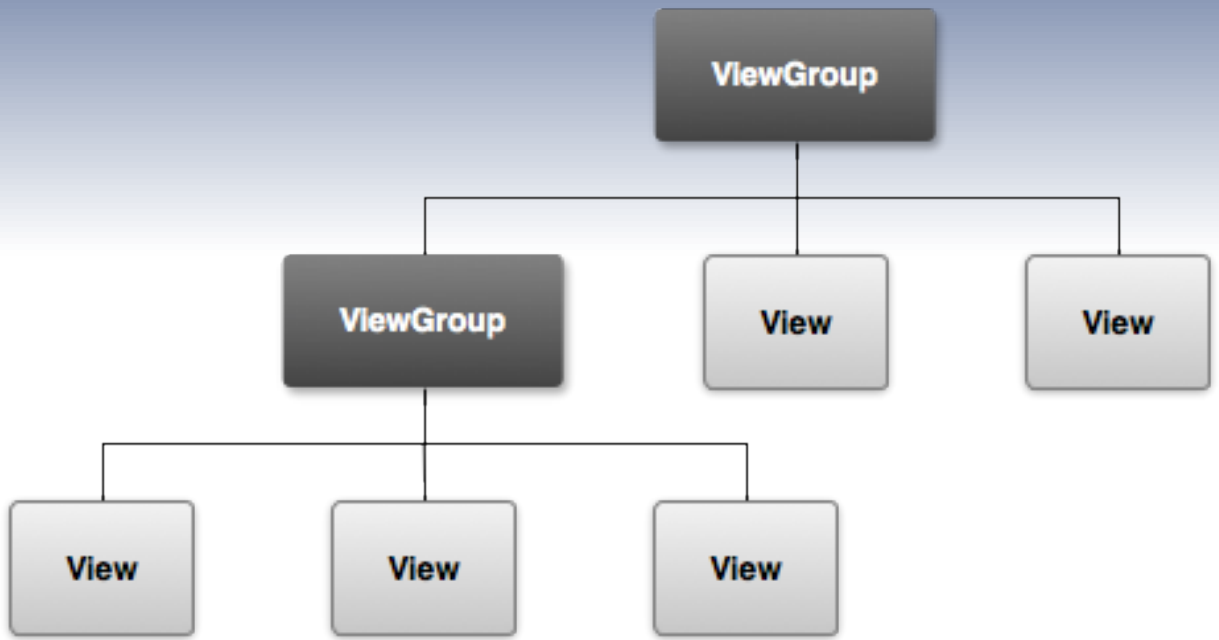
Programmation Android

II. GUI

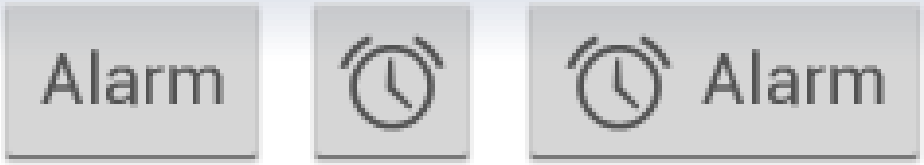
Plan

Organisation d'une GUI Android



- **View** : élément d'interface type *widget* (bouttons, champ texte, etc.)
- **ViewGroup** : un type de **View** contenant d'autres View, gérées par un même gestionnaire de mise en page : positionnement des éléments les uns par rapport aux autres (grille, liste verticale, etc.).

Exemple : les différents types de boutons



Les boutons sont définis grâce à du code xml, par exemple dans *./res/layout/content_main.xml* qui définit le contenu (une **View**) utilisé dans le **ViewGroup** global défini par *./res/layout/activity_main.xml* (la vue hiérarchiquement la plus haute de l'activité, i.e. qui contient les différents éléments)

Texte uniquement

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

Icône uniquement

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

Texte et icône

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

Gestion clique, solution 1 : XML

```
<Button
    android:id="@+id/button_send"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:onClick="sendMessage"
    android:text="@string/button_send" />
```

- Ajout de l'attribut **android :onClick** à l'élément *Button*
- valeur : méthode définie dans l'activité contenant la *view*
- signature standardisée : ***public void*** et un paramètre de type ***View***

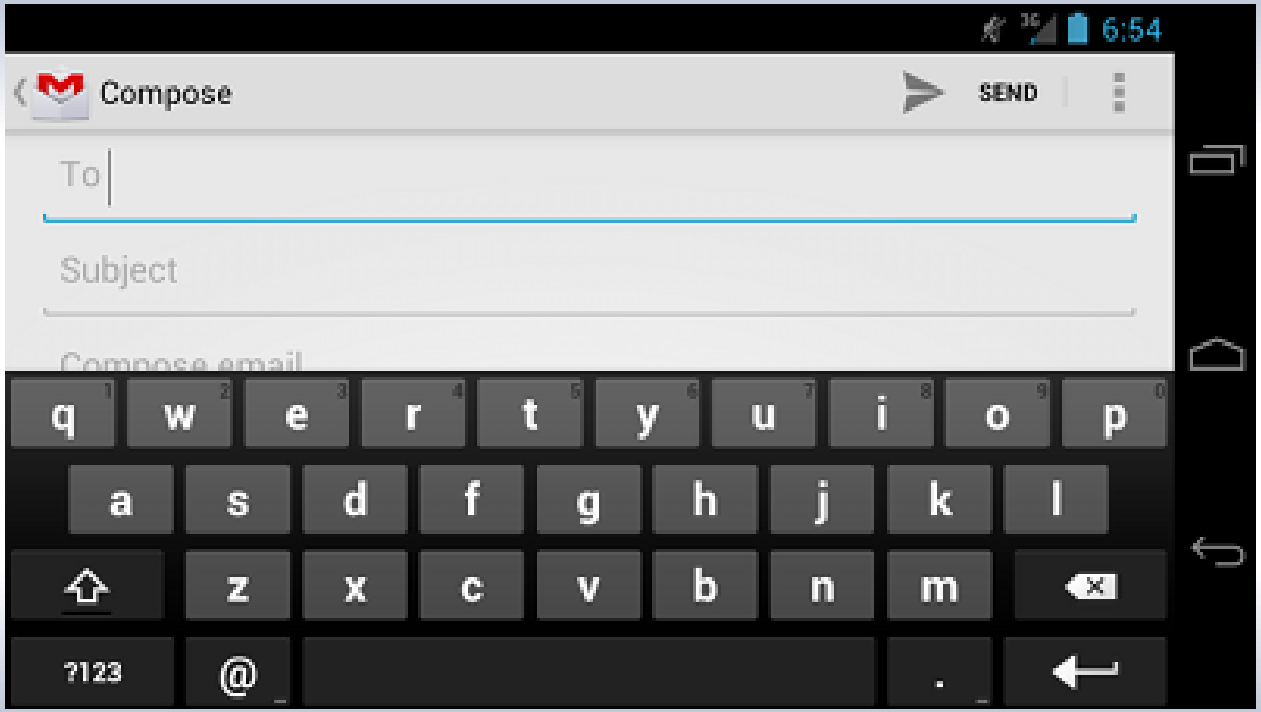
```
/** Called when the user touches the button */
public void sendMessage(View view) {
    // Do something in response to button click
}
```

Gestion clique, solution 2 : dans le code Java

Ajout d'un écouteur (listener) au bouton, par exemple au moment de la création de l'activité :

```
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
    }
});
```

Exemple 2, *Text Fields* : <EditText>



type de clavier : *android :inputType*

```
<EditText
    android:id="@+id/defaultTextField"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:inputType="text" />
```

Clavier par défaut :

type de clavier : *android :inputType*

Pour la saisie d'un email :

```
<EditText
    android:id="@+id/email_address"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:hint="@string/email_hint"
    android:inputType="textEmailAddress" />
```

android :inputType = “phone”

Autres comportements du clavier

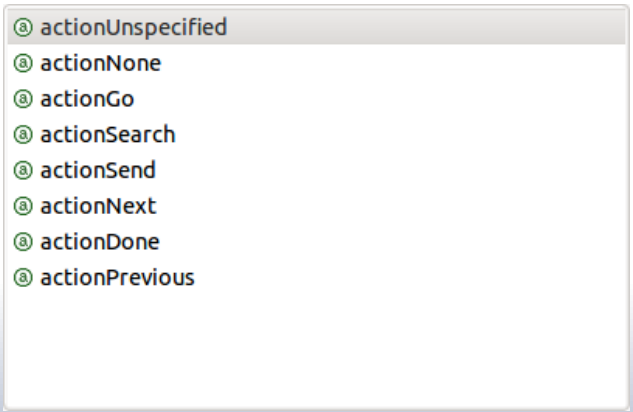
```
<EditText
    android:id="@+id/postal_address"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:hint="@string/postal_address_hint"
    android:inputType="textPostalAddress|
                       textCapWords|
                       textNoSuggestions" />
```

- Tous les possibilités pour android:inputType

Action en fin de saisie :

```
<EditText
    android:id="@+id/search"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:hint="@string/search_hint"
    android:inputType="text"
    android:imeOptions="actionSend" />
```

Autocomplétion sous Eclipse :



▸ Toutes les possibilités pour *android:imeOptions*

Si rien n'est précisé, le focus passe à la *view* suivante dans l'UI : *textField* suivant, bouton, etc.

Lancer une action après validation de la saisie :

À rajouter au moment de la créaction de l'activité :

```
EditText editText = (EditText) findViewById(R.id.search);
editText.setOnEditorActionListener(new OnEditorActionListener() {
    @Override
    public boolean onEditorAction(TextView v, int actionId, KeyEvent event) {
        boolean handled = false;
        if (actionId == EditorInfo.IME_ACTION_SEND) {
            sendMessage();
            handled = true;
        }
        return handled;
    }
});
```

Layouts : gestion de la mise en page

Layout : sous classe de ViewGroup

- Un **Layout** définit la manière dont les **Views** contenues sont disposées les unes par rapport aux autres.
- Des **ViewGroup** standards peuvent être créés avec du code XML

Exemples :

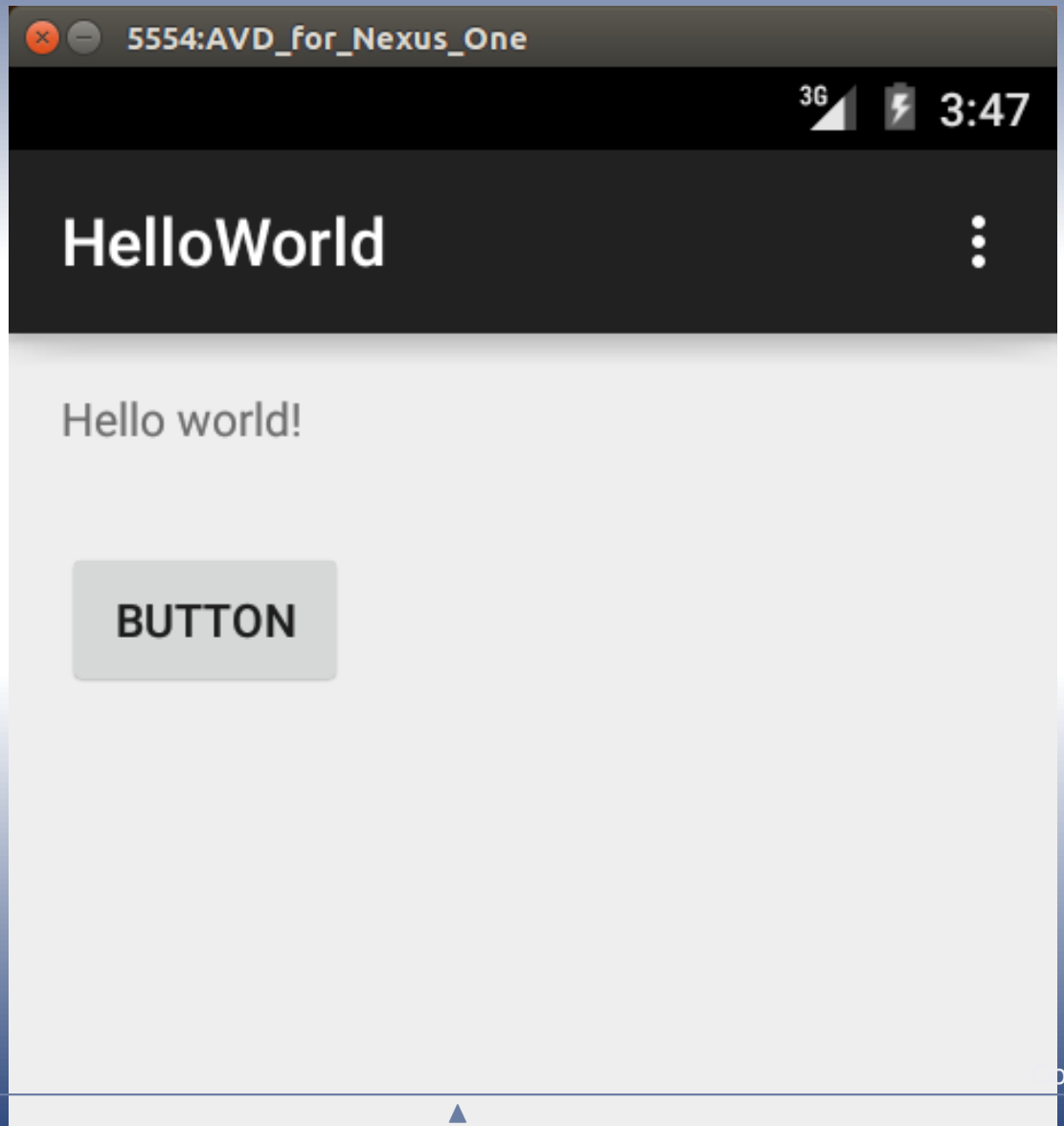
- **RelativeLayout** : chaque **View** définit son déplacement par rapport à une autre **View**
- **LinearLayout** : disposition des éléments en 1 ligne ou 1 colonne dans l'ordre où ils sont définis dans le XML

RelativeLayout

*activity_main.xml

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    tools:context="fr.iutmontp.helloworld.MainActivity" >

    <TextView
        android:id="@+id/textView1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/hello_world" />

    <Button
        android:id="@+id/button1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignLeft="@+id/textView1"
        android:layout_below="@+id/textView1"
        android:layout_marginTop="28dp"
        android:text="Button" />

</RelativeLayout>
```

RelativeLayout



LinearLayout

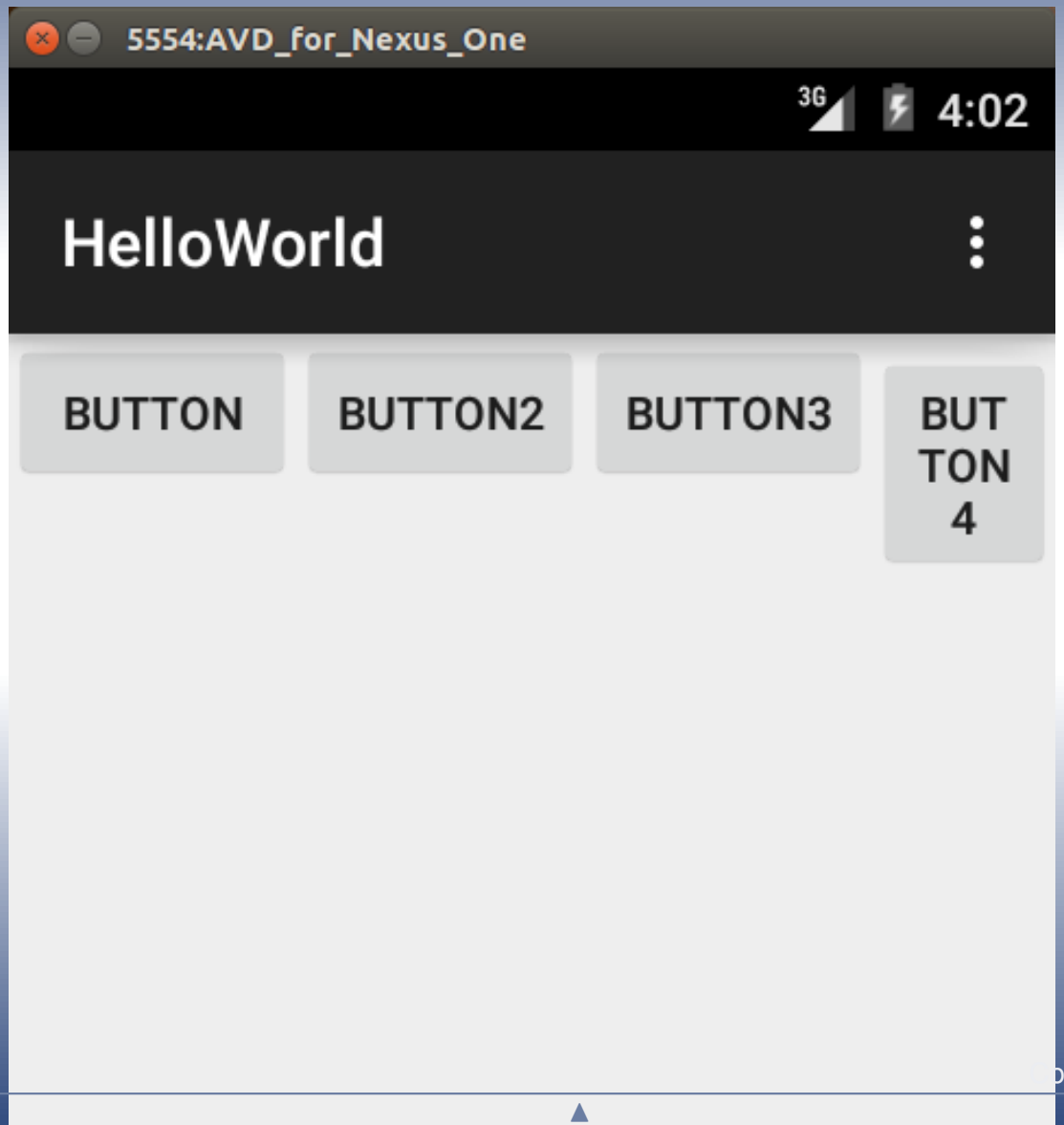
```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="horizontal" >
</LinearLayout>
```

LinearLayout : Ajout de boutons

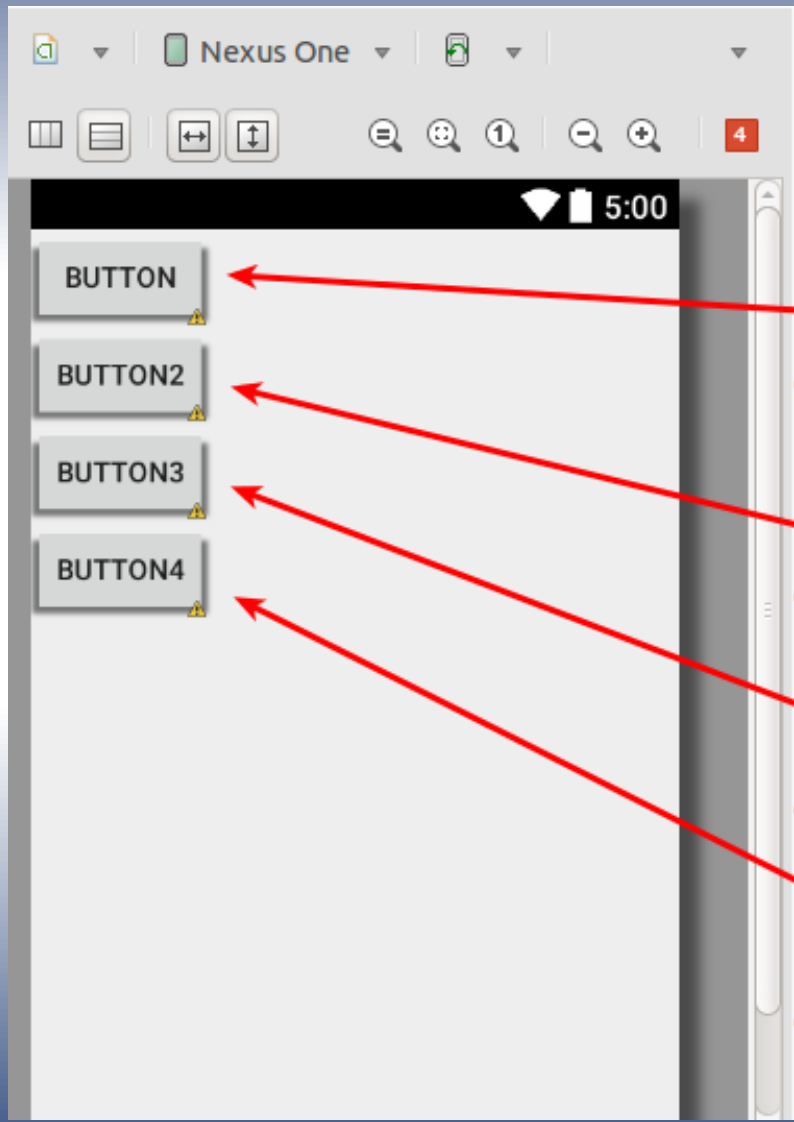
activity_main.xml

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

    <Button
        android:id="@+id/button2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Button" />

    <Button
        android:id="@+id/button1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Button2" />

    <Button
        android:id="@+id/button3"
        android:layout_width="wrap_content"
```

LinearLayout : résultat



LinearLayout en vertical




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

    <Button
        android:id="@+id/button2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Button" />

    <Button
        android:id="@+id/button1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Button2" />

    <Button
        android:id="@+id/button3"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Button3" />

    <Button
        android:id="@+id/button4"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Button4" />

</LinearLayout>
```

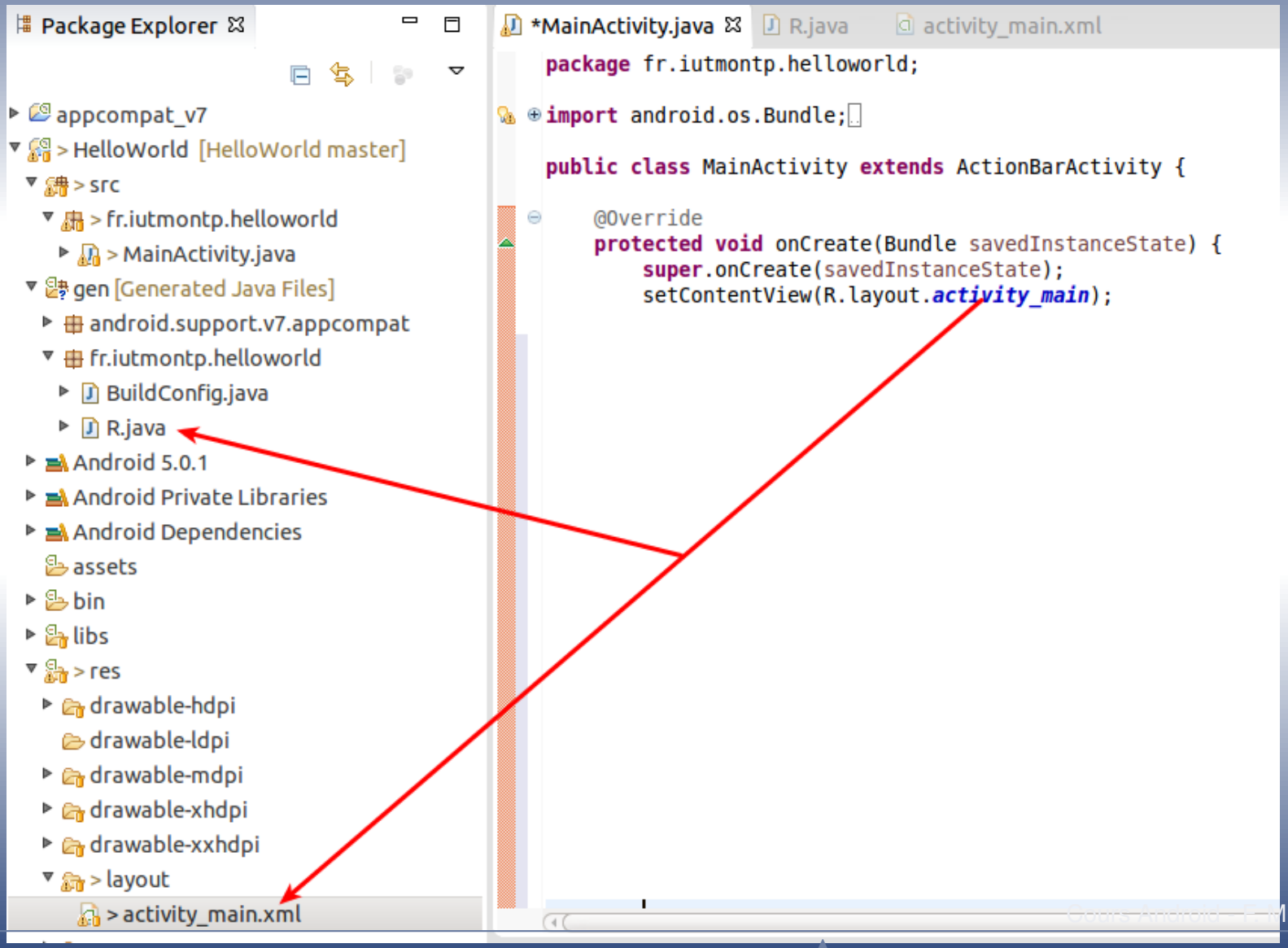
Utilisation des vues créées en XML

- La méthode **OnCreate** est appelée automatiquement à la création de l'activité
- utilisation de la méthode **setContentView** pour définir la GUI principale de l'activité, c'est-à-dire un fichier XML définissant un **ViewGroup** (identifié par un entier unique via la classe R.layout).

*MainActivity.java

```
package fr.iutmontp.helloworld;

import android.os.Bundle;

public class MainActivity extends ActionBarActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

    }
```

Utilisation des vues créées en XML



Identifiant d'une ressource

```
<EditText android:id="@+id/edit_message"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:hint="@string/edit_message" />
```

android :id

- identifiant unique pour l'élément : permet l'accès à l'élément dans le code
- @ est utilisé pour faire référence à une ressource dans du code XML
- **+** est nécessaire uniquement lors de la première déclaration d'une ressource $\rightarrow$ provoque la génération automatique d'un identifiant dans la classe **R**, cette dernière ne doit jamais être modifiée à la main.

Taille d'une View

```
<EditText android:id="@+id/edit_message"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:hint="@string/edit_message" />
```

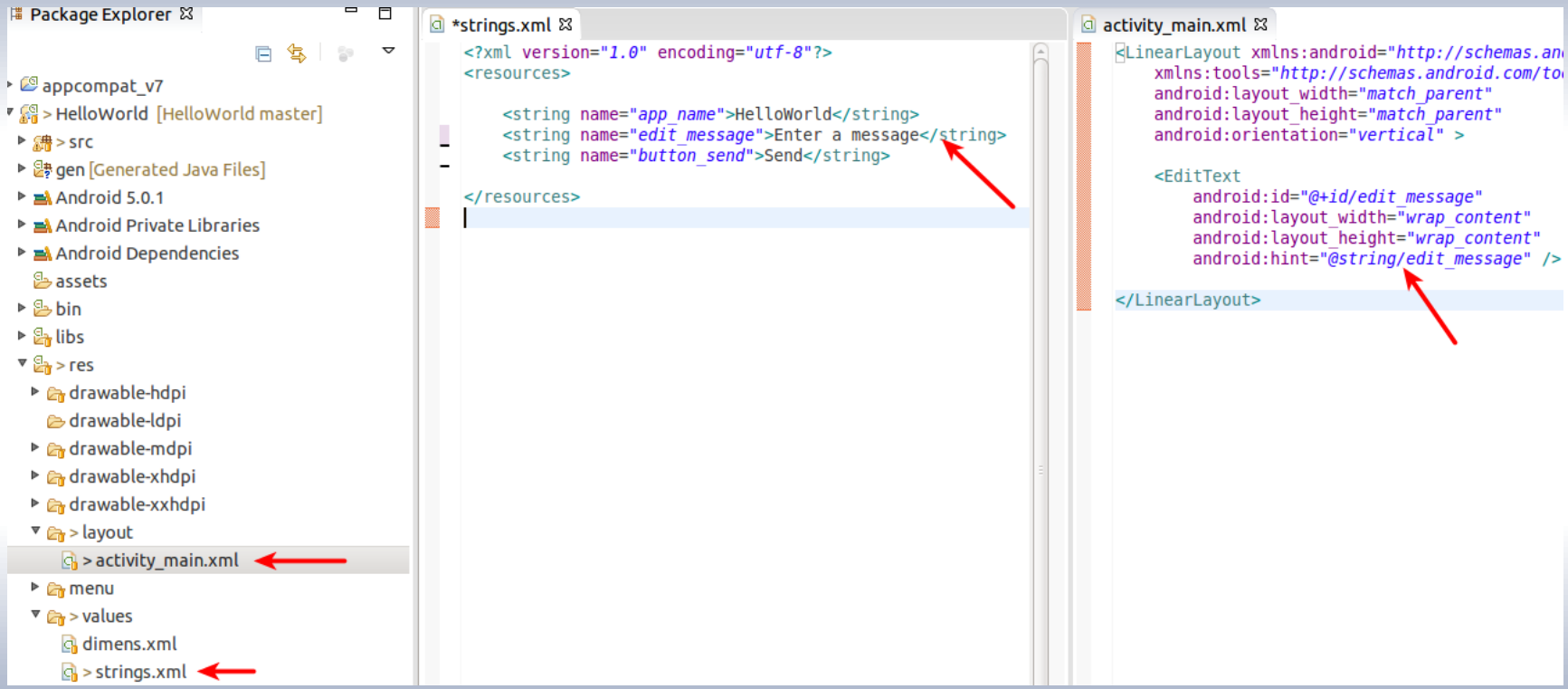
android :layout_width et android :layout_height

- permet de spécifier la taille d'une **View** :
  - relativement à la taille du contenu : **wrap_content**
  - relativement à la taille du **ViewGroup** parent : **match_parent**

android :hint

- texte par défaut dans le TextField
- **Note** : pas de conflit de noms sur *edit_message*, **string** et **id** sont des types de ressources différents

Parenthèse : ressources de type string

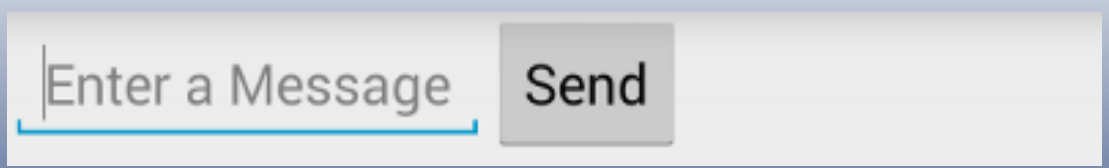


Intérêts : **centralisation** des strings de l'UI et **internationalisation**

Taille d'une View

activity_main.xml | main.xml | MainActivity.java

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

    <EditText
        android:id="@+id/edit_message"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:hint="@string/edit_message" />
    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/button_send" />

</LinearLayout>
```



Adaptation de la taille d'une View à l'espace disponible

```
<EditText
    android:layout_weight="1"
    ... />
```

android :layout_weight

- permet de spécifier le ratio d'espace restant à occuper, en fonction du poids des autres View de même niveau
- 0 est la valeur par défaut de toutes les vues.
- si cette valeur est > 0, la View prend tout l'espace innocupé restant

Adaptation de la taille : optimisation

```
<EditText
    android:layout_weight="1"
    android:layout_width="0dp"
    ... />
```

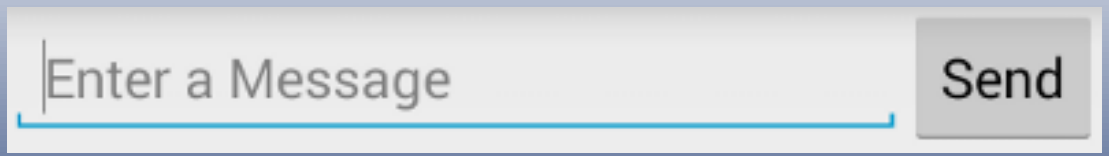
Spécifier la taille à 0 permet d'éviter un calcul qui ne sera pas utilisé, cest-à-dire la valeur de **wrap_content**.

Adaptation de la taille

activity_main.xml | main.xml | MainActivity.java

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

    <EditText
        android:id="@+id/edit_message"
        android:layout_weight="1"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:hint="@string/edit_message" />
    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/button_send" />

</LinearLayout>
```

⇒



Aspect mobile : adaptation UI au contexte

Propriétés d'un écran

- **size** $\rightarrow$ **small**, **normal**, **large** ou **xlarge**
- **density** $\rightarrow$ **low** (ldpi), **medium** (mdpi), **high** (hdpi), **extra high** (xhdpi)

Principe et gestion de l'adaptation

- Chaque layout ou bitmap est placé dans un sous répertoire de **res** ayant un nom lié à la taille et/ou à la résolution correspondantes.
- Note : le **changement d'orientation** (portrait ou paysage) est une **modification de la taille de l'écran**

Gestion de différents layout

```
MyProject/
    res/
        layout/
            main.xml
        layout-large/
            main.xml
```

Un layout par configuration

- Pour chaque taille à supporter : **un fichier layout de même nom**.
- Chaque configuration est placée dans un sous répertoire de **res** correspondant à la taille : **./res/layout-*<screen_size>***/
  e.g. **./res/layout-large**.
- Par défaut, **layout/** est utilisé pour l'orientation portrait.

Gestion de différentes orientation

```
MyProject/
    res/
        layout/                 # default (portrait)
            main.xml
        layout-land/            # landscape
            main.xml
        layout-large/           # large (portrait)
            main.xml
        layout-large-land/      # large landscape
            main.xml
```

Gestion de différentes résolutions

Exemple pour les images

- Il est important de fournir des images avec différentes résolutions
- À partir d'une image vectorielle, on génère 4 images ayant les rapports suivants : xhdpi : 2.0, hdpi : 1.5, mdpi : 1.0, ldpi : 0.75
- e.g. 200x200 xhdpi, 150x150 hdpi, 100x100 mdpi, 75x75 ldpi

Gestion de différentes résolutions

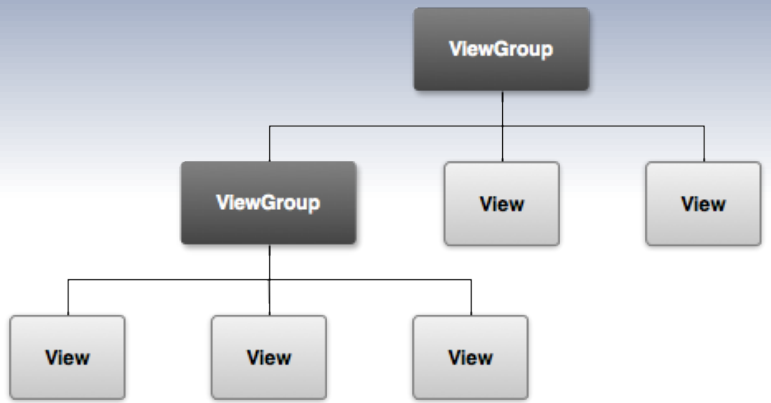
Idem, par sous répertoires de **res** :

```
MyProject/
    res/
        drawable-xhdpi/
            awesomeimage.png
        drawable-hdpi/
            awesomeimage.png
        drawable-mdpi/
            awesomeimage.png
        drawable-ldpi/
            awesomeimage.png
```

Accès automatique à la ressource correspondante

- Dans le code, **`@drawable/awesomeimage`** sélectionnera automatiquement la ressource correspondant à la résolution courante.

Résumé global



- **View** : boutons, textField.
- **ViewGroup** : View contenant d'autres **Views** (layout)
- Adaptation à diverses tailles et résolutions d'écran : gestion par répertoires standardisés

Ce cours reprend largement les tutoriaux en ligne proposés par Google : ▸ Android developers