CRÉATION D'INTERFACE UTILISATEUR (2)

CRÉATION D'APPLICATIONS ET DÉCOUVERTE DES ACTIVITÉS ET DES INTENTS

DÉVELOPPEMENT DES APPLICATIONS MOBILES (ANDROID)

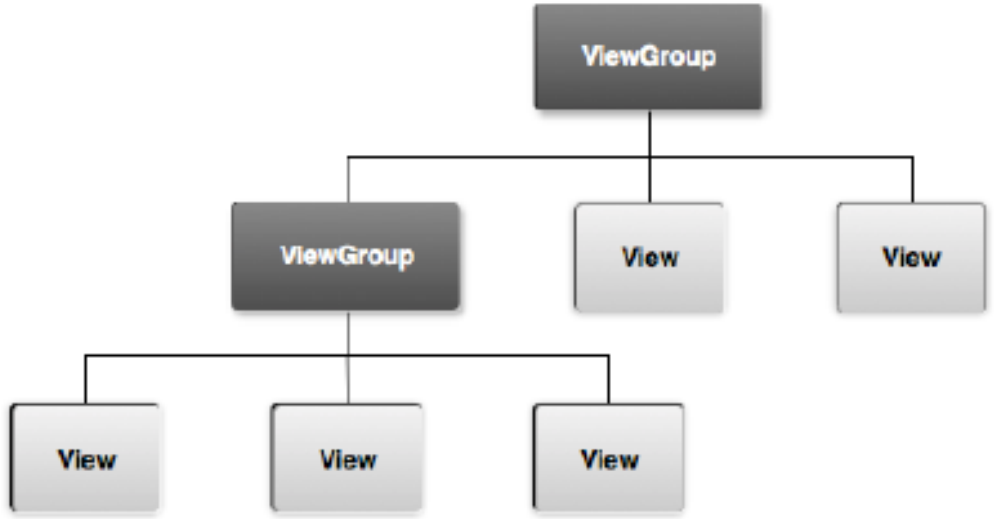
E.I. Djebbar

Département de Génie des systèmes

Ecole Nationale Polytechnique d'Oran

L'INTERFACE GRAPHIQUE

- Éléments d'interface (UI) construits à base d'objets View et ViewGroup
- View = un élément (zone de texte, bouton, etc)
- Hiérarchie sous forme d'arbre



VOCABULAIRE DE L'INTERFACE

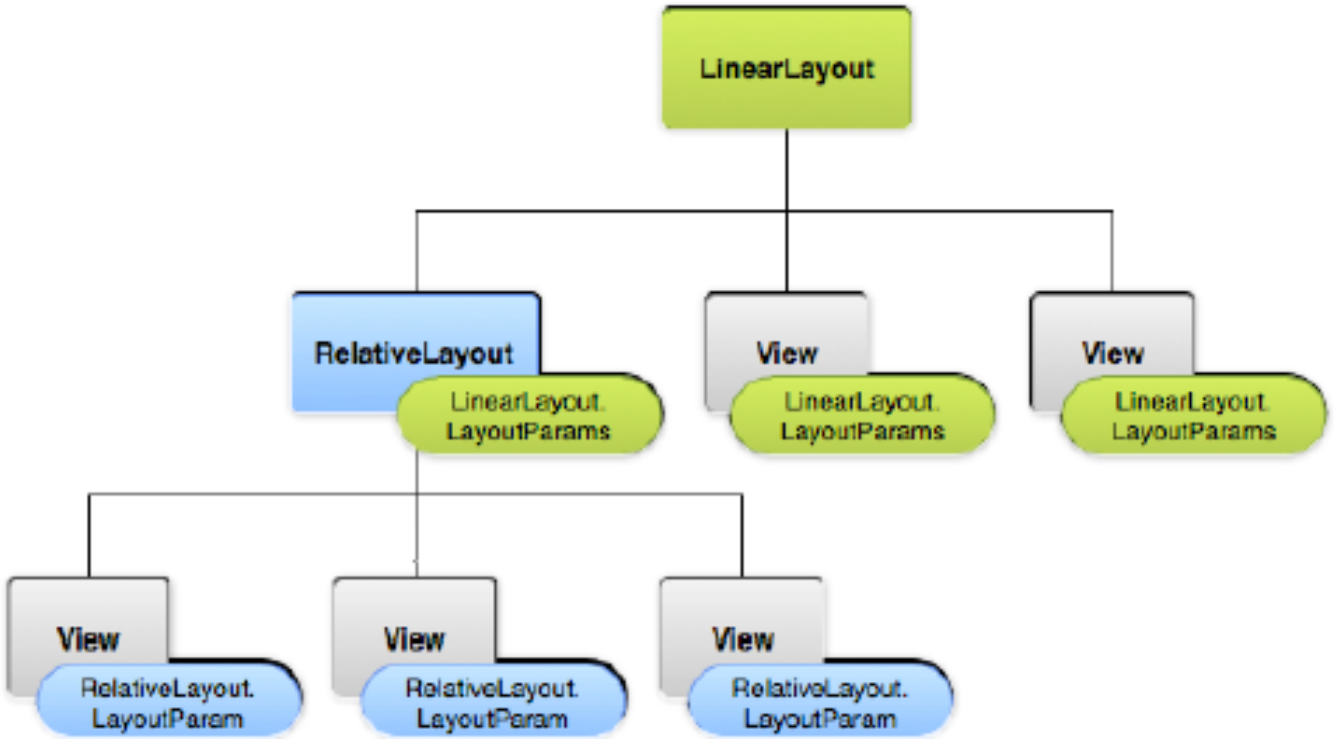
- View, widget, control : un élément d'interface sous la forme d'un ou plusieurs objets View
- Container : View contenant plusieurs autres View, par exemple un tableau
- Layout : arrangement visuel de vues, par exemple linéaire, en colonnes, en "grid", etc

LAYOUTS PRÉDÉFINIS

- Utilisation des layouts:
  - Positions relatives
  - Lignes ou colonnes
  - Tableaux et “Grid”
  - autres moins utilisés

ATTRIBUTS DU LAYOUT

- Paramétrage de l'élément dans le groupe qui le contient



ATTRIBUTS DU LAYOUT

- layout_width, layout_height sont obligatoires
- Valeurs : wrap_content, match_parent ou longueur
- wrap_content : prend les dimensions du contenu
- match_parent : prend toute la place que le parent lui autorise
- Attention changement dans Android 2.2 : match_parent si API ≥ 8, fill_parent si API < 8

RELATIVELAYOUT

- Tous les éléments sont disposés relativement au autres ou au parent
- Pas évident à créer mais rendu visuel rapide en exécution
- Attributs par exemple
- layout_alignParentTop : haut = haut du parent
- layout_centerVertical : aligné par rapport au parent
- layout_above
- layout_alignBaseline
- Etc

EXEMPLE DE RELATIVE LAYOUT

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content" >
    <TextView
        android:id="@+id/label"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="Enter email address" />
    <EditText
        android:id="@+id/inputEmail"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:layout_below="@id/label" />
    <Button
        android:id="@+id/btnLogin"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentLeft="true"
        android:layout_below="@id/inputEmail"
        android:layout_marginRight="10px"
        android:text="Login" />
    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignTop="@id/btnLogin"
        android:layout_toRightOf="@id/btnLogin"
        android:text="Cancel" />
    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentBottom="true"
        android:layout_centerHorizontal="true"
        android:text="Register" />
</RelativeLayout>
```

TextView with normal properties

EditView with normal properties

Button aligned left to the parent and also below the inputEmail EditView control

Button aligned at top of the parent and also right to the btnLogin

Button aligned at bottom of the parent and also center horizontally to the parent

LINEAIRE LAYOUT

- Éléments alignés
- Fils empilés les uns après les autres
- Attribut orientation pour horizontal ou vertical
- Attributs pour les fils :
- layout_weight spécifie son “importance” en terme de place occupée (valeur par défaut = 0)
- layout_gravity définit son positionnement (x/y) dans son contenant

EXEMPLE DE LINEAIRE LAYOUT

```xml
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical" >
    <EditText
        android:id="@+id/editText1"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:ems="10"
        android:hint="Username" >
        <requestFocus />
    </EditText>
    <EditText
        android:id="@+id/editText2"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:ems="10"
        android:hint="Password"
        android:inputType="textPassword" />
    <Button
        android:id="@+id/button1"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Button" />
    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent" >
        <TextView
            android:id="@+id/textView1"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Gender"
            android:textAppearance="?android:attr/textAppearanceMedium" />
        <RadioButton
            android:id="@+id/radioButton1"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Male" />
        <RadioButton
            android:id="@+id/radioButton2"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Female" />
    </LinearLayout>
</LinearLayout>
```

Vertical Layout

Horizontal Layout

LINEAIRELAYOUT

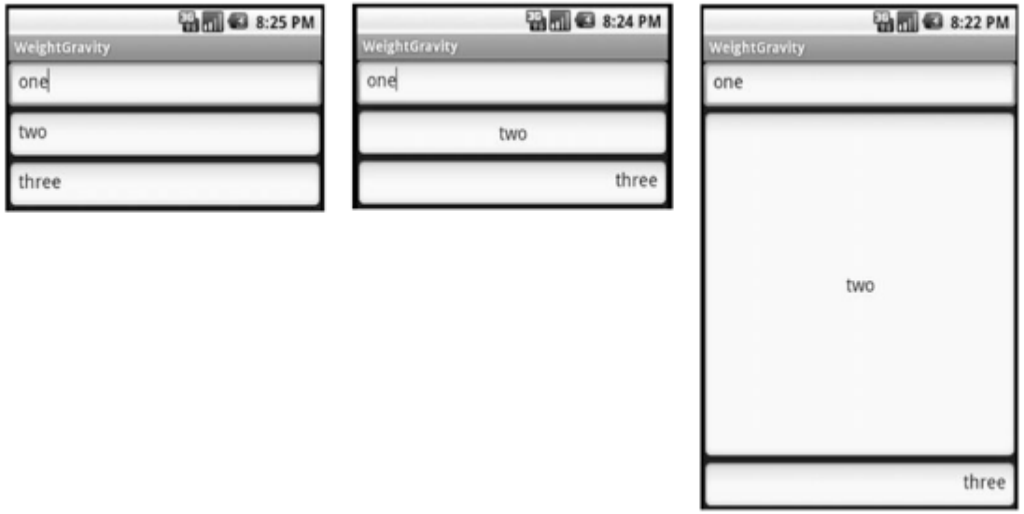
```
<LinearLayout xmlns:android="http://
schemas.android.com/apk/res/android"
android:orientation="vertical"
android:layout_width="fill_parent"
android:layout_height="fill_parent">

  <EditText android:layout_width="fill_parent"
  android:layout_height="wrap_content"
  android:text="one"/>

  <EditText android:layout_width="fill_parent"
  android:layout_height="wrap_content"
  android:text="two"/>

  <EditText android:layout_width="fill_parent"
  android:layout_height="wrap_content"
  android:text="three"/>

</LinearLayout>
```

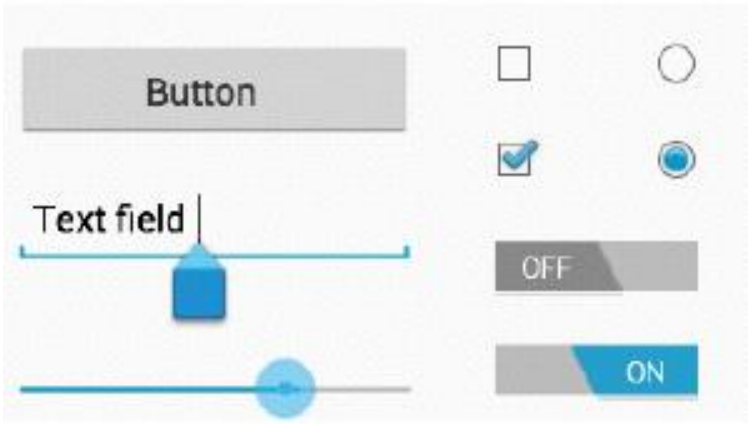


GRID LAYOUT

- Similaire à TableLayout mais plus simple car inutile de spécifier toutes les “cellules”
- Intéressant en termes de performances
- Attribut columnCount pour spécifier le nombre de colonnes (donc attribut rowCount à priori inutile)
- Contenu : layout_column et layout_row pour spécifier la cellule à remplir
- layout_rowSpan et layout_columnSpan (“span” du contenu)
- gravity à utiliser, weight non utilisé par GridLayout

OBJECTS VIEW

- TextView, EditText, etc
- Button, ToggleButton,
- CheckBox, RadioButton, etc
- TimePicker, DatePicker
- ImageView
- ProgressBar, Chronometer, etc



ATTRIBUTS COMMUNS

- Namespace android
- id référence indispensable pour utiliser l'objet View :

(TextView) findViewById(R.id.newText);

- Attributs layout_qquechose : paramétrage de l'élément dans le groupe qui le contient (layout_marginBottom, layout_marginLeft…)
- layout_width, layout_height avec valeurs wrap_content, match_parent
- Autres attributs pour définitions diverses : contenu (text), style…

EXEMPLE DE TEXTVIEW

```
<TextView
android:id="@+id/hello_text"
android:layout_width="match_parent"
android:layout_height=“wrap_content"
android:padding="5dp"
android:text="@string/hello_world"
android:textColor="#FF0000"
android:typeface="monospace" />
```

TEXTVIEW

- Affiche un texte :

attribut text android.text="@string/id_du_texte"

méthode setText('le texte');

- Classe parente de EditText et Button
- Attributs gravity, hint, ellipsize, ems, fontFamily, etc
- Attribut autoLink pour "activer" automatiquement liens, adresses emails, n° téléphone, etc :

<TextView ... android:autoLink="email|web" ... />

INTÉGRER UNE BOÎTE DE SAISIE DE TEXTE

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
xmlns:android="http://schemas.android.com/apk/res/android"
android:orientation="vertical"
android:layout_width="fill_parent"
android:layout_height="fill_parent">
<EditText
android:id="@+id/monEditText"
android:layout_height="wrap_content"
android:hint="Taper votre nom"
android:layout_width="fill_parent">
</EditText>
```

RÉCUPÉRER LA SAISIE DE L'UTILISATEUR

```
...
public void onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
setContentView(R.layout.main);
((Button)findViewById(R.id.monBouton)).setOnClickListener(new OnClickListener() {
@Override
public void onClick(View v) {
EditText texte = ((EditText)findViewById(R.id.monEditText));
String nom = texte.getText().toString();
Toast.makeText(Main.this, nom, Toast.LENGTH_SHORT).show();
}
});
}
}
...
```

IMAGEVIEW

- Créer une zone contenant une image
- Spécifier l'image avec attribut src
- Source = drawable, couleur

INTÉGRER UNE IMAGE DANS VOTRE INTERFACE

- Définition XML, via l'attribut src
- Utiliser la méthode setImageResource

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
xmlns:android="http://schemas.android.com/apk/res/android"
android:layout_height="fill_parent"
android:layout_width="fill_parent"
android:gravity="center_vertical|center_horizontal"
>
<ImageView
android:id="@+id/monImage"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:src="@drawable/icon"
>
</ImageView>
</LinearLayout>
```

BUTTON

- Créer un bouton, avec l'attribut text
- Dérive de TextView
- Attribut onClick="go" associé à la méthode
- public void go (View v)

- Capter le clic par programme

```
Button button = (Button) findViewById(R.id.button_send);
button.setOnClickListener(new View.OnClickListener() {
public void onClick(View v) {
// Do something in response to button click
}
});
```

IMAGEBUTTON

- Créer un bouton avec une image (attribut src)

```
<ImageButton
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:src="@drawable/button_icon"
... />
```

- Créer un bouton avec texte et image

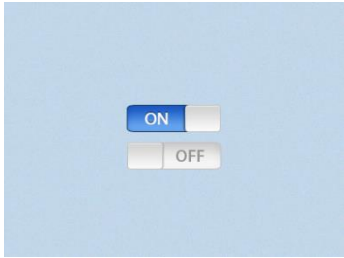
```
<Button
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:text="@string/button_text"
android:drawableLeft="@drawable/button_icon"
... />
```

- Fichier button_icon.* dans res/drawable*

TOGGLEBUTTON

- Passer d'un état à un autre
- Objet Switch pour Android 4.0 (API ≥ 14)


```
<ToggleButton
android:id="@+id/togglebutton"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:textOn="@string/day"
android:textOff="@string/night"
android:onClick="onToggleClicked"/>
```



- Méthodes isChecked(), setChecked(bool), toggle()

CHECKBOX

- Cases à cocher

<CheckBox android:id=“@+id/chickenCB"

android:text=“Chicken"

android:layout_width=“wrap_content"

android:layout_height="wrap_content"

android:checked="true" />

!<

CheckBox android:id="@+id/fishCB" android:text="Fish"

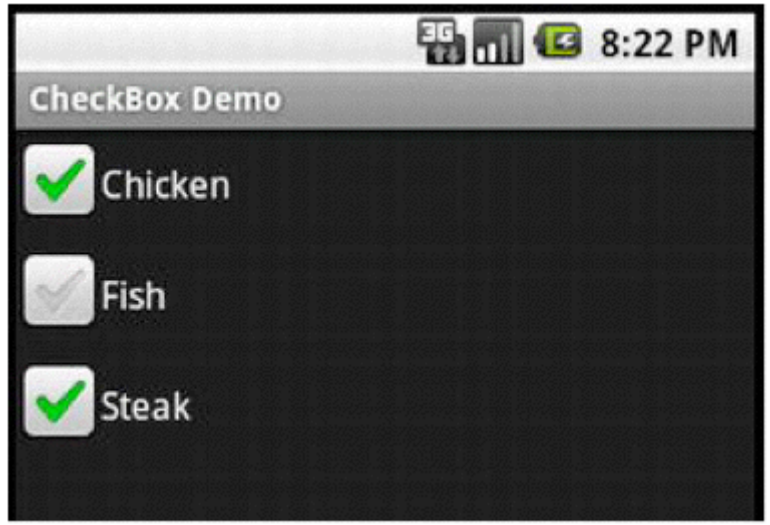
android:layout_width=“wrap_content" android:layout_height="wrap_content"
/>

!<

CheckBox android:id="@+id/steakCB" android:text="Steak"

android:layout_width=“wrap_content" android:layout_height="wrap_content"

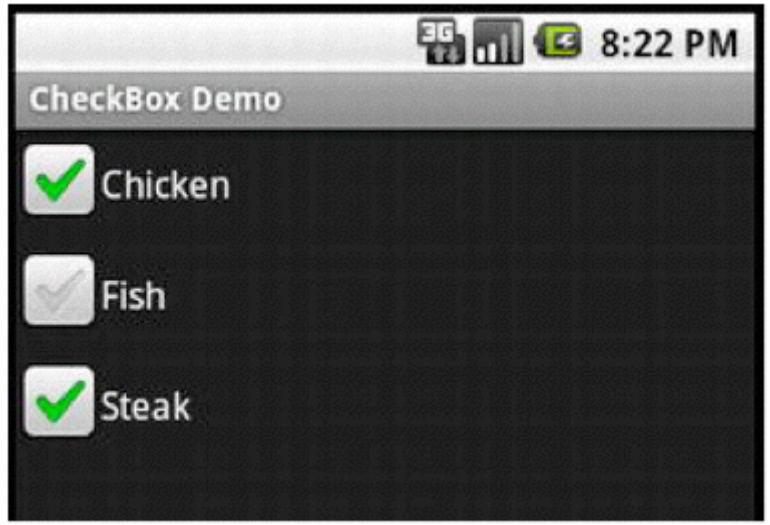
android:checked="true" />



CHECKBOX (2)

- Attribut onClick
- Méthodes isChecked(), setChecked(bool), …
- Capter un changement :

```
CheckBox fishCB = (CheckBox)findViewById(R.id.fishCB);
fishCB.setOnCheckedChangeListener(
new CompoundButton.OnCheckedChangeListener() {
@Override
public void onCheckedChanged(CompoundButton arg0, boolean isChecked) {
// faire quelque chose
}});
```

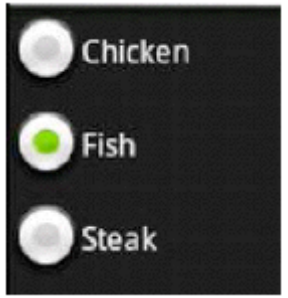


RADIOGROUP ET RADIOBUTTON

- Un seul bouton coché parmi un groupe

- Éléments RadioGroup et RadioButton

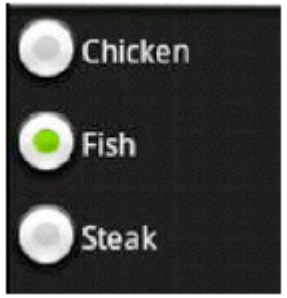
```
<RadioGroup android:id=“@+id/radGrp” android:layout_width="wrap_content"
android:layout_height="wrap_content" android:orientation="vertical" >
<RadioButton android:id="@+id/chRBtn" android:text="Chicken"
android:layout_width="wrap_content" android:layout_height="wrap_content"/>
<RadioButton android:id="@+id/fishRBtn" android:text="Fish" android:checked="true"
android:layout_width="wrap_content" android:layout_height="wrap_content"/>
<RadioButton android:id="@+id/stkRBtn" android:text="Steak"
android:layout_width="wrap_content" android:layout_height="wrap_content"/>
</RadioGroup>
```



RADIOGROUP ET RADIOBUTTON (2)

- Capter un changement

```
radGrp.setOnCheckedChangeListener(
new RadioGroup.OnCheckedChangeListener() {
@Override
public void onCheckedChanged(RadioGroup arg0, int id) {
// id vaut -1 sin aucun bouton coché
}
}
);
```



INTÉGRER DES ONGLETS À L'INTERFACE UTILISATEUR

- TabHost et ajouter un TabWidget

```
import android.app.Activity;
import android.os.Bundle;
import android.widget.TabHost;
import android.widget.Toast;
import android.widget.TabHost.TabSpec;
public class Main extends Activity {
private TabHost monTabHost;
@Override
public void onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
setContentView(R.layout.main7);
monTabHost =(TabHost) findViewById(R.id.TabHost01);
monTabHost.setup();
TabSpec spec = monTabHost.newTabSpec("onglet_1");
spec.setIndicator("Onglet 1",getResources().getDrawable(R.drawable.icon));
spec.setContent(R.id.Onglet1);
monTabHost.addTab(spec);
monTabHost.addTab(monTabHost.newTabSpec("onglet_2
").setIndicator(
X "Onglet 2").setContent(R.id.Onglet2));
monTabHost.addTab(monTabHost.newTabSpec("onglet_3
").setIndicator( "Onglet 3").setContent(R.id.Onglet3));
monTabHost.setOnTabChangedListener(
new TabHost.OnTabChangeListener (){
public void onTabChanged(String tabId){
Toast.makeText(Main.this, "L'onglet avec l'identifiant : "+ tabId + " a
été cliqué",Toast.LENGTH_SHORT).show();
}
}
);
}
}
```