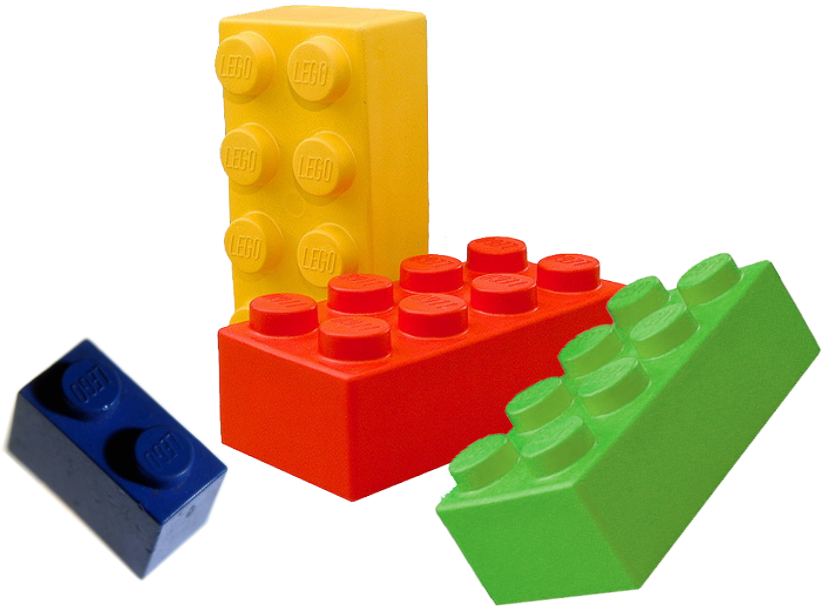
Code d'Armor

Anatomie d'une application Android

Blocs types

•Une application Android se décompose en 4 types de blocs, déclarés dans le AndroidManifest.xml

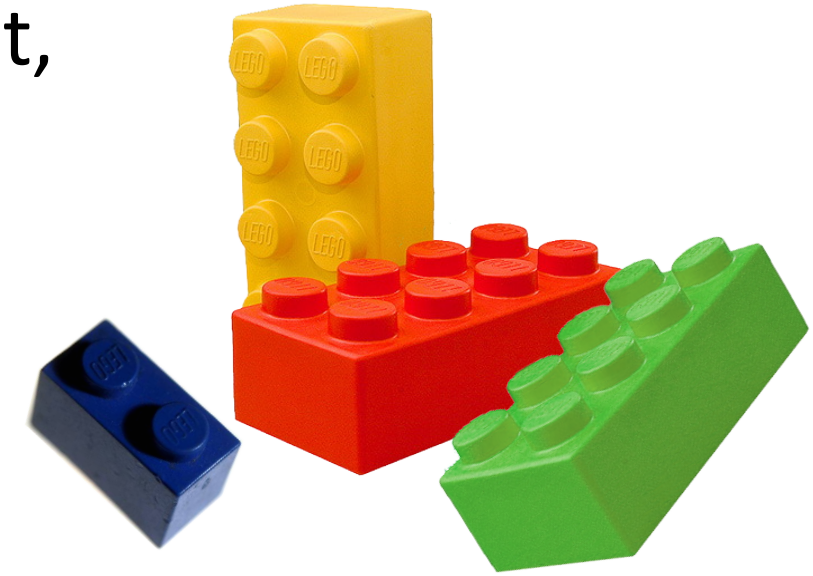
- Activity
- Broadcast Receiver
- Service
- Content Provider



AndroidManifest.xml

Chaque application Android possède un fichier AndroidManifest.xml. Ce fichier répertorie :

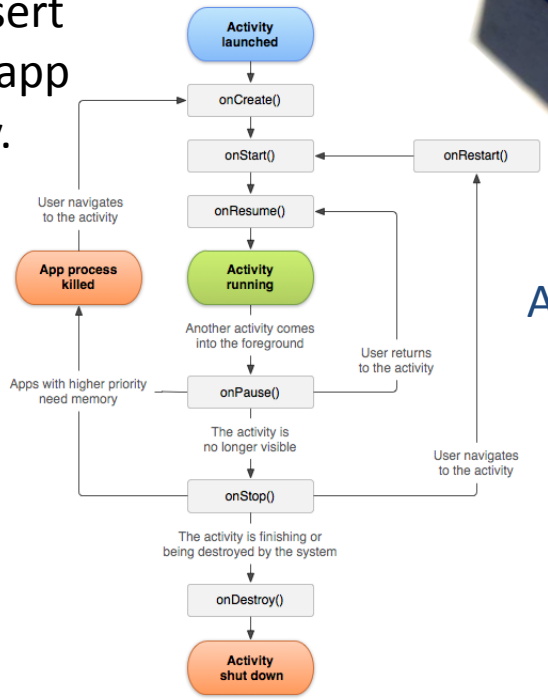
- les composants de l'application,
- les conditions pour qu'ils s'activent,
- les services qu'ils fournissent,
- les permissions requises
- Les caractéristiques exigées.



Blocs types

- Une Activity correspond à un écran. Une application comporte autant d'activités qu'elle comporte d'écrans.
- Déclaré dans le fichier Manifest
- Généralement une Activity sert de « point d'entrée » dans l'app et appelle les autres Activity.
- Cycle de vie à respecter.

Activity



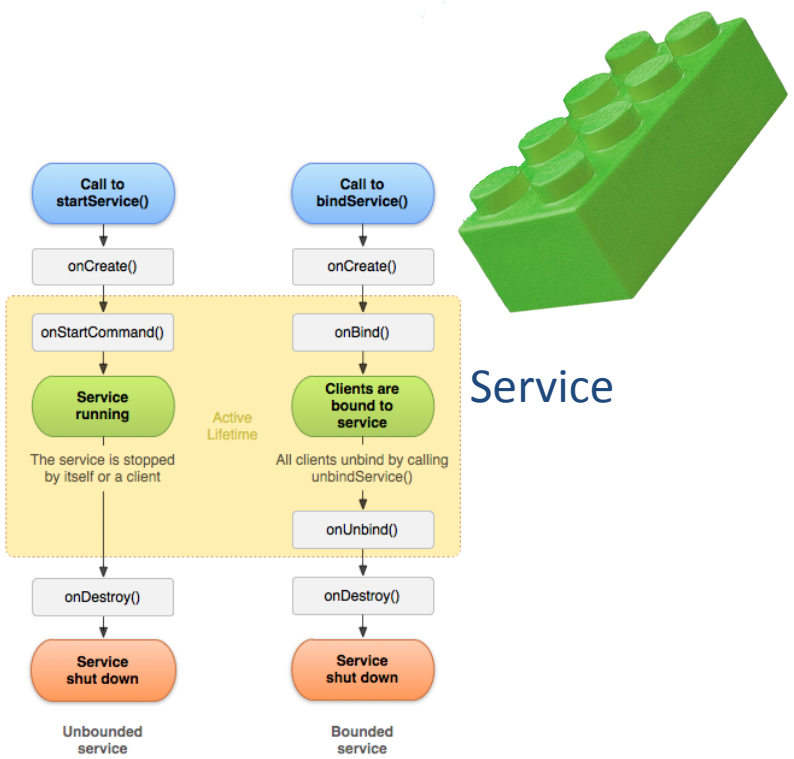
Blocs types

- Un Broadcast Receiver réagit à des événements particuliers : appel entrant, données disponibles, variable à une valeur particulière,…
- Processus à priorité haute dans le système, ne PAS lui confier de tâches lourdes ni asynchrones
- Souvent utilisé pour piloter un Service ou un NotificationManager

Broadcast Receiver

Blocs types

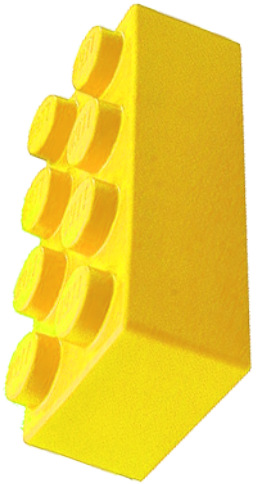
- Une application qui tourne en tâche de fond et qui ne requière par d'interface est un Service.
- Un cycle de vie différent selon la façon dont il a été lancé
- Exécuté sur le thread principal sauf si spécifié autrement:
  - Tâches longues doivent être dans un thread à part
  - Priorité basse sauf spécifié autrement



Service

Blocs types

- Une application qui tourne fourni des données persistante à d'autres applications est un Content Provider.
- Mécanisme d'URI et de type MIME pour toutes les opérations CRUD
- Gestion des permissions pour partager les données ou les restreindre à notre application
- Souvent adossé à une base de données SQLite (mais ça n'est pas obligatoire)
- Données manipulées à travers un Cursor



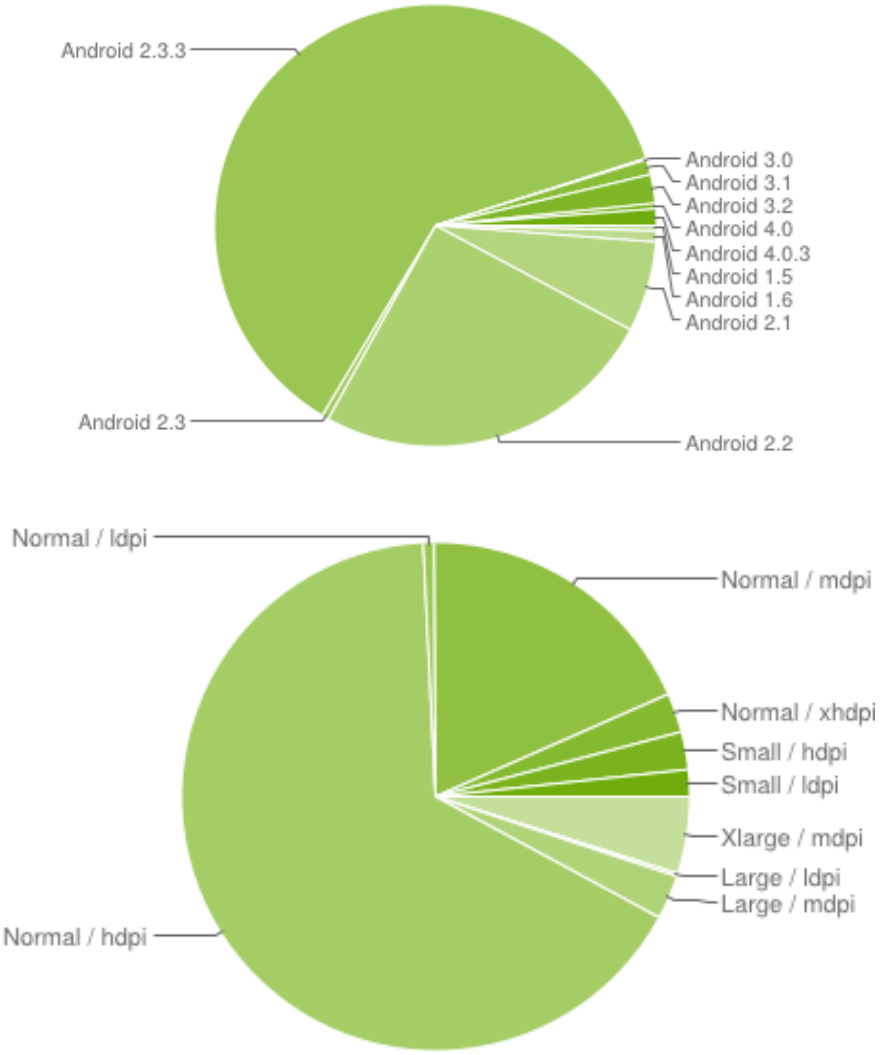
Content Provider

Code d'Armor

Une application pour les gouverner tous

Fragmentaquoi?

- 11 versions du SDK en circulation
- 9 tailles et densités d'écrans
- Plus d'une centaine de terminaux différents



Source : http://developer.android.com/ 4/3/2012

Stratégies

Cibler tous les terminaux

- Ressources graphiques pour toutes les densités et tailles
- 9-patch
- Tester si une fonctionnalité est dispo au runtime
- ActionBar
- Fragments!

Cibler un sous-ensemble

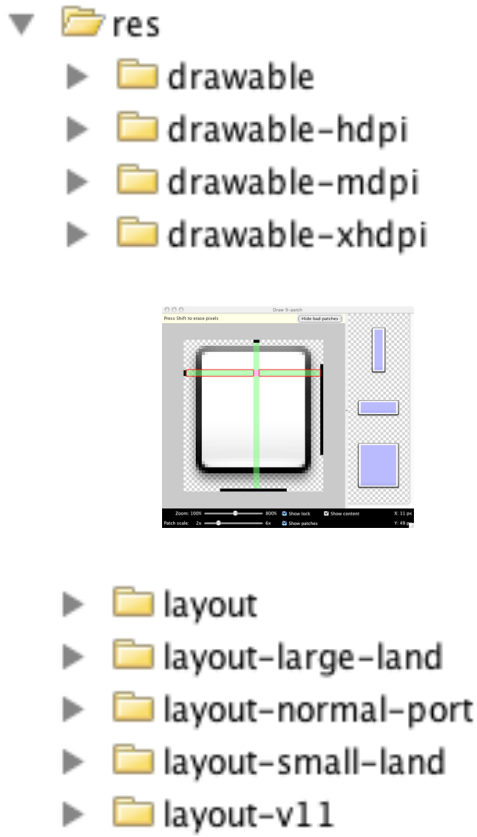
- Limiter à un ensemble de tailles d'écrans
- Limiter aux tablettes ou au téléphones
- Limiter à l'aide des caractéristiques techniques

Cibler un sous-ensemble

- Limiter à ensemble de tailles d'écran
  -
```
<supports-screens
  android:smallScreens="false"
  android:normalScreens="true"
  android:largeScreens="true"
  android:xlargeScreens="false"
  android:anyDensity="true"/>
```
- Limiter à certaines caractéristiques technique
  - `<uses-feature android:name="android.hardware.camera"/>`
  - Peut être optionnelle avec l'attribut `android:required="false"`

Cibler tous les terminaux

- Fournir les bitmaps en différentes densités
  - Des bitmaps par défaut
  - Des bitmaps pour chaque densité

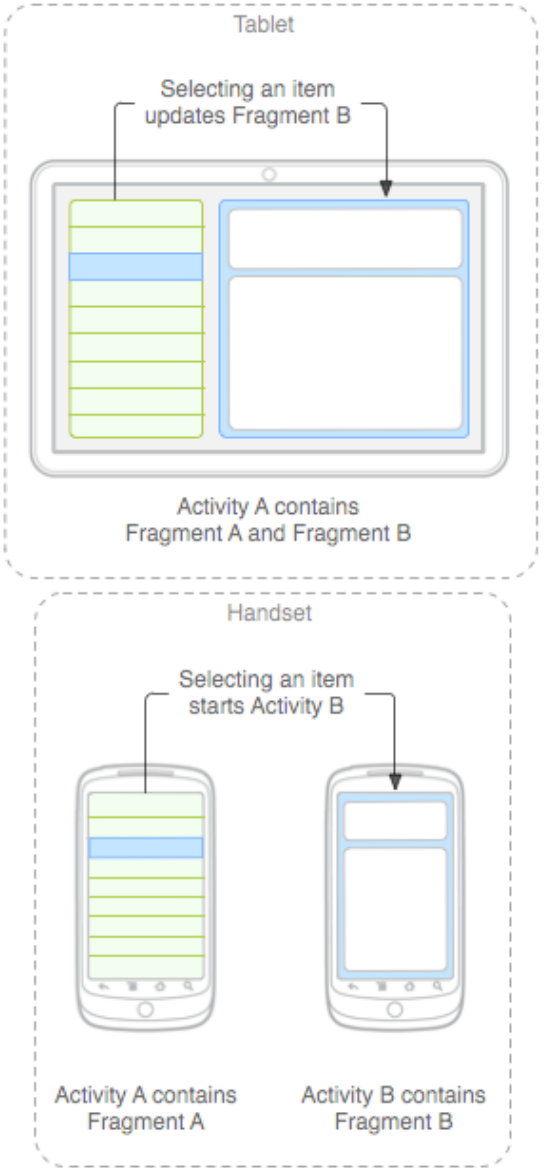


- 9-patch
  - Un bitmap avec des zones étirables

- Fournir différents layouts
  - Mieux exploiter l'espace disponible
  - Configuration qualifiers

Fragments!

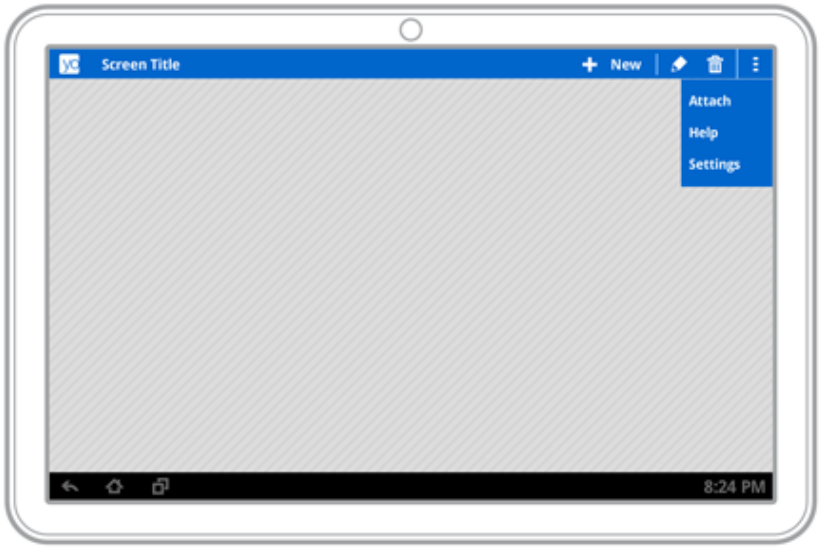
- 9 ” != 3,7”
- Plusieurs actions sur un écran
- Combiner des composants
- Détection au runtime
- Disponible depuis SDK 11
- Librairie de rétrocompatiblité



Source : http://developer.android.com/

ActionBar

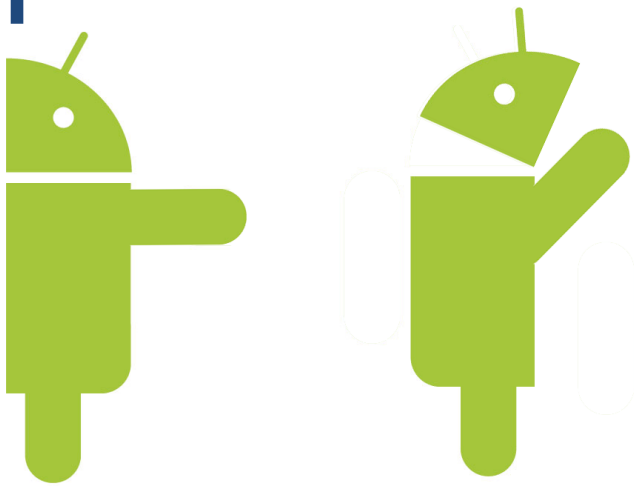
- Navigation dans l'application
- Disponible depuis SDK 11
- Librairie de rétrocompatiblité
- Menus
- Adaptative
- Customisable



Source : http://developer.android.com/

Code d'Armor

Communiquer entre applications



startActivityForResult

- Appeler une Activity pour une tâche spécifique
- Deux méthodes:
  - **`startActivityForResult (Intent intent, int requestCode)`**
  - **`onActivityResult (int requestCode, int resultCode, Intent data)`**

```
static final String INTENT_ACTION_SCAN = "com.google.zxing.client.android.SCAN";
static final String INTENT_EXTRA_SCAN_MODE = "SCAN_MODE";
static final String INTENT_EXTRA_PRODUCT_MODE = "PRODUCT_MODE";
/*
…
*/
        try {
            final Intent intent = new Intent(INTENT_ACTION_SCAN);
            intent.putExtra(INTENT_EXTRA_SCAN_MODE,INTENT_EXTRA_PRODUCT_MODE);
            startActivityForResult(intent, REQUEST_SCAN);
        } catch (ActivityNotFoundException e) {}
/*
…
*/
```

```
@Override
protected void onActivityResult(int requestCode, int resultCode, Intent data) {
        if (resultCode == RESULT_OK) {
            switch (requestCode) {
                case REQUEST_SCAN:
                    onScanAdd(data);
                    break;
                case REQUEST_SCAN_FOR_CHECK:
                    onScanCheck(data);
                    break;
            }
        }
}
```

Broadcast…

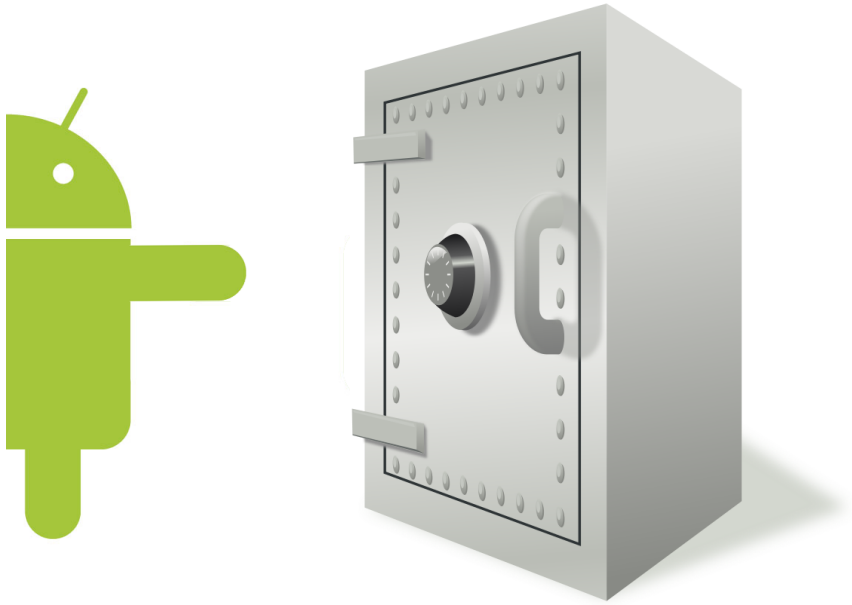
- Diffusion « à l'aveugle »
- Asynchrone
- Trois méthodes:
  - `sendBroadcast(Intent intent)`
  - `sendBroadcast(Intent intent, String receiverPermission)`
  - `sendOrderedBroadcast(Intent intent, String receiverPermission, BroadcastReceiver resultReceiver, Handler scheduler, int initialCode, String initialData, Bundle initialExtras)`

…Receiver

- Déclaration statique dans le Manifest
  - ```
    <receiver android:name=".receivers.SMSInterceptor">
          <intent-filter android:priority="0">
              <action
    android:name="android.provider.Telephony.SMS_RECEIVED" />
          </intent-filter>
    </receiver>
    ```
- Déclaration dynamique dans le code
  - `registerReceiver (BroadcastReceiver receiver, IntentFilter filter)`
- Une méthode:
  - `onReceive (Context context, Intent intent)`
- Broadcaster au sein d'une application:
  - `LocalBroadcastManager`
    ```

Code d'Armor

Stocker des données



Code d'Armor

SharedPreferences

- Stockage de clefs/valeurs
- Abstraction du mécanisme, le système gère
- Un seul fichier:
- `SharedPreferences getPreferences (int mode)`
- Plusieurs fichiers:
- `SharedPreferences getSharedPreferences (String name, int mode)`
- Modes:
  - privé,
  - public en lecture,
  - Public en écriture

Filesystem

- FileInputStream:
  - write,
  - close!
- FileOutputStream:
  - read,
  - close!
- Modes:
  - privé,
  - public en lecture,
  - public en écriture.
- Stockage local ou sur support externe:
  - `Environment.getExternalStorageState()`
  - `Environment.getExternalFilesDir()`

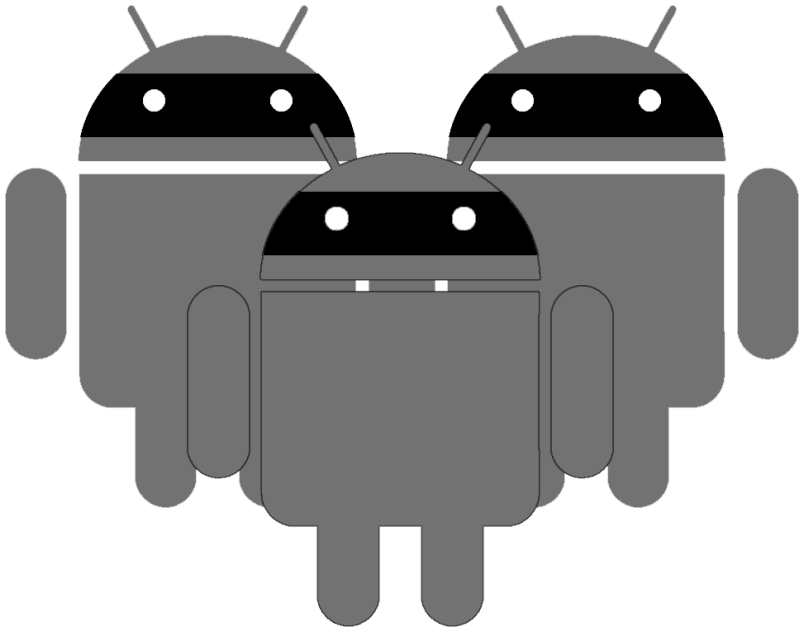
Code d'Armor

SQlite

- Base de données SQLite 3
- Du SQL mais pas complètement:
  - PAS de jointures à droite (mais gauche oui)
  - PAS de Alter Table (rename table et add column)
  - VIEW uniquement en lecture
  - Pas de GRANT/REVOKE
- Souvent derrière un ContentProvider

Code d'Armor

Travailler dans l'ombre…



Code d'Armor

Service!

- Tâches à exécuter sans IHM:
  - Lire un mp3
  - Synchroniser des données sur le réseau
  - Enregistrer des données de capteur…
- Deux modes de vie:
  - Démarré par un tiers et s'arrêtant quand il le souhaite
  - Démarré par binding IPC et exécuté tant qu'un tiers est connecté

AlarmManager

- Pour réveiller une app à un moment donné
  - Programmer l'envoi d'une Intent
- Peut-être récurrent
- Appelé même si le terminal est en veille:
  - Au receveur de l'Intent de gérer le wake lock

```
// Create an IntentSender that will launch our service, to be scheduled
// with the alarm manager.
PendingIntent mAlarmSender = PendingIntent.getService(AlarmService.this,
        0, new Intent(AlarmService.this, AlarmService_Service.class), 0);
// We want the alarm to go off 30 seconds from now.
long firstTime = SystemClock.elapsedRealtime();
// Schedule the alarm!
AlarmManager am = (AlarmManager)getSystemService(ALARM_SERVICE);
am.setRepeating(AlarmManager.ELAPSED_REALTIME_WAKEUP,firstTime, 30*1000, mAlarmSender);
```

Code d'Armor

Threads et AsyncTasks

- Ne JAMAIS bloquer le thread principal!
- Gare à l'ANR
- Perception de freeze à partir de 150ms

Pas de retour vers l'IHM, Thread

```
public void onClick(View v) {
    new Thread(new Runnable() {
        public void run() {
            final NetworkManager nm = NetworkManager.getInstance();
            nm.downloadInternet();
        }
    }).start();
}
```

Retour vers l'IHM, AsyncTask

```
public void onClick(View v) {
    new DownloadInternetTask().execute();
}

private class DownloadInternetTask extends AsyncTask<Void, Void, Void> {
    @Override
    protected Void doInBackground(Void... voids) {
        final NetworkManager nm = NetworkManager.getInstance();
        nm.downloadInternet();
        return null;
    }
    protected void onPostExecute() {
        tvResult.setText("I can haz internet!");
    }
}
```

Code d'Armor

Etre sur le devant de la scène

Widgets

- Mini-app incluse dans un conteneur
- AppWidgetHost le plus connu : la Homescreen
- Interaction avec un Service
- Affichage du contenu d'un Provider
- Lance des instances d'Activity

Widgets

- Classe AppWidgetProvider
  - Pilote les instances
  - Reçoit les Intents


```
<receiver android:name=".DofusAppWidgetProvider">
    <intent-filter>
        <action android:name="android.appwidget.action.APPWIDGET_UPDATE" />
    </intent-filter>
    <meta-data android:name="android.appwidget.provider"
               android:resource="@xml/dofus_appwidget_info" />
</receiver>
```

- Activity de config
- Fichier de layout
- Fichier XML AppWidgetProviderInfo

```
<?xml version="1.0" encoding="utf-8"?>
<appwidget-provider xmlns:android="http://schemas.android.com/apk/res/android"
                    android:minWidth="294dp"
                    android:minHeight="72dp"
                    android:updatePeriodMillis="86400000"
                    android:previewImage="@drawable/preview"
                    android:initialLayout="@layout/dofus_appwidget"
                    android:configure="org.marcpoppleton.dofus.activity.DofusAppWidgetConfigureActivity"
                    android:resizeMode="horizontal|vertical">
</appwidget-provider>
```

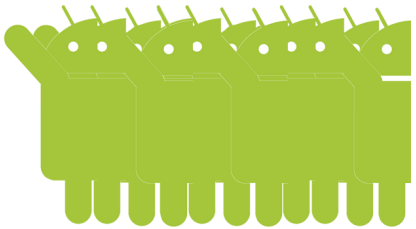
Code d'Armor

Widgets

- Mini-app donc mini-capacités
- Basées sur RemoteView:
  - 3 layouts possibles (Frame, Linear, Relative)
- Pas de OnClickListener:
  - PendingIntent sur des éléments de la Widget
- Doivent être rafraichies périodiquement:
  - Rôle de l'AppWidgetProvider
- Attention à l'empreinte mémoire!

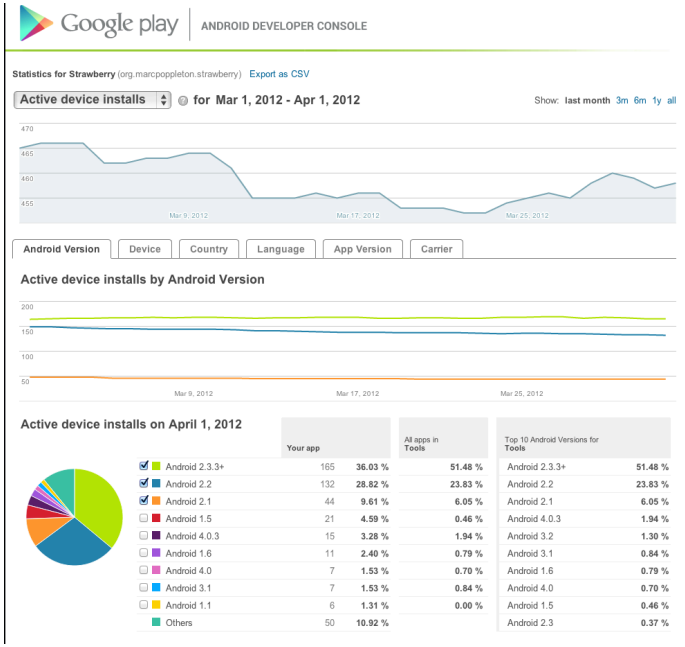
Code d'Armor

Distribuer ses apps



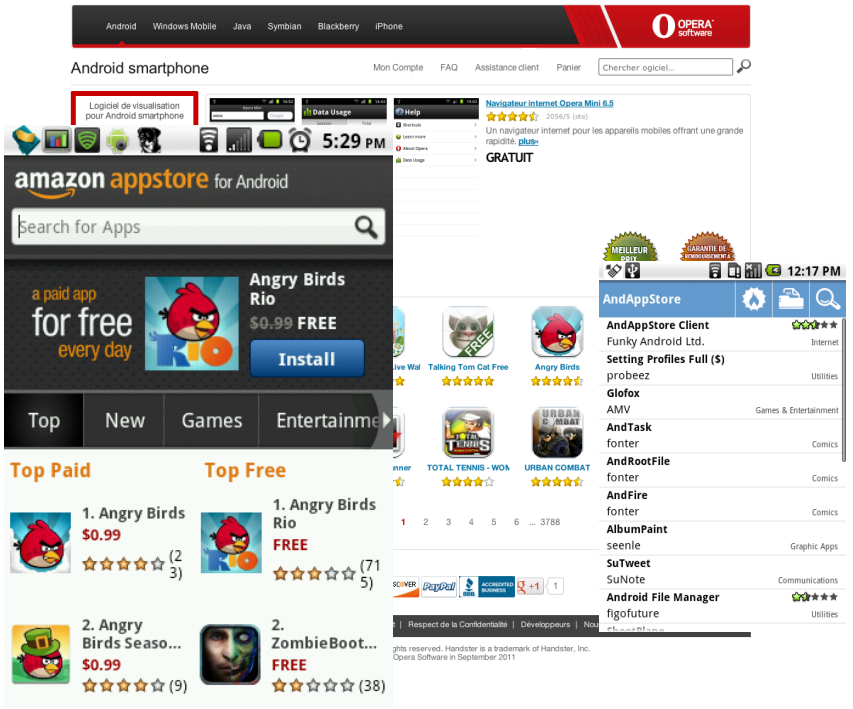
Come out and Play!

- Google play (feu Android Market)
- Inscription de 25$ (une fois, pour toute la vie)
- Publication d'applications
- Gestion du marketing
- Filtrage des cibles assuré
- Console colaborative
- In-app billing



Et les autres?

- Marketplace alternatifs:
  - Amazon App Store
  - Opera App Store
  - AndAppStore
- Distribution ad'hoc:
  - APK sur un serveur
  - APK de la main à la main
  - …



Q&R

http://developer.android.com
@marcpoppleton
marc@marc-poppleton.info