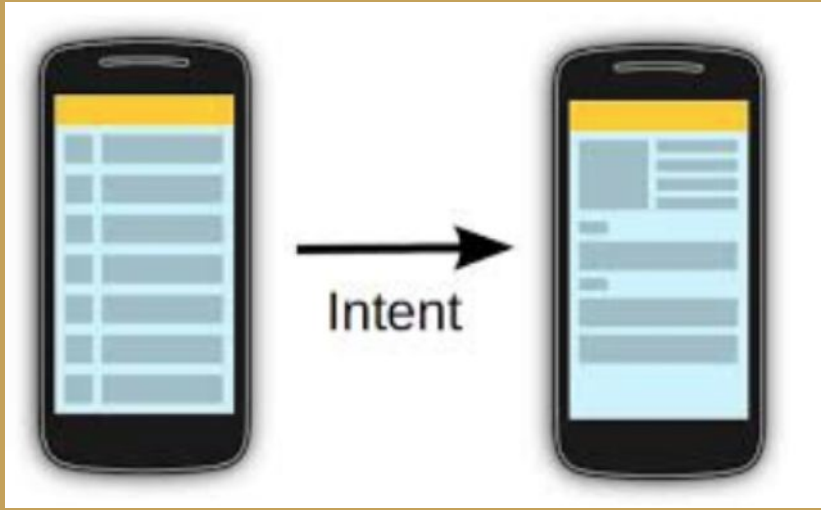
Chapitre VI : La navigation entre les composants

Intents



Introduction au Fichier Manifest

- Le fichier **AndroidManifest.xml** est le fichier central où chaque activité est déclarée pour être reconnue par Android.

- Il permet de gérer la structure de navigation et de spécifier quelles activités sont accessibles.

- Précise les paramètres essentiels des activités, comme les orientations d'écran et les modes de lancement.

- Dans le fichier Manifest, vous trouverez un ensemble de nœuds pour décrire votre projet.

Fichier Manifest

- la structure générale du fichier AndroidManifest.xml

```
1<?xml version="1.0" encoding="utf-8"?>
2<manifest xmlns:android="http://schemas.android.com/apk/res/android"
3      package="com.androidapp.basicElements"
4      android:versionCode="1"
5      android:versionName="1.0">
6    <application android:icon="@drawable/icon" android:label="@string/app_name">
7        <activity android:name=".BasicElements"
8                  android:label="@string/app_name">
9            <intent-filter>
10               <action android:name="android.intent.action.MAIN" />
11               <category android:name="android.intent.category.LAUNCHER" />
12           </intent-filter>
13       </activity>
14
15   </application>
16   <uses-sdk android:minSdkVersion="2" />
17
18</manifest>
```

Fichier Manifest

- Le nœud **« application »** décrit les attributs et les différents composants qui caractérisent votre application.
- Par défaut, votre application n'a qu'un composant, **l'activité principale**.
- **Les composants:** sont les éléments qui composeront vos projets.
- Votre application sera au final un ensemble de composants qui interagissent entre eux et avec le reste du système.

<Activity>

```
<activity android:name=".BasicElements"
          android:label="@string/app_name">
```

- Le nœud ***<activity>*** permet de décrire chaque activité (ou écran) de l'application.
- **Attributs principaux :**
  - **android:name** :
    - Indique la classe Java qui implémente l'activité.
    - Sert d'identifiant unique pour Android afin de repérer l'activité parmi d'autres composants de l'application.
  - **android:label** :
    - Permet de définir un nom visible pour l'activité.
    - Ce nom s'affiche en haut de l'écran lorsque l'activité est ouverte.
-

<intent-filter>

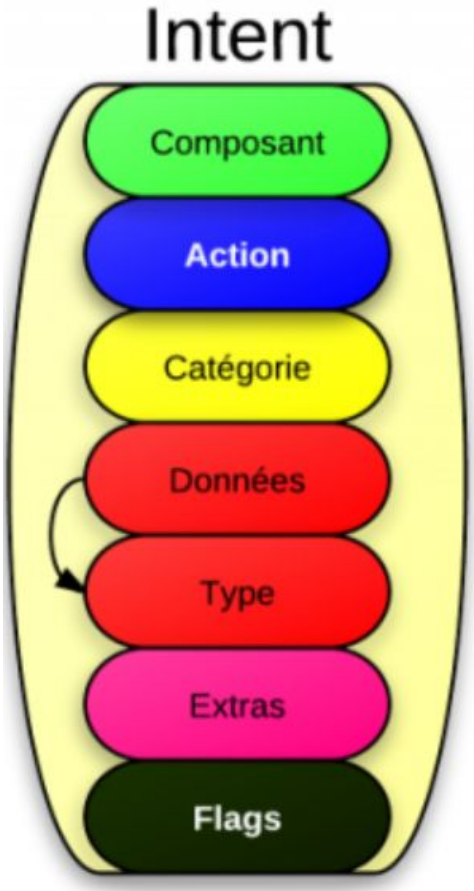
```
<intent-filter>
    <action android:name="android.intent.action.MAIN" />
    <category android:name="android.intent.category.LAUNCHER" />
</intent-filter>
```

- Indique comment cette activité se lancera.
- L'activité qui sera lancée depuis le menu principal d'Android contiendra toujours ces deux lignes dans son Manifest.
    - **Action** : ***MAIN*** indique que cette activité est le point d'entrée de l'application.
    - **Category** : ***LAUNCHER*** spécifie que cette activité doit apparaître dans le lanceur d'applications (l'écran principal).

Intent

- **Informations essentielles** :
  - **Action** : décrit ce que le destinataire doit faire (ex. "ACTION_VIEW" pour afficher une page).
  - **Données** : précise les données sur lesquelles l'action doit être effectuée (ex. une URL ou un numéro de téléphone).

- **Informations supplémentaires :** D'autres champs peuvent être inclus pour enrichir l'intent, bien qu'ils ne soient pas obligatoires (ex. catégories, options).



Intent

- **Catégorie** : fournit des détails supplémentaires sur l'action à exécuter et aide à identifier le type de composant pouvant gérer l'intent (ex. **CATEGORY_LAUNCHER** pour indiquer une activité de démarrage).
- **Type** : définit le type de données incluses dans l'intent, ce qui permet de restreindre les données à un format spécifique (ex. **image/jpeg** pour une image).
- **Extras** : utilisé pour ajouter du contenu supplémentaire aux intents, facilitant le passage de données entre les composants (ex. **putExtra("key", valeur)**).
- **Flags** : permettent de modifier le comportement de l'intent, influençant sa gestion dans la pile des activités (ex. **FLAG_ACTIVITY_NEW_TASK** pour démarrer une activité dans une nouvelle tâche).

Intent

- Il existe deux principaux types d’intent :
  - **Intent explicite :**
    - Spécifie directement la classe du composant cible (comme une activité spécifique).
    - Utilisé principalement pour la communication entre composants de la même application.
  - **Intent implicite :**
    - Déclenche une action sans spécifier de composant cible précis.
    - Utilisé pour des actions pouvant être gérées par d'autres applications (envoyer un e-mail depuis votre application).

Intent explicite

- Un intent explicite cible directement un composant précis en spécifiant son **contexte** et la classe de destination. Il est généralement utilisé pour démarrer une activité ou un service interne à l'application.

  Code Java:

```
Intent intent = new Intent(Activite_de_depart.this,
Activite_de_destination.class);
```

- Pour lancer l'intent, il existe **deux façons**:
  - Sans Retour
  - Avec Retour (le composant de destination nous renvoie une réponse).

Intent explicite:

1) Démarrer une activité sans retour

Méthode utilisée : **void startActivity(Intent intent)**

- Cette méthode lance une nouvelle activité sans attendre de retour.

Exemple de code :

```
Intent intent = new intent(this, Activity2.class);
startActivity(Intent);
```

- Ici, un nouvel intent est créé pour lancer **Activity2** depuis l'activité actuelle.
- Avant de démarrer une activité, il faut la déclarer dans le fichier **AndroidManifest.xml**. Sans cette déclaration, une erreur **ActivityNotFoundException** sera générée.

Intent explicite:

1) Démarrer une activité sans retour

- ***<application ...> <activity android:name=".Activity2"></activity> </application>***

Exemple: dans une première activité, vous allez mettre un bouton de sorte qu'appuyer sur ce bouton lance une seconde activité.

Intent explicite:

2) Démarrer une activité avec retour

Méthode utilisée : **void startActivityForResult(Intent intent, int requestCode)**

- La méthode **startActivityForResult** permet de communiquer une valeur de retour à l'activité parent.
- La sous-activité est identifiée avec un **requestCode**. I.e. identifier de manière unique un intent.
- La méthode **setResult** dans la sous-activité est utilisée pour renvoyer des informations à l'activité parent.
- Le retour peut inclure un code prédéfini comme **RESULT_OK** ou **RESULT_CANCELED**, ou un code personnalisé pour des résultats spécifiques.

Intent explicite:

2) Démarrer une activité avec retour

Exemple de code :

```
// Démarrage de l'activité avec retour
Intent intent = new Intent(this, Activity2.class);
startActivityForResult(intent, 1); // 1 est le requestCode

// Dans Activity2 pour renvoyer un résultat
setResult(RESULT_OK, resultIntent);
finish();
```

Exemple pratique :

- Une activité A peut lancer une activité B pour sélectionner une photo. Une fois la photo choisie, l'activité B retourne à A avec un résultat (RESULT_OK) et l'image sélectionnée en extra.

Intent implicite

- Un intent implicite permet de déléguer au système Android le choix de l'application qui va traiter la requête.
- **Exemple :**
  - Par exemple, au lieu de spécifier une application particulière pour ouvrir un lien, on laisse le système déterminer quelle application peut gérer cette action.
- L’application émettrice envoie une requête sans connaître explicitement le destinataire.
- Android examine toutes les applications installées sur l'appareil qui peuvent gérer le type d'action demandé.
- Ces intents peuvent être gérés par des applications natives d’Android ou par des applications téléchargées depuis le Play Store.

Intent implicite : exemples

- **Navigateur** :
  - Permet d'ouvrir des liens URL. En envoyant un intent implicite avec une action Intent.ACTION_VIEW et une URI de type URL, le système proposera d'ouvrir le lien avec le navigateur installé.
- **Appareil photo** :
  - Peut être invoqué pour capturer une photo ou une vidéo via un intent implicite. Par exemple, avec Intent.ACTION_IMAGE_CAPTURE, le système ouvre l'application appareil photo par défaut.
- **Contacts** :
  - Pour visualiser ou choisir un contact, vous pouvez utiliser l'intent Intent.ACTION_PICK avec l'URI ContactsContract.Contacts.CONTENT_URI, ce qui ouvrira l'application Contacts native.

Intent implicite : exemples

- **Messagerie SMS** :
  - L'intent Intent.ACTION_SENDTO avec l'URI SMS :12345 permet d'ouvrir l'application de messagerie pour envoyer un SMS au numéro spécifié.
- **E-mail** :
  - Utiliser Intent. ACTION_SEND avec des extras comme Intent.EXTRA_EMAIL, Intent.EXTRA_SUBJECT et Intent.EXTRA_TEXT ouvre l'application de messagerie pour envoyer un e-mail.

Intent implicite : actions natives

- **Autres exemples d'actions d'intentions implicites**

ACTION_EDIT	Permet de modifier les données identifiées par l'intention.
ACTION_SEND	Permet d'envoyer des données à une autre application, telle qu'une application de messagerie ou de courrier électronique.
ACTION_PICK	Permet de sélectionner un élément dans une liste, par exemple un contact ou une photo.
ACTION_SEARCH	Permet d'effectuer une recherche sur les données identifiées par l'intention
ACTION_DIAL	Permet de composer un numéro de téléphone.
ACTION_CALL	Permet de lancer un appel téléphonique.

Les Intents dans Android

- **Exemple d'intent qui lance le navigateur Web**

```
@Override
public void onClick(View v) {
    if(v.getId()==R.id.btn1){
        Uri uri = Uri.parse("tel:0774482316");
        Intent intent = new Intent(Intent.ACTION_DIAL, uri);
        startActivity(intent);
    }
```

URI: Uniform Resource Identifier

est une courte chaîne de caractères identifiant une ressource.

Les Intents dans Android

- **Exemple d'intent qui lance le navigateur Web**

```
@Override
public void onClick(View v) {
    if(v.getId()==R.id.btn1){
        Uri uri = Uri.parse("tel:0774482316");
        Intent intent = new Intent(Intent.ACTION_DIAL, uri);
        startActivity(intent);
    }
```

URI: Uniform Resource Identifier

est une courte chaîne de caractères identifiant une ressource.