Les intents

Interaction entre activités

- Une application Android peut contenir plusieurs activités.
- Chaque activité utilise la méthode **setContentView** pour s'associer avec une interface graphique.
- Les activités sont indépendantes les unes des autres, cependant, elles peuvent collaborer pour échanger des données et des actions.

Interaction entre activités

- Typiquement, l'une des activités est désignée comme étant la première à être présentée à l'utilisateur quand l'application est lancée : on l'appelle **l'activité de démarrage**
- Le passage d'une activité à une autre est réalisé en demandant à l'activité en cours d'exécuter **un Intent**.
- Le but des Intents est de déléguer une action à un autre composant, une autre application ou une autre activité de l'application courante.

Construction d’un intent

Un objet Intent contient les information suivantes:

- **le nom** du composant à démarrer
- **l'action à réaliser**: ACTION-VIEW, ACTION_SEND…
- **les données** : URI référençant la donnée sur laquelle l’action va agir
- **Les Extras**: des informations additionnelles pour réaliser l’action demandée sous forme de paires de clef/valeur
- **une catégorie** pour cibler un type d'application: CATEGORY-BROWSABLE, CATEGORY-LAUNCHER…
- **des drapeaux** (information supplémentaire)

Types d’intents

Il existe deux principaux types d’Intents :

- **Intent Explicite** : indique le composant qui fournit la classe exacte à exécuter. Il permet de démarrer un composant de votre propre application, car le nom de la classe est connu
- **Intent Implicite** : ne définit pas le composant mais déclare une action à réaliser. Il permet à un composant d’une application d’appeler un composant d’une autre application.

Intent implicite

- Un intent implicite ne définit pas le composant cible, mais possède assez d’informations pour aider le système à déterminer le composant le mieux approprié pour réaliser cette intention.
- Les composants récepteurs sont fournis par Android, ou par d'autres applications téléchargées sur le Play Store.
- Le système cherche une application ayant la capacité à effectuer l’action demandée.

Intent implicite

Les principaux arguments d'un Intent implicite sont :

- Action : l'action à réaliser, peut être prédéfinie (ACTION_VIEW, ACTION_EDIT, ACTION_MAIN, etc.) ou créée par l'utilisateur.
- Données : Les données principales (data) sur lesquelles on va agir, tel qu'un url ou un numéro de téléphone à appeler.

Il est donc typiquement appelé comme suit:

```
Intent myActivityIntent = new Intent (<action>, <données>) ;
startActivity (myActivityIntent) ;
```

Intent implicite: data

- Il s'agit de **Uniform Resource Identifier** (URI).
- Un URI ((Unified Resource Identifier) peut être un Url (http://), un numéro de téléphone (tel://), un emplacement géographique (geo://). etc.
- La méthode statique Uri.parse permet de construire un objet URI à partir d'une chaîne de caractères.

```
Uri webpage = Uri.parse("http://www.google.fr");
Uri tel = Uri.parse("tel :+21698123456") ;
Uri geo = Uri.parse("geo :52.312405,20.918330") ;
```

Intent implicite

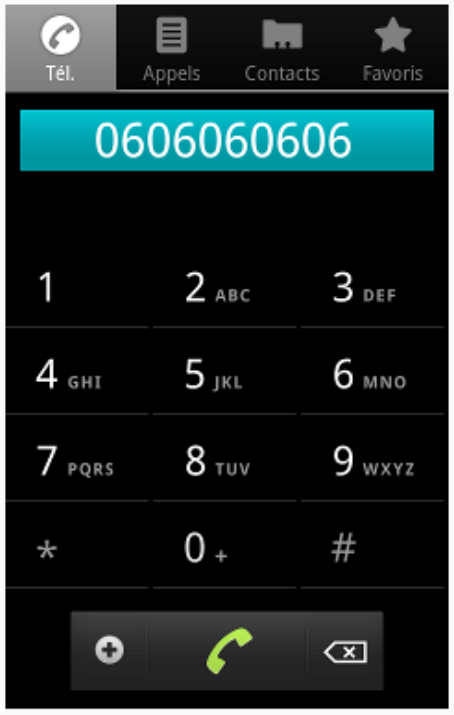
Plusieurs actions natives existent par défaut sur Android:

- ACTION_VIEW: d'appeler une application pour visualiser un contenu dont on donne l'URI
- ACTION_CALL (ANSWER, DIAL) : passer/réceptionner/afficher un appel
- ACTION_EDIT (DELETE) : éditer/supprimer une donnée
- ACTION_SEND : envoyer des données par SMS ou E-mail
- ACTION_WEB_SEARCH : rechercher sur internet

Intent implicite

Par exemple, pour passer un appel téléphonique, les paramètres de l'intent sont composés d'une action native et d'un URI comportant le numéro à appeler.

```
b.setOnClickListener(new View.OnClickListener() {
  @Override
  public void onClick(View v) {
    Uri telephone = Uri.parse("tel:0606060606");
    Intent secondeActivite = new Intent(Intent.ACTION_DIAL,
telephone);
    startActivity(secondeActivite);
  }
});
```



Intent implicite: intent Filter

- Un Intent Filter est une expression dans le fichier Manifest d'une application qui spécifie le type d'intents que le composant veut recevoir.
- L'activité principale a toujours, et par défaut, l'action et la catégorie ci-dessous.

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

Intent explicite

Le lancement d'une nouvelle Activity de façon explicite s'effectue en deux lignes:

1. Intent i = new Intent(ActivityA.this, ActivityTarget.class).

   ActivityA.this:activité courante

   ActivityTarget.class: Activité appelée

2. **StartActivity(i)** : Pour lancer une activité sans attendre de retour

Intent Explicite

- L'activité démarrée reste à l'écran jusqu'à ce que l'utilisateur appuie sur le bouton Précédent (Back) de l'appareil. Cette activité est alors fermée et reprise par le système.
- L'activité d'origine revient au premier plan.
- l'activité démarrée peut être fermée manuellement en réponse à une action de l'utilisateur (comme un clic sur un bouton, par exemple) avec la méthode finish()

Les extras

- Un extra est un couple clé/valeur, basé sur le système Bundle
- L'échange des données se fait par l'attribution d'une clé unique (un identifiant sous forme d'une chaîne de caractères) à une valeur à échanger.
- L'envoi de données se fait par la méthode putExtra
- La méthode putExtra supporte tous les types de données primitifs: int, float, String, etc.

```
Intent i = new Intent(this, SecondActivity.class);
i.putExtra("nom", "Ali");
i.putExtra("age", 24);
startActivity(i);
```

Les extras

- Pour récupérer l'Intent qui véhicule les données envoyées, on utilise la méthode getIntent().
- Les données sont encapsulées dans un objet de type Bundle. Les données de Bundle sont récupérées par la méthode getExtras.
- Les données à récupérer peuvent être de différents types (String, double…) et peuvent être récupérés à travers des méthodes (getDouble(), getString()…)

```
Bundle extras=getIntent().getExtras();
 String n= extras.getString("nom");
 String p= extras.getString("prenom");
```

Lancement d'une activité avec ou sans retour

- pour lancer Activ2 à partir d'une activité courante :

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

- On peut demander la terminaison de l'activité courante après lancement de l'activité à l'aide de la méthode finish(), L'utilisateur ne pourra pas faire back dessus, car elle disparaît de la pile. :

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

Lancer une activite avec resultat

- Si l'activité a été lancée avec la méthode startActivity, l'appelant n’attend pas de résultat ou de retour de l'activité.
- Si l'activité a été lancée avec la méthode startActivityForResult, l’appelant attend un retour de la sous-activité.
- Dès que la sous-activité se termine, la méthode onActivityResult de la sous-activité est appelée et il est possible d’ exécuter des actions suivant le résultat.

Lancement avec attente de résultat

Etapes

Activité parent

1. Création de l'intent

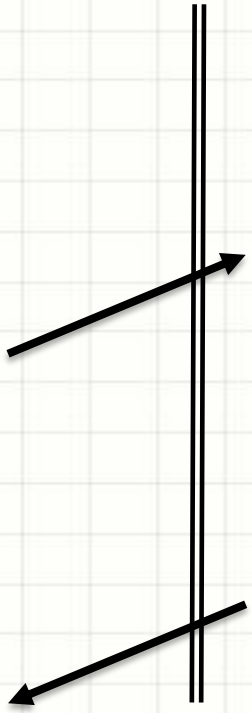
2. Lancement de la sous activité

startActivityForResult(intent,ID_REQUEST_CODE);

7. Gestionnaire pour la réception du mesage en Retour

protected void onActivityResult(int requestCode, int resultCode,

Intent data)



Activité fille

3. Récupération de l'Intent + extraction

4. Traitement

5.Création du Bundle retour

6. Fin activité et renvoie Bundle

Retour d’une activité

Au sein d'une application, une activité peut vouloir récupérer un code de retour de l'activité "enfant".

On utilise pour cela la méthode qui envoie un code de retour à l'activité enfant.

```
startActivityForResult(Intent i,int requestCode)
```

Lorsque l'activité parent reprend la main, il devient possible de filtrer le code de retour dans la méthode pour savoir si l'on revient ou pas de l'activité enfant.

```
onActivityResult(int requestCode, int resultCode, Intent data)
```

Lancement avec attente de résultat

- Pour une activité avec attente de résultat, Il faut définir un code d'appel RequestCode fourni au lancement.

```
private static final int APPEL_ACTIV2 = 1;
Intent intent = new Intent(this, Activ2.class);
startActivityForResult(intent, APPEL_ACTIV2);
```

- Ce code identifie l'activité lancée, afin de savoir plus tard que c'est d'elle qu'on revient.

  Par exemple, on pourrait lancer au choix plusieurs activités : édition, copie, suppression d'informations. Il faut pouvoir les distinguer au retour.

Méthode onActivityResult

Quand on revient dans l'activité appelante, Android lui fait exécuter cette méthode :

onActivityResult(int requestCode, int resultCode,Intent data)

- requestCode est le code d'appel de startActivityForResult
- resultCode vaut soit RESULT_CANCELED soit RESULT_OK ,
- data est fourni par l'activité appelée et qui vient de se terminer.

Ces deux dernières viennent d'un appel à **setResult(resultCode,data)**

Retour d’une activité

- Code d’appel dans l’activité appelante

```
Intent login = new Intent(this, ActiviteSecond.class);
startActivityForResult(login,1);
```

- Code de traitement par l’activité appelante

```
protected void onActivityResult(int resultCode, Intent data)
{
 if (requestCode == 1)
  {
  Toast.makeText(this, "Coup de tel terminé",Toast.LENGTH_LONG).show();
  }
}
```

Retour d’une activité

L’activité appelée fabrique un nouvel Intent

```
Intent intent = new Intent();
intent.putExtra("resultat", ""+result);
setResult(Activity.RESULT_OK, resultIntent);
```

L’activité appelante récupère les données

```
public void onActivityResult(int requestCode, int resultCode, Intent data)
{  super.onActivityResult(requestCode, resultCode, data);
if (requestCode == 1)
  {  if (resultCode == Activity.RESULT_OK)
      {
       // TODO Extract the data
      }
  }
}
```