Fondamentaux d'Android

6. Intents implicites

Hatem Aziza - October 2024

Developers

Introduction

Dans une intention (Intent) explicite, vous effectuez une activité dans votre application, ou dans une autre application, en envoyant une intention avec le nom de classe complet de l'activité.

Avec une intention **implicite**, vous lancez une activité **sans savoir** quelle application ou activité gérera la tâche. Par exemple, si vous souhaitez que votre application prenne une photo, envoie un e-mail ou affiche un emplacement sur une carte, vous ne vous souciez généralement pas de l'application ou de l'activité qui effectue la tâche.

Pour faire correspondre votre demande avec une application installée sur l'appareil, le système Android fait correspondre votre intention implicite avec une activité dont les filtres d'intention indiquent qu'il peut effectuer l'action. Si plusieurs applications correspondent, l'utilisateur se voit présenter un sélecteur d'application qui lui permet de sélectionner l'application qu'il souhaite utiliser pour gérer l'intention.

Dans cet atelier, vous créez une application qui envoie une intention implicite pour effectuer chacune des tâches suivantes :

- Ouvrez une URL dans un navigateur Web.
- Ouvrez un emplacement sur une carte.
- Partagez du texte.

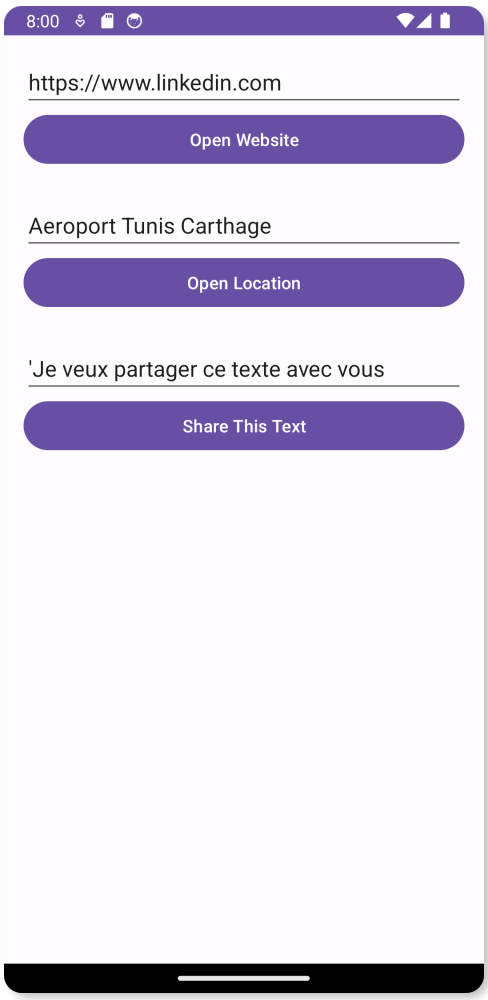
Ce que vous apprendrez

- Comment créer un Intent implicite et utiliser ses actions et catégories.
- Comment utiliser la classe d'assistance **ShareCompat.IntentBuilder** pour créer un Intent implicite pour le partage de données.
- Comment annoncer que votre application peut accepter un Intent implicite en déclarant des filtres d'intention dans le fichier AndroidManifest.xml.

Aperçu de l'application

Dans cette section, vous créez une nouvelle application avec une activité et trois options d'actions :

- ouvrir un site Web,
- ouvrir un emplacement sur une carte et
- partager un extrait de texte.



Tâche 1 : Créer le projet et le Layout

Dans cette tâche, créez le layout de l'application. Utilisez un LinearLayout, trois boutons et trois EditText, comme ceci :

1. Ajoutez les ressources de chaîne suivantes dans **strings.xml**:

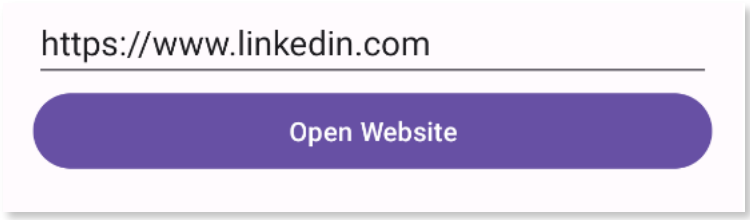
```
<string name="edittext_uri">https://www.linkedin.com</string>
<string name="button_uri">Open Website</string>
<string name="edittext_loc">Aeroport Tunis Carthage</string>
<string name="button_loc">Open Location</string>
```

```
<string name=“edittext_share”>\'Je veux partager ce texte avec vous</string>
<string name="button_share">Share This Text</string>
```

2. Dans activity_main.xml, remplacez **android.support.constraint.ConstraintLayout** par **LinearLayout**.
3. Supprimez le TextView qui affiche "Hello World".
4. Ajoutez les attributs **android:orientation** et **android:padding** comme suit:

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:padding="16dp"
    tools:context=“packageName.MainActivity”>
```

Partie 1

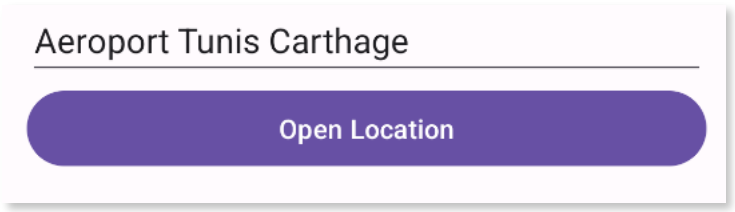


Pour la première partie du layout, Ajoutez un **EditText** et un **bouton** ayant les attributs suivants:

Attributs de l’EditText	Valeurs
android:id	"@+id/website_edittext"
android:layout_width	"match_parent"
android:layout_height	"wrap_content"
android:text	"@string/edittext_uri"

Attributs du bouton	Valeurs
android:id	"@+id/open_website_button"
android:layout_width	"wrap_content"
android:layout_height	"wrap_content"
android:layout_marginBottom	"24dp"
android:text	"@string/button_uri"
android:onClick	"openWebsite"

Partie 2



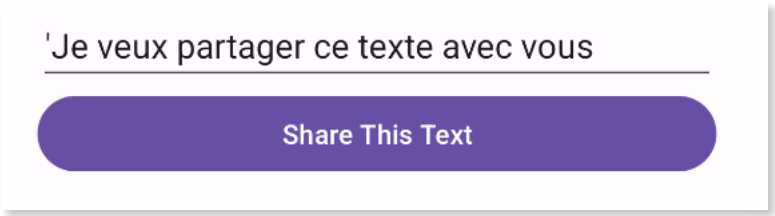
Pour la deuxième partie du layout, Ajoutez un **EditText** et un **bouton** ayant les attributs suivants:

Attributs de l'EditText	Valeurs
android:id	"@+id/location_edittext"
android:text	"@string/edittext_loc"

Attribut du bouton	Valeur
android:id	"@+id/open_location_button"

android:text	"@string/button_loc"
android:onClick	"openLocation"

Partie 3



Pour la deuxième partie du layout, Ajoutez un **EditText** et un **bouton** ayant les attributs suivants:

Attribut de l'EditText	Valeurs
android:id	"@+id/share_edittext"
android:text	"@string/edittext_share"

Attribut du bouton	Valeurs
android:id	"@+id/share_text_button"
android:text	"@string/button_share"
android:onClick	"shareText"

Le code du fichier **activity_main.xml**:

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
```

```
android:padding="16dp"
tools:context="com.example.android.implicitintents.MainActivity">

<EditText
    android:id="@+id/website_edittext"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
     android:text="@string/edittext_uri"/>

<Button
    android:id="@+id/open_website_button"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
     android:layout_marginBottom="24dp"
    android:text="@string/button_uri"
    android:onClick="openWebsite"/>

<EditText
    android:id="@+id/location_edittext"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:text="@string/edittext_loc"/>

<Button
    android:id="@+id/open_location_button"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginBottom="24dp"
    android:text="@string/button_loc"
    android:onClick="openLocation"/>

<EditText
    android:id="@+id/share_edittext"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:text="@string/edittext_share"/>

<Button
    android:id="@+id/share_text_button"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginBottom="24dp"
    android:text="@string/button_share"
    android:onClick="shareText"/>
```

</LinearLayout>

Tâche 2 : Implémenter le bouton Ouvrir le site Web

Dans cette tâche, vous implémentez la méthode de gestion au clic pour le premier bouton de la présentation, **Open Website**. Cette action utilise un intent implicite pour envoyer l'URI donné à une activité qui peut gérer cet Intent implicite (comme un navigateur Web).

Définir openWebsite( )

1. Ajoutez la méthode openWebsite dans la classe MainActivity.

```
public void openWebsite(View view) {
}
```

2. Dans MainActivity, ajoutez une variable privée en haut de la classe pour récupérer le texte saisi dans l'EditText.

```
private EditText mWebsiteEditText;
```

3. Dans **onCreate()**, utilisez **findViewById()** pour obtenir une référence à l'instance de l'EditText et l'attribuer à cette variable privée :

```
mWebsiteEditText = findViewById(R.id.website_edittext);
```

Ajoutez du code à openWebsite( )

4. Ajoutez une instruction à la nouvelle méthode openWebsite() qui obtient la valeur de chaîne de l'EditText :

```
String url = mWebsiteEditText.getText().toString();
```

5. Encodez et analysez cette chaîne dans un objet Uri :

```
Uri webpage = Uri.parse(url);
```

6. Créez un nouvel Intent ayant l'action Intent.ACTION_VIEW et l'URI comme suit :

```
Intent intent = new Intent(Intent.ACTION_VIEW, webpage);
```

Ce constructeur de Intent est différent de celui que vous avez utilisé pour créer un Intent explicite.
Dans ce constructeur, vous spécifiez une action et les données pour cette action. Les actions sont définies par la classe Intent et peuvent inclure **ACTION_VIEW**(pour afficher

les données), **ACTION_EDIT**(pour modifier les données) ou **ACTION_DIAL**(pour composer un numéro de téléphone).
Dans ce cas, on choisit l'action **ACTION_VIEW** car vous souhaitez afficher la page Web spécifiée par l'URI.

7. Dans le bloc **if**, appelez **startActivity()** pour envoyer l'Intent.

```
startActivity(intent);
```

La méthode **openWebsite()** devrait maintenant ressembler à ceci.

```
public void openWebsite(View view) {
  // Récupère le texte de l'URL
  String url = mWebsiteEditText.getText().toString();
  // Analyse l'URI et crée l'intention.
  Uri webpage = Uri.parse(url);
  Intent intent = new Intent(Intent.ACTION_VIEW, webpage);
  // Démarrez cette activité.
     startActivity(intent);
 }
```

Tâche 3 : implémenter le bouton Ouvrir l'emplacement

Dans cette tâche, vous implémentez la méthode de gestion au clic pour le deuxième bouton de l'interface utilisateur, **Open Location**. Cette méthode est presque identique à la méthode **openWebsite()**. La différence réside dans l'utilisation d'un URI géographique pour indiquer un emplacement sur la carte. Vous pouvez utiliser un URI géographique avec la latitude et la longitude, ou utiliser une chaîne de requête pour un emplacement général.

Définir openLocation()

1. Ajoutez la méthode **openLocation** dans la classe MainActivity.

```
public void openLocation(View view) {
}
```

2. Dans MainActivity, ajoutez une variable privée en haut de la classe pour récupérer le texte saisi dans l'EditText.

```
private EditText mLocationEditText;
```

3. Dans **onCreate()**, utilisez findViewByID() pour obtenir une référence à l'instance de l'EditText et l'attribuer à cette variable privée :

```
mLocationEditText = findViewById(R.id.location_edittext);
```

Ajouter du code à openLocation( )

4. Dans la nouvelle méthode **openLocation()**, ajoutez une instruction pour obtenir la valeur de chaîne du **mLocationEditText**.

```
String loc = mLocationEditText.getText().toString();
```

5. Analysez cette chaîne dans un objet Uri avec une requête de recherche géographique :

```
Uri addressUri = Uri.parse("geo:0,0?q=" + loc);
```

6. Créez un nouvel Intent ayant l'action Intent.ACTION_VIEW et l'URI comme suit :

```
Intent intent = new Intent(Intent.ACTION_VIEW, addressUri);
```

7. Démarrez l'activité:

```
startActivity(intent);
```

La méthode **openLocation()** devrait maintenant ressembler à ceci :

```
public void openLocation(View view) {
 // Récupère la chaîne indiquant un emplacement. La saisie n'est pas validée;
 // c'est transmis au gestionnaire d'emplacement.
 String loc = mLocationEditText.getText().toString();
 // Analyse l'emplacement et crée l'intention.
 Uri addressUri = Uri.parse("geo:0,0?q=" + loc);
 Intent intent = new Intent(Intent.ACTION_VIEW, addressUri);
 // Démarre cette activité.
  startActivity(intent);
}
```

Tâche 4 : implémenter le bouton Partager ce texte

Une action de partage est un moyen simple pour les utilisateurs de partager des éléments de votre application avec les réseaux sociaux et d'autres applications. Bien que vous puissiez créer une action de partage dans votre propre application à l'aide d'un Intent implicite, Android fournit la classe d'assistance **ShareCompat.IntentBuilder** pour faciliter la mise en œuvre du partage.
Vous pouvez utiliser **ShareCompat.IntentBuilder** pour créer l'Intent et lancer un sélecteur pour permettre à l'utilisateur de choisir l'application de destination à partager.

Dans cette tâche, vous implémentez le partage d'un morceau de texte, à l'aide de la classe **ShareCompat.IntentBuilder**.

Définir shareText( )

1. Ajoutez la méthode **shareText** dans la classe MainActivity.

```
public void shareText(View view) {
}
```

2. Ajoutez une variable privée en haut de MainActivity pour contenir l'EditText.

```
private EditText mShareTextEditText;
```

3. Dans **onCreate()**, utilisez **findViewById()** pour obtenir une référence à l'instance de l'EditText et l'attribuer à cette variable privée :

```
mShareTextEditText = findViewById(R.id.share_edittext);
```

Ajouter du code à shareText( )

1. Dans la nouvelle méthode shareText(), ajoutez une instruction pour obtenir la valeur de chaîne de mShareTextEditText.

```
String txt = mShareTextEditText.getText().toString();
```

2. Définissez le type **MIME** du texte à partager :

```
String mimeType = "text/plain";
```

Le type **M**ultipurpose **I**nternet **M**ail **E**xtensions (MIME) est un standard permettant d'indiquer la nature et le format d'un document.

3. Appelez **ShareCompat.IntentBuilder** avec ces méthodes :

```
new ShareCompat.IntentBuilder(MainActivity.this)
        .setType(mimeType)
        .setChooserTitle(R.string.share_text_with)
        .setText(txt)
        .startChooser();
```

4. Extrayez la valeur de **.setChoosterTitle** dans une ressource de chaîne (**strings.xml**).

L'appel à **ShareCompat.IntentBuilder** utilise ces méthodes :

Méthode	Description
setType()	Le type MIME de l'élément à partager.
setChooserTitle()	Le titre qui apparaît dans le sélecteur d'application système.
setText()	Le texte réel à partager
startChooser()	Affichez le sélecteur d'application système et envoyez le fichier Intent.

Ce format, avec toutes les méthodes de définition du constructeur regroupées dans une seule instruction, constitue un moyen simple et abrégé de créer et de lancer l'Intent. Vous pouvez ajouter n'importe laquelle des méthodes supplémentaires à cette liste.

La méthode **shareText()** devrait maintenant ressembler à ceci :

```
public void shareText(View view) {
    String txt = mShareTextEditText.getText().toString();
    String mimeType = "text/plain";
    new ShareCompat.IntentBuilder(MainActivity.this)
            .setType(mimeType)
            .setChooserTitle(R.string.share_text_with)
            .setText(txt)
            .startChooser();
}
```