# Annexe 5B – Intents implicites

Le téléphone / AVD contient plusieurs applications de base qu'on peut utiliser à partir de nos propres applications ( internet, géolocalisations, liste de contacts, le téléphone lui-même )

Afin de performer ces opérations dans nos projets, il est préférable de recourir à ces applications plutôt que de les recoder nous-mêmes. Étant donné l’Action demandée, Android répondra avec la ou les applications permettant de répondre à cette Action. On utilise des Intents implicites pour ce faire par opposition à des intents explicites utilisés lorsqu’on dit nous-mêmes vers quelle autre activité on veut se rendre.

Les objets Intent implicites qu'on doit créer dans ces situations sont constitués de deux paramètres la plupart du temps :

* Une Action : une constante représentant ce qu'on veut faire
* Des données ( data ) : les données qu'on a besoin de fournir afin de réaliser l'action ci-haut

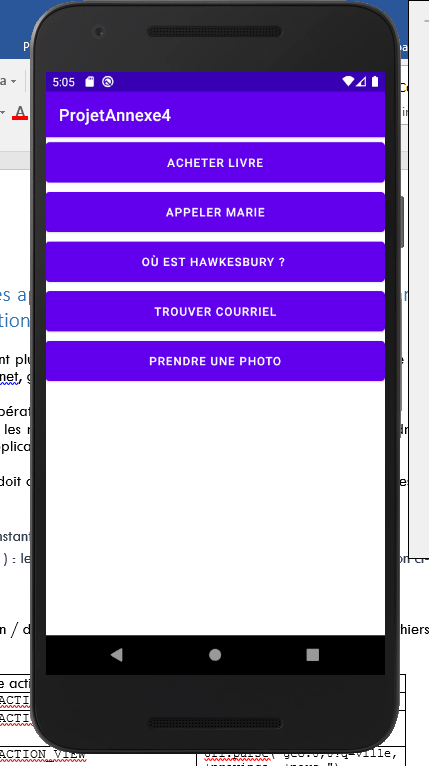
Voici quelques couples action / données utiles; il en existe un très grand nombre dans les fichiers de l'API:

|  |  |  |
| --- | --- | --- |
| Type de l'action | Constante action | Données |
| Ouvrir une page web | Intent.ACTION\_VIEW | Uri.parse( "http://…") |
| Faire un appel téléphonique | Intent.ACTION\_DIAL | Uri.parse( "tel:+789789") |
| Faire afficher un endroit sur maps.google ( **l'AVD doit supporter Google APIS** ) | Intent.ACTION\_VIEW | Uri.parse("geo:0,0?q=ville, +province, +pays ") |
|  |  |  |
| Prendre une photo | MediaStore.ACTION\_IMAGE\_CAPTURE | Pas de données à passer, l’appareil photo est démarré et attend notre prise |
|  |  |  |
| Choisir dans sa liste de contacts | Intent.ACTION\_PICK | Pas de données à passer, on doit choisir un contact |

On peut également passer les données nécessaires à la réussite de la demande à l'aide de la méthode setData .

On peut avoir à déterminer le type de données qu'on s'attend à recevoir suite à une intention. On peut le faire à l'aide de la méthode setType.

Ex. : intention.setType ( ContactsContract.Contacts.CONTENT\_TYPE) retournera les résultats ( data ) sous un type MIME permettant de regrouper toutes les infos d'un contact.



**Exercice : Quoi faire aujourd'hui ?**

1) Développez l'application ci-contre ( Version 1 ) composée de 5 Buttons et d’un ImageView en bas ( pour la photo ). Partagez l'espace avec des poids et un LinearLayout

2) Dans le fichier .java, gérez les événements onClick à l'aide de notre bonne vieille méthode **des 3 étapes.**

3) Faites en sorte qu'un clic sur le premier bouton fasse apparaître le site **leslibraires.ca**

4) Faites en sorte qu'un clic sur le 2ème bouton démarre un appel téléphonique

5) Faites en sorte qu'un clic sur le 3 ème bouton fasse apparaître une carte de la ville de Hawkesbury (**Google APIs doit être installé sur le téléphone / émulateur** )

6) Envoyer la requête pour prendre une photo ( 5e bouton ) à l’aide d’un nouvel Intent en utilisant un Intent implicite et avec la technique du « boomerang » On démarre donc l’appareil photo et on attend le résultat ( la photo sous forme de miniature ) dans l’activité présente.

Les permissions : [site d’Android Developers](https://developer.android.com/guide/topics/permissions/overview.html#normal-dangerous)

Attention, bug : <https://issuetracker.google.com/issues/37063818>

**NB On a besoin de mentionner que notre app utilisera la caméra du téléphone. On le fait dans le fichier manifest.mf**

<uses-feature android:name="android.hardware.camera"

android:required="true" />

**On doit également demander explicitement la permission à l’usager depuis la version API23 :**

<**uses-permission android:name="android.permission.CAMERA"** />

**Demander la permission….on fait la même procédure qu’avec les fichiers musicaux :**

1. Dans le onCreate, lors de l’installation :

**if** (Build.VERSION.***SDK\_INT*** >= Build.VERSION\_CODES.***M***) *//depuis l'API 23*{

**if** (checkSelfPermission(Manifest.permission.***CAMERA***) != PackageManager.***PERMISSION\_GRANTED***)  
 requestPermissions(**new** String[]{Manifest.permission.***CAMERA***}, 666);  
 *//demander permission*}

1. On attend la réponse de l’usager qui arrive dans cette méthode ( un callback )

**public void** onRequestPermissionsResult ( **int** requestCode, String[]permissions, **int** [] grantResults)  
{  
  
 **if** ( requestCode == 666 )  
 **if** ( grantResults[0] == PackageManager.***PERMISSION\_GRANTED***)  
 Toast.*makeText*(**this**, **"Oui, utiliser la caméra"**, Toast.***LENGTH\_LONG***).show();  
}

Codez les lignes suivantes au retour du boomerang

Bundle extras = data.getExtras();

Bitmap imageBitmap = (Bitmap) extras.get("data");

mImageView.setImageBitmap( imageBitmap ); // pour l’afficher, mImageView est le nom de mon ImageView

Si vous testez avec votre téléphone, ça fonctionne bien.

Pour plus d’infos sur les photos : <http://developer.android.com/training/camera/photobasics.html>

Une image contenant texte, moniteur, équipement électronique, intérieur

Description générée automatiquement

**EXTRA / BONUS TERMINER L’ANNEXE AVANT**

**7) Trouver courriel**

Ajoutez 2 contacts à votre AVD

Faites en sorte qu'un clic sur le bouton « trouver courriel » permette d'afficher la liste de contacts présents sur cet AVD.

**NB On a besoin d’avoir une permission pour indiquer qu’on va « fouiller » dans les contacts de l’usager. On doit le déclarer dans le fichier manifest.mf :**

<uses-permission android:name="android.permission.READ\_CONTACTS" />

**De la même manière que dans l’annexe 3B, depuis l’API 23, on doit également demander explicitement la permission à l’usager; ajoutez donc la condition ainsi que la méthode onRequestPermissionsResult.**

* On a besoin de déclencher l’activité externe mais on a également besoin de revenir sur notre activité présente afin d’afficher le résultat ( le courriel )
* **Pour ce faire, on doit utiliser la technique du boomerang**
* On doit donner à l’intent un type : intention.setType ( ContactsContract.Contacts.CONTENT\_TYPE) retournera les résultats ( data ) sous un type MIME permettant de regrouper toutes les infos d'un contact.
* Vous recevez le résultat dans la méthode onActivityResult de la classe Callback tel que vu précédemment. On peut retrouver l’Uri représentant l’ensemble de toutes les informations du contact comme ceci :

Uri result = data.getData();

* On peut appeler la méthode getLastPathSegment sur l’Uri afin de récupérer son id :

String id = result.getLastPathSegment(); //id = 1, 2 dans l'ordre des contacts

Ensuite, on va pouvoir récupérer les champs du contact qu’on a choisi à l’aide d’une requête sur le contentProvider de la même manière que dans les MP3 !

L’Uri à utiliser cette fois est : ContactsContract.CommonDataKinds.Email.CONTENT\_URI

En consultant l’API pour cet URI :

[**https://developer.android.com/reference/android/provider/ContactsContract.CommonDataKinds.Email**](https://developer.android.com/reference/android/provider/ContactsContract.CommonDataKinds.Email)

* obtenez le Cursor résultant à partir du id précédent qui est la valeur du champ CONTACT\_ID et récupérez les valeurs des champs nom ( DISPLAY\_NAME ) et email ( ADDRESS )

}

* Utilisez un Toast pour afficher les infos sur le nom et le email du contact sélectionné.