

Vie des activités

Chantal Keller

10 septembre 2015

Plan

- 1 Divers
- 2 Plusieurs activités
- 3 Cycle de vie d'une activité
- 4 Conclusion

Bonne nouvelle

Le rendu fonctionne sur les machines de l'IUT !

Planning

Ce que je propose :

- Aujourd'hui : plusieurs activités au sein d'une même application, TD3
- Demain : fin des TD 2 et 3
- Bases qui doivent être bien acquises
- À partir de la semaine prochaine : changement d'air (en l'occurrence, applications client/serveur)

Plan

- 1 Divers
- 2 Plusieurs activités
- 3 Cycle de vie d'une activité
- 4 Conclusion

Rappels

Lancement d'une activité :

- interface en XML : arbre de boîtes imbriquées les unes dans les autres
- boîtes : classes héritant de `View`, boîtes usuelles prédéfinies
- interface lancée lors de `onCreate`
- méthode de la classe `Activity`

Interaction :

- méthodes Java appelées lorsque l'utilisateur effectue une action
- lire, modifier, ... etc. les objets de l'interface

↔ lancer une nouvelle activité

Création d'une deuxième activité

Comme jusqu'à présent :

- classe héritant d'Activity
- sa propre interface, sa propre méthode onCreate, ...

Déclaration dans le manifeste :

```
<activity
    android:name=".SecondActivity"
    android:label="@string/title_activity_second" >
</activity>
```

Activité principale et activités secondaires

Dans le manifeste :

```
<application
    android:allowBackup="true"
    android:icon="@mipmap/ic_launcher"
    android:label="@string/app_name"
    android:theme="@style/AppTheme" >
    <activity
        android:name=".FirstActivity"
        android:label="@string/app_name" >
        <intent-filter>
            <action android:name="android.intent.action.MAIN" />
            <category android:name="android.intent.category.LAUNCHER" />
        </intent-filter>
    </activity>
    <activity
        android:name=".SecondActivity"
        android:label="@string/title_activity_second" >
    </activity>
</application>
```


Exemple

Nos deux activités :

- `FirstActivity` et `SecondActivity` : deux classes héritant d'`Activity`, avec chacune leur interface, leur `onCreate`, ...
- F veut lancer S (par exemple, lorsque l'utilisateur appuie sur un bouton)
- en lui passant des données
- on revient sur F lorsque S a fini
- en lui passant des résultats

Lancement de S

Dans une méthode de `FirstActivity` :

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

Intentions

Objets de la classe `Intent` :

- demander une action à une autre activité (ou à une autre application)
- ici : lancement
- autres possibilités : communication avec le système, avec d'autres applications, utilisation d'un service (activité en arrière-plan), ...
- possibilité de passer des données

Passage de données

Avant de lancer l'intention :

```
intent.putExtra("DATA", data);
```

Passage de données

Avant de lancer l'intention :

```
intent.putExtra(EXTRA_DATA, data);
```

Attribut de la classe :

```
final static String EXTRA_DATA = "EXTRA_DATA";
```

Récupération des données par S

À la fin de onCreate :

```
Intent intent = getIntent();  
int data = intent.getIntExtra(FirstActivity.EXTRA_DATA, 1);
```

Arrêt d'une activité

Une activité “s'arrête” :

- lors d'un appel à la méthode `finish` de la classe `Activity` (que l'on peut surcharger !)
- lorsque l'utilisateur appuie sur le bouton “retour” (sauf si c'est l'activité principale)
- lorsque l'interface doit se redessiner (ex : on fait tourner la tablette)
- lorsque l'activité a été stoppée et que le système a besoin de mémoire

Mais pas :

- lorsqu'on quitte l'activité
- lorsqu'on lance une nouvelle activité

↪ elle est seulement stoppée

Récupération de résultats : dans l'activité F

Au moment du lancement :

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


Récupération de résultats : dans l'activité F

Au moment du lancement :

```
Intent intent = new Intent(this, SecondActivity.class);  
startActivityForResult(intent, SOME_REQUEST);
```

Récupération de résultats : dans l'activité F

Au moment du lancement :

```
Intent intent = new Intent(this, SecondActivity.class);  
startActivityForResult(intent, SOME_REQUEST);
```

Récupération :

```
@Override  
protected void onActivityResult(int requestCode, int resultCode, Intent data) {  
    if (requestCode == SOME_REQUEST && resultCode == RESULT_OK) {  
        int res = data.getIntExtra(EXTRA_DATA, 0);  
    }  
}
```

Récupération de résultats : dans l'activité S

On surcharge la méthode `finish` :

```
@Override
public void finish() {
    Intent data = new Intent();
    data.putExtra(FirstActivity.EXTRA_DATA, 42);
    setResult(RESULT_OK, data);
    super.finish();
}
```

Conclusion sur le passage de données

Passage de données :

- **toujours** *via* les intentions
- dans un sens (lancement) comme dans l'autre (arrêt)
- **jamais** par “effet de bord” (jamais par modification d'une variable partagée) : on ne sait pas dans quel ordre ont lieu les opérations

Pile des activités

Au sein d'une application :

- lancements et arrêts des activités gérés par une pile

Pile des activités

Au sein d'une application :

- lancements et arrêts des activités gérés par une pile

Au départ

F

Pile des activités

Au sein d'une application :

- lancements et arrêts des activités gérés par une pile

F lance S

S

F

Pile des activités

Au sein d'une application :

- lancements et arrêts des activités gérés par une pile

S quitte

F

Pile des activités

Au sein d'une application :

- lancements et arrêts des activités gérés par une pile

F relance S

S

F

Pile des activités

Au sein d'une application :

- lancements et arrêts des activités gérés par une pile

	T
S lance T	S
	F

Pile des activités

Au sein d'une application :

- lancements et arrêts des activités gérés par une pile

T quitte

S

F

Pile des activités

Au sein d'une application :

- lancements et arrêts des activités gérés par une pile

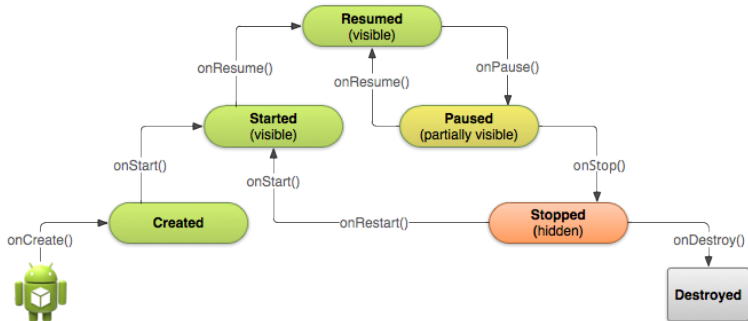
S quitte

F

Plan

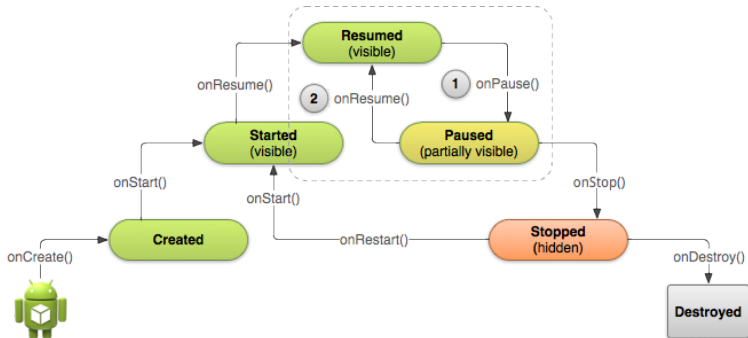
- 1 Divers
- 2 Plusieurs activités
- 3 Cycle de vie d'une activité
- 4 Conclusion

Cycle



©android.com

Mettre en pause et reprendre



©android.com

Mettre en pause et reprendre

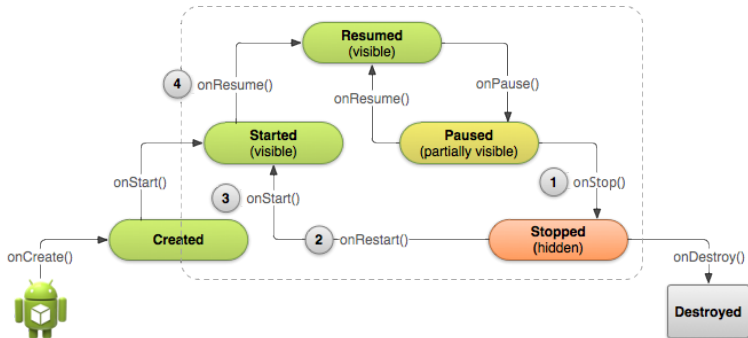
Lorsque le système appelle `onPause` :

- stopper les actions en cours (vidéos, ...)
- libérer les ressources coûteuses (webcam, ...)
- sauvegarder les données **persistantes**

Lorsque le système appelle `onResume` :

- relancer les actions en cours
- reprendre les ressources

Stopper et relancer



©android.com

Stopper et relancer

Lorsque le système appelle `onStop` :

- libérer toutes les ressources (webcam, réseau, ...)
- sauvegarder toutes les données

Lorsque le système appelle `onStart` :

- reprendre les ressources

Lorsque le système appelle `onRestart` :

- actions qui ne sont faites que dans ce cas, mais pas au premier démarrage

Plan

- 1 Divers
- 2 Plusieurs activités
- 3 Cycle de vie d'une activité
- 4 Conclusion

Conclusion

- Déclarer plusieurs activités
- Lancer et arrêter une activité
- Passer des messages entre activités : les intentions
- Cycle de vie d'une activité

À vous

Jeu “click the button” : interface simple (déjà donnée)

- créer une deuxième activité et la lancer
- observer le cycle de vie de ces deux activités
- passer des messages
- en solo : troisième activité