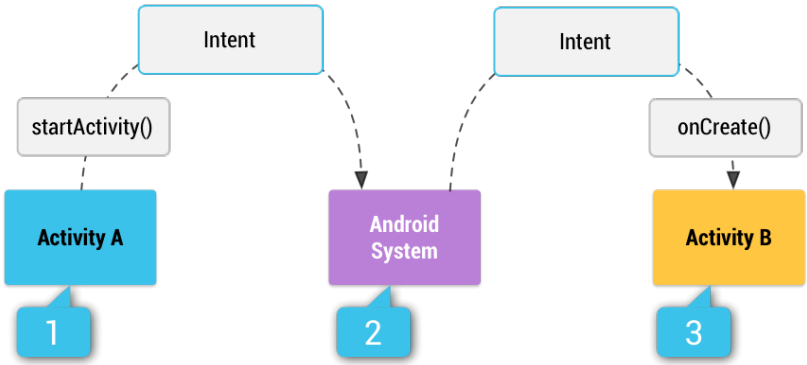
TP04 - Intents

Principe

- On crée un objet Intent
- On insérer les éventuels paramètres dans cet objet
- On envoie cet objet Intent au système (méthode startActivity)
- Le système démarre l'activité adéquate



Source : https://developer.android.com/guide/components/intents-filters.html

1. Démarrer une seconde activité depuis une première activité

On utilise la classe Intent (*intention*).

```
public class Intent
extends Object implements Parcelable, Cloneable
public Intent(Context packageContext, Class<?> cls)
```

(d'autres constructeurs existent)

TP: Créer un projet avec 2 activités

- Activité principale
  - Créez un projet avec une activité de type *Empty Activity*
    - MainActivity.java et activity_main.xml
  - Ajouter un bouton qui permettra de lancer l'activité 2
- Seconde Activité
  - Utilisez le menu File/New/Activity/Empty Activity
    - MainActivity2.java et. activity_main2.xml
  - Ajoutez un textView avec un texte différent de la première

1.1. Le fichier Manifest

- Chaque activité doit être référencée dans le fichier manifest

```
<activity android:name=".MainActivity2"></activity>
```

- a priori c'est fait automatiquement si vous avez utilisé l'assistant (le menu File/New/Activity) pour créer la seconde activité.

TP : Vérifiez en ouvrant le fichier Manifest.xml

1.2. Lancer la seconde activité

- On va d'abord créer un objet intent :

```
Intent intent = new Intent(this, MainActivity2.class);
```

TP04 - Intents

- Ensuite on démarre la seconde activité :

```
startActivity(intent);
```

TP

- Où allons-nous insérer ces lignes, dans le TP en cours ?
- Que se passe-t-il lors de l'exécution ?

Résumé

- Créez un projet avec 2 activités
- Vérifiez que la seconde activité est bien référencée dans le fichier Manifest.xml
- Dans la première activité placez un bouton « Go »
- Dans la méthode onClick, insérez créez un intent et lancez la seconde application :

```
Intent intent = new Intent(this, MainActivity2.class);
startActivity(intent);
```

- Exécutez l'application

2. Envoyer des paramètres à la seconde activité

Insérer les données à transmettre dans l'Intent en utilisant les méthodes putExtra de la classe Intent.

2.1. Préliminaire

- Reprenez le projet précédent (ou bien dupliquez-le)
- Ajoutez un champ de saisie sur la première activité pour saisir un message à transmettre
- Ajouter un textView à la seconde activité pour afficher le message transmis

2.2. Ajouter un « extra » à l'intent

La classe Intent possède de nombreuses méthodes putExtra permettant d'insérer dans l'intent des données de différents types.

Quelques exemples:

```
Intent putExtra(String name, int value)
Intent putExtra(String name, int[] value)
Intent putExtra(String name, byte value)
Intent putExtra(String name, double value)
Intent putExtra(String name, boolean value)
Intent putExtra(String name, String value)
Intent putExtra(String name, String[] value)
```

Il en existe 24 dans l'API 27, dont une pour chacun des types simples, pour les tableaux de types simples, et pour un certain nombre de classes de base (e.g. String)

Dans notre exemple, avant de lancer le startActivity(intent)

- On va récupérer le message dans le champ de texte :

```
EditText editText = (EditText) findViewById(R.id.editText);
String message = editText.getText().toString();
```

- puis on l'insère dans l'Intent

```
intent.putExtra("MSG_KEY", message);
```

TP04 - Intents

2.3. Récupérer le message dans la seconde activité

- On récupère l'intent et on va chercher l'extra

```
Intent intent = getIntent();
String message = intent.getStringExtra("MSG_KEY");
```

TP

- Où allons-nous insérer ces lignes, dans le TP en cours ?
- Que fait exactement getIntent

2.4. Méthodes de récupération des extras

Il en existe pour chacune des types de données pour lesquels existe un putExtra

```
int getIntExtra(String name, int defaultValue)
int getIntArrayExtra(String name)
byte getByteExtra(String name, byte defaultValue)
double getDoubleExtra(String name, double defaultValue)
boolean getBooleanExtra(String name, boolean defaultValue)
String getStringExtra(String name)
String[] getStringArrayExtra(String name)
```

Résumé

- Ajouter un champ de saisie à l'activité 1
- Avant le startActivity(intent) :

```
intent.putExtra("MSG_KEY", message);
```

- On récupère la valeur dans l'activité 2
- Dans le onCreate de la seconde activité :

```
Intent intent = getIntent();
String message = intent.getStringExtra("MSG_KEY");
```

3. Récupérer une valeur de retour lorsqu'on revient à la première activité

- On lance l'intent avec la méthode startActivityForResult
- On implémente la méthode onActivityResult, qui sera exécutée lorsque la seconde activité est terminée, et qu'on retourne à la première.

3.1. La méthode startActivityForResult

```
void startActivityForResult (Intent intent, int requestCode)
```

Le requestCode sera renvoyé à l'activité lors du retour.

TP : À quoi sert-il ?

3.2. La méthode onActivityResult

```
void onActivityResult (int requestCode, int resultCode, Intent data)
```

- Le resultCode est assigné dans l'activité 2 par la méthode setResult(int)
  - Par défaut (si rien n'a été assigné ou crash) ce sera la constante Activity.RESULT_CANCELED
  - Une autre valeur prédéfinie est Activity.RESULT_OK

TP04 - Intents

- Toute autre valeur est possible (code perso)

3.3. Renvoyer des données

On utilise de nouveau un Intent.

- Créer un objet Intent avec son constructeur par défaut
- Ajouter des données (putExtra)
- Assigner l'intent au résultat qui sera renvoyé, avec setResult(int, Intent)

```
Intent resultIntent = new Intent();
resultIntent.putExtra("RET_KEY", message);
setResult(Activity.RESULT_OK, resultIntent);
finish();
```

TP

- Reprenez le projet précédent (ou bien dupliquez-le de nouveau)
- Ajoutez dans la seconde activité :
  - un champ de saisie
  - un bouton OK
- Ajoutez dans la première activité un champ de texte pour recevoir le message de retour
- Terminez l'implémentation pour qu'un message saisi dans le champs de saisie soit placé dans le champs de texte
- Utilisez le code de retour (resultCode) pour savoir si l'activité 2 a terminé via le bouton OK ou via le bouton retour du système (utilisez un Toast pour indiquer le résultat).

4. Intent explicite vs implicite

- Jusqu'à maintenant on a utilisé des Intent explicites, c'est à dire qu'on a indiqué explicitement l'activité à lancer.
- Dans le cas d'Intent implicites
  - on indique seulement une action à réaliser
  - le système doit choisir quelle activité lancer
  - si plusieurs sont possibles il demande à l'utilisateur

4.1. Exemple d'intent implicite

- Envoyer une chaîne de texte
- Le système propose toutes les activités capables de recevoir une chaîne de texte, il affiche cet app chooser

TP04 - Intents

source: https://developer.android.com/guide/components/intents-filters.html

TP

- Cherchez dans la documentation comment créer un intent implicite.
- Ajoutez à votre application un bouton permettant d'envoyer un texte à une application capable de recevoir du texte

4.2 Recevoir un Intent implicite

Maintenant nous souhaitons que notre application soit capable de recevoir un Intent implicite. C'est à dire que lorsqu'une application enverra un Intent avec une action donnée, notre application soit proposée parmi les possibilités. Il faut pour cela la déclarer dans le fichier Manifest.

TP

- Retrouvez dans la documentation (même pas que précédemment) les éléments à ajouter au fichier Manifest.
- Déclarez que la seconde activité de votre appli est capable de recevoir des actions de type ACTION_SEND
- Testez, est-ce que le message est bien reçu par l'activité (retrouvez-vous bien le texte). Si non d'où peut venir le problème ? Corrigez.