Application Android

Sommaire

- **Les Conteneurs**
  - LinearLayout
  - RelativeLayout
  - TableLayout
  - ScrollView
- **Widgets de base**
  - Textview
  - Button
  - ImageView
  - EditText
  - CheckBox
  - RadioGroup-RadioButton
- **Widgets de selection**
  - ListView
  - Spinner
  - Gallery
  - GridView
- **Intent**

Les Conteneurs

- **LinearLayout**
  - Les widgets ou les conteneurs fils sont alignés en colonnes ou en lignes les uns après les autres.
- **RelativeLayout**
  - Permet de placer les widgets relativement aux autres widgets du conteneur et de son conteneur parent
- **TableLayout**
  - Permet de positionner les widgets dans une grille
- **ScrollView**
  - Permet de fournir un défilement à son contenu

Les conteneurs:LinearLayout

- **Implémentation**

```
<?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"
    android:background="#CCCCCC"
    >
</LinearLayout>
```

LinearLayout

- **android:orientation :** la façon dont seront **alignés** les éléments contenus dans ce Layout.
  Deux options sont disponibles :
  - **Verticale** : Oriente les éléments sur une ligne verticale.
  - **Horizontale** : Oriente les éléments sur une ligne horizontale.

- **android:layout_width :** définit la largeur du conteneur
- **android:layout_height :** définit la hauteur du conteneur

Ces deux propriétés peuvent prendre 3 types de valeur :

- **Une taille fixe** : par exemple par px ou dip.
- **fill_parent** : le conteneur occuper tout l'espace disponible chez son conteneur parent .
- **wrap_content** : Pour demander au conteneur d'occuper une taille naturelle.

LinearLayout

- **android:layout_gravity :** Dans un **LinearLayout** les éléments sont alignés de gauche à droite et de haut en bas alors si vous voulez placer notre élément tout en bas ou a droite? On pourra utiliser les propriétés siuvante :
  - **left :** gauche
  - **center_horizontal** : centre
  - **top** : haut
  - **bottom :** bas
  - **right** : droite
- **android:padding :** Les différents éléments que vous créez sont par défaut serrés les uns contre les autres. Vous pouvez augmenter l'espacement à l'aide de cette propriété.
  La valeur précise l'espace situé entre le contour de l'élément et son contenu réel.

  Les valeurs possibles :**(padding top, left, right et bottom**).

```
<?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"   >
 <TextView
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:text="HelloCognicap" />

  <Button android:layout_width="wrap_content"
 android:id="@+id/button1"
 android:layout_height="wrap_content"
  android:text="Button1"
  android:paddingTop="10dip"> </Button>

  <Button android:layout_width="wrap_content"
 android:id="@+id/button2"
 android:layout_height="wrap_content"
  android:text="Button2"
  android:layout_gravity="center_horizontal"> </Button>
    <Button android:layout_width="wrap_content"
 android:id="@+id/button3"
 android:layout_height="wrap_content"
  android:text="Button3"> </Button>
  </LinearLayout>
```

Relativelayout

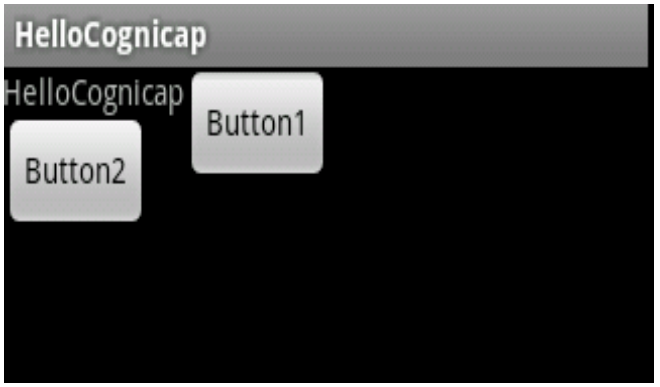
- **Android:layout_above** : Indique que l'élément sera placé au-dessus de celui indiqué par son id.
- **Android:layout_below** : Indique que l'élément sera placé en dessous de celui indiqué par son id.
- **Android:layout_toLeftOf** : Indique que l'élément sera placé à gauche de celui indiqué par son id.
- **Android:layout_toRightOf** : Indique que l'élément sera placé à droite de celui indiqué par son id.
- **Android:layout_alignTop** : Indique que le haut de notre élément est aligné avec le haut de l'élément indiqué.
- **Android:layout_alignBottom** : Indique que le bas de notre élément est aligné avec le bas de l'élément indiqué.
- **Android:layout_alignLeft** : Indique que le côté gauche de notre élément est aligné avec le côté gauche de l'élément indiqué.
- **Android:layout_alignRight** : Indique que le côté droit de notre élément est aligné avec le côté droit de l'élément indiqué.
- **Android:layout_alignBaseLine** : Indique que les lignes de base des 2 éléments sont alignées.

RelativeLayout :Exemple d'utilisation

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android=
"http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent" >
<TextView
   android:id="@+id/text1"
   android:layout_width="wrap_content"
   android:layout_height="wrap_content"
   android:text="HelloCognicap" />

   <Button android:layout_width="wrap_content"
  android:id="@+id/button1"
  android:layout_height="wrap_content"
  android:text="Button1"
    android:layout_toRightOf="@id/text1"> </Button>

 <Button android:layout_width="wrap_content"
  android:id="@+id/button2"
  android:layout_height="wrap_content"
  android:text="Button2"
  android:layout_below="@id/text1" > </Button>
</RelativeLayout>
```



RelativeLayout

Positionnement relatif au conteneur : Vous pouvez lier un élément à son conteneur :

- **android:layout_alignParentTop** (true / false) : Cette option permet de préciser si le haut de l'élément doit être aligné avec celui de son conteneur.
- (**android:layout_alignParentBottom**,
- **android:layout_alignParentLeft** et
- **android:layout_alignParentRight)**.

- **android:layout_centerHorizontal** : Indique si l'élément doit être **centré horizontalement** dans son conteneur. (**android:layout_centerVertical)**.

- **android:layout_centerInParent** : Vous permet d'indiquer que l'élément doit être **centré horizontalement et verticalement** dans le conteneur.

Position relative aux autres éléments Afin de pouvoir référencer le positionnement d'un élément par rapport à un autre, vous disposez d'un moyen simple et efficace, il s'agit des identificateurs (ID).

Donc voilà comment vous pouvez utiliser un ID :

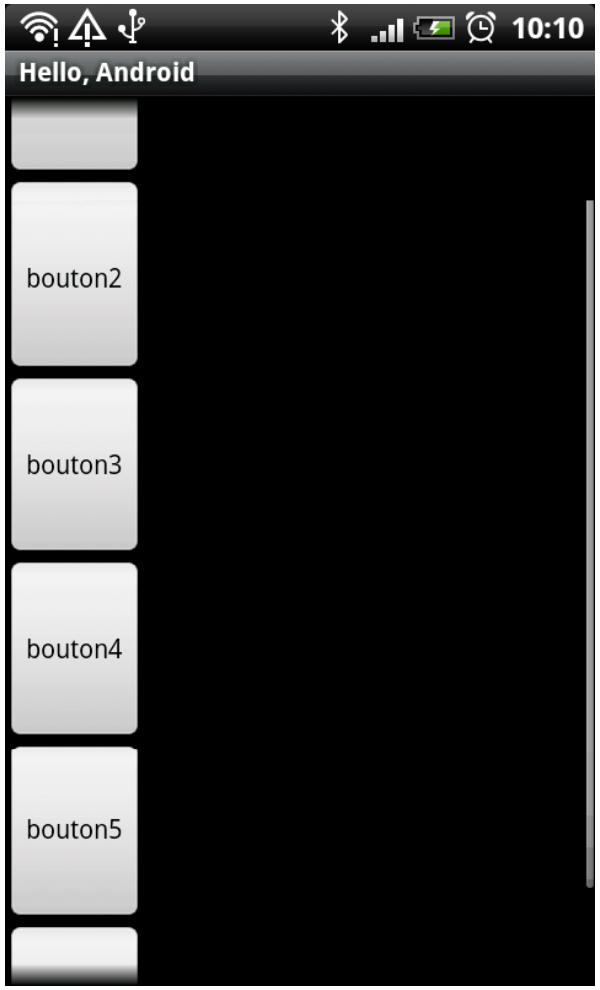
- **A la déclaration d'un élément : android:id= « @+id/idElem »**
- **A l'utilisation : @id/idElem**

TableLayout

```
<?xml version="1.0" encoding="utf-8"?>
 <TableLayout xmlns:android=
 "http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:stretchColumns="1">
    <TableRow>
        <TextView android:text="Test TableLayout :"></TextView>
       <EditText></EditText>
    </TableRow>
    <TableRow>
        <Button android:id="@+id/stop"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Stop"/>
        <Button android:id="@+id/start"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Bonjour"/>
        <Button android:id="@+id/pause"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Bonsoir"/>
    </TableRow>
 </TableLayout>
```

ScrollView

```
<?xml version="1.0" encoding="utf-8"?>
<ScrollView
        xmlns:android="http://schemas.android.com/apk/res/android"
        android:orientation="vertical"
        android:layout_width="fill_parent"
        android:layout_height="fill_parent">
    <LinearLayout
        android:orientation="vertical"
        android:layout_width="fill_parent"
        android:layout_height="fill_parent">
        <TextView
            android:id="@+id/text"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="@string/hello"
            />
         <Button
            android:layout_width="wrap_content"
            android:layout_height="100dip"
            android:text="bouton1"/>
         <Button
            android:layout_width="wrap_content"
            android:layout_height="100dip"
            android:text="bouton2"/>
         <Button
            android:layout_width="wrap_content"
            android:layout_height="100dip"
            android:text="bouton3"/>
```



Les widgets de base

- Widgets de base
  - Textview
  - Button
  - ImageView
  - EditText
  - CheckBox
  - RadioGroup-RadioButton

Widgets : TextView

- Implémentation XML

```
<TextView
android:id="@+id/TextView"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
 android:text="Tester le TextView" />
```

- [illegible]

```
@Override
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.main);
    TextView  texteView = (TextView) this.findViewById(R.id.TextView);
     texteView.setText("Réalisation du premier Tp");
    }
```

Widgets : Button

- Implémentation XML

```
<Button android:id="@+id/bouton"
     android:text="Tester le Click"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"/>
```

```
Button bouton = (Button) this.findViewById(R.id.bouton);
bouton.setOnClickListener(new OnClickListener() {
    @Override
    public void onClick(View v) {
Toast.makeText(HelloCognicap.this, "Vous Avez Clické sur le bouton ",10)
.show();
      }
});
```

Widgets : ImageView

- Implémentation Xml

```
<ImageView
  android:id="@+id/image_artiste"
  android:src="@drawable/cognicap"
  android:layout_width="200dip"
  android:layout_height="150dip" />
```

- Implémentation Java

```
final ImageView imageView = (ImageView) this.findViewById(R.id.image_artiste);

imageView.setOnClickListener(new OnClickListener() {
    @Override
    public void onClick(View v) {
        imageView.setImageResource(R.drawable.images);
Toast.makeText(HelloCognicap.this, "Vous avez aimé la nouvelle photos? ",10)
.show();
      }
});
```

Widgets : CheckBox

- Implémentation Xml

```
<CheckBox
android:id="@+id/checkbox"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:orientation="vertical"
android:text="Tester Checkbox : n'est pas cochée"/>
```

- Implémentation Java

```
final CheckBox check = (CheckBox) this.findViewById(R.id.checkbox);

check.setOnCheckedChangeListener(new OnCheckedChangeListener() {

 @Override
 public void onCheckedChanged(CompoundButton buttonView, boolean isChecked) {
     // TODO Auto-generated method stub
     if(isChecked)
     {check.setText("CheckBox est cochée") ;}
     else
     {check.setText("CheckBox est décochée") ;}
 }
});
```

Widgets : RadioButton

- Implémentation Xml

```
<RadioGroup
android:id="@+id/rgGroup1"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:orientation="vertical">

<RadioButton android:id="@+id/RB1" android:text="Java" />
<RadioButton android:id="@+id/RB3" android:text="Android" />
</RadioGroup>
```

- Implémentation Java

```
RadioGroup radio = (RadioGroup)this.findViewById(R.id.rgGroup1);
radio.setOnCheckedChangeListener(new OnCheckedChangeListener() {

    @Override
    public void onCheckedChanged(RadioGroup group, int checkedId) {
        // TODO Auto-generated method stub
    RadioButton radio = (RadioButton)findViewById(group.getCheckedRadioButtonId());
        Toast.makeText(HelloCognicap.this,radio.getText() , 10).show();
    }
});
```

Widgets : Résultats

Quand vous cliquez sur l'image elle change avec un message qui apparait

Quand vous cliquez sur le bouton un toast apparait

A chaque fois que vous cochez un radio Bouton un toast apparait avec le texte du radio coché

Le texte du checkbox change en débit si il est cochée ou non

- DatePicker
- TimePicker

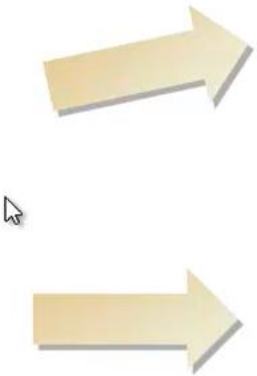
```
<DatePicker
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:id="@+id/datePicker"
    android:calendarViewShown="false"
    android:datePickerMode="spinner"/>
```

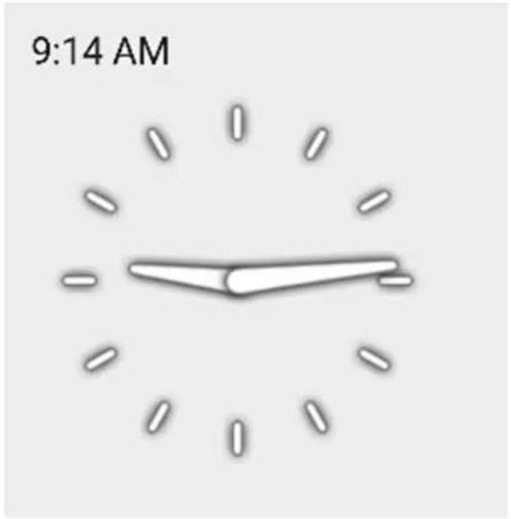

```
<TimePicker
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:id="@+id/timePicker" />
```

- TextClock
- AnalogClock

```
<TextClock
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:id="@+id/textClock" />

<AnalogClock
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:id="@+id/analogClock" />
```





TP N°1

Réalisez les deux interfaces suivantes :