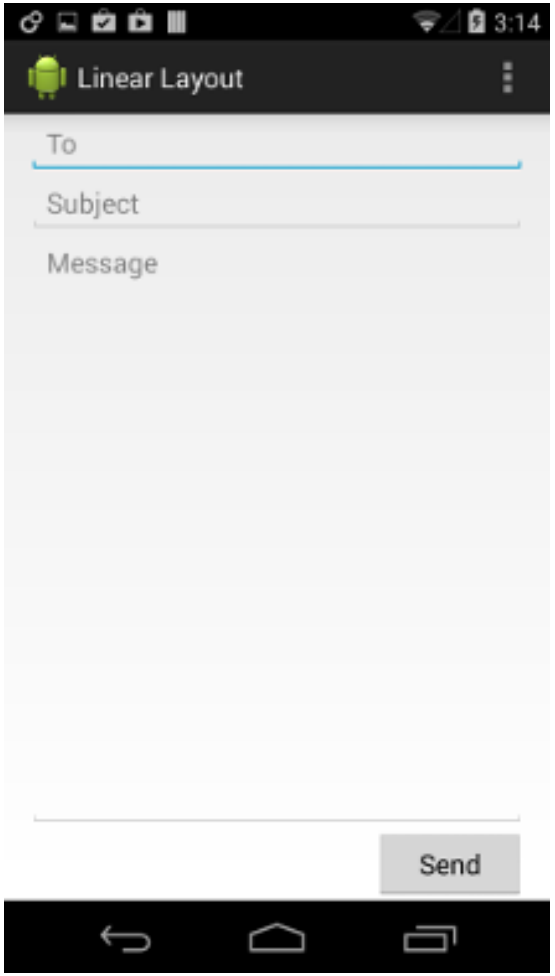
android

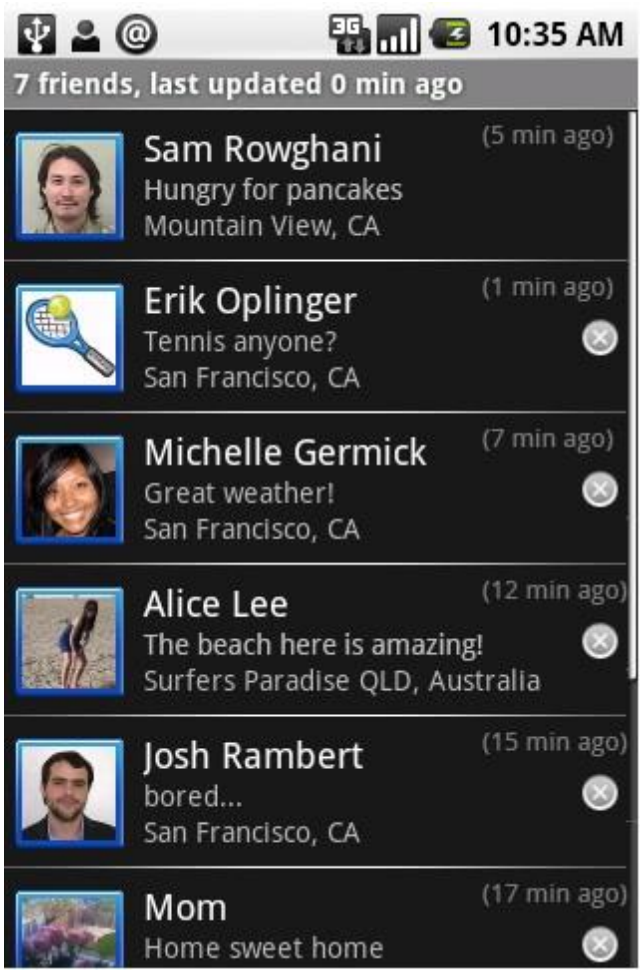
An Open Handset Alliance Project

Les vues

G.Seront

Les vues





https://developer.android.com/guide/topics/ui/overview.html

Les vues

- Dans Android, une ihm est composée de vues.
- Ces vues sont des objets de type *View* ou *ViewGroup.*
- Pour qu'une vue soit affichée, elle doit être associée à une activité:
  - *activité.setContentView(uneVue)*

https://developer.android.com/guide/topics/ui/overview.html

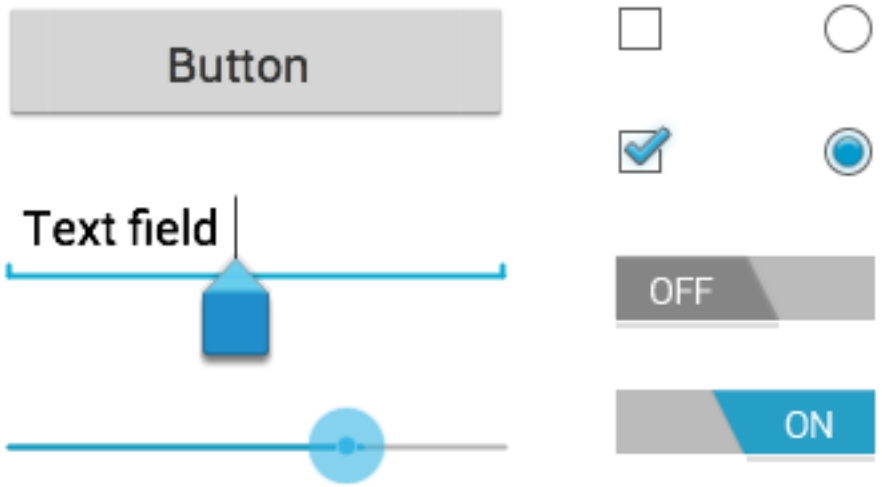
View

- Une vue est un objet de la classe *android.view.View* qui gère :
  - l'affichage d'une partie rectangulaire de l'écran;
  - la disposition des composants graphiques qu'il contient *(widgets);*
  - le changement de focus;
  - le scrolling;
  - les interactions de l'utilisateur avec la vue, avec les composants de cette vue;

https://developer.android.com/guide/topics/ui/overview.html

View

- Une vue peut contenir des composants graphiques interactifs, appelés *widgets:*
  - *TextView;*
  - *EditText;*
  - *Button;*
  - *RadioButton;*
  - *Checkbox;*
  - *…*

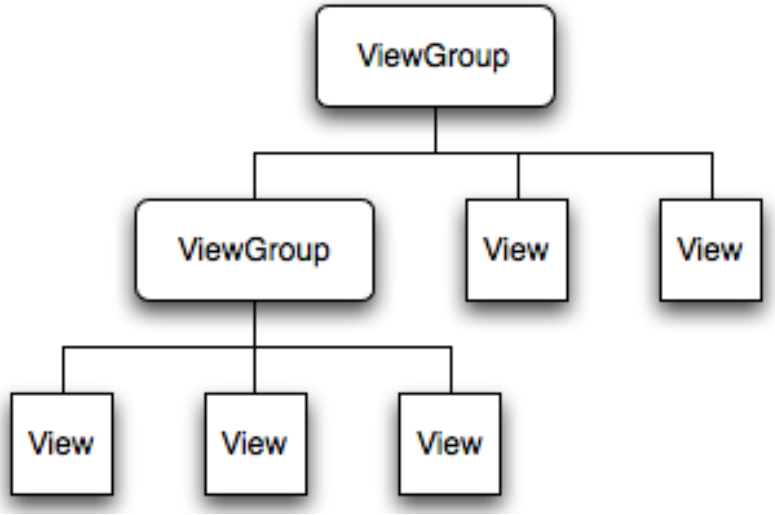


- Ces *widgets* sont eux-même des *View.*

https://developer.android.com/guide/topics/ui/controls.html

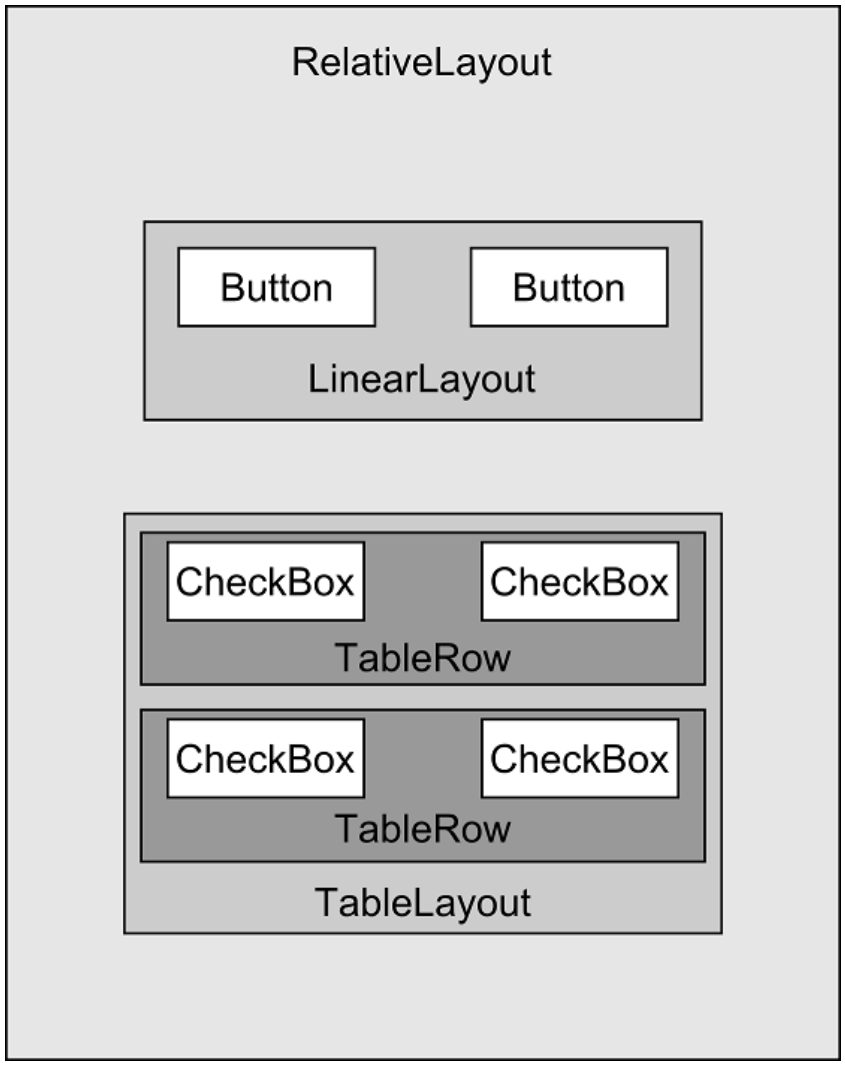
Viewgroup

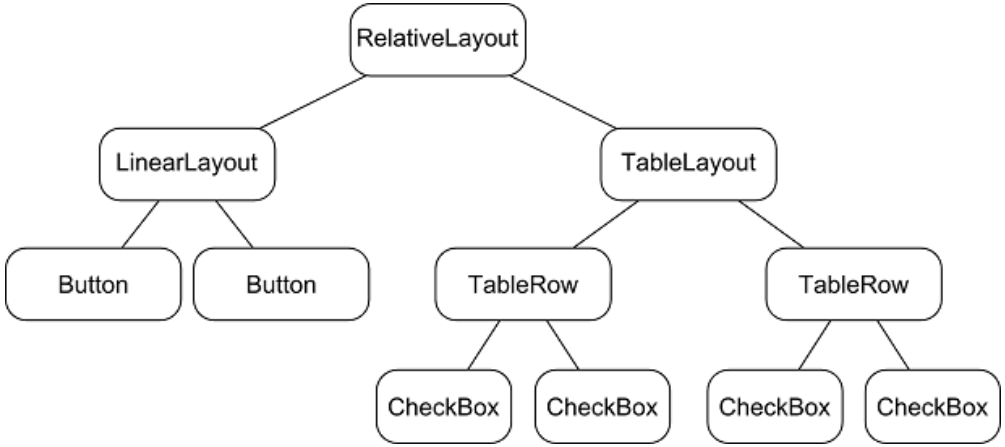
- Type spécial de vue dont le rôle est de gérer plusieurs vues.
- Permet la construction d'interfaces utilisateurs sophistiquées.



https://developer.android.com/guide/topics/ui/overview.html

ViewGroup - exemple





http://www.techotopia.com/index.php/Understanding_Android_Views%2C_View_Groups_and_Layouts_in_Android_Studio

En pratique

- Les écrans des activités sont définis dans des fichiers XML
- Séparation code ⇔ ressources !

```
<?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" >
    <TextView android:id="@+id/text"
              android:layout_width="wrap_content"
              android:layout_height="wrap_content"
              android:text="Hello, I am a TextView" />
    <Button android:id="@+id/button"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Hello, I am a Button" />
</LinearLayout>
```

http://https://developer.android.com/guide/topics/ui/declaring-layout.html

En pratique

- On lie l'écran à l'activity dans sa méthode *onCreate*

```
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.main_layout);
}
```

http://https://developer.android.com/guide/topics/ui/declaring-layout.html

En pratique

- On lie l'écran à l'activity dans sa méthode *onCreate*

```
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.main_layout);
}
```

- Ceci créera la hiérarchie d'objets java.

http://https://developer.android.com/guide/topics/ui/declaring-layout.html

Designer

- On peut aussi créer les écrans à l'aide du Designer
- Cela génère en fait du XML
- Démo designer, …

http://www.techotopia.com/index.php/Designing_a_User_Interface_using_the_Android_Studio_Designer_Tool

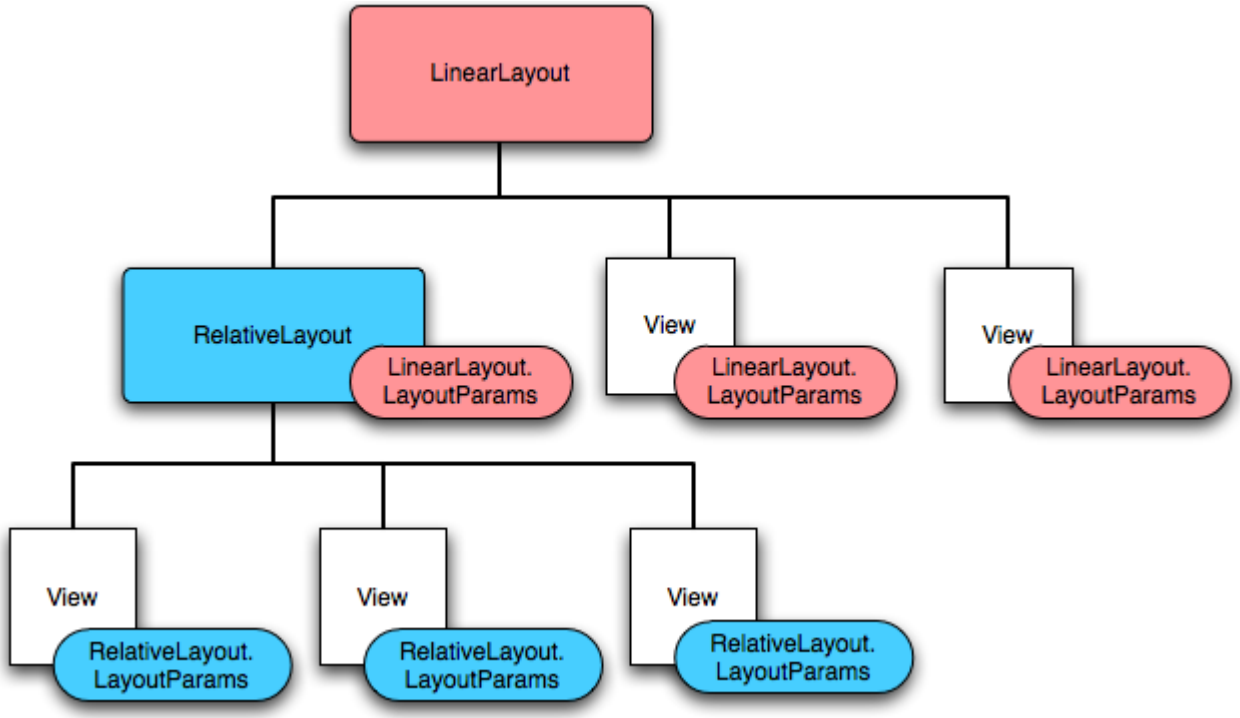
View, Viewgroup, widgets

- Chaque vue est responsable de son affichage et de ce qu'elle contient:
  - si c'est une *View*, de son affichage;
  - si c'est une *Viewgroup*, de l'affichage des vues et/ou groupes de vues qu'elle contient.
- La vue va appeler automatiquement et successivement la méthode *onDraw()* de chacun de ses enfants.
- Le processus d'affichage va donc se propager en parcourant l'arbre.

https://developer.android.com/guide/topics/ui/overview.html

View et Viewgroup

- Chaque view mémorise son type de layout et ses propriétés : taille, position, bordure, marge,…



https://developer.android.com/guide/topics/ui/overview.html

Taille des *Views* et *widgets*

- Souvent la taille d'une vue ou d'un *widget* est définie :
  - en fonction de ce qu'il contient :
    - *WRAP_CONTENT*
  - ou en fonction de la taille de son parent :
    - *FILL_PARENT*

https://developer.android.com/guide/topics/ui/overview.html

Quelques Viewgroups

- Les *Viewgroups* les plus souvent utilisés :
  - *FrameLayout*
  - *LinearLayout*
  - *TableLayout*
  - *AbsoluteLayout*
  - *RelativeLayout*
  - *ScrollView*

https://developer.android.com/guide/topics/ui/declaring-layout.html

FrameLayout

- Le *FrameLayout* est le plus simple des layouts.
- C’est un espace réservé de l’écran qui ne peut contenir qu’un seul composant (une image par exemple).
- Ce composant sera « accroché » en haut à gauche. On ne peut pas spécifier une autre position.
- Ce composant peut être remplacé par un autre.

https://developer.android.com/guide/topics/ui/declaring-layout.html

QuelquesViewgroups

- Les *Viewgroups* les plus souvent utilisés :
  - *FrameLayout*
  - *LinearLayout*
  - *TableLayout*
  - *AbsoluteLayout*
  - *RelativeLayout*

https://developer.android.com/guide/topics/ui/declaring-layout.html

LinearLayout

- Le *LinearLayout* permet d'afficher des composants de façon linéaire :
  - soit horizontalement (par défaut)
  - soit verticalement
- Les composants sont affichés les uns après les autres.
  - Si c'est un layout vertical, il n'y aura qu'un seul composant par ligne quelque soit sa taille.

https://developer.android.com/guide/topics/ui/declaring-layout.html

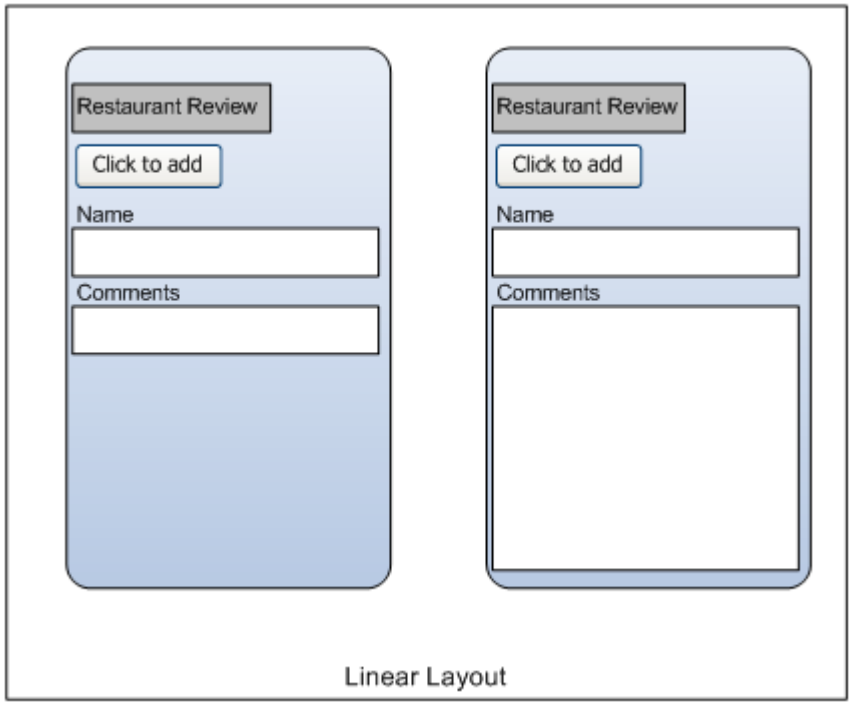
LinearLayout

- Permet d'attribuer un « poids » spécifique à chaque composant, le *« weight ».*
- Afin de s'étirer en utilisant l'espace restant disponible.
- Le « poids » par défaut est 0;
- Supposons que l'on attribue les « poids » suivants aux composants qui doivent se partager l'espace d'une vue:
  - *poids des composants A, B et C : 0*
  - *poids du composant D : 1*

=>*Ce composant D va utiliser tout l'espace restant disponible de la vue* (voir figure de droite page suivante)

https://developer.android.com/guide/topics/ui/declaring-layout.html

LinearLayout



- Dans la figure de gauche, tous les composants ont le même « poids »: 0
- Dans la figure de droite le composant « *Comments* » utilise l'espace restant disponible car il a un poids de 1;
- Si le composant « *Name* » avait aussi un poids de 1, il aurait la même hauteur que le dernier composant.

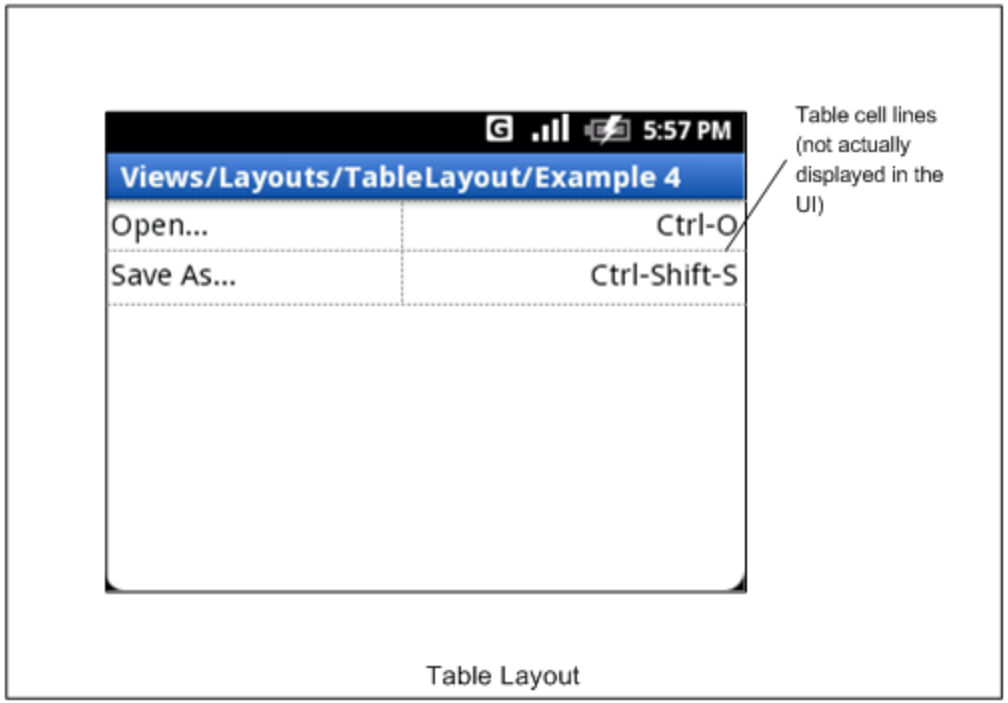
https://developer.android.com/guide/topics/ui/declaring-layout.html

QuelquesViewgroups

- Les *Viewgroups* les plus souvent utilisés :
  - *FrameLayout*
  - *LinearLayout*
  - *TableLayout*
  - *AbsoluteLayout*
  - *RelativeLayout*

https://developer.android.com/guide/topics/ui/declaring-layout.html

TableLayout



Les composants sont disposés en lignes et en colonnes.

https://developer.android.com/guide/topics/ui/declaring-layout.html

TableLayout

- Le *TableLayout* contient plusieurs instances de *TableRow*, une par ligne.
- Chaque *TableRow* contient les composants d'une ligne, une vue par cellule.
- La table possède autant de colonnes que la ligne ayant le plus de cellules.
- Des cellules peuvent rester vides.
- Une cellule ne peut s'étendre sur plusieurs colonnes.

https://developer.android.com/guide/topics/ui/declaring-layout.html

QuelquesViewgroups

- Les *Viewgroups* les plus souvent utilisés :
  - *FrameLayout*
  - *LinearLayout*
  - *TableLayout*
  - *AbsoluteLayout*
  - *RelativeLayout*

https://developer.android.com/guide/topics/ui/declaring-layout.html

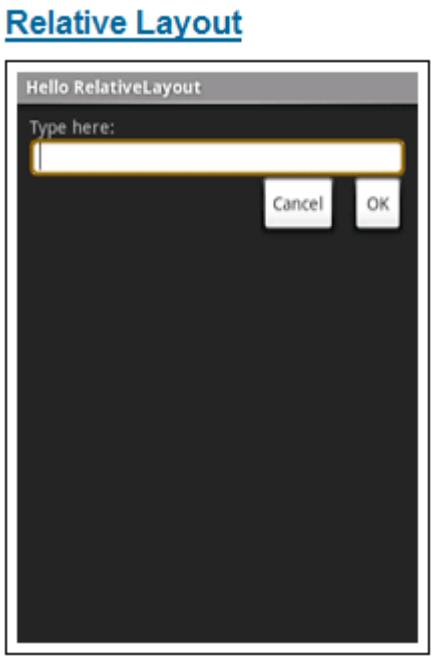
AbsoluteLayout

- L'*AbsoluteLayout* permet de spécifier l'emplacement exacte des composants en renseignant leurs coordonnées x,y;
- L'origine (0,0) est située en haut à gauche de la vue;
- Les composants peuvent se superposer;
- **Il est déconseillé de l'utiliser** pour des raisons de portabilité d'un appareil à l'autre.

https://developer.android.com/guide/topics/ui/declaring-layout.html

QuelquesViewgroups

- Les *Viewgroups* les plus souvent utilisés :
  - *FrameLayout*
  - *LinearLayout*
  - *TableLayout*
  - AbsoluteLayout
  - *RelativeLayout*



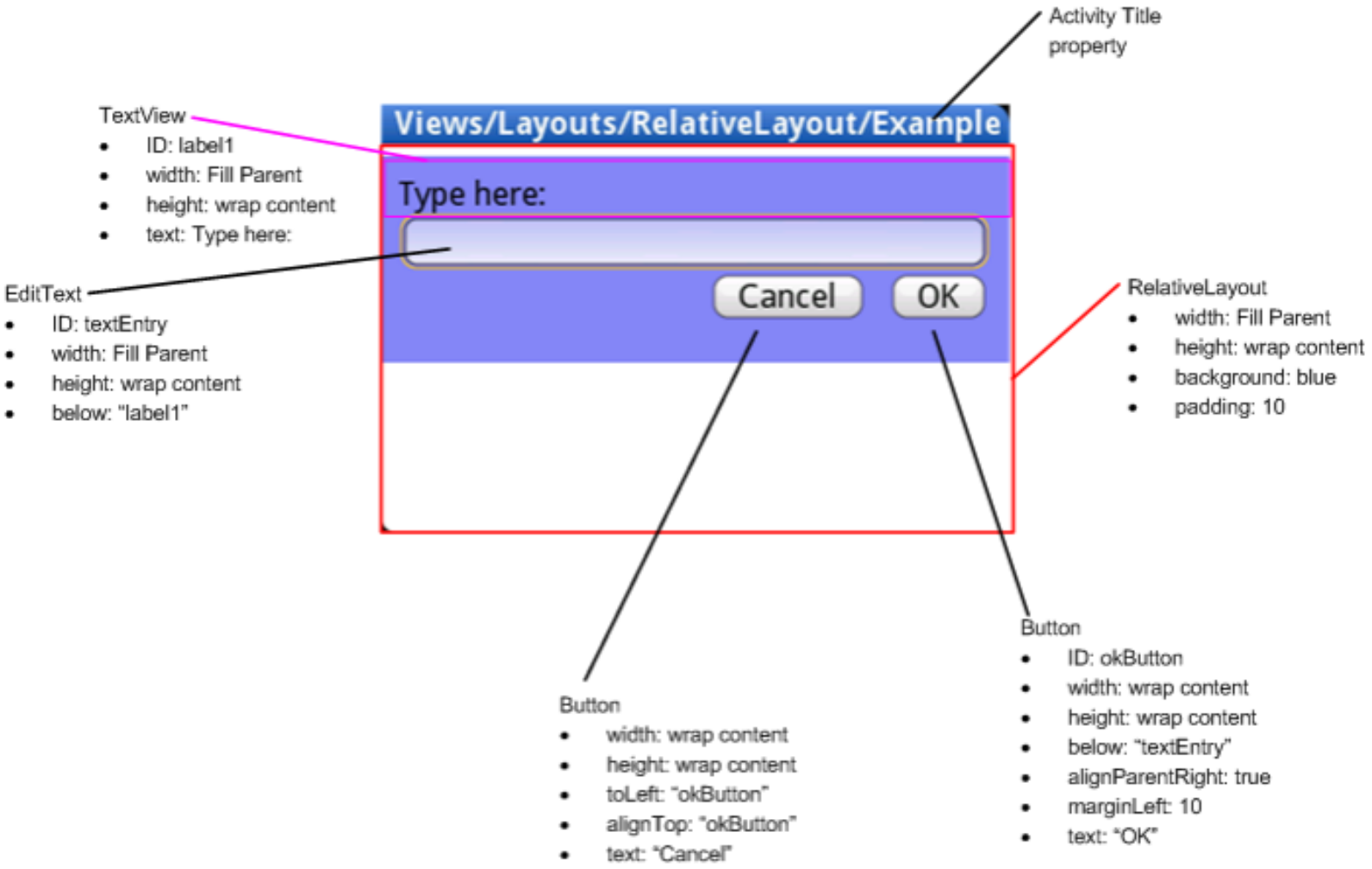
https://developer.android.com/guide/topics/ui/declaring-layout.html

RelativeLayout

- Le *RelativeLayout* permet de positionner les composants :
  - les uns par rapport aux autres;
  - ou par rapport au parent;
- On peut par exemple placer :
  - un composant à la droite d'un autre;
  - un composant au centre de la vue;
  - un composant en dessous d'un autre;
- Les composants sont dessinés suivant leur ordre d'apparition dans le fichier décrivant la vue.

https://developer.android.com/guide/topics/ui/declaring-layout.html

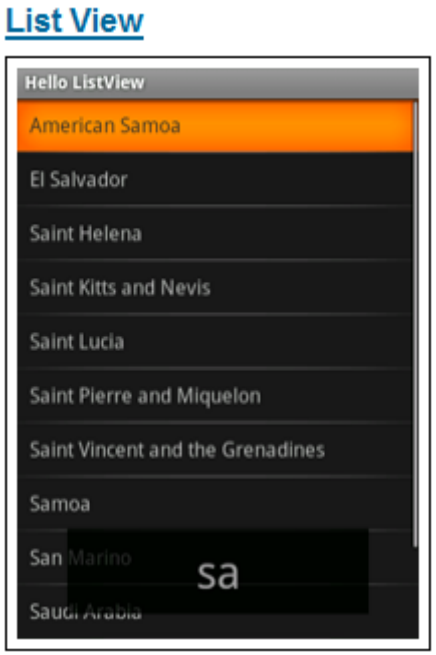
RelativeLayout - exemple



https://developer.android.com/guide/topics/ui/declaring-layout.html

Autres Viewgroups

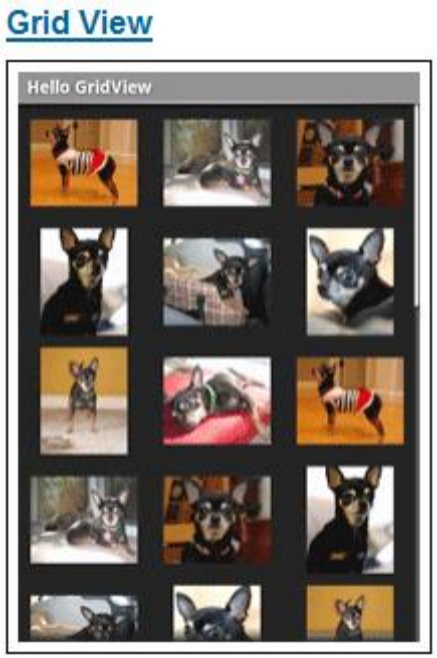
- *ListView :*
  - permet l'affichage et le parcours d'une liste verticale de composants;



https://developer.android.com/guide/topics/ui/declaring-layout.html

Autres Viewgroups

- *GridView :*
  - permet l'affichage et le parcours d'éléments dans une grille



https://developer.android.com/guide/topics/ui/declaring-layout.html

Autres Viewgroups

- *SurfaceView :*
  - permet l'affichage de composants et le dessin de pixels dans la même vue;
  - utilise un thread en arrière plan pour accélérer le rafraichissement de la vue
  - utilise openGL pour les graphiques 3D
- *ViewFlipper :*
  - permet l'affichage d'un composant à la fois;
  - peut servir pour un diaporama.

https://developer.android.com/guide/topics/ui/declaring-layout.html