LA VIE D'UNE APPLICATION

UVCI

11 FÉVRIER 2019

Table des matières

Objectifs

A la fin de cette leçon, l'étudiant sera capable de :

- *Décrire* les composantes d'une application ;
- *Décrire* le fonctionnement des activités.

Composantes d'une application

Une application ne se résume pas à un ou plusieurs programmes utilisés par des clients. Un logiciel naît, vit et meurt.

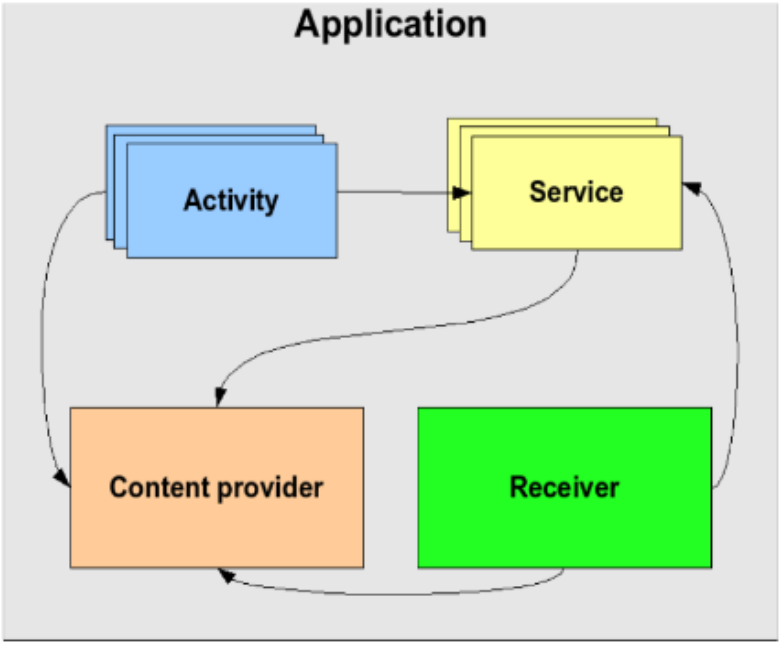
C'est pourquoi on parle de cycle de vie d'un logiciel. Le cycle de vie d'une application est constitué de toutes les étapes jalonnant son parcours, depuis l'idée de départ jusqu'à l'abandon du logiciel par ses utilisateurs, en passant par la conception, la réalisation et la maintenance.

1. Composantes d'une application android

Les composants d'une application sont liés par le fichier manifeste (`AndroidManifest.xml`) .Ce fichier permet de décrire les interactions entre les composants.

Une application est composée d'une ou plusieurs *activités*. Chacune gère un écran d'interaction avec l'utilisateur et est définie par une classe Java.Une application complexe peut aussi contenir :

- *des services :* ce sont des processus qui tournent en arrière-plan
- *des fournisseurs de contenu* : ils représentent une sorte de base de données
- *des récepteurs d'annonces* : pour gérer des événements globaux envoyés par le système à toutes les applications.



Les composants d'une application

Déclaration d'une application

Le fichier *AndroidManifest.xml* déclare les éléments d'une application, avec un '.' devant le nom de classe

```
<?xml version="1.0" encoding="utf-8"?>
<manifest ... >
    <application android:icon="@drawable/app_icon.png" ...>
        <activity android:name=".MainActivity"
                  ... />
        <activity android:name=".EditActivity"
                  ... />
        ...
    </application>
</manifest>
```

`<application>` est le seul élément sous la racine `<manifest>` et ses filles sont des `<activity>`.

Complément : Sécurité des applications

Chaque application est associée à un UID (compte utilisateur Unix) unique dans le système. Ce compte les protège les unes des autres. Cet UID peut être défini dans le fichier *AndroidManifest.xml*

```
<?xml version="1.0" encoding="utf-8"?>
<manifest ...
        android:sharedUserId="fr.iutlan.demos">
    ...
</manifest>
```

Définir l'attribut *android : sharedUserId* avec une chaîne identique à une autre application, et signer les deux applications avec le même certificat, permet à l'une d'accéder à l'autre.

Autorisations d'une application

Une application doit déclarer les autorisations dont elle a besoin : accès à internet, caméra, carnet d'adresse, GPS, etc.Cela se fait en rajoutant des éléments dans le manifeste :

```
<manifest ... >
    <uses-permission
        android:name="android.permission.INTERNET" />
    ...
</manifest>
```

Remarque

NB: les premières activités que vous créerez n'auront besoin d'aucune permission.

Démarrage d'une application

L'une des activités est marquée comme démarrable de l'extérieur :

```
<activity android:name=".MainActivity" ...>
  <intent-filter>
    <action android:name="android.intent.action.MAIN"/>
    <category android:name="android.intent.category.LAUNCHER"/>
  </intent-filter>
</activity>
```

Un `<intent-filter>` déclare les conditions de démarrage d'une activité, ici il dit que c'est l'activitéprincipale.

Démarrage d'une activité et Intents

Les activités sont démarrées à l'aide d'Intents. Un Intent contient une demande destinée à une activité, par exemple, composer un numéro de téléphone ou lancer l'application.

- *action :* spécifie ce que l'Intent demande. Il y en a de très nombreuses
  - *VIEW* pour afficher quelque chose, *EDIT* pour modifier une information,*SEARCH*. . .
- *données :* selon l'action, ça peut être un numéro de téléphone, l'identifiant d'une information. . .
- *catégorie :* information supplémentaire sur l'action, par exemple,...*LAUNCHER* pour lancer une application.

Une application a la possibilité de lancer certaines activités d'une autre application, celles qui ont un intent-filter

Lancement d'une activité par programme

Soit une application contenant deux activités : *Activ1* et *Activ2*. La première lance la seconde par :

```
Intent intent = new Intent(this, Activ2.class);
startActivity(intent);
```

Complément : Application Android

Une activité = un programme + une interface

Un service = un programme sans interface

Une application =

- Une activité principale Eventuellement
- une ou plusieurs activités secondaires Eventuellement
- un ou plusieurs services Eventuellement
- un ou plusieurs écouteurs d'intentions diffusées Eventuellement
- un ou plusieurs fournisseurs de contenu

2. Exercice

Exercice

Choisissez la bonne réponse

Une application android est constituée d'au moins une activité ?

- ○ Vrai
- ○ Faux

Exercice

Une intention est un message système qu'on peut qualifier d' évènement ?

- ○ Vrai
- ○ Faux

Exercice

Un fournisseur de contenu / Content Provider permet :

- ☐ de gérer et de partager des informations
- ☐ d'accéder à des données au sein d'une application et entre applications
- ☐ des interactions entre applications et avec le système
- ☐ de faire passer des messages contenant de l'information entre composants principaux

3. le fonctionnement des activités.

Il est indispensable de connaître le cycle de vie d'une application afin de savoir les points d'entrée et de réaction d'une application

En fonction du type d'application le cycle de vie ne sera pas le même .

On distingue deux (02) cycles différents :

- Cycle de vie d'une Activité
- Cycle de vie d'un Service

3.1. Activités

Une activité (*Activity*) représente le bloc de base d'une application.

- Elle correspond à la partie présentation de l'application et fonctionne par le biais de *vues* qui affichent des interfaces graphiques.
- Elle répond aux actions utilisateur
- Réagit à son cycle de vie.

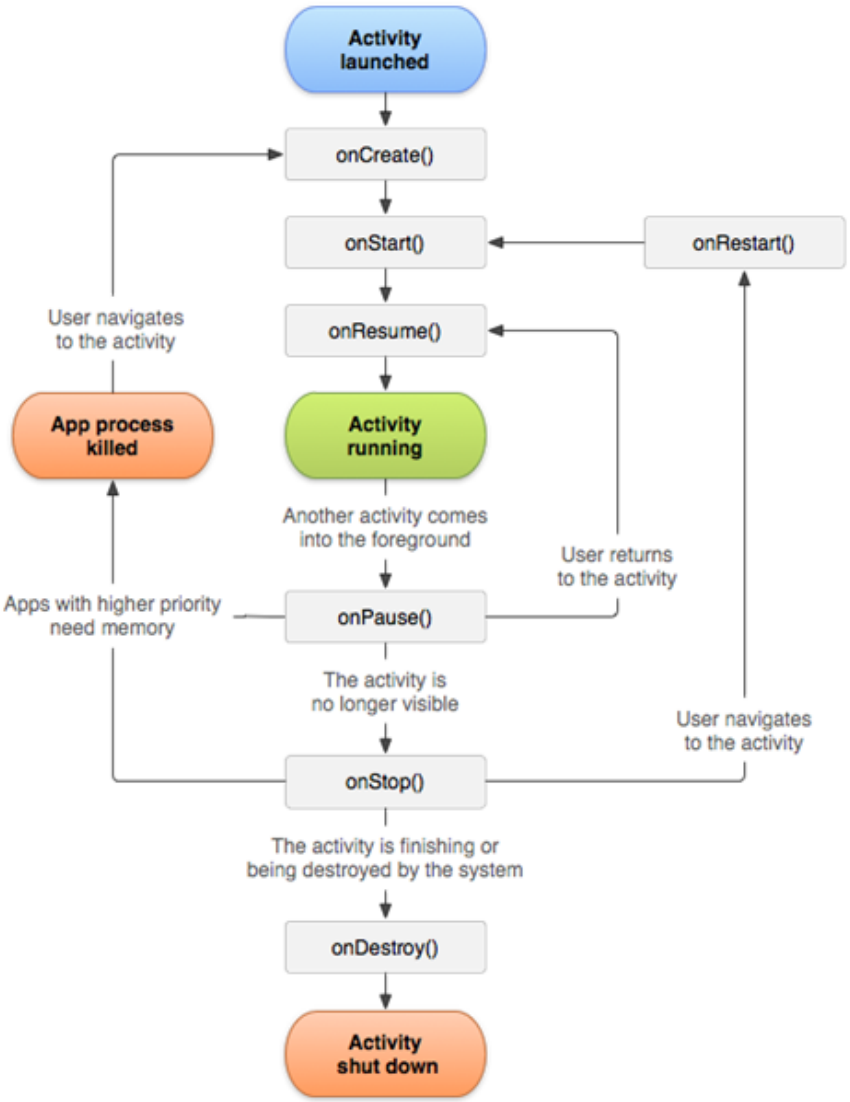
Par extension, on dit que l'activité est la forme disponible sur l'écran car elle offre à l'utilisateur l'interface (écran) pour interagir avec l'application.

- Chaque écran de l'application représente donc une activité.
- Une application Android est composée d'une ou de plusieurs activités (un jeu sur plusieurs écrans).
- Une seule activité est lancée à la fois.

Dans un cycle de vie d'une activité on compte *7 états*.

Ces états sont déclenchés soit par le système soit par l'utilisateur.

Il peut être très utile pour une application de pouvoir réagir à ces états.



Fondamental

Les *7 états* du cycle de vie sont accessibles grâce à *7 callback* définis dans la classe Activity qui sont :

1. *onCreate()*
2. *onStart()*
3. *onResume()*
4. *onRestart()*
5. *onPause()*
6. *onStop()*
7. *onDestroy()*

Chaque classe d'écrivant une activité doit hériter de la classe Activity ou de toutes autres classes héritant elles-mêmes de cette classe.

Dans « Java », chaque activité est une classe qui étend par héritage la classe « Activity ».

Lors du démarrage d'une activité, 3 méthodes sont appelées automatiquement : « onCreate », « onStart » et « onResume ».

Fondamental

En général, le mouvement dans le cycle de vie d'une activité ressemble à ceci:

Méthodes	Description	Killable	Suivant
onCreate()	Appelé lors de la création de l'activité. C'est ici que vous devez effectuer toute votre configuration statique normale: créer des vues, lier des données à des listes, etc. Cette méthode vous fournit également un Bundle contenant l'état précédemment gelé de l'activité, le cas échéant. Toujours suivi par *onStart()*.	Non	*onStart()*
onRestart()	Appelé après l'arrêt de votre activité, avant son redémarrage. Toujours suivi par *onStart()*	Non	*onStart()*

onStart()	Appelé lorsque l'activité devient visible pour l'utilisateur. Suivi de *onResume()* si l'activité passe au premier plan ou *onStop()* si elle devient masquée.	Non	*onResume()* ou *onStop()*
onResume()	Appelé lorsque l'activité commencera à interagir avec l'utilisateur. À ce stade, votre activité se situe au sommet de la pile d'activités, avec la saisie de l'utilisateur. Toujours suivi par *onPause()*.	Non	*onPause()*
onPause()	Appelé lorsque le système est sur le point de commencer une activité précédente. Ceci est généralement utilisé pour valider des modifications non sauvegardées dans des données persistantes, arrêter des animations et d'autres éléments susceptibles de consommer de la CPU, etc. Les implémentations de cette méthode doivent être très rapides car l'activité suivante ne sera reprise qu'après le retour de cette méthode. Suivi par onResume()si l'activité retourne au début, ou onStop()si elle devient invisible pour l'utilisateur.	Pré-Build.VERSION_CODES. HONEYCOMB	*onResume()* ou *onStop()*

onStop()	Appelé lorsque l'activité n'est plus visible par l'utilisateur, car une autre activité a été reprise et couvre celle-ci. Cela peut se produire soit parce qu'une nouvelle activité est en cours de démarrage, qu'une activité existante est présentée devant celle-ci ou que celle-ci est en cours de destruction. Suivi par *onRestart()* si cette activité revient pour interagir avec l'utilisateur, ou *onDestroy()* si cette activité s'en va.	Oui	*onRestart()* ou *onDestroy()*
onDestroy()	Le dernier appel que vous recevez avant que votre activité soit détruite. Cela peut se produire soit parce que l'activité est terminée (quelqu'un a appelé *finish()* dessus), soit parce que le système est en train de détruire temporairement cette instance de l'activité pour économiser de l'espace. Vous pouvez faire la distinction entre ces deux scénarios avec la *isFinishing()* méthode.	Oui	Rien

Remarque

la colonne "*Killable*" dans le tableau ci-dessus - pour les méthodes marquées comme pouvant être tuées, après le retour de cette méthode, le processus hébergeant l'activité peut être tué par le système à tout moment sans qu'une autre ligne de son code ne soit exécutée.

Exemple : Terminaison d'une activité

Voici la prise en compte de la terminaison définitive d'une activité, avec la fermeture d'une base de données :

```
@Override
public void onDestroy() {
    super.onDestroy();

    // fermer la base
    db.close();
}
```

Exemple : Pause d'une activité

Cela arrive quand une nouvelle activité passe devant, exemple : un appel téléphonique. Il faut libérer les ressources qui consomment de l'énergie (animations, GPS. . .).

```
@Override public void onPause() {
    super.onPause();
    // arrêter les animations sur l'écran
    ...
}
@Override public void onResume() {
    super.onResume();
    // démarrer les animations
    ...
}
```

Généralités sur le cycle de vie d'une activité

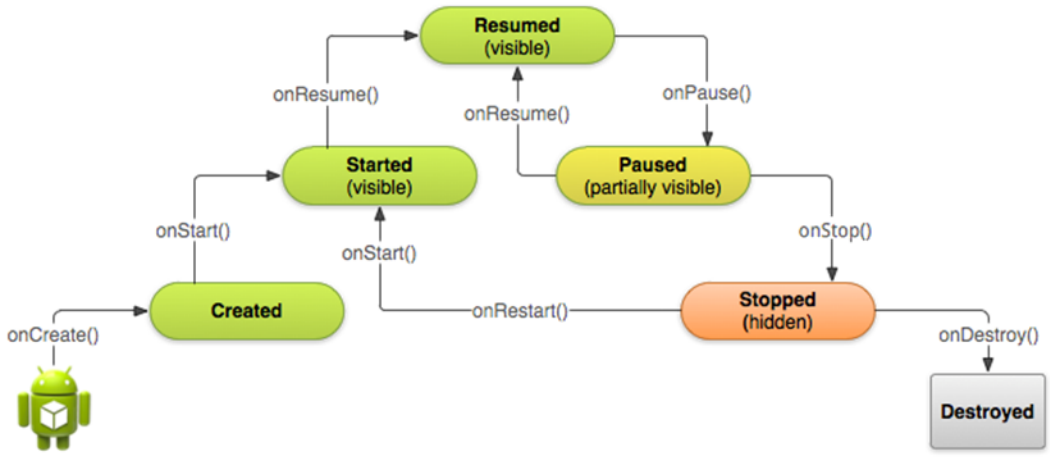
- Aucune méthode *Main* dans un programme Android
- Android exécute le code d'une activité en appelant des *callbacks*
- Ces *callbacks* correspondent aux phrases de la vie d'une activité
- Il n'est pas nécessaire d'implémenter toutes les *callbacks*

Exemple

Activité : code source

```java
package com.ieconcept.firstapp;

import android.app.Activity;
import android.os.Bundle;

public class MainActivity extends Activity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }

    @Override
    protected void onResume() {
        super.onResume();
    }

    @Override
    protected void onDestroy() {
        super.onDestroy();
    }
}
```

Vision simplifiée du cycle de vie d'une activité



Les états d'une activité

Les trois états durables d'une activité :

- *Resumed* : L'activité est au premier plan et l'utilisateur peut interagir avec elle. On dit aussi qu'elle est en train d'être exécutée.
- *Paused* : L'activité est partiellement recouverte par une autre activité qui se trouve au premier plan. L'activité en pause ne peut pas recevoir d'action de l'utilisateur.
- *Stopped* : L'activité est totalement cachée et ne peut plus exécuter de code. En revanche, toutes ses informations sont conservées.

Les deux états transitoires d'une activité :

- *Created* : L'activité vient d'être créée
- *Started* : L'activité vient de devenir visible.

3.2. Services

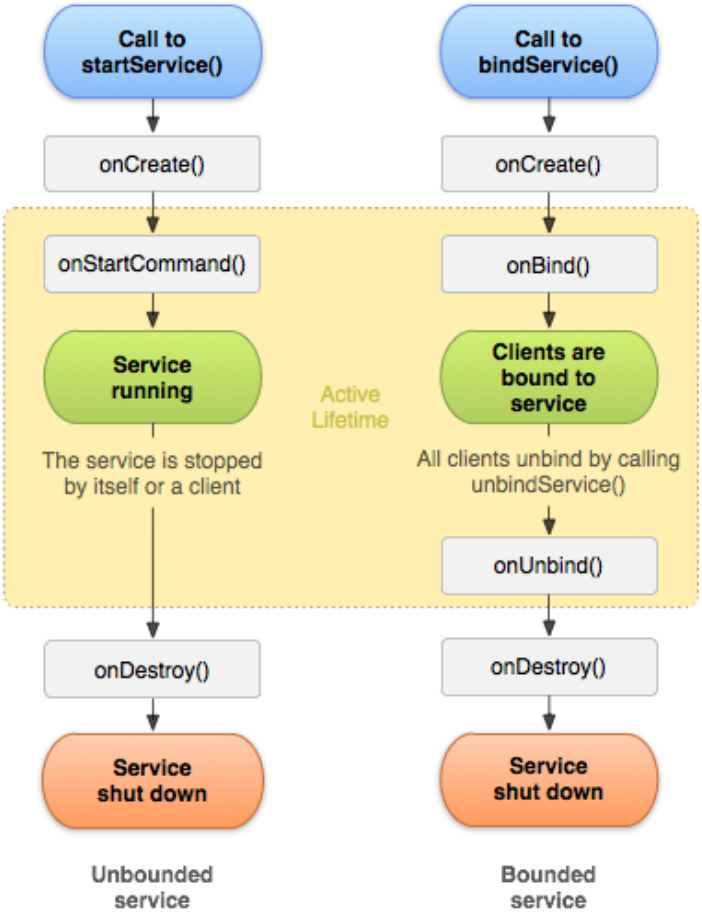
Un service est un composant qui fonctionne en tâche de fond, de manière invisible.

- Il ne nécessite pas d'interface utilisateur
- Ses principales utilisations sont la mise à jour de sources de données ainsi que d'activités visibles et le déclenchement de notifications
- Possède un cycle de vie différent d'une activité

Cycle de vie d'un Service

Dans un cycle de vie d'un service on compte 5 états.

1. *onCreate()*
2. *onStartCommand()*
3. *onBind()*
4. *onUnbind()*
5. *onDestroy()*



Cycle de vie d'un Service

Généralités sur le cycle de vie d'un service

- Il y a moins de *callbacks* pour un service qu'une activité
- Il y a deux types de service
  Les services locaux, ils sont lancés depuis le même processus.
  Les services distants, ils sont lancés depuis un processus différent.

Les états d'un service

Il n'y a qu'un état durable pour un service :

- Service running pour un service local, le service est en cours d'exécution.
- Client are bound to service pour un service distant, le service est en cours d'exécution tant qu'il y a des clients connectés

Exemple

Service : code source

```java
package com.ieconcept.firstapp;

import android.app.Service;
import android.content.Intent;
import android.os.IBinder;
import android.support.annotation.Nullable;

public class TestService extends Service {

    @Override
    public void onCreate() {
        super.onCreate();
    }

    @Override
    public void onDestroy() {
        super.onDestroy();
    }

    @Nullable
    @Override
    public IBinder onBind(Intent intent) {
        return null;
    }
}
```

3.3. Généralités sur les cycles de vie

Fondamental

- Il est important d'implémenter les *callbacks* afin que votre application fonctionne correctement :
  - Réception d'un appel et basculement sur une autre application.
  - Ne pas consommer trop de ressources système.
  - Ne pas avoir de problème lors de la création / restauration de l'application par le système (par exemple lors d'une rotation de l'écran).
- Il est donc important d'avoir fait des tests dans les situations évoquées ci-dessus, avant de distribuer une application.

4. Exercice

Exercice

un cycle de vie d'une activité on compte :

- 7 états
- 5 états
- 10 états
- 3 états

Exercice

l'état *onCreate()* est appelé lorsque le système est sur le point de commencer une activité précédente

- Faux
- Vrai

Exercice

..... est Appelé après l'arrêt de votre activité, avant son redémarrage.

- onRestart()
- onPause()
- onBind()
- onStop()

Exercice

un cycle de vie d'un service compte .....

- 5 états
- 6 états
- 7 états
- 3 états