PROGRAMMATION DE COMPOSANTS MOBILES (ANDROID)

WIESLAW ZIELONKA

WWW.IRIF.UNIV-PARIS-DIDEROT.FR/~ZIELONKA

LAYOUTS

layout : une view (invisible) qui sert à positionner d'autre view

- LinearLayout
- GridLayout
- RelativeLayout
- ConstraintLayout
- CoordinatorLayout
- FrameLayout

LinearLayout

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    ...
>

</LinearLayout>
```

LinearLayout affiche les views une après l'autre soit rangées horizontalement soit verticalement.

layout_width et layout_height sont obligatoires. Les valeurs possibles :

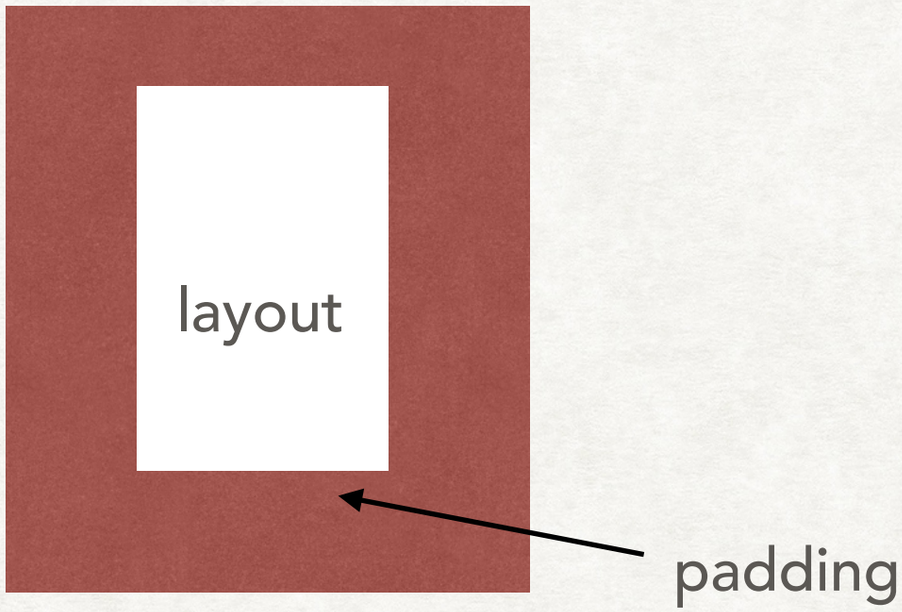
match_parent - la taille du parent
wrap_content - la taille suffisante pour afficher le contenu
20dp - la taille en density-independent pixels

android:orientation -- vertical ou horizontal : affichage en colonne oui en ligne

*

padding : android:padding="16dp"

l'espace entre le composant et son parent



Si padding différent de chaque côté :

android:paddingBottom = "16dp"
android:paddingLeft="8dp"
android:paddingRight="8dp"
android:paddingTop="32dp"

padding -dimensions dans dimes.xml

Dans le fichier res/values/dimens.xml

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
<dimen name="activity_horizontal_margin">16dp</dimen>
<dimen name="activity_vertical_margin">16dp</dimen>
</resources>
```

et ensuite dans le fichier layout :

```
<LinearLayout ...
android:paddingLeft="@dimen/activity_horizontal_margin"
android:paddingRight="@dimen/activity_horizontal_margin"
android:paddingTop="@dimen/activity_vertical_margin" a
android:paddingBottom="@dimen/activity_vertical_margin"/>
```

margins

android:layout_margin

android:layout_marginLeft

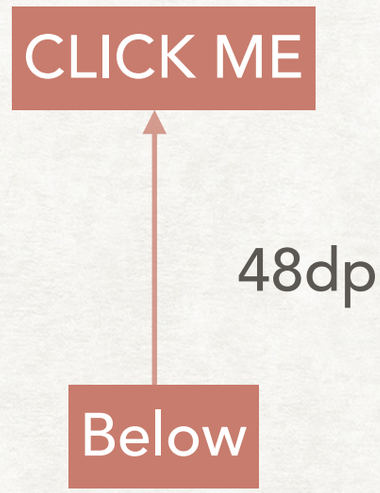
android:layout_marginRight

android:layout_marginTop

android:layout_marginBottom

extra espace autour de composant

```
<LinearLayout ... >
    <Button
        android:id="@+id/button_click_me" ... />
   <Button
        android:id="@+id/button_below"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginTop="48dp"
        android:text="@string/button_below" />
</LinearLayout>
```



le poids - layout_weight

android:layout_weight="number"

layout détermine combien d'espace nécessaire pour chaque view et l'espace restant est distribué entre les views dont layout_weight est >0, proportionnellement au poids.

Pour que ça marche l'attribut

android:layout_height ou android:layout_width doit prendre

la valeur "0dp"

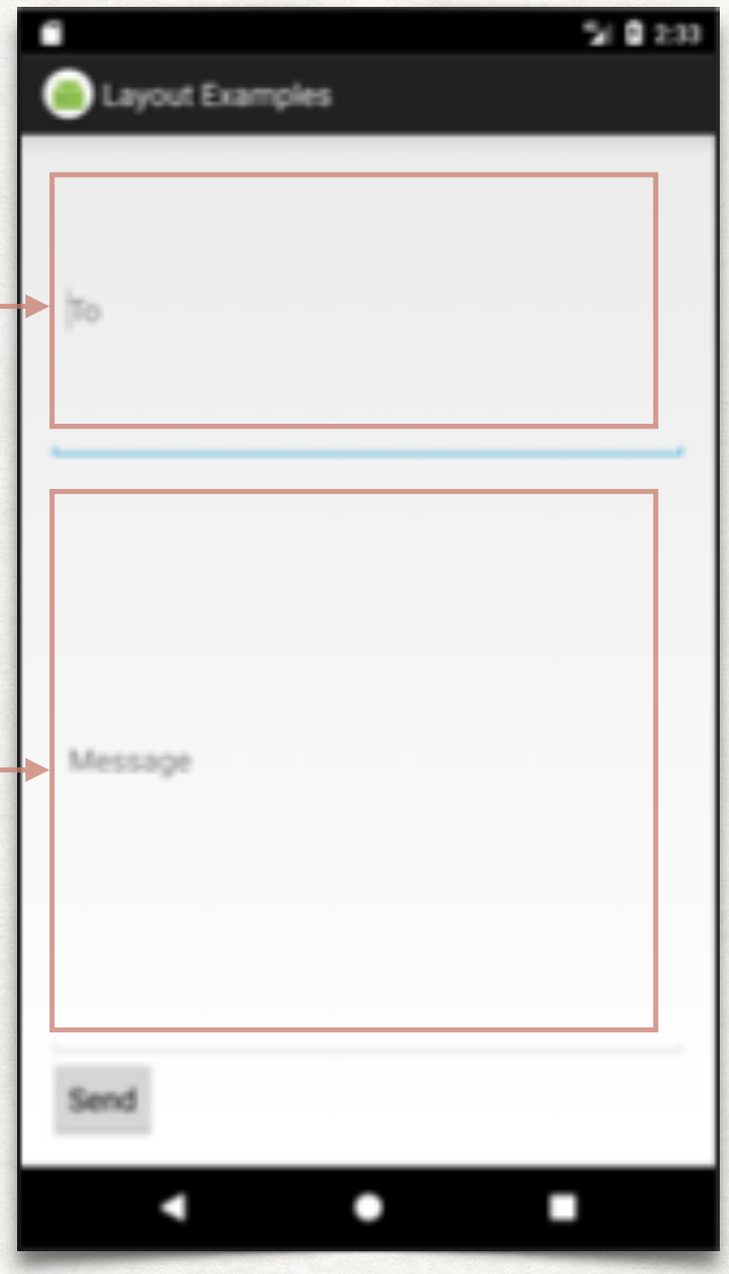
le poids - layout_weight

```
<LinearLayout ... >
    <EditText
          android:layout_width="match_parent"
           android:layout_height="wrap_content"
           android:hint="@string/to" />
       <EditText
           android:layout_width="match_parent"
           android:layout_height="0dp"
           android:layout_weight="1"
           android:hint="@string/message" />
       <Button
           android:layout_width="wrap_content"
           android:layout_height="wrap_content"
           android:text="@string/send" />
</LinearLayout>
```

EditText au milieu est le seul élément avec le poids. Donc occupe tout espace libre.

le poids - layout_weight

```
<LinearLayout ... > ...
 <EditText
  android:layout_width="match_parent"
  android:layout_height="0dp"
  android:layout_weight="1"
  android:hint="@string/to" />
 <EditText
  android:layout_width="match_parent"
  android:layout_height="0dp"
  android:layout_weight="2"
  android:hint="@string/message" />
... </LinearLayout>
```



Les deux EditText partagent l'espace libre en proportion 2:1

android:gravity

android:gravity indique comment positionner le contenu à l'intérieur du view


```
<EditText
        android:layout_width="match_parent"
        android:layout_height="0dp"
        android:layout_weight="1"
        android:gravity="top"
        android:hint="@string/message" />
```

EditText avec le hint en haut (top).

Il est possible d'indiquer plusieurs valeurs :
android:gravity="bottom|end"

android:layout_gravity

`android:layout_gravity` indique la position de view à l'intérieur de layout.

```
<Button
    android:layout_gravity = "end"
/>
```

android:gravity pour contrôler le placement de contenu de view,

andoid:layout_gravity pour contrôler la position de view dans le layout parent

FrameLayout

`FrameLayout` permet de placer les views l'un sur l'autre.

Par exemple pour placer un text sur une image :

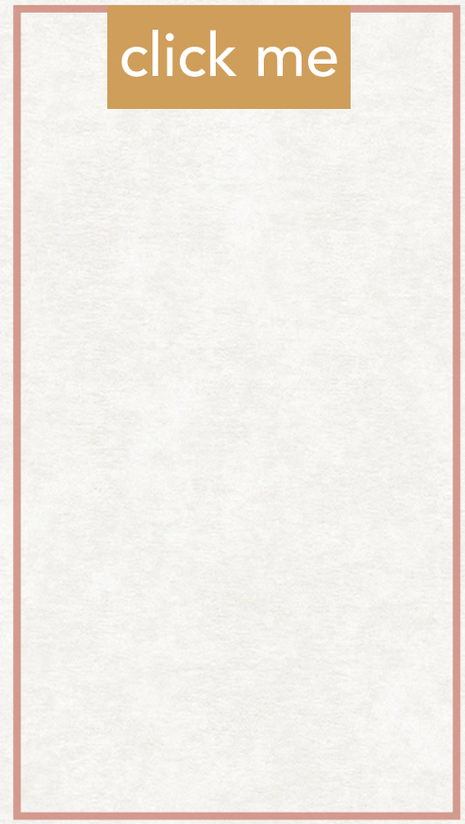
```
<FrameLayout
        .........
>

      <ImageView ...>
      <TextView  ....>
<FrameLayout>
```

Les views sont empilés un sur l'autre dans l'ordre de définition dans FrameLayout

RelativeLayout

RelativeLayout permet de placer les views permet de positionner les view relativement au parent et relativement à d'autres views dans le layout.

```
<RelativeLayout ... >
    <Button
          android:layout_width="wrap_content"
          android:layout_height="wrap_content"
          android:text="@string/click_me"
          android:layout_alignParentTop="true"
          android:layout_centerHorizontal="true" />
</RelativeLayout>
```



RelativeLayout

positions vis à vis du parent :

layout_alignParentBottom

layout_alignParentLeft

layout_alignParentRight

layout_alignParentTop

layout_centerInParent

layout_centerHorizontal

layout_centerVertical

RelativeLayout

positions vis à vis à d'autres enfants : :

```
<RelativeLayout ... >

<Button

android:id="@+id/button_click_me"

android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:layout_centerInParent="true" android:text="Click
Me" />

<Button
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:layout_alignStart="@id/button_click_me"
android:layout_below="@id/button_click_me" android:text="New
Button" />

</RelativeLayout>
```

Le deuxième bouton est au-dessous du premier (**android:layout_below**) et aligné sur le bord gauche du premier bouton (**android:layout_alignStart**)

RelativeLayout

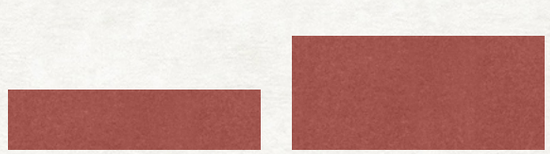
positions vis à vis d'autres enfants : :

layout_above

layout_below

layout_alignTop

layout_alignBottom

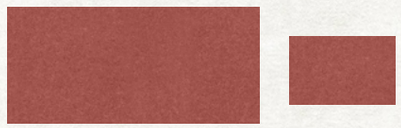


layout_alignLeft, layout_alignStart

layout_alignRight, layout_alignEnd

layout_toLeftOf, layout_toStartOf

layout_toRightOf, layout_toEndOf



GridLayout

```
<GridLayout xmlns:android="http://
schemas.android.com/apk/res/android"
android:layout_width="match_parent"
android:layout_height="match_parent"
android:columnCount="2"
android:rowCount="3"
>
```

columnCount - nombre de colonnes

rowCount - nombre de ligne

GridLayout - placer les enfants

```
<GridLayout...>

  <TextView
        ...
        android:layout_row="0"
        android:layout_column="0"
        android:text="@string/to" />
    <EditText
        android:layout_row="0"
        android:layout_column="1"
        android:layout_columnSpan="2"
        android:hint="@string/to_hint" />
</GridLayout>
```

ligne 0 et colonne 0

ligne 0 et colonne 1

EditText s'étend sur 2 colonnes