Développement d'applications mobiles : le système et la boîte à outils *Android*.

Renaud Blanch <blanch@imag.fr>

Université Joseph Fourier, Polytech'Grenoble & UFR IMAG

janvier 2010

Plan

Plan

Ressources en ligne

Le page consacrée au cours est ici :
<http://iihm.imag.fr/blanch/RICM4/MIM/>.

Agenda

- 2 février 2010, 13h30–15h, cours #0
  Introduction, le système *Android* (R. Blanch)
- 3 février 2010, 13h30–, TP #0
  Découverte de l'environnement, *Hello android*
  (A. Harbaoui)

Agenda

- 2 février 2010, 13h30–15h, cours #0
  Introduction, le système *Android* (R. Blanch)
- 3 février 2010, 13h30–, TP #0
  Découverte de l'environnement, *Hello android*
  (A. Harbaoui)
- 9 février 2010, 13h30–15h, cours #1
  Les outils de développement,
  la boîte à outils graphique (R. Blanch)
- 10 février 2010, 13h30–, TP #1
  Application interactive (A. Harbaoui)

Objectifs du cours

Après avoir suivi ce cours, vous saurez :

- **décrire** l'architecture du système d'exploitation *Android* ;
- **utiliser** la boîte à outils java d'*Android* pour créer des interfaces graphiques ;
- **développer** et **tester** une application en utilisant les outils de développement et le simulateur fournis par *Google* ; et
- **déployer** vos applications sur un terminal.

Un écosystème en pleine mutation

- **Janvier 2007**, **Apple** annonce l'iPhone avec la possibilité de développer des **applications web**.
- **Novembre 2007**, **Google** annonce la constitution de l'Open Handset Alliance et présente Android avec la possibilité de développer des **applications natives en Java**.
- **Février 2008**, **Apple** ouvre l'iPhone au développement natif et distribue son SDK.

Un écosystème en pleine mutation

Cette lutte annonce **la mort** promise **du modèle de développement** d'applications sur mobiles qui prévalait :
un univers **clos** avec des outils **payants**,
de qualité "professionnelle".

Un écosystème en pleine mutation

Cette lutte annonce **la mort** promise **du modèle de développement** d'applications sur mobiles qui prévalait : un univers **clos** avec des outils **payants**, de qualité "professionnelle".

Janvier 2010, **Symbian** annonce l'ouverture du code de son système d'exploitation.

La "ruée vers l'or"

- **Décembre 2009**, 20 000 applications dans l'Android Market.
- **Janvier 2010**, 140 000 applications dans l'App Store d'Apple, 3 millions de téléchargements revendiqués (la barre du million de téléchargements franchie le 23 avril 2009).

L'histoire d'*Android*

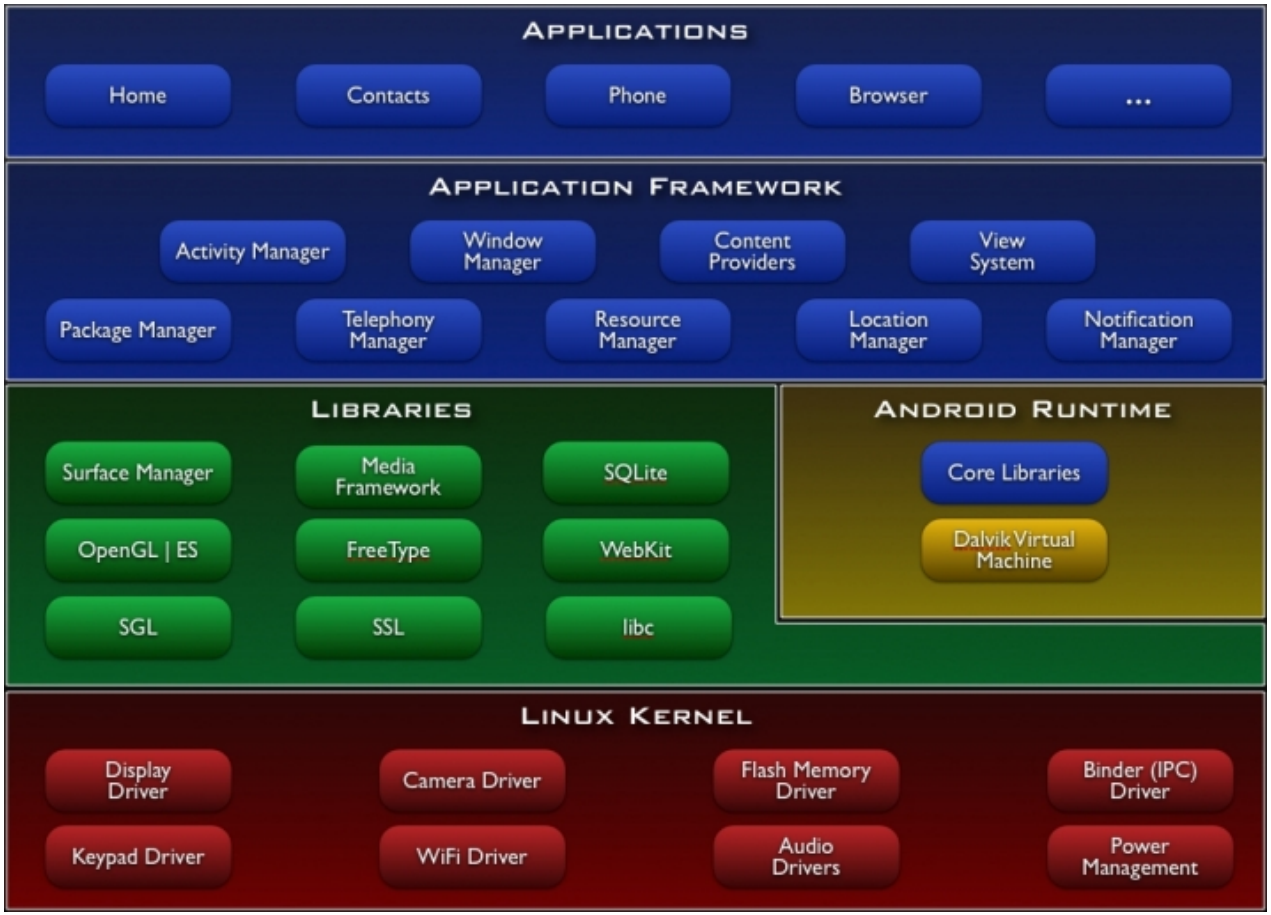
- **2005**, Google rachète Android, Inc., une start-up californienne qui réalise des logiciels pour mobiles.
- **Fin 2007**, Google dévoile l'Open Handest Alliance, un consortium de 47 membres (opérateurs, fabricants de terminaux, fabricants de processeurs, éditeurs de logiciels).
- **2008**, le code d'Android est ouvert (logiciel libre sous licence Apache), un SDK est distribué gratuitement.
- **2009**, "*dev phone*" 1.1, SDK 1.5 puis 1.6.
- **2010**, premier terminal commercialisé, le Nexus One du taïwanais HTC Corp.

Les terminaux

FIGURE : Terminaux fonctionnant sous *Android*, (gauche) *dev phone*, (droite) Nexus One.

Plan

Android vu du ciel



Un noyau Linux

Le système Android est basé sur un **noyau Linux** (kernel 2.6) qui sert d'**abstraction de la plate-forme matérielle**.

Un noyau Linux

Le système Android est basé sur un **noyau Linux** (kernel 2.6) qui sert d'**abstraction de la plate-forme matérielle**.

Linux fournit des **modèles éprouvés**

- de pilotes (*drivers*) ;
- de gestion de la mémoire ;
- de gestion des processus ;
- de sécurité ;
- de gestion du réseau ;
- ...

Des bibliothèques

Au dessus du noyau, un ensemble de **bibliothèques** sont **fournies**. Elles sont **natives** (i.e. développées en C ou en C++, et compilées dans le jeu d'instructions propre au processeur qui équipe chaque mobile) pour des **considérations de performance**.

Des bibliothèques “bas niveau”

libc

fournit des services minimaux d’**interface avec le système d’exploitation** (par exemple open, malloc, printf, etc.)
android n’utilise pas la *GNU C Library* (glibc) qui est pourtant éprouvée, mais une implémentation alternative (bionic), ce qui peut poser des problèmes de compatibilité.

Des bibliothèques “bas niveau”

libc

fournit des services minimaux d’**interface avec le système d’exploitation** (par exemple open, malloc, printf, etc.)
android n’utilise pas la *GNU C Library* (glibc) qui est pourtant éprouvée, mais une implémentation alternative (bionic), ce qui peut poser des problèmes de compatibilité.

SSL

gère la **sécurité des communications réseau** (couche transport).

Des bibliothèques graphiques (1/2)

Surface Manager

gère la **composition** à l'écran du **graphique** rendu par différents processus.

Des bibliothèques graphiques (1/2)

Surface Manager

gère la **composition** à l'écran du **graphique** rendu par différents processus.

SGL

permet l'affichage de **graphique 2D**, c'est la base graphique de la plupart des applications.

Des bibliothèques graphiques (1/2)

Surface Manager

gère la **composition** à l'écran du **graphique** rendu par différents processus.

SGL

permet l'affichage de **graphique 2D**, c'est la base graphique de la plupart des applications.

OpenGL|ES 1.x

permet l'affichage de **graphique 3D** (une implémentation logicielle est fournie, elle peut être accélérée par le matériel).

Des bibliothèques graphiques (2/2)

FreeType

permet le **rendu typographique** de qualité, en proposant en particulier des algorithmes d'*antialiasing* particulièrement adaptés aux spécificités du texte.

Des bibliothèques graphiques (2/2)

FreeType

permet le **rendu typographique** de qualité, en proposant en particulier des algorithmes d'*antialiasing* particulièrement adaptés aux spécificités du texte.

WebKit

permet le *layout et le rendu de pages HTML*, ainsi que l'**exécution** du langage **javascript** (WebKit est la branche du projet libre KHTML initiée est maintenue par Apple pour son navigateur Safari).

Encore des bibliothèques

Media Framework

fournit le support pour le décodage d'un ensemble de **formats audio et vidéo** (MPEG 4, H.264, MP3, AAC, etc.)

Encore des bibliothèques

Media Framework

fournit le support pour le décodage d'un ensemble de **formats audio et vidéo** (MPEG 4, H.264, MP3, AAC, etc.)

SQLite

fournit le service de **stockage de données** au sein d'une base de donnée.

Java *or not* Java

Les applications pour android sont écrites en **Java**.

Java *or not* Java

Les applications pour android sont écrites en **Java**.

Cependant, pour différentes raisons (performances, mais aussi licences), les mobiles android n'embarquent **pas de machine virtuelle Java, mais une machine virtuelle Dalvik**.

Java *or not* Java

Les applications pour android sont écrites en **Java**.

Cependant, pour différentes raisons (performances, mais aussi licences), les mobiles android n'embarquent **pas de machine virtuelle Java, mais une machine virtuelle Dalvik**.

En pratique, il n'y a pas de différence pour le programmeur, les outils fournis compilent le code source Java (.java) en *byte code* Java (.class/.jar), ce dernier étant ensuite transformé en *byte code* Dalvik (.dex).

Java, support à l'exécution

Avec la machine virtuelle est offerte une **bibliothèque standard** de classes **Java** :

- les conteneurs standard (*collections*) ;
- les entrées/sorties (*io*) ;
- xml, dom, sql, zip, http ;
- ...

Java, support à l'exécution

Avec la machine virtuelle est offerte une **bibliothèque standard** de classes **Java** :

- les conteneurs standard (*collections*) ;
- les entrées/sorties (*io*) ;
- xml, dom, sql, zip, http ;
- ...

mais **pas la boîte à outils de construction d'interfaces graphiques standard (AWT/Swing)**.

Services aux applications

Un ensemble de **services d'intérêt général** sont fournis aux applications sous la forme de classes Java, en particulier :

Services aux applications

Un ensemble de **services d'intérêt général** sont fournis aux applications sous la forme de classes Java, en particulier :

Window Manager

transpose les fonctionnalités du *Surface Manager* dans le langage Java.

Services aux applications

Un ensemble de **services d'intérêt général** sont fournis aux applications sous la forme de classes Java, en particulier :

Window Manager

transpose les fonctionnalités du *Surface Manager* dans le langage Java.

Package Manager

gère l'**installation des applications**, et centralise ce que chaque application est capable de faire.

Services aux applications (cont.)

Activity Manager

gère l'**enchaînement entre les applications** au cours de l'interaction avec l'utilisateur.

Services aux applications (cont.)

Activity Manager

gère l'**enchaînement entre les applications** au cours de l'interaction avec l'utilisateur.

Content Provider

permet aux applications de **partager des données** entre-elles (par exemple l'application "carnet d'adresses" peut mettre à disposition des autres applications ses informations (les numéros de téléphone pour l'application "téléphone" ou les adresses électroniques pour l'aplication "*email*".)

Services aux applications (cont.)

Ressource Manager

permet d'accéder aux **ressources externes** des applications (texte localisé, images composant l'interface, informations de placement, etc.)

Services aux applications (cont.)

Ressource Manager

permet d'accéder aux **ressources externes** des applications (texte localisé, images composant l'interface, informations de placement, etc.)

View System

propose une **boîte à outils de construction d'interfaces graphiques** avec les **widgets** standard et la possibilité de les assembler pour composer les interfaces des applications interactives.

Services aux applications (cont.)

Telephony Manager

fournit l'accès au **service de téléphonie** (c'est cette classe qui est utilisée pour l'application "téléphone" fournie en standard).

Services aux applications (cont.)

Telephony Manager

fournit l'accès au **service de téléphonie** (c'est cette classe qui est utilisée pour l'application "téléphone" fournie en standard).

Location Manager

fournit un service de **géolocalisation** du mobile qui permet de connaître sa position grâce au GPS s'il est présent, à l'information de l'antenne relais la plus proche ou encore de la connexion wifi.

Plan

Anatomie d'une application

Une application se décompose en :

Activity partie de l'**interface graphique** (typiquement une page) ;

Anatomie d'une application

Une application se décompose en :

Activity partie de l'**interface graphique** (typiquement une page) ;

IntentReceiver **répond à une notification** extérieure (en réveillant éventuellement le processus) ;

Anatomie d'une application

Une application se décompose en :

Activity partie de l'**interface graphique** (typiquement une page) ;

IntentReceiver **répond à une notification** extérieure (en réveillant éventuellement le processus) ;

Service **tâche de fond** sans interface graphique ;

Anatomie d'une application

Une application se décompose en :

Activity partie de l'**interface graphique** (typiquement une page) ;

IntentReceiver **répond à une notification** extérieure (en réveillant éventuellement le processus) ;

Service **tâche de fond** sans interface graphique ;

ContentProvider permet de **partager des données** entre applications.

Mashup d'applications

Le mécanisme d'*intents* permet de **réutiliser des applications**, sans présumer lesquelles exactement (par exemple, je demande au système de "sélectionner une photo" et il me lance l'application "galerie photo" par défaut choisie par l'utilisateur).

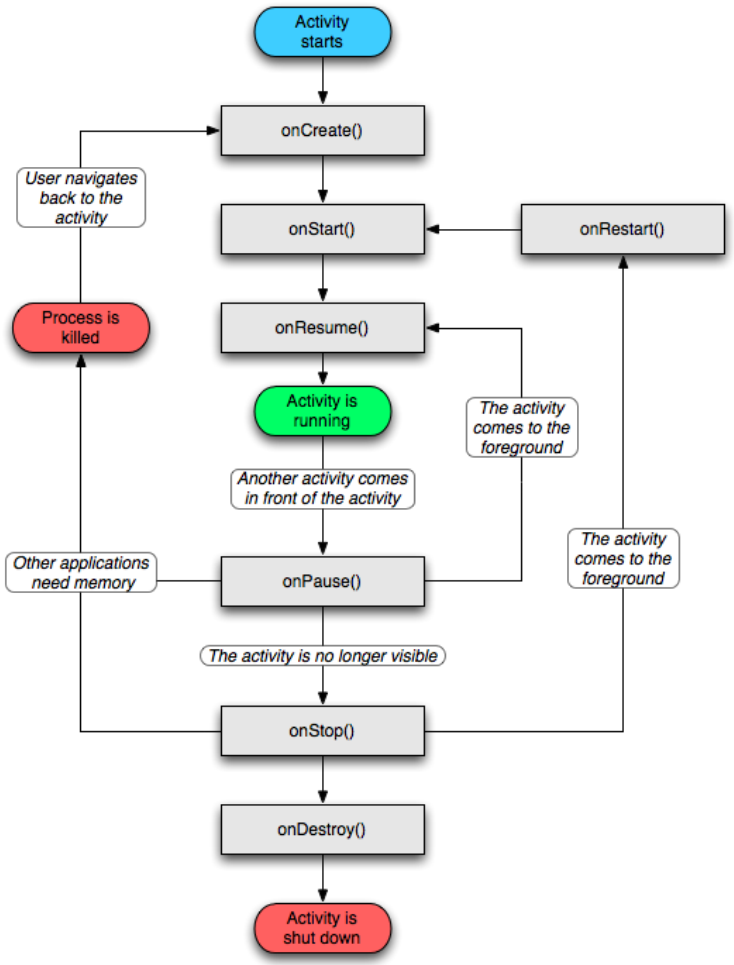
exemples

go home, show location, pick photo, edit contacts, send email, send SMS, show web page, etc.

référence

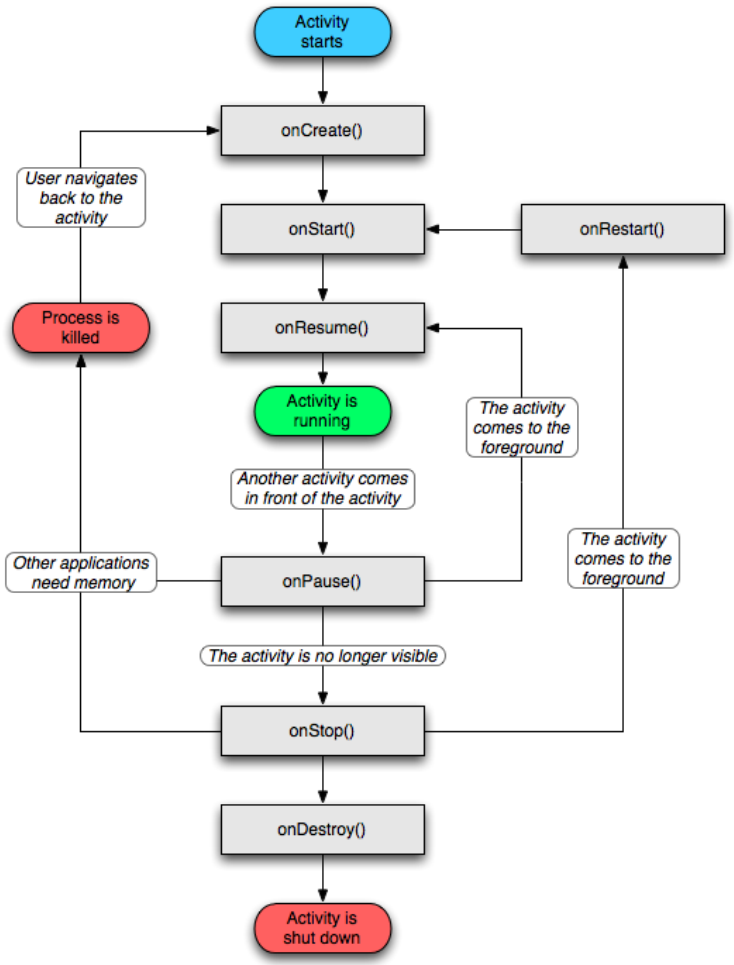
`<Context.startActivity(Intent intent)>`

Vie et mort d'une application



L'*Activity Manager* gère une **pile d'activités** qui peuvent appartenir à différentes applications au gré de l'interaction.

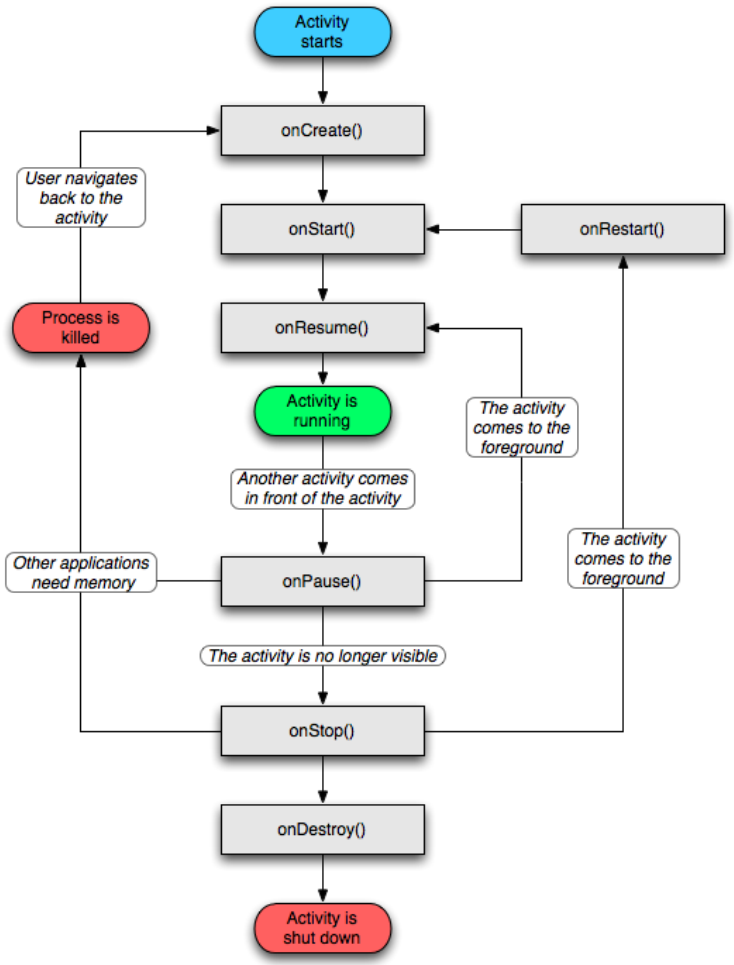
Vie et mort d'une application



L'*Activity Manager* gère une **pile d'activités** qui peuvent appartenir à différentes applications au gré de l'interaction.

En fonction des ressources nécessaires, une **activité** peut être **interrompue**, son **processus** peut même se voir **détruit**.

Vie et mort d'une application



L'*Activity Manager* gère une **pile d'activités** qui peuvent appartenir à différentes applications au gré de l'interaction.

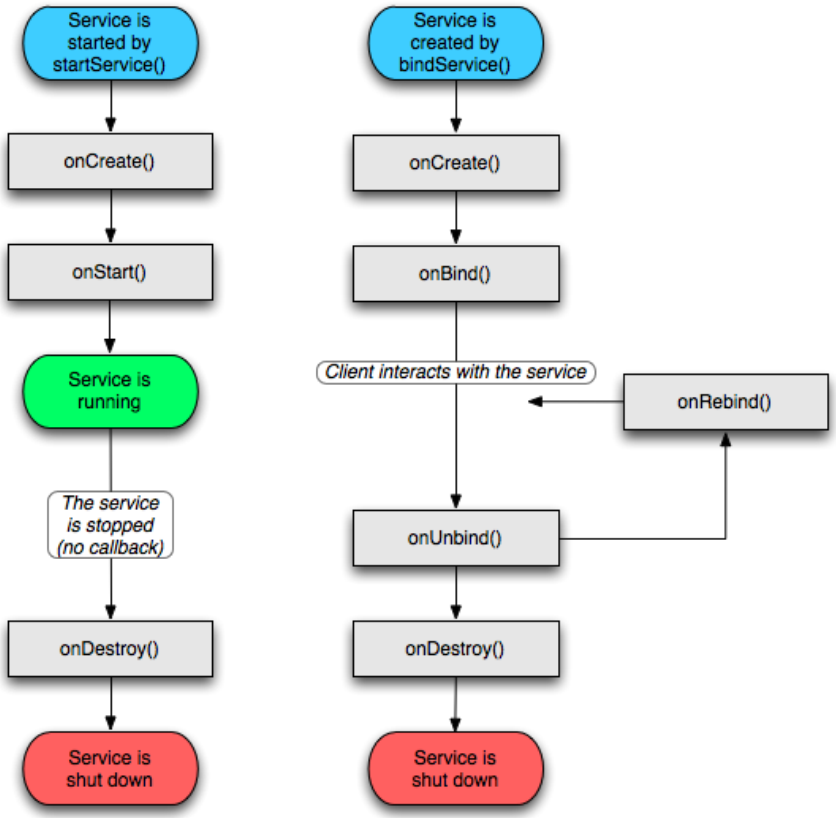
En fonction des ressources nécessaires, une **activité** peut être **interrompue**, son **processus** peut même se voir **détruit**.

Il est de la **responsabilité du programmeur** d'assurer la **restauration de l'état** de l'activité lorsque celle-ci est de retour sur le dessus de la pile.

Vie et mort d'une application (cont.)

Androidology - Part 2 of 3 - Application Lifecycle
<http://www.youtube.com/watch?v=fL6gSd4ugSI>

Vie et mort d'un service



Suivant leur nature, les **services peuvent** ou non **interagir** avec le monde extérieur par un mécanisme de **liaison à l'exécution** (*binding*) qui permet d'ouvrir un canal de communication entre le service et une autre application.

Description d'une application

Le fichier **AndroidManifest.xml**
contient les **métadonnées** décrivant l'application.

```
<?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"
                  . . .  >
            <intent-filter . . . >
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
            <intent-filter . . . >
                <action android:name="com.example.project.BOUNCE" />
                <data android:mimeType="image/jpeg" />
                <category android:name="android.intent.category.DEFAULT" />
            </intent-filter>
        </activity>
        . . .
    </application>
</manifest>
```

Widgets

Android fournit un ensemble de *widgets* standard.
Ils étendent tous la classe `android.view.View`.

Widgets

Android fournit un ensemble de *widgets* standard.
Ils étendent tous la classe `android.view.View`.

Ils peuvent être créés par du **code** :

```
Button button = new Button(context, attrs);
```

les **attributs** du boutons étant spécifiés à l'aide d'une configuration donnée en **XML**.

Widgets (cont.)

La manière standard de procéder est de **décrire l'interface graphique dans un fichier XML** :

```
<Button id="@+id/button"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/my_button_text"/>
```

Puis de **lier le code** à l'interface par le biais des **identifiants uniques** :

```
Button button = (Button) findViewById(R.id.button);
```

Widgets (cont.)

La **gestion des évènements** se fait à l'aide du mécanisme classique d'**écouteurs** d'évènements *(listeners)* :

```
button.setOnClickListener(new View.OnClickListener() {
    public void onClick(View v) {
        // Perform action on click
    }
});
```

Layout

Les *widgets* peuvent être **composés hiérarchiquement** en utilisant des **conteneurs**. Ils étendent tous la classe `android.view.ViewGroup`.

Layout

Les *widgets* peuvent être **composés hiérarchiquement** en utilisant des **conteneurs**. Ils étendent tous la classe `android.view.ViewGroup`.

La plupart *(FrameLayout, LinearLayout, RelativeLayout, TableLayout)* ont essentiellement pour fonction le **placement** *(layout)* de leur fils.
D'autres *(Gallery, GridView, ListView, ScrollView, Spinner, SurfaceView, TabHost, ViewFlipper, ViewSwitcher)* fournissent en plus des **interactions spécifiques**.

Layout

Les *widgets* peuvent être **composés hiérarchiquement** en utilisant des **conteneurs**. Ils étendent tous la classe `android.view.ViewGroup`.

La plupart *(FrameLayout, LinearLayout, RelativeLayout, TableLayout)* ont essentiellement pour fonction le **placement** *(layout)* de leur fils.
D'autres *(Gallery, GridView, ListView, ScrollView, Spinner, SurfaceView, TabHost, ViewFlipper, ViewSwitcher)* fournissent en plus des **interactions spécifiques**.

Comme pour le reste de l'interface, ils peuvent être **instanciés par du code** mais la manière standard de procéder et de **décrire l'interface par un fichier XML**.

Layout (cont.)

exemple de LinearLayout

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
              android:layout_width="fill_parent"
              android:layout_height="fill_parent"
              android:orientation="vertical" >
    <TextView android:id="@+id/text"
              android:layout_width="wrap_content"
              android:layout_height="wrap_content"
              android:text="Hello, I am a TextView" />
    <Button android:id="@+id/button"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Hello, I am a Button" />
</LinearLayout>
```

Layout (cont.)

exemple (simplifié) de RelativeLayout

```
<RelativeLayout layout_width="fill_parent" layout_height="wrap_content"
                background="@drawable/blue" padding="10px" >

    <TextView id="@+id/label" text="Type here:"
              layout_width="fill_parent" layout_height="wrap_content" />

    <EditText id="@+id/entry" background="@drawable/editbox_background"
              layout_width="fill_parent" layout_height="wrap_content"
              layout_below="@id/label" />

    <Button id="@+id/ok" text="OK" layout_marginLeft="10px"
            layout_width="wrap_content" layout_height="wrap_content"
            layout_below="@id/entry"
            layout_alignParentRight="true" />

    <Button text="Cancel"
            layout_width="wrap_content" layout_height="wrap_content"
            layout_toLeftOf="@id/ok"
            layout_alignTop="@id/ok" />
</RelativeLayout>
```

Ressources

Les fichiers XML de description des interfaces sont des **ressources** qui sont compilées avec l'applications. Ce mécanisme permet d'inclure de la même manière des données dans les applications :

- des **chaînes de caractère** (à placer dans res/values/strings.xml) ;
- des **couleurs** (res/values/colors.xml) ;
- des **layouts** (res/layout/toto.xml) ;
- des **images**, des **animations**, des **menus**, etc.

Ressources (cont.)

Chaque ressource est **identifiée**,
par un nom ou un identifiant unique :

```
<resources>
    <string name="welcome_message">Welcome!</string>
</resources>
```

Ressources (cont.)

Chaque ressource est **identifiée**,
par un nom ou un identifiant unique :

```
<resources>
   <string name="welcome_message">Welcome!</string>
</resources>
```

et peut être **référencée** par d'autres ressources :

```
<TextView android:layout_width="fill_parent"
          android:layout_height="wrap_content"
          android:textAlign="center"
          android:text="@string/welcome_message"/>
```

Ressources (cont.)

Android génère lors de la compilation de l'application une classe particulière nommée **R** qui permet d'**accéder aux ressources** depuis le code :

```
// Assign a styled string resource to a TextView
// on the current screen.
CharSequence str = getString(R.string.welcome_message);
TextView tv = (TextView)findViewByID(R.id.text);
tv.setText(str);
```

Plan

Outils du *Software Development Kit*

Un certain nombre d'outils sont fournis dans le *SDK* proposé.

Outils du *Software Development Kit*

Un certain nombre d'outils sont fournis dans le *SDK* proposé.

Android Emulator
émulateur basé sur QEMU permettant de tester ses applications sur station de travail.

Outils du *Software Development Kit*

Un certain nombre d'outils sont fournis dans le *SDK* proposé.

Android Emulator
émulateur basé sur QEMU permettant de tester ses applications sur station de travail.

Android Virtual Devices (AVDs)
outil de configuration permettant de créer des dispositifs virtuels tournant dans l'émulateur.

Outils du *Software Development Kit* (cont.)

android
un script qui permet d'automatiser AVD et de générer des fichiers dirigeant la compilation *(Ant build files)* des applications.

Outils du *Software Development Kit* (cont.)

android
un script qui permet d'automatiser AVD et de générer des fichiers dirigeant la compilation *(Ant build files)* des applications.

dx
compilateur transformant les classes java (.class) en language de la machine virtuelle Dalvik (.dex).

Outils du *Software Development Kit* (cont.)

android
un script qui permet d'automatiser AVD et de générer des fichiers dirigeant la compilation *(Ant build files)* des applications.

dx
compilateur transformant les classes java (.class) en language de la machine virtuelle Dalvik (.dex).

Android Debug Bridge (adb)
outil pour débuguer les applications sur émulateur ou sur cible.

Outils du *Software Development Kit* (cont.)

Android Asset Packaging Tool (aapt)

outil permettant de créer les paquets distribuables contenant les applications (.apk).

Outils du *Software Development Kit* (cont.)

Android Asset Packaging Tool (aapt)

outil permettant de créer les paquets distribuables contenant les applications (.apk).

zipalign

outil pour optimiser les paquet d'application (.apk).

Outils du *Software Development Kit* (cont.)

Android Asset Packaging Tool (aapt)

outil permettant de créer les paquets distribuables contenant les applications (.apk).

zipalign

outil pour optimiser les paquet d'application (.apk).

Hierarchy Viewer

permet de débuguer les layouts.

Outils du *Software Development Kit* (cont.)

Android Asset Packaging Tool (aapt)

outil permettant de créer les paquets distribuables contenant les applications (.apk).

zipalign

outil pour optimiser les paquet d'application (.apk).

Hierarchy Viewer

permet de débuguer les layouts.

layoutopt

outil pour optimiser les layouts.

Intégration dans *Eclipse*

La plupart des outils sont intégrés à l'*IDE Eclipse* grâce au composant **Android Development Tools** (ADT).

Intégration dans *Eclipse*

La plupart des outils sont intégrés à l'*IDE Eclipse* grâce au composant **Android Development Tools** (ADT).

Celui-ci offre :

- un **assistant** pour créer des **squelettes d'applications** ;
- une automatisation de la **compilation**, du **déploiement** vers simulateur ou cible, et du **débuguage** ;
- un **éditeur** pour les divers **fichiers de ressources** ;
- les fonctionnalités pour **packager les applications** ;
- ...