

PROGRAMMATION

MOBILE

3 - Intents

NIDHAL JELASSI

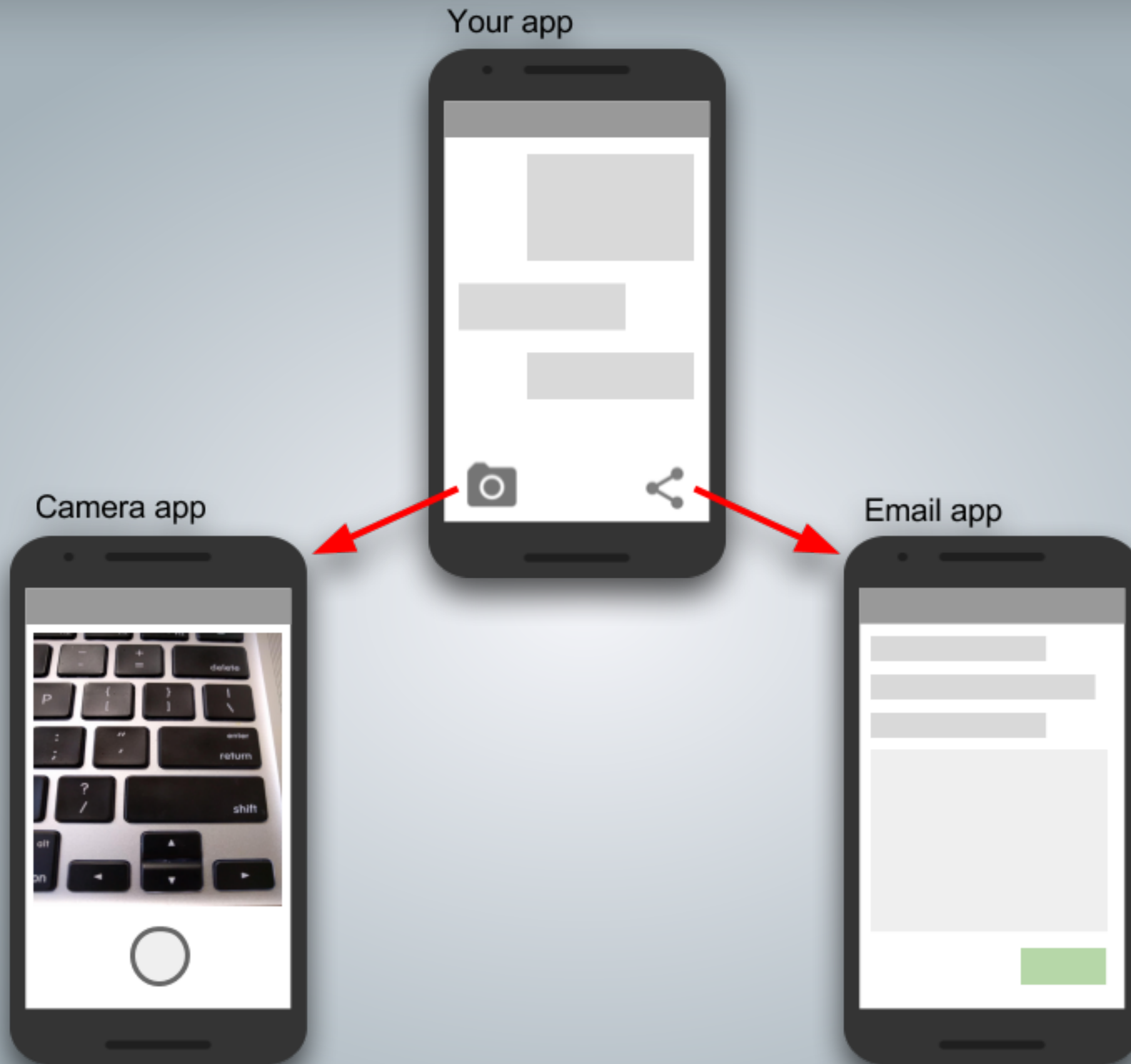
nidhal.jelassi@fsegt.utm.tn

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
 - Les activités interagissent en mode **asynchrone**.
 - Le passage d'une activité à une autre est réalisé en demandant à l'activité en cours d'exécuter un **Intent**.



Définition

- Un intent est un message qui peut être utilisé pour demander une action à partir d'un autre composant de l'application.
- Permet invoquer des Activités, des Broadcast Receivers ou des Services. Les différentes méthodes utilisées pour appeler ces composantes sont :
 - `startActivity(intent)` : lance une activité
 - `sendBroadcast(intent)` : envoie un intent à tous les composants Broadcast Receivers intéressés
 - `startService(intent)` ou `bindService(intent, ...)` : communiquent avec un service en arrière plan.

Construction d'un intent

- Un intent comporte des informations qu'Android utilise
 - **Nom du composant à démarrer**
 - **Action à réaliser**
 - ACTION-VIEW, ACTION_SEND...
 - **Donnée**
 - URI référençant la donnée sur laquelle l'action va agir
 - **Catégorie**
 - Information sur le type de composants qui va gérer l'intent
 - CATEGORY-BROWSABLE, CATEGORY-LAUNCHER...
 - **Extras**
 - Paires clef-valeur qui comportent des informations additionnelles pour réaliser l'action demandée
 - **Drapeaux (Flags)**
 - Définissent la classe qui fonctionne comme métadonnée pour cet intent

Types

- Il existe deux types d'intents :
 - **Intents Explicites**
 - Spécifient le composant à démarrer par nom (nom complet de la classe).
 - Permettent de démarrer un composant de votre propre application, car le nom de la classe est connu
 - **Intents Implicites**
 - Ne nomment pas un composant spécifique, mais déclarent une action à réaliser.
 - Permet à un composant d'une application d'appeler un composant d'une autre application.

Intent explicite

- Démarrer un 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);

Activité courante

Activité appelée

2. StartActivity(i)

Intent explicite

Les principaux arguments d'un Intent explicite sont :

- Le contexte déclenchant l'intent, soit :
 - This, si on le lance à partir de l'activité de départ,
 - `<Activity_class_name>.this`, sinon
 - La classe destination : `<Activity_class_name>.class`
- Il est donc appelé comme suit:
 1. `Intent myActivityResult = new Intent (StartClass.this, EndClass.class) ;`
 2. `startActivity (myActivityResult) ;`


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.
- Vous pouvez également fermer manuellement l'activité démarrée en réponse à une action de l'utilisateur (comme un clic sur un bouton, par exemple) avec la méthode **finish()**

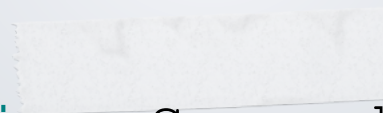
Les Extras

- Un extra est un couple clé/valeur, basé sur le système Bundle :
 - La clé est l'identifiant, on peut y mettre ce qu'on veut sous la forme d'une chaîne de caractères ;
 - La valeur est celle de la donnée. Elle sera associée à la clé.
- Ainsi, quand on crée un extra, on lui donne une clé et une valeur. En revanche, à la récupération de notre extra, c'est à travers la clé qu'on obtiendra la valeur associée.

Les Extras : Exemple




```
Intent i = new Intent(this, SecondActivity.class);
i.putExtra("zoneTexte", chaine);
i.putExtra("nbreDeMots", mots);
startActivity(i);
```




```
Intent i = new Intent(this, SecondActivity.class);
Bundle bundle = new Bundle();
bundle.putString("zoneTexte « , chaine);
bundle.putString("nbreDeMots", mots);
i.putExtras(bundle);
startActivity(i);
```


Les Extras : Exemple suite



```
Intent i = getIntent();  
if (i.hasExtra("zoneTexte")) {  
    String str = i.getStringExtra("zoneTexte");  
}  
int nbMots = i.getIntExtra("Mots", 0);
```



```
Intent i = getIntent();  
Bundle bundle = i.getExtras();  
str = bundle.getString("zoneTexte");
```

Les Datas

- L'envoi des données se fait à travers la méthode setData() de la classe Intent.
- setData() prend comme arguments un objet URI, qui représente l'emplacement de la donnée qu'on compte passer à l'intent.
- Un URI ((Unified Resource Identifier) peut être un Url (http://), un numéro de téléphone (tel://), un emplacement géographique (geo://). etc.

Les Datas : Exemple



```
Intent i = new Intent(this, SecondActivity.class);

// adresse URL
i.setData(Uri.parse("http://www.google.com"));

// un fichier
i.setData(Uri.fromFile(new File("/sdcard/sample.jpg")));

// un contenu data
i.setData(Uri.parse("content://mysample.provider/data"));

// type personnalisé
i.setData(Uri.parse("custom:" + dataID + buttonID));
```

La méthode statique `Uri.parse` permet de construire un objet URI à partir d'une chaîne de caractères.

Data et Extra

- En résumé donc : dans la première activité (d'envoi), il faut :
 1. Créer l'objet d'Intent.
 2. Mettre des **données** ou des **extras** dans cet Intent.
 3. Commencer la nouvelle activité avec startActivity().
- Dans la deuxième activité (réception), il faut :
 4. Obtenir l'objet d'Intent avec lequel l'activité a été démarrée.
 5. Récupérer les **données** ou les **extras** de l'objet Intent.

Data vs Extra

On opte pour les intent Data quand :

- Nous n'allons envoyer qu'une seule information à l'activité cible.
- L'information envoyée peut être représenté par un URI

On opte pour les intent Extras quand :

- On désire envoyer plusieurs informations (de différents types en général) à l'activité cible.
- Une de ces informations ne peut pas être envoyé à travers un URI.

Data vs Extra


- Les données des Intents et les extras ne sont pas exclusifs.
- Il est possible ainsi d'utiliser des données pour un URI et des extras pour toute information supplémentaire dont l'activité démarrée a besoin pour traiter les données dans cet URI.

StartActivityForResult


- Pour récupérer un résultat à partir d'une autre activité, il est possible d'établir un intent « bidirectionnel » entre deux activités grâce.
 - Invoquer `startActivityForResult` au lieu de `StartActivity`
- Evidemment, l'activité destination doit renvoyer un résultat une fois l'opération réalisée.
- Le résultat est envoyé sous forme d'Intent.
- L'activité principale le recevra dans un `onActivityResult`.

Intent implicite

- Démarrer un intent implicite :
 - Exemple :
 1. `String requete = "http://www.google.fr/search?q=Mobile"`
 2. `Intent intent = new Intent(Intent.ACTION_VIEW, Uri.parse(requete));`
 3. `startActivity(intent);`



Ne définit pas une application en particulier,
Android va tenter de chercher une
application s'étant définie comme capable de
répondre à l'action ACTION_VIEW



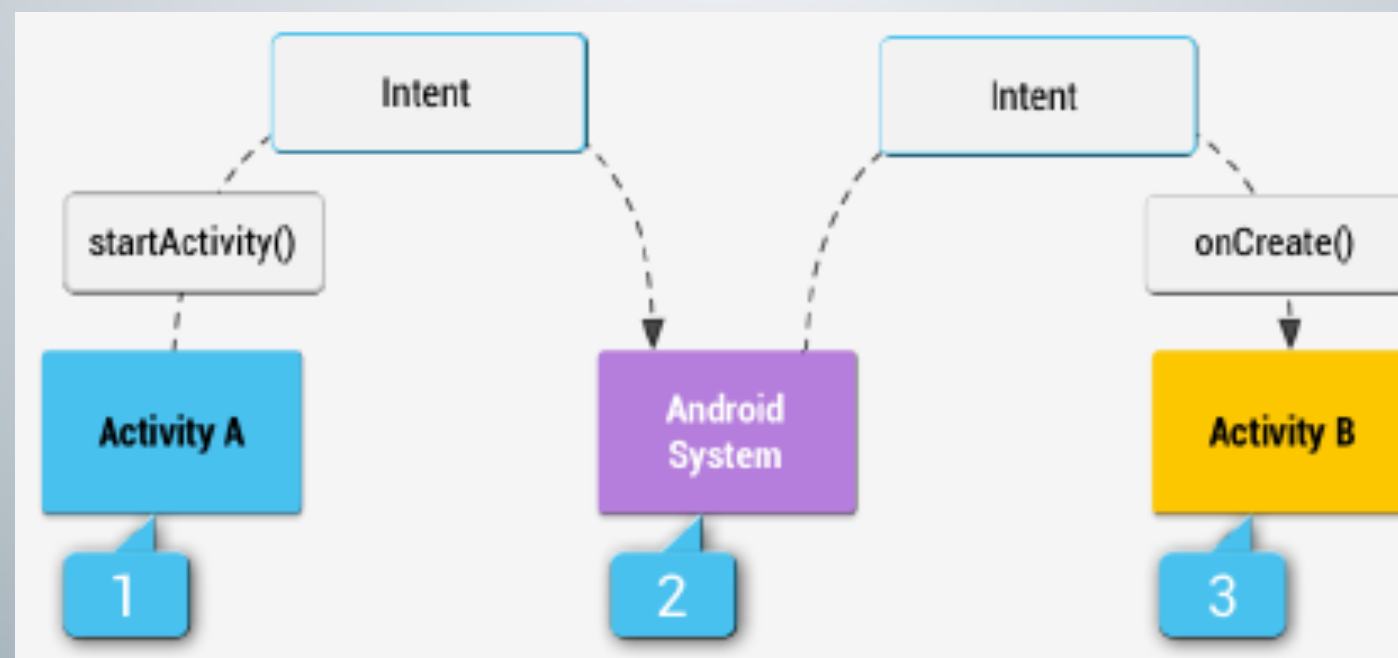
ACTION_VIEW : action définie par le
framework qui consiste à démarrer un
navigateur web sur l'Uri donnée

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 sur lesquelles on va agir, tel qu'un url ou un numéro de téléphone à appeler.
- Il est donc typiquement appelé comme suit:
 - `Intent myActivityResult = new Intent(<action>, <données>);`
 - `startActivity(myActivityResult);`

Intent implicite

- Un intent implicite se comporte comme suit:
 1. Activité A crée un Intent avec une action et le passe en paramètre à `startActivity`
 2. Le système Android cherchent toutes les applications pour trouver un **Intent Filter** qui correspond à cet Intent
 3. Quand une correspondance est trouvée, le système démarrent l'activité (Activity B) en invoquant `onCreate` et en lui passant l'intent



Actions prédéfinies

- Exemples d'actions prédéfinies communément utilisées

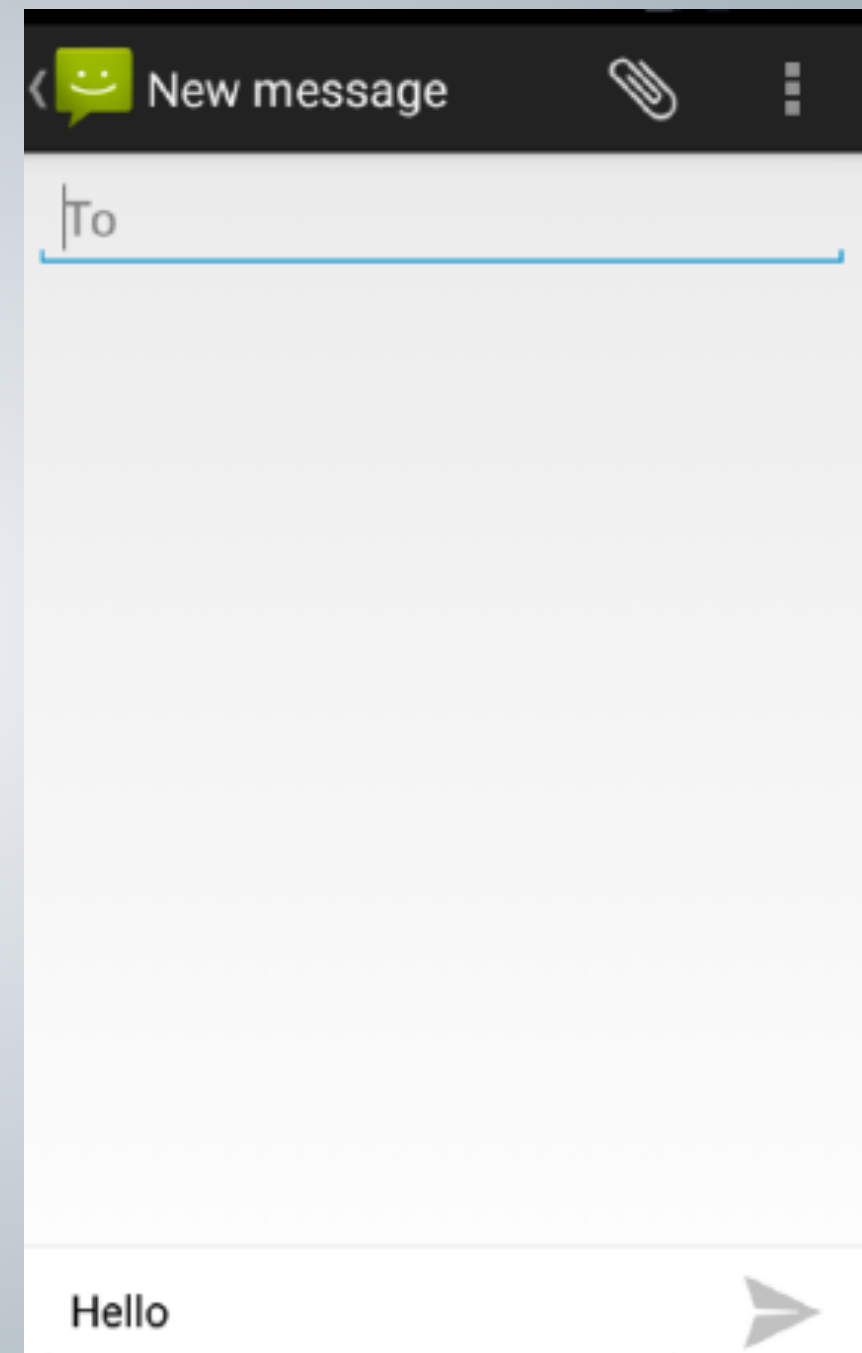
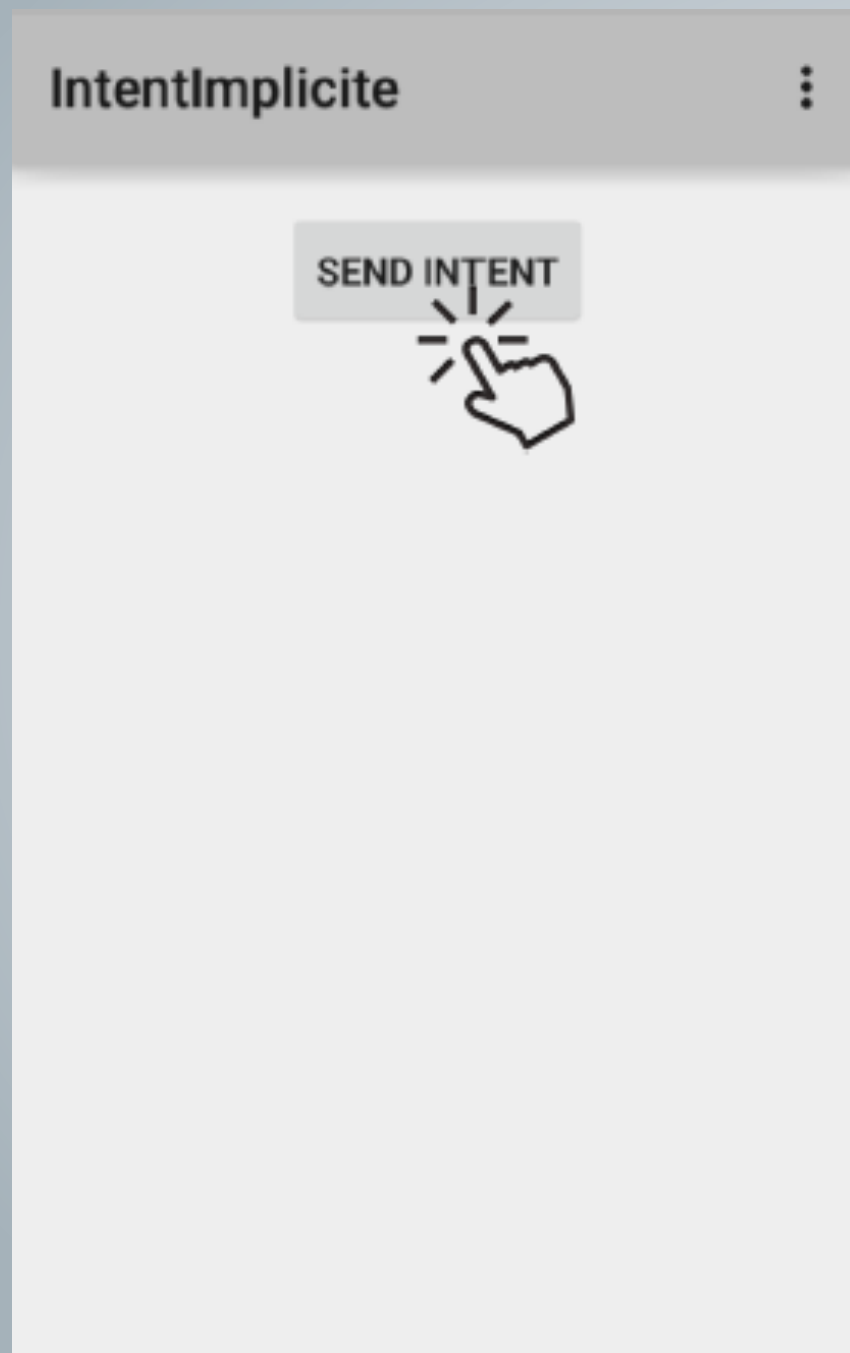
Action	Donnée	Description
ACTION_DIAL	tel:123	Affiche le numéroteur téléphonique avec le numéro (123) rempli
ACTION_VIEW	http://www.google.com	Affiche la page Google dans un navigateur.
ACTION_EDIT	content://contacts/people/2	Edite les informations sur la personne dont l'identifiant est 2 (de votre carnet d'adresse)
ACTION_VIEW	content://contacts/people/2	Utilisé pour démarrer une activité qui affiche les données du contact numéro 2
ACTION_VIEW	content://contacts/people	Affiche la liste des contacts, que l'utilisateur peut parcourir. La sélection d'un contact permet de visualiser ses détails dans un nouvel Intent.

Exemple

```
public void send(View v){  
    // Create the text message with a string  
    Intent sendIntent = new Intent();  
    sendIntent.setAction(Intent.ACTION_SEND);  
    sendIntent.putExtra(Intent.EXTRA_TEXT, "Hello");  
    sendIntent.setType("text/plain");  
  
    // Verify that the intent will resolve to an activity  
    if (sendIntent.resolveActivity(getPackageManager()) != null) {  
        startActivity(sendIntent);  
    }else{  
        Toast.makeText(this, "The send action could not be performed!", Toast.LENGTH_SHORT).show();  
    }  
}
```

Eviter que l'application crash si l'activité
appelée n'existe pas

Exemple



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.
- Si vous ne déclarez pas d'Intent Filters à votre activité, elle ne pourra pas être déclenchée par un Intent Implicite.
- Un composant d'une application doit avoir autant de filtres que de capacités de traitement. S'il peut gérer **N** types d'intent, il doit avoir **N** filtres.

Intent Filter

- La correspondance entre un intent et un filtre se fait selon trois critères :
 - L'action
 - La catégorie
 - Les données

```
<activity
  android:name=".MainActivity"
  android:label="TP3"
  android:theme="@style/AppTheme.NoActionBar">
  <intent-filter>
    <action android:name="android.intent.action.MAIN" />
    <action android:name="android.intent.action.SENDTO" />
    <category android:name="android.intent.category.LAUNCHER" />
    <data android:mimeType="text/plain" android:scheme="http" />
    <data android:mimeType="text/plain" android:scheme="https" />
  </intent-filter>
</activity>
```

Intent Filter

- 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>
```

- L'action spécifie qu'il s'agit de l'activité principale de notre application.
- La catégorie spécifie que l'activité est répertoriée dans le System's application launcher.