Polycopié de cours pour le module Applications mobiles

Chapitre V

Les intents et intents filters

V.1 Les intentions (Intents)

Une intention (un "Intent") permet de lancer une activité dans une autre application en décrivant une action simple qu'on souhaite effectuer. il s'agit ici d'un intent dit implicit, car on ne spécifie pas le nom de l'activité mais l'action. En appelant `statrtActivity()` ou `statrtActivityForResult()` , Le système fait la **résolution** de l'intent.

On décrira ici quelques intents implicites organisées suivant le type d'application. Chaque section montre également comment créer un filtre d'intention pour rendre l'application susceptible à recevoir ce même intent et donc capable à effectuer la même action.

- Note : on doit Vérifier d'abord qu'il existe une application pour recevoir l'intention en faisant appel à `resolveActivity()` sur l'objet Intent.

Lorsque vous créez un explicite intent pour démarrer une activité ou un service, le système lance immédiatement le composant spécifié dans l'objet Intent. Lorsque vous créez un implicite intent, le système Android trouve le composant approprié à démarrer en comparant le contenu de l'intention aux filtres d'intention déclarés dans le fichier manifeste d'autres applications sur le périphérique. Si l'intention correspond à un filtre d'intention particulier, le système démarre ce composant et lui délivre l'objet Intent.

V.2 les filtres d'intention

Un filtre d'intention est une expression dans le fichier manifeste d'une application qui spécifie le type d'intentions que le composant souhaite recevoir. Par exemple, en déclarant un filtre d'intention dans une activité, vous permettez à d'autres applications de démarrer directement votre activité avec un type spécifique d'intention. De même, si vous ne déclarez aucun filtre d'intention pour une activité, il ne peut être lancé qu'avec une intention explicite.

Un objet Intent contient des informations que le système Android utilise pour déterminer le composant à démarrer (par exemple, le nom exact du composant ou la catégorie de composant qui doit recevoir l'intention), ainsi que les informations utilisées par le composant destinataire pour effectuer correctement l'action désiré (par exemple les données sur lesquelles on doit agir).

Les principales informations contenues dans une intention sont les suivantes:

- **Name** : Nom du composant à relancer. On peut définir le nom du composant avec `setComponent()` , `setClass()` , `setClassName()` ou avec le constructeur `Intent()` .
- **Action** : une chaine de caractère qui spécifie l'action générique à prendre.
- **Data** : L'URI (un objet Uri) qui fait référence aux données à traiter.
- **Category** : Chaîne contenant des informations supplémentaires sur le type de composant ciblé par l'intention : CATEGORY_BROWSABLE, CATEGORY_LAUNCHER, ...
- **Extras** : Paires valeur-clé qui portent les informations supplémentaires requises pour accomplir l'action demandée.

- **Exemple**
  Intention implicite pour lancer d'autres applications qui peuvent effectuer une action particulière
  (ici l'action est `ACTION_SEND`)

```
Intent send_intent = new Intent();
sendIntent.setAction(Intent.ACTION_SEND);
sendIntent.putExtra(txt_key,"This is my text to send.");
sendIntent.setType("text/plain");
```

V.3 Les actions

V.3.1 Standards Actions pour Activités

`ACTION_MAIN`

Action: signifie que l'activité est un point d'entrée principal. On ne s'attend pas à ce que l'activité va recevoir des données.

`ACTION_VIEW`

Action: affichez les données à l'utilisateur. C'est l'action la plus courante effectuée sur les données. Lorsqu'elle est utilisée sur un mailto:URI, elle affichera une fenêtre de composition remplie avec les informations fournies par l'URI; lorsqu'elle est utilisée avec un tel:URI, elle appellera le numéroteur.

`ACTION_EDIT`

Action: Fournir un accès modifiable explicite aux données.

`ACTION_PICK`

Action: sélectionnez un élément dans un ensemble de données, et renvoyer ce qui a été sélectionné.

`ACTION_DIAL`

Action: composez un numéro tel que spécifié par les données. Cela montre une interface utilisateur avec le numéro en cours de composition, permettant à l'utilisateur d'initier explicitement l'appel.

`ACTION_CALL`

Action: Effectuez un appel à quelqu'un spécifié par les données.

`ACTION_SEND`

Action: Transmettre des données à quelqu'un d'autre. il appartient au destinataire de cette action de demander à l'utilisateur où les données doivent être envoyées. Lorsque vous lancez une intention SEND, vous devez généralement l'insérer dans un sélecteur (via `createChooser(Intent, CharSequence)`), ce qui donnera à l'utilisateur la bonne interface pour choisir comment envoyer vos données.

`ACTION_SENDTO`

Action: Envoyez un message à quelqu'un spécifié par les données.

`ACTION_ANSWER`

Action: Répondre à un appel téléphonique entrant.

`ACTION_INSERT`

Action: insérer un élément vide dans le conteneur spécifié.

`ACTION_SEARCH`

Action: Effectuer une recherche.

V.3.2 Standards Actions pour Broadcast

`ACTION_TIME_CHANGED`

Action du Broadcast: l'heure a été mise à jour.

`ACTION_BOOT_COMPLETED`

Action du Broadcast: il est diffusé une fois, après qu'on a fini le boot. Il peut être utilisé pour effectuer une initialisation spécifique à l'application, telle que l'installation d'alarmes. Vous devez détenir la permission RECEIVE_BOOT_COMPLETED pour recevoir cette diffusion.

`ACTION_BATTERY_LOW`

Action du Broadcast: Indique que la charge batterie est faible. Ce Broadcast correspond à la boîte de dialogue du système "Low battery warning".

`ACTION_BATTERY_OKAY`

Action du Broadcast: Indique que la batterie est maintenant chargée après avoir été faible. Renvoyé après `ACTION_BATTERY_OKAY` et une fois que la batterie est revenue à un état correct.

`ACTION_POWER_CONNECTED`

Action du Broadcast: Une alimentation externe a été connectée à l'appareil. Ceci est destiné aux applications qui souhaitent s'inscrire spécifiquement à cette notification. Contrairement à `ACTION_BATTERY_CHANGED` , les applications seront réveillées pour cela et n'auront donc pas à rester actives pour recevoir cette notification. Cette action peut être utilisée pour implémenter des actions qui attendent que l'alimentation soit disponible pour se déclencher.

V.3.3 Autres Actions

`ACTION_POWER_CONNECTED,ACTION_CALL_BUTTON,`
`ACTION_CAMERA_BUTTON, ACTION_DATE_CHANGED,`
`ACTION_DREAMING_STARTED, ACTION_DREAMING_STOPPED,`
`ACTION_NEW_OUTGOING_CALL, ACTION_PASTE,`
`ACTION_SCREEN_OFF, ACTION_SCREEN_ON.`

V.4 Le champ DATA

L'URI (un objet Uri) qui référence les données à traiter et/ou le type **MIME** de ces données. Le type de données fournies est généralement dicté par l'action de l'intention. Par exemple, si l'action est **ACTION_EDIT** , les données doivent contenir l'URI du document à modifier/éditer.

Lors de la création d'une intention, il est souvent important de spécifier le type de données (son type MIME) en plus de son URI. Par exemple, une activité capable d'afficher des images ne sera probablement pas capable de lire un fichier audio, même si les formats URI peuvent être similaires. La spécification du type MIME de vos données aide le système Android à trouver le meilleur composant pour recevoir votre intention. Cependant, le type MIME peut parfois être déduit de l'URI, en particulier lorsque les données sont de la forme "content:URI" . Cela indique que les données se trouvent sur le device et sont contrôlées par un ContentProvider, ce qui rend le type MIME de données visible pour le système.

Quelques exemples pour **mimeType**

application/octet-stream, application/pdf, application/zip, audio/mpeg, audio/x-ms-wma, image/jpeg, image/png, text/plain, text/xml, video/mp4, video/webm

V.5 Le champ Category

Chaîne contenant des informations supplémentaires sur le type de composant associé à l'intention. Plusieurs descripteurs de catégorie peuvent être placés dans une intention, mais la plupart des intentions n'ont pas besoin d'une spécification pour Category.

Exemples de valeurs pour le champ Category:

- `CATEGORY_BROWSABLE`
  L'activité cible peut être démarrée par un navigateur Web pour afficher des données référencées par un lien ou un message électronique.
- `CATEGORY_LAUNCHER`
  L'activité est l'activité initiale d'une tâche et est répertoriée dans le module de lancement des applications.

Pour recevoir des intentions implicites, on doit assigner `CATEGORY_DEFAULT` comme attribut pour Category dans le filtre d'intention

Code dans le manifeste :

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

Autres Exemples de Category :

- `CATEGORY_DEFAULT`
- `CATEGORY_BROWSABLE`
- `CATEGORY_TAB`
- `CATEGORY_ALTERNATIVE`
- `CATEGORY_SELECTED_ALTERNATIVE`
- `CATEGORY_INFO`
- `CATEGORY_HOME`
- `CATEGORY_PREFERENCE`

V.6 Le champ Extras

Des paires clé-valeur contenant des informations supplémentaires requises pour accomplir l'action demandée. Tout comme certaines actions qui utilisent des types particuliers d'URI de données, certaines actions utilisent également des extras particuliers On peut ajouter des données supplémentaires de diverses méthodes telles que `putExtras()` , chacune acceptant deux paramètres: le nom de la clé et la valeur. On peut également créer un objet Bundle avec toutes les données supplémentaires, puis insérer l'ensemble dans l'intention avec `putExtras()` .

V.7 Permettre à d'autres applications de faire démarrer votre activité

Si votre application peut assurer une action utile à d'autres applications, elle pourra répondre à des demandes d'activation en spécifiant un filtre d'intention approprié.

Par exemple, une application réseau social qui doit partager des messages ou des photos avec les amis de l'utilisateur, doit prendre en charge l'intention `ACTION_SEND`. Si, les utilisateurs lancent une action "partage" à partir d'autres applications, votre application apparaît en tant qu'option dans la boîte de dialogue du sélecteur (également appelée "boîte de dialogue de désambiguïsation").

On a à ajouter un <intent-filter> dans le fichier manifeste pour l'élément <activity> correspondant.

Code dans le manifeste :

```
<activity android:name="ShareActivity">
     <intent-filter>
          <action android:name="android.intent.action.SEND"/>
          <category
android:name="android.intent.category.DEFAULT"/>
          <data android:mimeType="text/plain"/>
          <data android:mimeType="image/*"/>
     </intent-filter>
</activity>
```

Si deux paires d'actions et de données s'excluent mutuellement dans leurs comportements, vous devez créer des filtres d'intention distincts pour spécifier les actions acceptables lorsqu'elles sont associées à des types de données.

Par exemple, supposons que votre activité traite à la fois le texte et les images pour les intentions `ACTION_SEND` et `ACTION_SENDTO`. Dans ce cas, vous devez définir deux filtres d'intention distincts pour les deux actions, car une intention `ACTION_SENDTO` doit utiliser les données Uri pour spécifier l'adresse du destinataire à l'aide du schéma d'envoi ordinaire ou d'envoi `URI`.

```
   <activity android:name="ShareActivity">
        <!-- filter for sending text; -->
        <!-- accepts SENDTO action with sms URI schemes -->
        <intent-filter>
            <action android:name="android.intent.action.SENDTO"/>
            <category
android:name="android.intent.category.DEFAULT"/>
            <data android:scheme="sms"/>
            <data android:scheme="smsto"/>
        </intent-filter>

        <!-- filter for sending text; -->
        <!-- accepts SEND action and text or image data -->
        <intent-filter>
            <action android:name="android.intent.action.SEND"/>
            <category
android:name="android.intent.category.DEFAULT"/>
            <data android:mimeType="image/*/>
            <data android:mimeType="text/plain/>
        </intent-filter>
   </activity>
```

V.8 Intents communs

V.8.1 Création d'alarme

Pour créer une nouvelle alarme, utilisez l'action `ACTION_SET_ALARM` et spécifiez les détails de l'alarme tels que l'heure et le message en utilisant des extras.

Action

ACTION_SET_ALARM

Data URI

None.

MIME Type

None.

Extras

EXTRA_HOUR	
EXTRA_MINUTES	
EXTRA_MESSAGE	
EXTRA_DAYS	Pour une alarme unique "hebdomadaire", ne spécifiez pas cet extra.
EXTRA_RINGTONE	Un contenu:URI spécifiant une sonnerie à utiliser avec l'alarme.
EXTRA_VIBRATE	Un boolean spécifiant s'il faut vibrer pour cette alarme.
EXTRA_SKIP_UI	Un boolean spécifiant si l'application devrait ignorer son IU lors de la mise d'alarme.

◦ Exemple

```
  public void createAlarm(String message,
                                int hour, int minutes) {

    Intent alrm_intent = new
Intent(AlarmClock.ACTION_SET_ALARM);
    alrm_intent.putExtra(AlarmClock.EXTRA_MESSAGE),message);
    alrm_intent.putExtra(AlarmClock.EXTRA_HOUR),hour);
    alrm_intent.putExtra(AlarmClock.EXTRA_MINUTES),minutes);
    if(alrm_intent.resolveActivity(getPackageManager()))!=null)
        {
            startActivity(alrm_intent);
         }
    }
```

◦ **Remarque**

L'application doit avoir l'autorisation SET_ALARM via la déclaration suivante dans le manifeste

```
<uses-permission
 android:name="com.android.alarm.permission.SET_ALARM" />
```

◦ Et pour le filtre d'intention

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

V.8.2 Création d'un timer

Pour créer un compte à rebours, utilisez l'action `ACTION_SET_TIMER`

Action

`ACTION_SET_TIMER`

Data URI

None.

MIME Type

None.

Extras

`EXTRA_LENGTH`	Durée du timer en secondes.
`EXTRA_MESSAGE`	A custom message to identify the timer.
`EXTRA_SKIP_UI`	Un boolean spécifiant si l'application devrait ignorer son IU lors du réglage de l'alarme.

- **Exemple**

```
            public void startTimer(String message, int seconds)
{

    Intent timr_intent = new
Intent(AlarmClock.ACTION_SET_TIMER);
    timr_intent.putExtra(AlarmClock.EXTRA_MESSAGE),message);
    timr_intent.putExtra(AlarmClock.EXTRA_LENGTH),seconds);
    timr_intent.putExtra(AlarmClock.EXTRA_SKIP_UI),true);
    if(timr_intent.resolveActivity(getPackageManager()))!=null)
        {
            startActivity(timr_intent);
         }
    }
```

- **Et pour le filtre d'intention**

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

V.8.3 Le calendrier

Pour ajouter un nouvel événement au calendrier de l'utilisateur, utilisez l'action ACTION_INSERT et spécifiez l'URI de données avec Events.CONTENT_URI. Vous pouvez ensuite spécifier différents détails d'événements en utilisant les extras définis ci-dessous.

Action

ACTION_INSERT

Data URI

Events.CONTENT_URI

Extras

EXTRA_EVENT_ALL_DAY

boolean variable spécifiant si l'événement concerne tous les jours de la semaine

EXTRA_EVENT_BEGIN_TIME

Date du début pour l'l'événement (en millisecondes)

EXTRA_EVENT_END_TIME

Date de fin pour l'événement (en millisecondes).

TITLE

Titre

DESCRIPTION

Description

EVENT_LOCATION

lieu d'occurrence de l'événement

EXTRA_EMAIL

List des e-mails des invités

- **Exemple**

```
public void addEvent(String title,
                     String location,long begin,long end) {

  Intent evnt_intent = new Intent(Intent.ACTION_INSERT);
  evnt_intent.setData(Events.CONTENT_URI);
  evnt_intent.putExtra(Events.TITLE),title);
  evnt_intent.putExtra(Events.EVENT_LOCATION),location);
  evnt_intent.putExtra(CalendarContract.EXTRA_EVENT_BEGIN_TIME,
                                    begin);
  evnt_intent.putExtra(CalendarContract.EXTRA_EVENT_END_TIME,
                                    end);
  if(evnt_intent.resolveActivity(getPackageManager()))!=null)
     {
        startActivity(evnt_intent);
     }
  }
```

- **Et pour le filtre d'intention**

```
<activity ...>
    <intent-filter>
        <action android:name="android.intent.action.INSERT"/>
        <data android:mimeType="vnd.android.cursor.dir/event"/>
        <category android:name="android.intent.category.DEFAULT"/
    </intent-filter>
</activity>
```

V.8.4 Capture Photo et Vidéo avec La Camera

Utiliser l'action ACTION_IMAGE_CAPTURE ou ACTION_VIDEO_CAPTURE.

Action

ACTION_IMAGE_CAPTURE or ACTION_VIDEO_CAPTURE

Data URI

None

MIME Type

None

Extras

EXTRA_OUTPUT

L'emplacement URI où l'application de la caméra doit sauvegarder la photo ou le fichier vidéo (en tant qu'objet Uri).

Lorsque l'application camera (photo) rend le focus à l'activité (le callback onActivityResult () donc en action), vous pouvez accéder à la photo ou à la vidéo à l'URI que vous avez spécifié avec la valeur EXTRA_OUTPUT.

◦ Exemple

```
static final int REQUEST_IMAGE_CAPTURE = 1;
static final Uri mLocationForPhotos;

public void capturePhoto(String targetFilename) {

    Intent cam_intent = new Intent(Intent.ACTION_IMAGE_CAPTURE);
    cam_intent.putExtra(MediaStore.EXTRA_OUTPUT,
        Uri.withAppendedPath(mLocationForPhotos,targetFilename));
    if(cam_intent.resolveActivity(getPackageManager())!=null)
        {
            startActivityForResult(cam_intent,REQUEST_IMAGE_CAPTURE
         }
    }
```