



android

An Open Handset Alliance Project

Les applications

O.Legrand
G. Seront

Les applications

- Chaque application a son Linux uid
 - Inconnu de l'application
 - Utilisé pour les permissions des fichiers
- Chaque process a sa propre VM
- Chaque application a son propre process



Les applications

- Il existe 4 types de composants applicatifs:
 - *Activity*
 - *BroadcastReceiver*
 - *Service*
 - *Content Provider*
- Une application peut être constituée d'un ou plusieurs composants, éventuellement de types différents.



Activity

- Une application simple comprend une ou plusieurs *Activity*.
- Chaque écran étant associé à une activité, passer à un autre écran, fait démarrer l'activité associée à cet écran.
- L'exécution de l'application va entraîner le démarrage et l'arrêt d'activités qui vont s'enchaîner.



Activity et Intent

- Android utilise la classe *Intent* pour se déplacer d'une activité à l'autre.
- Soit l' *Intent* décrit ce que l'activité appelante veut faire, quelle est son intention : *intent implicite* ;
- Soit l' *Intent* spécifie qu'elle activité doit être lancée : *intent explicite*.



Intent implicite

- A la création d'un *Intent* implicite, on peut préciser :
 - son type d'action : *MAIN, VIEW, PICK, EDIT, ...*
 - les données concernées par cette action : une *Uri*
- Par exemple: si une application veut consulter les données d'une personne (un contact) :
 - elle doit construire un *Intent* avec l'action *VIEW* et une *Uri* qui pointe vers les données de ce contact.




Intent implicite

On va dire ce qu'on veut, pas comment on le fait

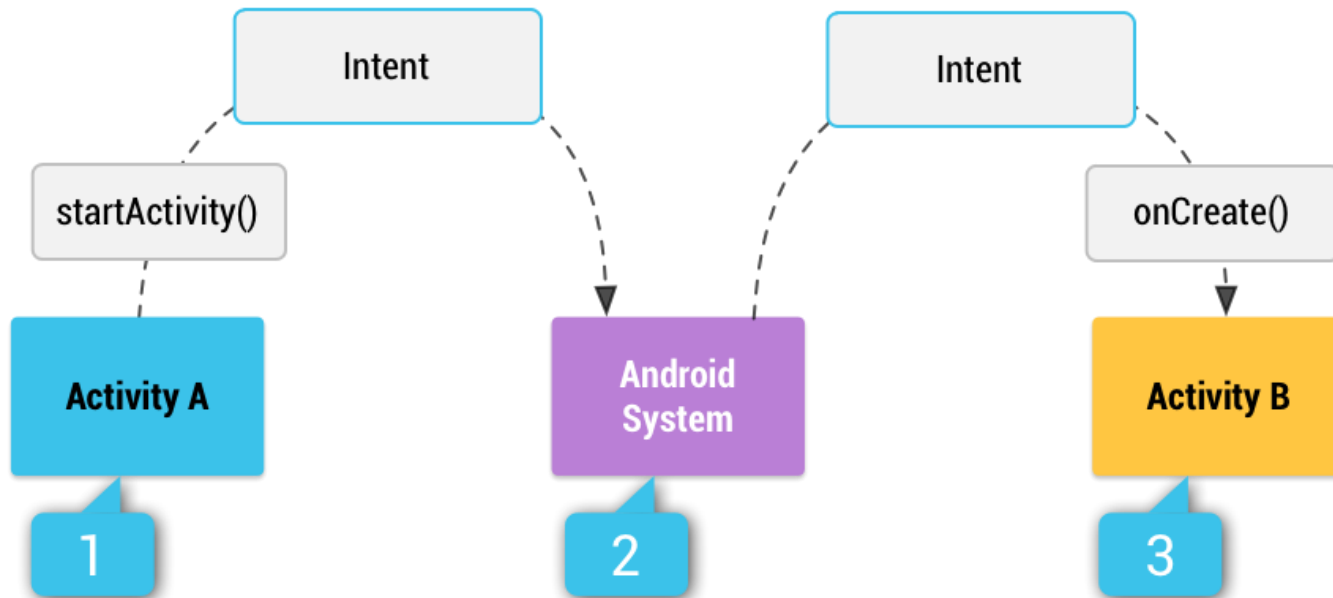
```
static final int REQUEST_IMAGE_CAPTURE = 1;

private void dispatchTakePictureIntent() {
    Intent takePictureIntent = new Intent(MediaStore.ACTION_IMAGE_CAPTURE);
    if (takePictureIntent.resolveActivity(getPackageManager()) != null) {
        startActivityForResult(takePictureIntent, REQUEST_IMAGE_CAPTURE);
    }
}
```




Intent implicite

Le système se charge de trouver la bonne activité installée sur le téléphone



Activity, Intent, IntentFilter

- Toute activité peut publier ce qu'elle est capable de faire sous forme d'un ou plusieurs *IntentFilters*.
- Ces *IntentFilters* sont publiés dans le fichier : *AndroidManifest.xml*.
- Lors de la navigation d'un écran à l'autre, le système (l'*ActivityManager*) recherche et démarre une activité dont l'*IntentFilter* correspond le mieux à l'*Intent* de l'activité appelante.



Intent Filter

Votre activité peut dire dans le manifest qu'elle traite certains intents implicites :

```
<activity android:name="ShareActivity">
  <!-- This activity handles "SEND" actions with text data -->
  <intent-filter>
    <action android:name="android.intent.action.SEND"/>
    <category android:name="android.intent.category.DEFAULT"/>
    <data android:mimeType="text/plain"/>
  </intent-filter>
```



Intent Filter

Vous avez déjà utilisé les filtres sans le savoir:

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



Intent standards

Il existe une liste complète d'intents standards
(voir [ref](#)).



Récupération du résultat

Après la prise d'image, l'activité appelante est notifiée:

```
@Override
protected void onActivityResult(int requestCode, int resultCode, Intent data) {
    if (requestCode == REQUEST_IMAGE_CAPTURE && resultCode == RESULT_OK) {
        Bundle extras = data.getExtras();
        Bitmap imageBitmap = (Bitmap) extras.get("data");
        mImageView.setImageBitmap(imageBitmap);
    }
}
```



Intent explicite

On dit qui on veut

Une activité précise

```
public void sendMessage(View view) {
    Intent intent = new Intent(this, DisplayMessageActivity.class);
}
```

Mais aussi un autre composant applicatif

```
// Executed in an Activity, so 'this' is the Context
// The fileUrl is a string URL, such as "http://www.example.com/image.png"
Intent downloadIntent = new Intent(this, DownloadService.class);
downloadIntent.setData(Uri.parse(fileUrl));
startService(downloadIntent);
```



Intent explicite: passer de l'information

Dans l'activité appelante

```
monIntent.putExtra("UneValeur", 36);
```

Dans le onCreate de l'activité appelée

```
Intent intent = getIntent();
int valeur = intent.getIntExtra("UneValeur", 0);
```

Best practice: définir une constante dans l'activité appelée

```
public static final String UNE_VALEUR = "UneValeur";
```



Activity, Intent, IntentFilter

- Avantages de ce système:
 - les applications peuvent utiliser les fonctionnalités d'activités déjà écrites;
 - une activité peut être remplacée à tout moment par une activité qui présente le même *IntentFilter*.



Activity, Intent

- Pour démarrer une activité :
 - *startActivity(myIntent) ;*
- Pour démarrer une activité avec retour à l'activité appelante:
 - *startActivityForResult(myIntent) ;*



Composants applicatifs

- Il existe 4 types de composants applicatifs:
 - *Activity*
 - *BroadcastReceiver*
 - *Service*
 - *Content Provider*



BroadcastReceiver

- Partie de code d'une application exécutée en réponse à un événement externe :
 - le téléphone sonne;
 - le réseau devient accessible;
 - il est minuit,...
- Un *BroadcastReceiver* ne possède pas d'UI. Il peut utiliser le *NotificationManager* pour prévenir l'utilisateur d'un événement (utilisation de la barre d'état, vibreur, son émis,...)



BroadcastReceiver

- L'application ne doit pas nécessairement tourner pour qu'un de ses *BroadcastReceiver* soit exécuté.
- L'application sera lancée par le système lorsqu'un événement survient, démarrant un de ses *BroadcastReceiver*.
- Ce *BroadcastReceiver* doit s'exécuter rapidement (5 " max).
- Il peut lancer une activité, un service, avertir l'utilisateur, ...



Composants applicatifs

- Il existe 4 types de composants applicatifs:
 - *Activity*
 - *BroadcastReceiver*
 - *Service*
 - *Content Provider*



Service

- Un *Service* est un process qui tourne en tâche de fond.
- Il ne possède pas d'UI.
- Exemple :
 - Une application de type music player présentera plusieurs écrans (activités) permettant de sélectionner le titre, le volume, etc. Ensuite, ces écrans ne seront plus nécessaires (les activités s'arrêtent). Un service peut démarrer. Il s'occupera de la diffusion de la musique.



Service

- Le service peut être démarré à partir de l'activité (le music player) en exécutant :
 - *context.startService(...);*
- Pour se connecter à un service :
 - *context.bindService(...)*
- Pour communiquer avec un service connecté, il faut utiliser les méthodes qu'il expose.



Composants applicatifs

- Il existe 4 types de composants applicatifs:
 - *Activity*
 - *BroadcastReceiver*
 - *Service*
 - *Content Provider*



ContentProvider

- Une application peut stocker ses données dans un fichier ou une base de données SQLite.
- Un *ContentProvider* permet l'échange de données entre applications.
- Dans une même application, il permet de découpler la couche applicative de la couche de persistance des données.



ContentProvider

- Un *ContentProvider* est une classe qui expose un ensemble de méthodes standards permettant à des composants applicatifs de sauver et lire des données d'un type donné propre à ce *ContentProvider*.
- Android propose des *ContentProvider* pour des types de données courants (audio, video, image, données de contacts, etc.).
- Ils sont disponibles dans le package : *android.provider*.



Application Manifest

- Fichier XML
- Pour assembler les différents composants
- Pour donner les permissions
- Démo, ...



Application Resources

- Autres ressources que code et manifest
- Images, sons
- Définitions des écrans et menus
- Constantes (Strings, ...)
- Sous le répertoire /res
- Sont référencées dans une classe générée: **R**



Application Context

- Le contexte représente l'état global de l'application
- Il est nécessaire pour la création de beaucoup d'objets
- Il permet l'accès aux ressources communes (GPS, écran, ...)
- Il est contenu dans toute Activity
- Démo R + contexte

