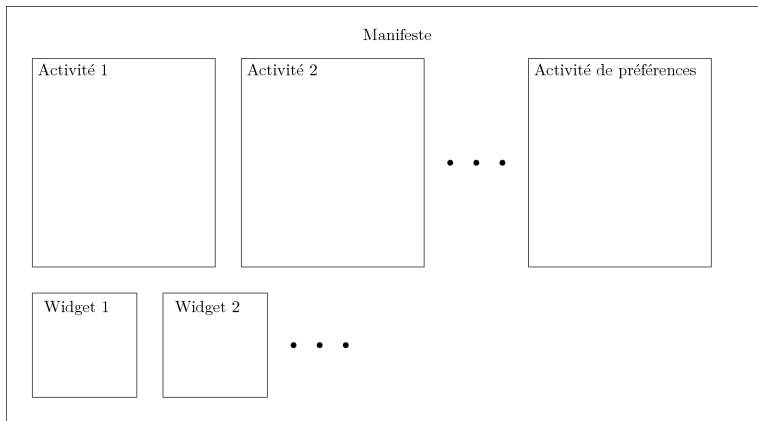


Vie des activités

Chantal Keller

Rappel sur l'architecture



Création d'une deuxième activité

Une activité comme une autre :

- sa propre vue : en XML
- son propre contrôleur et modèle : une classe héritant d'`Activity`
- sa propre 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>
```

Premier exemple

Deux activités F et S :

- 1 F veut lancer S (par exemple, lorsque l'utilisateur appuie sur un bouton)
- 2 on revient sur F lorsque S a fini

1. F veut lancer S

Dans la méthode correspondante du modèle :

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

1. F veut lancer S

Dans la méthode correspondante du modèle :

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

↪ `this` doit désigner l'objet courant de la classe implémentant F
(attention où on se trouve!)

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

2. On revient sur F lorsque S a fini

Lorsqu'une activité "s'arrête" :

- appel à plusieurs méthodes (suite du cours)
- on revient automatiquement sur l'activité qui l'a lancée (ou on quitte l'application si c'est l'activité principale)

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

S lance T

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

Deuxième exemple

Deux activités F et s :

- 1 F veut lancer s en lui passant des données
- 2 on revient sur F lorsque s a fini en nous donnant un résultat

1. En lui passant des données

Dans F :

```
final static String CLE_DONNEES = "CLE_DONNEES";  
  
..... {  
    Intent intent = new Intent(this, SecondActivity.class);  
    intent.putExtra(CLE_DONNEES, 17);  
    startActivity(intent);  
}
```

On associe des données à l'intention :

- par un système de clé/valeur
- clé : une chaîne de caractères (explicite) définie une fois pour toutes : attribut `final static`
- valeur : donnée que l'on veut transmettre (peut être de différents types : plusieurs `putExtra`)
- on peut passer plusieurs valeurs (**associées à des clés différentes**) avec plusieurs `putExtra`

Récupération des données par S

À la fin de onCreate de S :

```
protected void onCreate(Bundle savedInstanceState) {  
    super.onCreate(savedInstanceState);  
    setContentView(R.layout.activity_xml);  
    Intent intent = getIntent();  
    int valeur = intent.getIntExtra(F.CLE_DONNEES, 1);  
}
```

Récupération des données :

- `getIntent()` donne l'intention qui a lancé l'activité
- `getIntExtra`, `getStringExtra`, ... permettent de récupérer les valeurs associées aux clés dans l'intention
- prennent la clé (`F.CLE_DONNEES` : aller la chercher dans la classe où elle est définie) et une valeur par défaut (si pas de valeur dans l'intention)

2. En nous donnant un résultat : dans l'activité S

On redéfinit la méthode `finish` de S :

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

Fournir un résultat lorsque l'activité termine :

- en redéfinissant la méthode `finish`
- intention avec clé/valeur
- que l'on associe au résultat
- `RESULT_OK` : pour dire que tout s'est bien passé

Dans l'activité F (1/2)

Au moment du lancement :

```
final static String CLE_DONNEES = "CLE_DONNEES";

..... {
    Intent intent = new Intent(this, SecondActivity.class);
    intent.putExtra(CLE_DONNEES, 17);
    startActivity(intent);
}
```

Dans l'activité F (1/2)

Au moment du lancement :

```
final static String CLE_DONNEES = "CLE_DONNEES";  
final static int REQUETE = 1;  
  
..... {  
    Intent intent = new Intent(this, SecondActivity.class);  
    intent.putExtra(CLE_DONNEES, 17);  
    startActivityForResult(intent, REQUETE);  
}
```

Dans l'activité F (1/2)

Au moment du lancement :

```
final static String CLE_DONNEES = "CLE_DONNEES";  
final static int REQUETE = 1;  
  
..... {  
    Intent intent = new Intent(this, SecondActivity.class);  
    intent.putExtra(CLE_DONNEES, 17);  
    startActivityForResult(intent, REQUETE);  
}
```

Récupération :

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


Dans l'activité F (2/2)

Lancement :

- **utiliser** `startActivityForResult`

Récupération :

- **redéfinir** la méthode `onActivityResult` (réaction à un évènement)
- regarder quelle était la requête : `requestCode == REQUEST`
- vérifier que tout s'est bien passé : `resultCode == RESULT_OK`
- dans ce cas, récupérer le résultat se trouvant dans l'intention

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

Données dans les intentions

Méthodes :

- Ajout : `intent.putExtra(CLE, valeur)`
- Récupération : `intent.get[Int]Extra(CLE, default)`

Types :

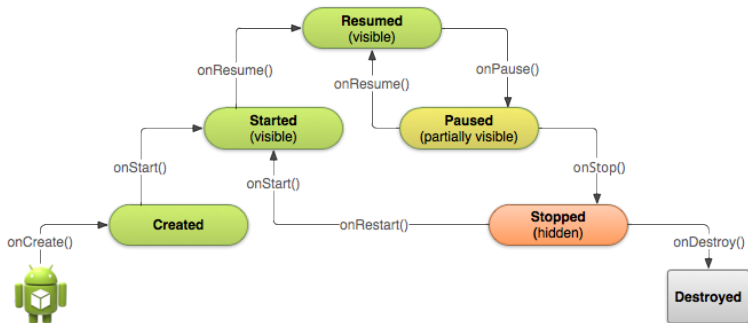
- CLE : `String`
- valeur, default : le même type au choix, se reflète dans `get[Int]Extra`

Aussi bien de F vers S que de S vers F .

Conclusion sur le lancement et l'arrêt des activités

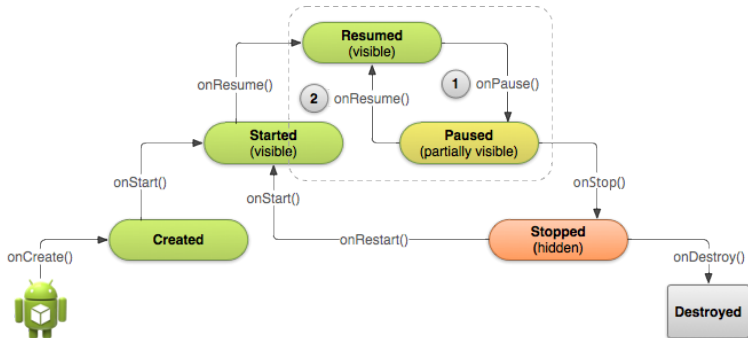
	Lancement	Arrêt
Sans données	utiliser <code>startActivity</code>	
Avec données	utiliser <code>startActivityForResult</code> redéfinir <code>onActivityResult</code>	redéfinir <code>finish</code>

Cycle



©android.com

Pause : pop-up, mise en veille



©android.com

Mettre en pause et reprendre

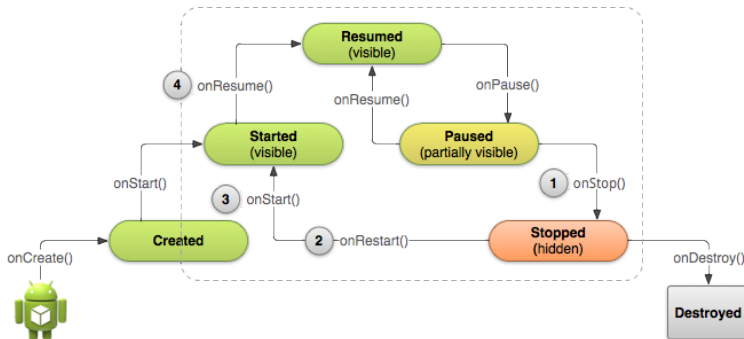
Lorsque le système appelle `onPause` :

- 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

Stop : application plus au premier plan



©android.com

Stopper et relancer

Lorsque le système appelle `onStop` :

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

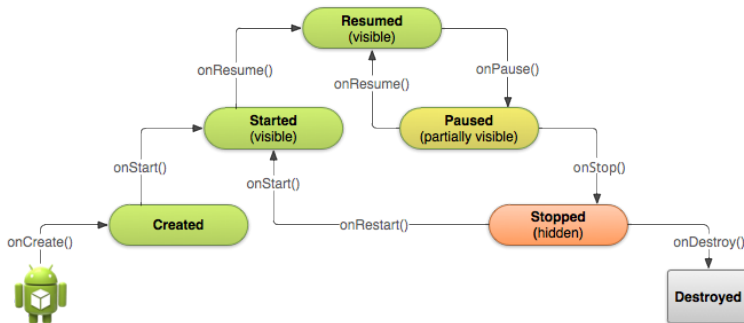
Lorsque le système appelle `onStart` :

- reprendre les ressources

Lorsque le système appelle `onRestart` :

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

Destruction : application arrêtée ou tablette tournée



©android.com

Détruire et créer

Lorsque le système appelle `onDestroy` :

- tout est déjà libéré et sauvegardé lors de `onPause` et `onStop`

Lorsque le système appelle `onCreate` :

- création de l'interface, ajout des écouteurs, ...
- gestion de la persistance courte des données : lorsque l'application est détruite pour être recrée instantanément (rotation)

Bundle

But :

- stocker des “petites” valeurs lorsqu’une activité est détruite pour une courte durée
- stockage : système de clé/valeur

Principe :

- lorsqu’une activité est détruite pour une courte durée,
`protected void onSaveInstanceState (Bundle outState)`
est appelée \Rightarrow la redéfinir pour sauvegarder nos données
- on récupère ce bundle à la re-crédation de l’activité :

`protected void onCreate(Bundle savedInstanceState)`

Sauvegarde ...

```
public class MainActivity extends Activity {  
  
    public final static String ENTIER_TRES_IMPORTANT = "ENTIER_TRES_IMPORTANT";  
    private int entierASauvegarder;  
    private TextView text;  
  
    ...  
  
    @Override  
    protected void onSaveInstanceState (Bundle outState) {  
        super.onSaveInstanceState(outState);  
        outState.putInt(ENTIER_TRES_IMPORTANT, entierASauvegarder);  
    }  
  
    @Override  
    protected void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        setContentView(R.layout.activity_main);  
  
        ...  
  
    }  
  
}
```

Sauvegarde ... et récupération

```
public class MainActivity extends Activity {

    public final static String ENTIER_TRES_IMPORTANT = "ENTIER_TRES_IMPORTANT";
    private int entierASauvegarder;
    private TextView text;

    ...

    @Override
    protected void onSaveInstanceState (Bundle outState) {
        super.onSaveInstanceState(outState);
        outState.putInt(ENTIER_TRES_IMPORTANT, entierASauvegarder);
    }

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

        ...

        text = (TextView) findViewById(R.id.text);

        if (savedInstanceState != null) {
            entierASauvegarder = savedInstanceState.getInt(ENTIER_TRES_IMPORTANT);
            text.setText(String.valueOf(entierASauvegarder));
        }
    }
}
```