**Faculté des sciences Ain Chock 2022-2023**

# Master BIG DATA & Cloud Computing

Développement sous Android

TP3

**Principe des Intents**

* Intent: description abstraite d’une opération à exécuter
* Permet la gestion d’envoie et de réception de messages afin de faire coopérer les applications au sein d’un seul device.
* Utilisation:
* Lancer une activité
* Communiquer avec des composants en background
* Faire appel à d’autres composants externes de l’application

**Pour lancer une activité:**

Intent intent= new Intent(this, SecondActivity.class);

startActivity(intent);

L’un des constructeurs de la classe Intent accepte 2 paramètres :

1. Contexte de la première activité qui appelle la deuxième activité. Dans certain cas, il ne faut pas mettre **this** mais faire appel à *getApplicationContext()* si l'objet manipulant l'intent n'hérite pas de la classe *Context*.
2. L'activité appelée.

*startActivity(intent);* méthode appartenant à la classe Activity qui permet de déclencher l’Intent passée en paramètre.

**Type des Intent : Action**

* Pour ce type d’Intent il suffit de lui passer en paramètre l’action à faire
* Ces types d'actions peuvent être les actions natives du système ou des actions définies par le développeur.
* Plusieurs actions natives existent par défaut sur Android. La plus courante est l'action Intent.ACTION\_VIEW qui permet d'appeler une application pour visualiser un contenu dont on donne l'URI.

**Exemple:**

*Uri telnumber=Uri.parse("tel:0123456789");*

*Intent call=new Intent(Intent.ACTION\_DIAL, telnumber);*

*startActivity(call);*

**Passer des informations entre les Activities**

*void putExtra(....)*

*Bundle getExtra(...);*

Les types supportés*:*

* *les types de base*
* *les objets serialisables (****implements Serializable****)*
* *Objets parcelables (****implements*** *Parcelable)*

Pour envoyer des données Extra:

*Intent callactivity2 = new Intent(getApplicationContext(), Activity2.class);*

*callactivity2.putExtra("login","fstUser");*

*startActivity(callactivity2);*

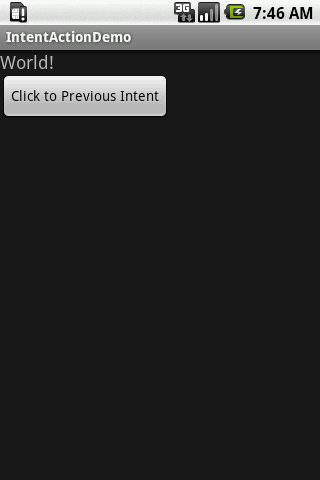
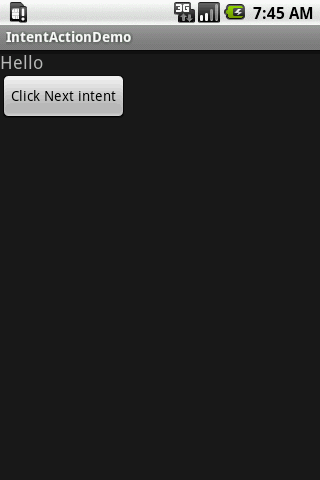
Pour récupérer cette donné:

*Bundle extras =getIntent().getExtras();*

*String s =new String(extras.getString("login"));*

**Exercices:**

I) Créer une application qui permet d'afficher les interfaces suivantes:



II) Passer un téléphone

Uri telnumber =Uri.parse("tel:0123456789");

Intent call new Intent(Intent.ACTION\_DIAL, telnumber);

startActivity(call);

III) Créer une activité permettant la saisie des infos d’un utilisateur

Créer une classe serialisable user

À l'aide des intents afficher les données saisies par une autre activité.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

public class User implements Serializable{  
 String fname;  
 String lname;  
 public void setFname(String fname) {  
 this.fname = fname;  
 }  
 public String getLname() {  
 return lname;  
 }  
 public String getFname() {  
 return lname;  
 }  
  
 public void setLname(String lname) {  
 this.lname = lname;  
 }  
  
 public User(String f, String l){  
 fname=f;  
 lname =l;  
  
 }  
  
}  
//////////////////////////////////////////////////////////

Button button=(Button)findViewById(R.id.saisir);  
button.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View v) {  
 EditText editText=(EditText)findViewById(R.id.fname);  
 EditText editText1=(EditText)findViewById(R.id.lname);  
 User user=new User(editText.getText().toString(), editText1.getText().toString());  
  
 Intent intent=new Intent(getApplicationContext(),ActivitySaisie.class);  
 intent.putExtra("user",user);  
 startActivity(intent);  
  
 }  
});

///////////////////////////////////////////

TextView textView=(TextView)findViewById(R.id.afficher);  
 Bundle extras= getIntent().getExtras();  
 User user=(User)extras.getSerializable("user");  
textView.setText("Bonjour "+user.getFname()+ " "+ user.getLname());

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**Mécanisme de référencement dans l’IU**

Pour générer un ID qui sera stocké automatiquement dans la classe R.java et ainsi relier le composant UI à son instance objet dans l’activité on utilise cette syntaxe sous XML:

"@+id/Ressource\_identifier"

TextView fName= (TextView)findViewById(R.id.Ressource\_identifier);

**Layouts**

L’SDK Android nous offre plusieurs type de layout pour pouvoir positionner les composants UI:

* LinearLayout.
* RelativeLayout.
* AbsoluteLayout (deprecated)
* FrameLayout.
* TableLayout.
* Grid View.

<LinearLayout

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:layout\_marginBottom="20dp"

android:gravity="center\_horizontal"

android:orientation="vertical">

</LinearLayout>

**LinearLayout : placer les éléments sur une ligne**

Comme son nom l'indique, ce layout se charge de mettre les vues sur une même ligne, selon une certaine orientation. L'attribut pour préciser cette orientation est android:orientation. On peut lui donner deux valeurs :

* vertical pour que les composants soient placés de haut en bas (en colonne) ;
* horizontal pour que les composants soient placés de gauche à droite (en ligne).

On va faire quelques expériences pour s'amuser !

##### Premier exemple

<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"

android:orientation="vertical"

android:layout\_width="fill\_parent"

android:layout\_height="fill\_parent" >

<Button

android:id="@+id/premier"

android:layout\_width="fill\_parent"

android:layout\_height="wrap\_content"

android:text="Premier bouton" />

<Button

android:id="@+id/second"

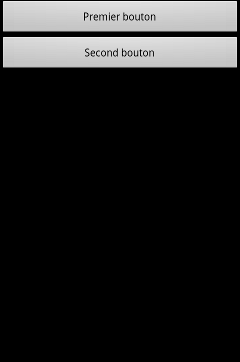
android:layout\_width="fill\_parent"

android:layout\_height="wrap\_content"

android:text="Second bouton" />

</LinearLayout>

Le rendu de ce code se trouve à la figure suivante.



Les deux boutons prennent toute la largeur

* Le LinearLayout est vertical et prend toute la place de son parent (vous savez, l'invisible qui prend toute la place dans l'écran).
* Le premier bouton prend toute la place dans le parent en largeur et uniquement la taille nécessaire en hauteur (la taille du texte, donc !).
* Le second bouton fait de même.

##### Deuxième exemple

<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"

android:orientation="vertical"

android:layout\_width="fill\_parent"

android:layout\_height="fill\_parent" >

<Button

android:id="@+id/premier"

android:layout\_width="wrap\_content"

android:layout\_height="fill\_parent"

android:text="Premier bouton" />

<Button

android:id="@+id/second"

android:layout\_width="wrap\_content"

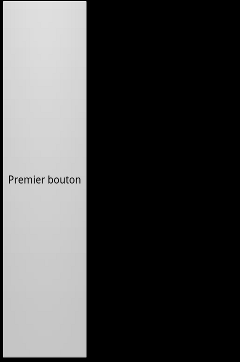
android:layout\_height="fill\_parent"

android:text="Second bouton" />

</LinearLayout>

Le rendu de ce code se trouve à la figure suivante.

Le premier bouton fait toute la hauteur, on ne voit donc pas le deuxième bouton

* 
* Le LinearLayout est vertical et prend toute la place de son parent.
* Le premier bouton prend toute la place de son parent en hauteur et uniquement la taille nécessaire en largeur.
* Comme le premier bouton prend toute la place, alors le pauvre second bouton se fait écraser.  C'est pour cela qu'on ne le voit pas.

##### Troisième exemple

<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"

android:orientation="vertical"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content" >

<Button

android:id="@+id/premier"

android:layout\_width="wrap\_content"

android:layout\_height="fill\_parent"

android:text="Premier bouton" />

<Button

android:id="@+id/second"

android:layout\_width="wrap\_content"

android:layout\_height="fill\_parent"

android:text="Second bouton" />

</LinearLayout>

Le rendu de ce code se trouve à la figure suivante.



* Le LinearLayout est vertical et prend toute la place en largeur mais uniquement la taille nécessaire en hauteur : dans ce cas précis, la taille nécessaire sera calculée en fonction de la taille des enfants.
* Le premier bouton prend toute la place possible dans le parent. Comme le parent prend le moins de place possible, il doit faire de même.
* Le second bouton fait de même.

##### Quatrième exemple

<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"

android:orientation="horizontal"

android:layout\_width="fill\_parent"

android:layout\_height="fill\_parent" >

<Button

android:id="@+id/premier"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="Premier bouton" />

<Button

android:id="@+id/second"

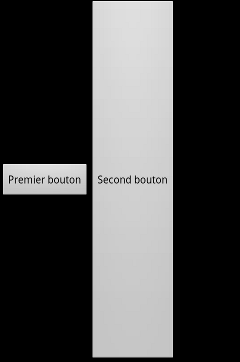
android:layout\_width="wrap\_content"

android:layout\_height="fill\_parent"

android:text="Second bouton" />

</LinearLayout>

Le rendu de ce code se trouve à la figure suivante.



Le premier bouton prend uniquement la place nécessaire et le deuxième toute la hauteur

* Le LinearLayout est horizontal et prend toute la place de son parent.
* Le premier bouton prend uniquement la place nécessaire.
* Le second bouton prend uniquement la place nécessaire en longueur et s'étend jusqu'aux bords du parent en hauteur.

Vous remarquerez que l'espace est toujours divisé entre les deux boutons, soit de manière égale, soit un bouton écrase complètement l'autre. Et si on voulait que le bouton de droite prenne deux fois plus de place que celui de gauche par exemple ?

Pour cela, il faut attribuer un poids au composant. Ce poids peut être défini grâce à l'attribut android:layout\_weight. Pour faire en sorte que le bouton de droite prenne deux fois plus de place, on peut lui mettre android:layout\_weight="1" et mettre au bouton de gauche android:layout\_weight="2". C'est alors le composant qui a la plus faible pondération qui a la priorité.

Et si, dans l'exemple précédent où un bouton en écrasait un autre, les deux boutons avaient eu un poids identique, par exemple android:layout\_weight="1" pour les deux, ils auraient eu la même priorité et auraient pris la même place. Par défaut, ce poids est à 0.

Une astuce que j'utilise souvent consiste à faire en sorte que la somme des poids dans un même layout fasse 100. C'est une manière plus évidente pour répartir les poids.

Dernier attribut particulier pour les widgets de ce layout, android:layout\_gravity, qu'il ne faut pas confondre avec android:gravity. android:layout\_gravity vous permet de déterminer comment se placera la vue dans le parent, alors que android:gravity vous permet de déterminer comment se placera le contenu de la vue à l'intérieur même de la vue (par exemple, comment se placera le texte dans un TextView ? Au centre, en haut, à gauche ?).

Vous prendrez bien un petit exemple pour illustrer ces trois concepts ?

<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"

android:orientation="horizontal"

android:layout\_width="fill\_parent"

android:layout\_height="fill\_parent" >

<Button

android:id="@+id/bouton1"

android:layout\_width="fill\_parent"

android:layout\_height="wrap\_content"

android:layout\_gravity="bottom"

android:layout\_weight="40"

android:text="Bouton 1" />

<Button

android:id="@+id/bouton2"

android:layout\_width="fill\_parent"

android:layout\_height="wrap\_content"

android:layout\_gravity="center"

android:layout\_weight="20"

android:gravity="bottom|right"

android:text="Bouton 2" />

<Button

android:id="@+id/bouton3"

android:layout\_width="fill\_parent"

android:layout\_height="wrap\_content"

android:layout\_gravity="top"

android:layout\_weight="40"

android:text="Bouton 3" />

</LinearLayout>

Le rendu de ce code se trouve à la figure suivante.



Trois boutons placés différemment

Comme le bouton 2 a un poids deux fois inférieur aux boutons 1 et 3, alors il prend deux fois plus de place qu'eux. De plus, chaque bouton possède un attribut android:layout\_gravity afin de que l'on détermine sa position dans le layout. Le deuxième bouton présente aussi l'attribut android:gravity, qui est un attribut de TextView et non layout, de façon à mettre le texte en bas (bottom) à droite (right).

***Exercice:***

***1) Ecrire le code XML d'une interface graphique d'une calculatrice.***

2) Ecrire le code d'une application de calcul de l'IMC et modifiez le layout pour obtenir quelque chose ressemblant à la figure suivante.



Les EditText prennent le plus de place possible, mais comme ils ont un poids plus fort que les TextView, ils n'ont pas la priorité.

<?xml version="1.0" encoding="utf-8"?>

<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"

android:layout\_width="fill\_parent"

android:layout\_height="fill\_parent"

android:orientation="vertical">

<LinearLayout

android:layout\_width="fill\_parent"

android:layout\_height="wrap\_content"

android:orientation="horizontal"

>

<TextView

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="Poids : "

android:textStyle="bold"

android:textColor="#FF0000"

android:gravity="center"

/>

<EditText

android:id="@+id/poids"

android:layout\_width="fill\_parent"

android:layout\_height="wrap\_content"

android:hint="Poids"

android:inputType="numberDecimal"

android:layout\_weight="1"

/>

</LinearLayout>

<LinearLayout

android:layout\_width="fill\_parent"

android:layout\_height="wrap\_content"

android:orientation="horizontal"

>

<TextView

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="Taille : "

android:textStyle="bold"

android:textColor="#FF0000"

android:gravity="center"

/>

<EditText

android:id="@+id/taille"

android:layout\_width="fill\_parent"

android:layout\_height="wrap\_content"

android:hint="Taille"

android:inputType="numberDecimal"

android:layout\_weight="1"

/>

</LinearLayout>

<RadioGroup

android:id="@+id/group"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:checkedButton="@+id/radio2"

android:orientation="horizontal"

>

<RadioButton

android:id="@+id/radio1"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="Mètre"

/>

<RadioButton

android:id="@+id/radio2"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="Centimètre"

/>

</RadioGroup>

<CheckBox

android:id="@+id/mega"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="Mega fonction !"

/>

<LinearLayout

android:layout\_width="fill\_parent"

android:layout\_height="wrap\_content"

android:orientation="horizontal"

>

<Button

android:id="@+id/calcul"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="Calculer l'IMC"

android:layout\_weight="1"

android:layout\_marginLeft="25dip"

android:layout\_marginRight="25dip"

/>

<Button

android:id="@+id/raz"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="RAZ"

android:layout\_weight="1"

android:layout\_marginLeft="25dip"

android:layout\_marginRight="25dip"

/>

</LinearLayout>

<TextView

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="Résultat:"

/>

<TextView

android:id="@+id/result"

android:layout\_width="fill\_parent"

android:layout\_height="fill\_parent"

android:text="Vous devez cliquer sur le bouton « Calculer l'IMC » pour obtenir un résultat."

/>

</LinearLayout>