Introduction au développement d'applications mobiles avec Androïd : Activités & Intents

IGR 201

James EAGAN

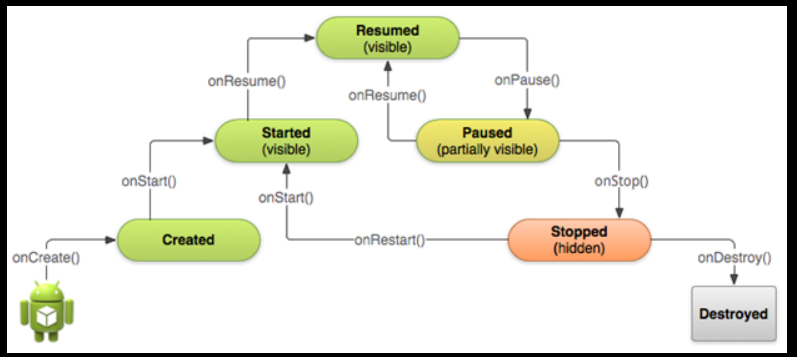
Agenda

- Introduction
- Cycle de vie d'une appli Androïd
- Gestion d'état entres instances
- Communication entre applis

Activity

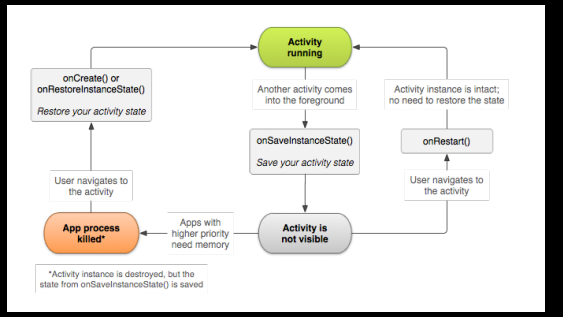
- Une activité est l'unité d'une application
- On peut le considérer un peu comme une fenêtre

Cycle de vie d'une Activity



Persistence d'état entre vies

- Même quand une `Activity` est tuée, son état peut être stocké dans un `Bundle`.
- Un `Bundle` est un paquet qui peut stocker des objets `Parcelable` (un peu comme `Serializable`).
- L'implementation par défaut appelle `onSaveInstanceState()` sur les éléments de l'IU.

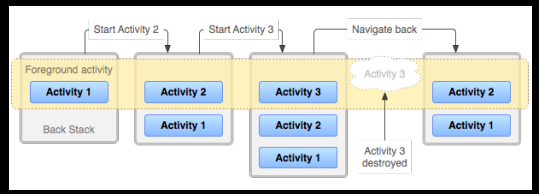


Transfert d'état entre activités

- Généralement, on sauvegarde l'état à chaque `onPause()`, car l'activité pourrait être tuée à n'importe quel moment après. Si l'activité en lance une autre, Androïd garantie que `onPause()` sera appelée avant les méthodes de la nouvelle `Activity` : `onCreate()`, `onStart()`, et `onResume()`.

Gestion du bouton ↩

- Androïd maintient une pile de toute tâche (avec une interface cohérente) pour le bouton retour-en-arrière.
- Parfois, il est souhaitable de modifier ce comportement :
  - Activités instanciées plusieurs fois ou pas du tout
  - Lancer une nouvelle activité implique (ou pas) lancer une nouvelle application



Intent : communication entre composants

- Une `Intent`ion exprime une action et ses paramètres :
  - Une `Intent`ion explicite cible un composant connu avec son package et nom de classe.
  - Une `Intent`ion implicite décrit une opération à faire, mais pas qui doit la faire.

Quand utiliser les deux

- À l'intérieur d'une appli, on utilise généralement des `Intent`s explicites pour naviguer entre `Activity`s.
- Pour scanner un code QR, on utilise généralement des `Intent`s implicites pour trouver un service installé dans le système.
- Pour partager quelque chose, on utiliser un chooser qui liste les composants capable de le partager (e.g. mail, sms, twitter, …)
- Il pourrait y en avoir plusieurs : pour trouver un chooser, on utiliser une `Intent` implicite en donnant l'`Intent` de partage comme paramètre.

Intent

Une `Intent` a plusieurs attributs :

Nom utilisé pour des `Intent`s explicites

Data un URI (si besoin) et un type MIME (e.g. text/plain)

Catégorie la catégorie des genres de composants qui pourraient gérer cette action (e.g. CATEGORY_LAUNCHER pour des activités qui peuvent être lancées depuis l'écran d'accueil).

Extras Paramètres supplémentaires (clé, valeur)

Drapeaux infos supplémentaires (e.g. s'il faut lancer une nouvelle tâche).

Exemple d'une Intent explicite

- Appel :

```
final Intent editIntent = new Intent(this, RecipeEditor.class);
editIntent.putExtra(EXTRA_RECIPE_ID, recipe.getId());
                    startActivity(editIntent);
```

- Définition :

```
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    final int recipe = getIntent().getIntExtra(EXTRA_RECIPE_ID, 0);
    // ...
```


Exemple d'une Intent implicite avec Chooser

```
// Créer une Intent implicite pour partager du texte
final Intent intent = (new Intent(Intent.ACTION_SEND))
        .setType("text/plain")
        .putExtra(Intent.EXTRA_TEXT, recipeText.toString());

// Lancer un chooser pour laisser choisir l'application à utiliser
// pour le partage.  Intent.createChooser() est une méthode statique
// qui construit une Intent implicite de type ACTION_CHOOSER.
final String title = res.getText(R.string.recipe_share);
startActivity(Intent.createChooser(intent, title));
```

Filtres d'Intent

- Pour recevoir une Intent, elle doit être déclarée dans le `AndroidManifest.xml`

```
<activity
    android:name=".RecipeManagerActivity"
    android:label="@string/app_name">
    <intent-filter>
        <action android:name="android.intent.action.MAIN"/>
        <category android:name="android.intent.category.LAUNCHER"/>
    </intent-filter>
</activity>
```

[https://developer.android.com/guide/components/intents-filters.html]

Un exemple plus complexe

```
<intent-filter>
    <action android:name="android.intent.action.VIEW"/>
    <category android:name="android.intent.category.DEFAULT"/>
    <data android:mimeType="text/xml"/>
    <data android:mimeType="application/xml"/>
    <data android:mimeType="application/zip"/>
    <data android:mimeType="application/x-compressed"/>
    <data android:mimeType="application/x-zip-compressed"/>
    <data android:mimeType="application/x-zip"/>
    <data android:mimeType="application/octet-stream"/>
    <data android:pathPattern=".*\\.recipe"/>
    <data android:pathPattern=".*\\.zip"/>
</intent-filter>
```


Recevoir le résultat d'une `Activity`

- Parfois, on veut lancer une `Activity` et récupérer le résultat (e.g. Scanner un flashcode)
  - Pour lancer l'Activity : `startActivityForResult()` avec un code de requête
  - Dans l'Activity lancé : `setResult()`
  - On reçoit le résultat dans la méthode `onActivityResult()` (avec le code de requête utilisé dans `startActivityForResult()`)

developer.android.com

[Transparents adaptés de ceux de Samuel Tardieu]