THE INNOVATIVE UNIVERSITY	Cours : Web Avancé et Développement Mobile	**Année universitaire :** 202./202.

Partie II : Développement Mobile

TP3 : Gestion des Activités

Objectif :

L'objectif de ce TP est de créer une application composée de plusieurs activités et de tester l'envoi des données entre activités/applications. Une activité est une tâche au sein de l'application (avec ou sans interface utilisateur) : On peut avoir plusieurs activités dans le même programme et elle doit être déclarée dans le fichier AndroidManifest.xml. L'activité est un programme soumis à plusieurs évènements comme le montre la figure 1.

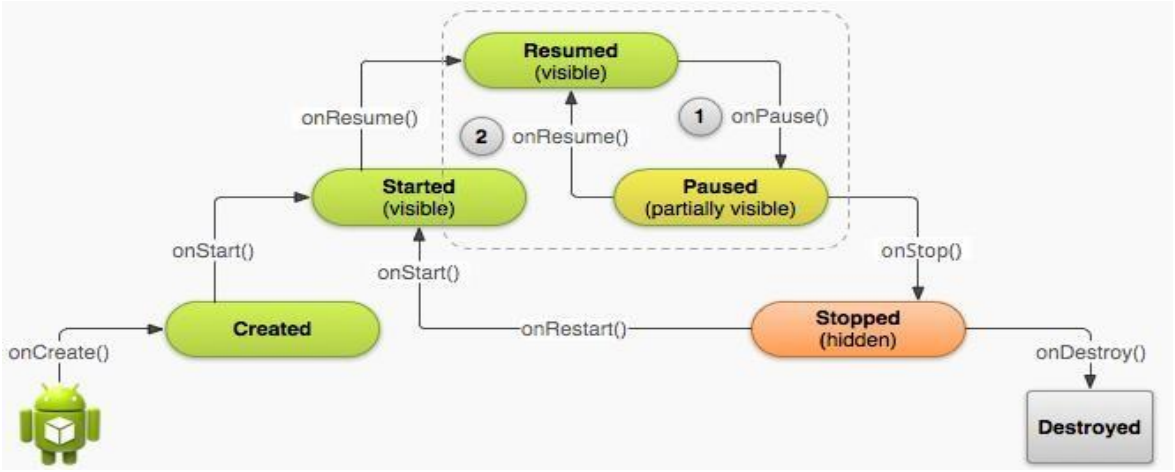

Figure 1 : Cycle de vie d'une activité

Une activité possède des callbacks (méthodes de rappel) appelés tout au long de son cycle de vie. Ces callbacks permettent de définir des passages d'un état à l'autre de l'activité par rapport à l'évolution des ressources et du comportement du système et de l'utilisateur. On peut ainsi définir des comportements particuliers à chaque état de celle-ci.

L'activité s'exécute sur un petit appareil, donc il faut que toute la mémoire disponible le soit lorsque l'utilisateur le demande pour une rapidité et une interaction immédiate avec l'utilisateur. À tout moment, une autre activité qui n'est pas de votre application peut prendre la main sur votre activité, par exemple, lorsqu'un appel est effectué pendant que votre activité est en cours d'exécution. Il existe plusieurs types d'activités :

- Activité de premier plan
- Activité visible
- Activité d'arrière-plan
- Processus sans interface graphique utilisateur (GUI) :

- Service.
- Broadcast Receivers.

Les applications Android gèrent leur état elles-mêmes, ainsi que leur mémoire, les ressources et les données. Le système Android peut mettre fin à une activité qui a été mise en pause, arrêt, ou détruite lorsqu'elle ne dispose plus de mémoire (Table 1). Cela signifie que toute activité qui n'est pas au premier plan est soumise à l'arrêt.

Méthode	Description
`onCreate()`	Appeler quand une activité démarre ou redémarre. Initialise les données statiques d'une activité. Crée des points de vue, liée des données à des listes··· Cette méthode a un argument de type `Bundle` contenant l'état précédent de l'activité. Définit l'interface graphique avec `setContentView()`.
`onRestart()`	Appeler après que l'activité s'arrête et juste avant qu'elle ne soit relancée. Toujours suivie par `onStart()`
`onStart()`	Appeler juste avant que l'activité devienne visible pour l'utilisateur. Suivi par `onResume()` si l'activité est au premier plan, ou `onStop()` s'elle est en arrière plan.
`onResume()`	Appeler juste avant que l'activité ne commence à interagir avec l'utilisateur. Toujours suivie par `onPause()`.
`onPause()`	Appeler lorsque le système est sur le point de commencer à reprendre une autre activité. Cette méthode est généralement utilisée pour valider les modifications non enregistrées de données persistantes··· Suivie, soit par `onResume()` si l'activité revient au premier plan, ou `onStop()` si elle est invisible pour l'utilisateur.
`onStop()`	Appeler lorsque l'activité n'est plus visible pour l'utilisateur. Cela peut se produire s'elle est détruite, ou si une autre activité (existante ou une nouvelle) a été repris et la couvrir. Suivie, soit par `onRestart()` si l'activité revient au premier plan, ou par `onDestroy()` s'elle est détruite.
`onDestroy()`	Appeler avant que l'activité soit détruite. Elle pourrait être appeler parceque le système a temporairement besoin d'économiser l'espace mémoire.

Table 1 : Méthodes relatives au cycle de vie d'une activité

Attribution d'écouteurs aux boutons :

Android offre deux possibilités :

- Soit vous définissez l'attribut android:onClick et vous créez une méthode dans l'activité concernée : C'est le plus facile.
- Soit vous définissez un écouteur (Listener) pour le bouton. Il faut implémenter deux taches : associer le bouton à un écouteur setOnClickListener(ecouteur) et définir l'écouteur lui-même. Pour mettre en place cette dernière taches, il y a plusieurs façons :
  - Cet écouteur est une classe privée anonyme.
  - Cet écouteur est l'activité elle-même.
  - Cet écouteur est une référence de méthode.

Écouteur mentionné dans le layout :

Il suffit de rajouter un attribut android:onClick="onValider" et une méthode du même nom dans l'activité :

```
//Il faut définir la méthode onValider dans l'activité
public void onValider(View btn) {
    ....
}
```

Écouteur classe privée anonyme :

Il faut créer une classe, lors de l'appel à setOnClickListener qui contient une seule méthode **onClick** : il faut employer la syntaxe NomActivity.this pour manipuler les variables et méthodes de l'activité sous-jacente.

```
Button button = userinter.buttonvalider;
button.setOnClickListener(
        new View.OnClickListener() {
            public void onClick(View button) {
        // faire quelque chose
            }
        });
```

Écouteur classe privée :

Il faut définir une classe privée dans l'activité qui implémente l'interface OnClickListener et à en fournir une instance en tant qu'écouteur.

```
private class EcBtnValider implements View.OnClickListener {
    public void onClick(View btn) {
    // faire quelque chose
    }
};
    public void onCreate(...) {
    ...
    Button btn = userinter.buttonvalider;
    btn.setOnClickListener(new EcBtnValider());
}
```

L'activité elle-même en tant qu'écouteur

Il faut mentionner this comme écouteur et indiquer qu'elle implémente l'interface OnClickListener: dans ce cas tous les boutons appelleront la même méthode.

```
public class EditActivity extends Activity implements View.OnClickListener {
    public void onCreate(...) {

    Button button = userinter.buttonvalider;
    button.setOnClickListener(this);
                    }
    public void onClick(View btn) {
    // faire quelque chose
    }
```

Tester les trois démarches

Notre application :

Il faut éditer la méthode onCreate de l'activité concernée et rajouter des instructions comme celles-ci. La classe Button provient de android.widget.Button et attribuer this en tant qu'écouteur des clics sur le bouton concerné:

```
Button loginButton = findViewById(R.id.loginButton);
loginButton.setOnClickListener(this);
```

Après, il faut implémenter l'interface OnClickListener :

```
public class MainActivity extends AppCompatActivity implements View.OnClickListener {
```

Ensuite il faut redéfinir la fonction onClick(View v). Son paramètre est une vue v d'un type général View, (en réalité c'est un Button.)

```
@Override
public void onClick(View v) {
    Intent intent;
    switch (v.getId()) {


        case R.id.loginButton:
            // lancer l'activité numero 1
            intent = ...
            break;
        case R.id.Anuller :
            // lancer l'activité numero 2
            intent = ...
            break;
    }
}
```

Lancement d'une activité : utiliser les Intents

Un Intent permet de définir des messages échangés avec le système Android (intentions) afin de :

- Demander l'activation d'une activité.
- Demander l'exécution d'une activité qui devra produire un résultat.
- Demander le lancement d'un service en mode "started".
- Envoyer un message à destination de BroadcastListeners.

On décompose les Intents en 2 grands types :

- Les Intents explicites : qui spécifient exactement la classe du composant applicatif cible.

```
Intent intent = new Intent( packageContext: this, NewActivity.class);
startActivity(intent);
```

  Si vous placez ces instructions dans un switch , la variable intent doit être déclarée avant le switch, ou alors créer autant de variables que de cas, ou encore : séparez l'écouteur en différentes méthodes spécifiques à chaque bouton.

- Les Intents implicites : qui ne spécifient pas la classe du composant cible, mais décrivent une cible par une action à réaliser, une catégorie, des données à manipuler… C'est le système Android qui choisit le composant applicatif approprié (exemple Je veux faire un appel téléphonique, le système va chercher une application qui me permet de le faire.)
  - Action : définit ce que le destinataire doit réaliser On peut utiliser une des actions fournies par le système, (exemple « ACTION_DIAL » pour afficher un composeur

de numéros téléphoniques,…)et il est possible aussi d'en définir de nouvelles actions propres à l'activité. Il existe de nombreuses actions standard

- ACTION_MAIN démarrage point d'entré principal
- ACTION_VIEW traitement sur données à afficher
- ACTION_DIAL clavier téléphonique avec numéro pré saisi
- ACTION_CALL effectue un appel téléphonique
- ACTION_ANSWER décrocher sur un appel entrant

- Data : les données sur lesquelles le traitement va avoir lieu, exprimées en « URI ». Il faudra donc utiliser « tel :0666 » pour téléphoner, « https ://www uir ma » pour afficher une page dans un navigateur, etc

```
btnSend.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        Intent intent = new Intent(Intent.ACTION_SEND);
        intent.setType("text/plain");
        intent.putExtra(Intent.EXTRA_TEXT,  value: "Salut tout le monde");
        intent.putExtra(Intent.EXTRA_SUBJECT,  value: "App Android");
        intent.putExtra(Intent.EXTRA_EMAIL, new String[]{"contact@uir.ma"});
        intent.putExtra(Intent.EXTRA_CC, new String[]{"utilisateur@uir.ma"});

        try {
            startActivity(intent.createChooser(intent,  title: "Choisir une application :"));
        }catch (ActivityNotFoundException e){
            e.getMessage().toString();
        }
    }
});

btnBrowse.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        Intent intent = new Intent(Intent.ACTION_VIEW, Uri.parse("https://www.uir.ma"));
        if (intent.resolveActivity(getPackageManager())!=null){
            startActivity(intent);
        }
    }
});
```

Figure 2 : Test des Intents implicites

Terminaison d'une activité

Une activité se termine si elle appelle la méthode **finish().** Dans ce cas, le bouton back ne pourra pas ramener sur elle. Relire les spécifications du début pour savoir dans quels cas les activités doivent se terminer.

Transmission d'informations entre activités :

Maintenant, pour implémenter la communication et passer une valeur d'une propriété (le nom de l'utilisateur) entre LoginActivity communique à NewsActivity : on peut soit utiliser les intents, soit définir un contexte d'application.

Extras d'intents

Dans le layout login.xml, rajoutez un EditText permettant de saisir un nom de compte. Ce qu'il faut maintenant, c'est que lorsque l'utilisateur clique sur le bouton login, l'écouteur lise d'abord le nom saisi dans l'EditText et l'envoie à NewsActivity. Pour cela, il faut utiliser un extra :

```
Intent intent = new Intent( packageContext: this, NewsActivity.class);
intent.putExtra("login", ... );
startActivity(intent);
```

De l'autre côté, dans NewsActivity, il faut récupérer cet extra et le placer dans une (nouvelle) variable membre de la classe. Pour vérifier que ça marche, rajoutez un TextView dans le layout news.xml et faites en sorte qu'il affiche le login saisi dans l'activité de connexion.
Normalement, ce login ne doit pas être oublié lors de la navigation vers DetailsActivity. Il n'est perdu que si on quitte l'activité NewsActivity.

Le problème est que ce login n'est pas disponible dans la troisième activité. Il faudrait la transmettre à chaque lancement dans un extra. Ce n'est pas demandé car il y a mieux pour cela, le contexte d'application.

Exercice 1 :

Création des activités :

Le projet comporte trois activités : chacune devra afficher un titre à son nom et des boutons permettant d'aller dans les autres activités de la manière suivante :

- LoginActivity : C'est l'activité qui est lancée au démarrage de l'application. Dans une application complète, son but serait de saisir un identifiant et un mot de passe d'utilisateur.
  - Elle doit contenir un bouton « login » envoyant sur l'activité NewsActivity.
  - Il faut faire en sorte que le bouton **back** ne permette pas de revenir dans cette activité : C'est à dire que si l'utilisateur va de LoginActivity à NewsActivity et qu'il appuie sur back, il revient directement sur l'écran d'accueil Android.
- NewsActivity Son rôle serait d'afficher une liste d'informations.
  - L'activité contient un bouton « details » qui envoie sur l'activité DetailsActivity.
  - Elle contient aussi un bouton « logout » qui fait revenir sur l'activité LoginActivity.
- DetailsActivity Elle pourrait servir à afficher les détails de l'une des informations de la liste.
  - Si on appuie sur **back**, on revient dans l'activité NewsActivity.
  - Cette activité contient un bouton libellé « ok » qui ramène sur l'activité NewsActivity.

Vous pouvez commencer par la finalisation de vos layouts : Rajoutez des titres et des boutons selon les spécifications. Définissez-leur un identifiant bien choisi, et le label spécifié (ne les appelez pas

button1, button2 mais avec un identifiant plus précis).

Exercice 2 :

- Créer un formulaire qui demande votre nom sur la première activité et l'affiche sur un deuxième activité (utilise putExtra pour envoyer le nom récupérer depuis la première activité et utilise getIntent pour l'afficher sur la deuxième activité).

Figure 1Formulaire de la première activité

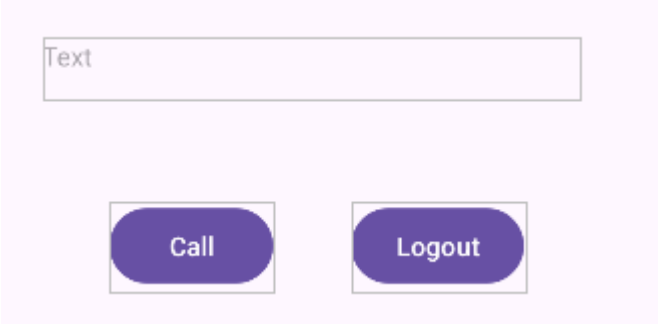

Figure 2 Zone d'affichage du texte récupéré + 2 boutton

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main2);

    TextView editText = findViewById(R.id.textView);
    String textToGet = getIntent().getStringExtra( name: "Text");
    editText.setText(textToGet);
```

Figure 3: Code a ajouté dans la première activité

```
1 usage
public void gotoSecond(View view) {

Intent intent = new Intent( packageContext: this, MainActivity2.class);
EditText editTxtZone = findViewById(R.id.editTextText);
String textToGet = editTxtZone.getText().toString();
intent.putExtra( name: "Text", textToGet);
startActivity(intent);

}
```

Figure 4Code a ajouté dans la deuxième activité

- Sur la deuxième activité ajouter un boutton call qui fait appel aux Intents implicites pour effectuer un appel vers un numéro de téléphone.
    - Il faut ajouter la permission dans le fichier manifest (chercher sur internet le code a ajouté).
    - Ajouter un Listener sur le boutton call.
    - Ajouter le code de demande de permission et appeler le intent avec l'action approprié.

```
Button btnCall = findViewById(R.id.btnCall);
btnCall.setOnClickListener(
        new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                callSomeone(v);
            }
        }
);
```

```
public void callSomeone(View view) {

    if (ContextCompat.checkSelfPermission( context: MainActivity2.this, Manifest.permission.CALL_PHONE) != PackageManager.PERMISSION_GRANTED) {
        ActivityCompat.requestPermissions( activity: MainActivity2.this, new String[]{Manifest.permission.CALL_PHONE}, REQUEST_PHONE_CALL);
    } else {
        // Change the phone number to the one you wish to call
        String phoneNumber = "tel:1234567890";
        startActivity(new Intent(Intent.ACTION_CALL, Uri.parse(phoneNumber)));
    }
}
```