

# Giao diện người dùng và xử lý sự kiện

---

GV: TRƯƠNG BÁ THÁI

Email: [truongbathai@tdc.edu.vn](mailto:truongbathai@tdc.edu.vn)

ĐT: 0932.577.765

# I MỤC TIÊU THỰC HIỆN

- Xây dựng được giao diện cho ứng dụng
- Xử lý được các sự kiện cho ứng dụng
- Sử dụng thành thạo IDE Android Studio để viết chương trình Android
- Hình thành thói quen thiết kế chương trình theo tiếp cận Top-Down



# Giao diện người dùng

# | Giao diện người dùng

- Giao diện người dùng là một trong những yếu tố quan trọng, quyết định sự thành công của một ứng dụng Android. Ứng dụng Android muốn thành công thì phải có giao diện trực quan, dễ hiểu và dễ sử dụng

# Các loại Layout trong Android

Stt	Layout & Miêu tả
1	<b>Linear Layout</b>  LinearLayout là một view group mà căn chỉnh các view con theo một hướng nào đó: chiều dọc hay chiều ngang
2	<b>Relative Layout</b>  RelativeLayout là một view group mà hiển thị các view con trong các vị trí cân xứng với nhau

# Các loại Layout trong Android

3	<b>Table Layout</b>  TableLayout là một view mà nhóm tất cả các view vào trong các hàng và các cột
4	<b>Absolute Layout</b>  AbsoluteLayout cho phép xác định vị trí chính xác của các view con
5	<b>Frame Layout</b>  FrameLayout là một placeholder trên màn hình mà có thể sử dụng để hiển thị một view đơn

# Các thuộc tính trong Android

Attribute	Miêu tả
<code>android:id</code>	Đây là ID mà nhận diện duy nhất View
<code>android:layout_width</code>	Đây là độ rộng của Layout
<code>android:layout_height</code>	Đây là chiều cao của Layout
<code>android:layout_marginTop</code>	Đây là không gian phụ (extra space) trên cạnh trên của Layout
<code>android:layout_marginBottom</code>	Đây là extra space trên cạnh dưới của Layout
<code>android:layout_marginLeft</code>	Đây là extra space trên cạnh trái của Layout
<code>android:layout_marginRight</code>	Đây là extra space trên cạnh phải Layout

# Các thuộc tính trong Android

<code>android:layout_gravity</code>	Xác định cách các view con được đặt tại đâu
<code>android:layout_weight</code>	Xác định có bao nhiêu extra space trong Layout nên được cấp phát tới View đó



# Các thuộc tính trong Android

<code>android:paddingLeft</code>	Đây là left padding được điền cho Layout
<code>android:paddingRight</code>	Đây là right padding được điền cho Layout
<code>android:paddingTop</code>	Đây là top padding được điền cho Layout
<code>android:paddingBottom</code>	Đây là bottom padding được điền cho Layout



# FrameLayout

# | FrameLayout

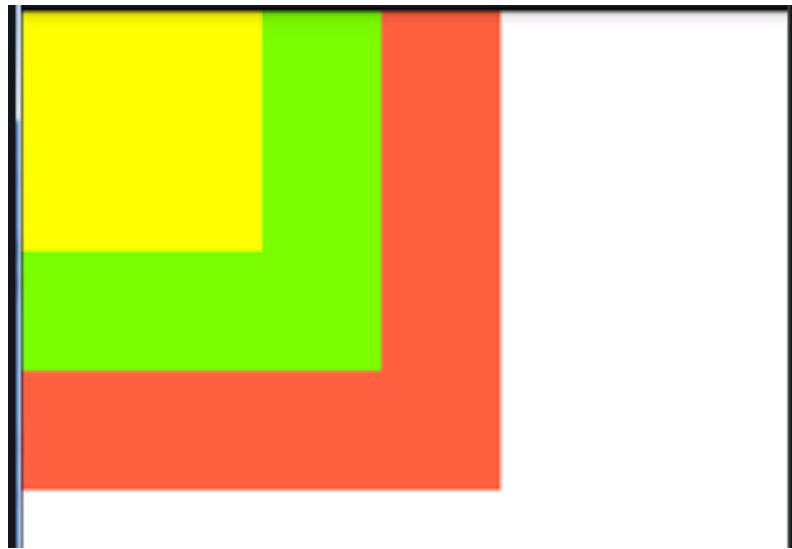
- FrameLayout là một ViewGroup được sử dụng rất nhiều trong android. Bởi vì nó là ViewGroup đơn giản nhất, và thời gian tính toán của nó để layout ra các view con trong nó là thấp nhất nên performance của ViewGroup này là cao nhất

```
<FrameLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <!--View Child-->

</FrameLayout>
```

- Quy tắc layout các view con trong FrameLayout là các view sẽ nằm chồng lên nhau, view thêm vào sau sẽ nằm đè lên view nằm phía dưới.



# Ví dụ:



**<FrameLayout**

**android:layout\_width="match\_parent"**  
**android:layout\_height="wrap\_content">**

**<ImageView**

**android:layout\_width="match\_parent"**  
**android:layout\_height="wrap\_content"**  
**android:src="@drawable/tdc"/>**

**<TextView**

**android:layout\_gravity="bottom|center"**  
**android:layout\_width="wrap\_content"**  
**android:layout\_height="wrap\_content"**  
**android:text="Trường Cao Đẳng Công Nghệ Thủ Đức"**  
**android:textColor="#16a085"**  
**android:textSize="18sp" />**

**</FrameLayout>**

# | Nhận xét:

- Ưu điểm: Là ViewGroup đơn giản nên thời gian tính toán để layout các view con nhanh.
- Nhược điểm: Không thiết kế được cái giao diện phức tạp.

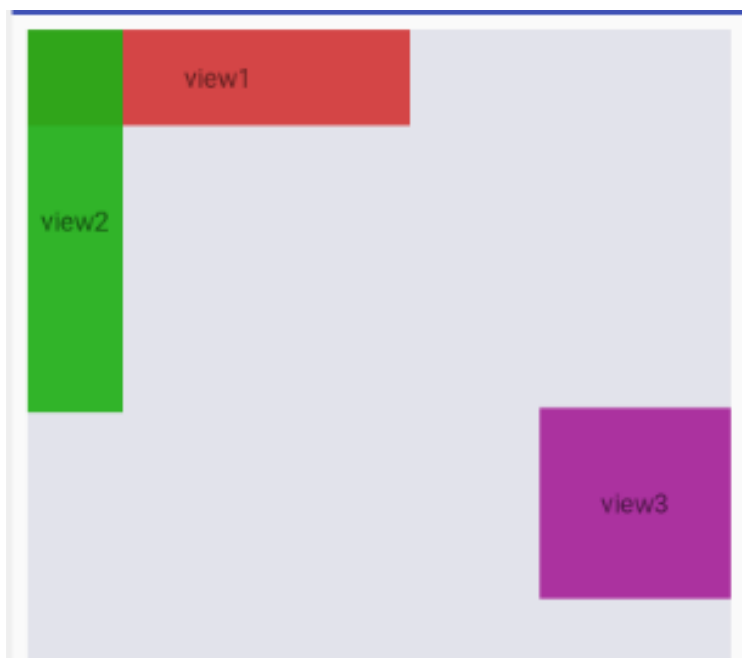




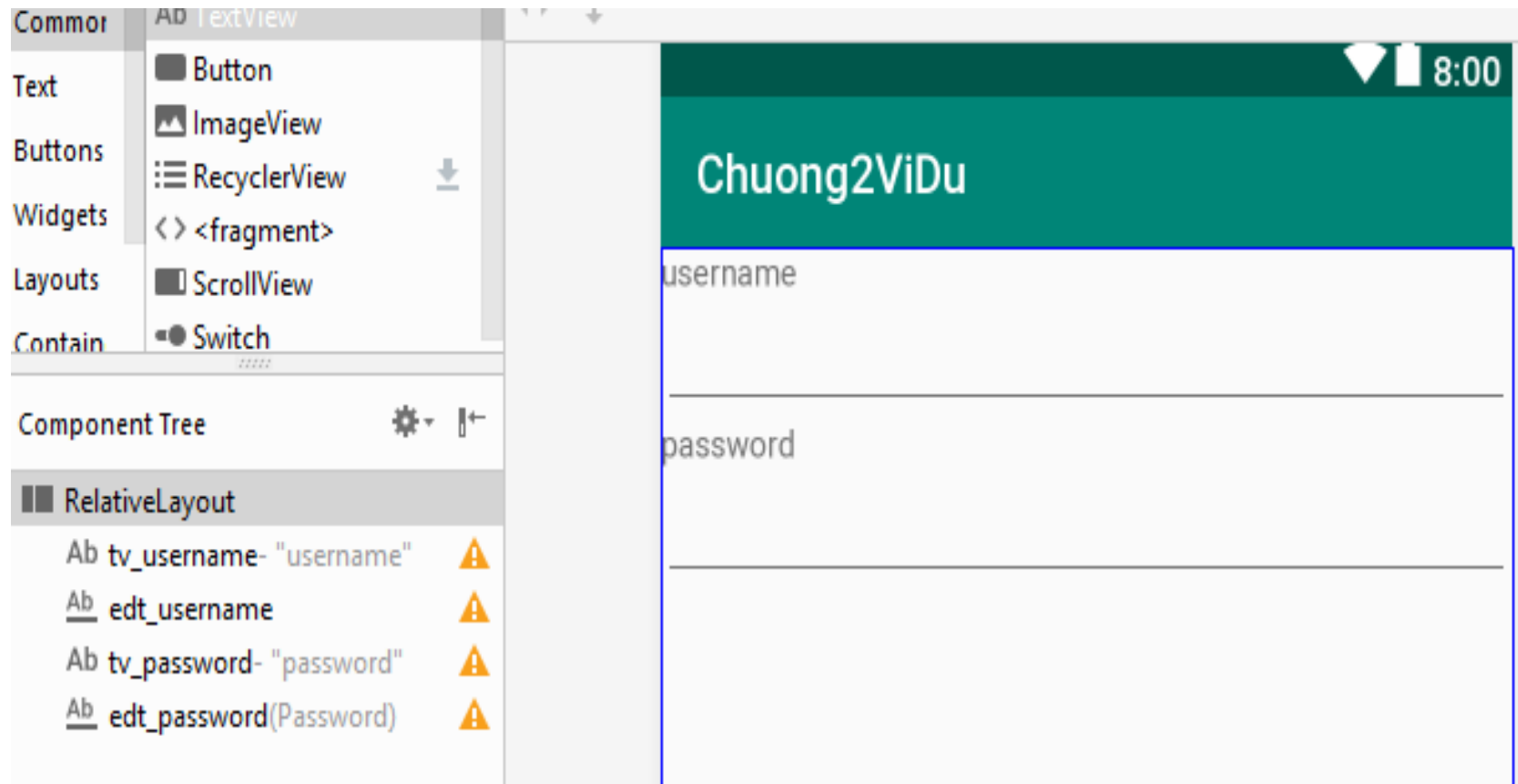
# RelativeLayout

# | RelativeLayout

- Các View khi được đặt lên Layout sẽ có vị trí phụ thuộc vào View đã đặt vào trước nó, do đó khi thay đổi vị trí của một View sẽ làm thay đổi vị trí của các View còn lại



# Ví dụ:



```
<?xml version="1.0" encoding="utf-8"?>
```

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
```

```
    android:layout_width="match_parent"
```

```
    android:layout_height="match_parent">
```

```
    <TextView
```

```
        android:id="@+id/tv_username"
```

```
        android:layout_width="wrap_content"
```

```
        android:layout_height="wrap_content"
```

```
        android:text="username" />
```

```
    <EditText
```

```
        android:id="@+id/edt_username"
```

```
        android:layout_width="match_parent"
```

```
        android:layout_height="wrap_content"
```

```
        android:layout_below="@+id/tv_username" />
```

*<TextView*

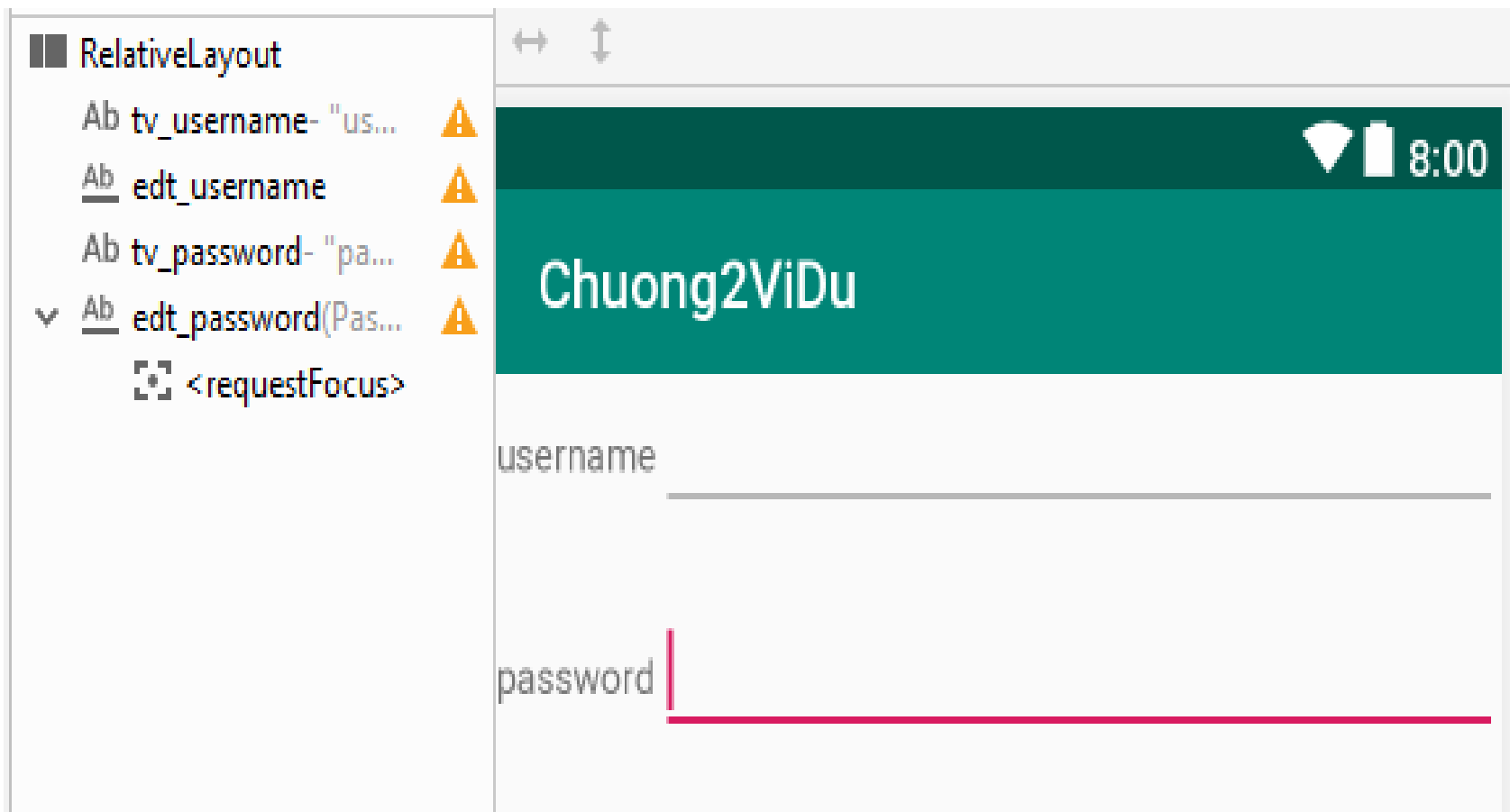
```
android:id="@+id/tv_password"  
android:layout_width="wrap_content"  
android:layout_height="wrap_content"  
android:layout_below="@id/edt_username"  
android:text="password" />
```

*<EditText*

```
android:id="@+id/edt_password"  
android:layout_width="match_parent"  
android:layout_height="wrap_content"  
android:layout_below="@id/tv_password"  
android:inputType="textPassword" />
```

*</RelativeLayout>*

# Ví dụ:



```
<?xml version="1.0" encoding="utf-8"?>
```

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
```

```
    android:layout_width="match_parent"
```

```
    android:layout_height="match_parent">
```

```
    <TextView
```

```
        android:id="@+id/tv_username"
```

```
        android:layout_width="wrap_content"
```

```
        android:layout_height="wrap_content"
```

```
        android:layout_alignBaseline="@+id/edt_username"
```

```
        android:text="username" />
```

```
    <EditText
```

```
        android:id="@+id/edt_username"
```

```
        android:layout_width="match_parent"
```

```
        android:layout_height="wrap_content"
```

```
        android:layout_toRightOf="@+id/tv_username" />
```

**<TextView**

```
android:id="@+id/tv_password"  
android:layout_width="wrap_content"  
android:layout_height="wrap_content"  
android:layout_alignBaseline="@id/edt_password"  
android:layout_below="@id/edt_username"  
android:text="password" />
```

**<EditText**

```
android:id="@+id/edt_password"  
android:layout_width="match_parent"  
android:layout_height="wrap_content"  
android:layout_alignLeft="@+id/edt_username"  
android:layout_alignParentRight="true"  
android:layout_below="@+id/edt_username"  
android:layout_marginTop="22dp"  
android:ems="5"  
android:inputType="textPassword" >
```

**</EditText>**

**</RelativeLayout>**



# | Nhận xét:

- Ưu điểm: Thiết kế được các giao diện phức tạp.
- Nhược điểm: Muốn sử dụng các thuộc tính như `android:layout_above`, `android:layout_toLeftOf` thì phải đặt id cho các view mà view hiện tại xác định vị trí tương đối đối với các view đó.



# LinearLayout

# LinearLayout

- LinearLayout sắp xếp các view con theo hai hướng:
- **Vertical:** Sắp xếp view con theo chiều dọc.
- **Horizontal:** Sắp xếp các view con theo chiều ngang:

`android:orientation="vertical"`



`android:orientation="horizontal"`

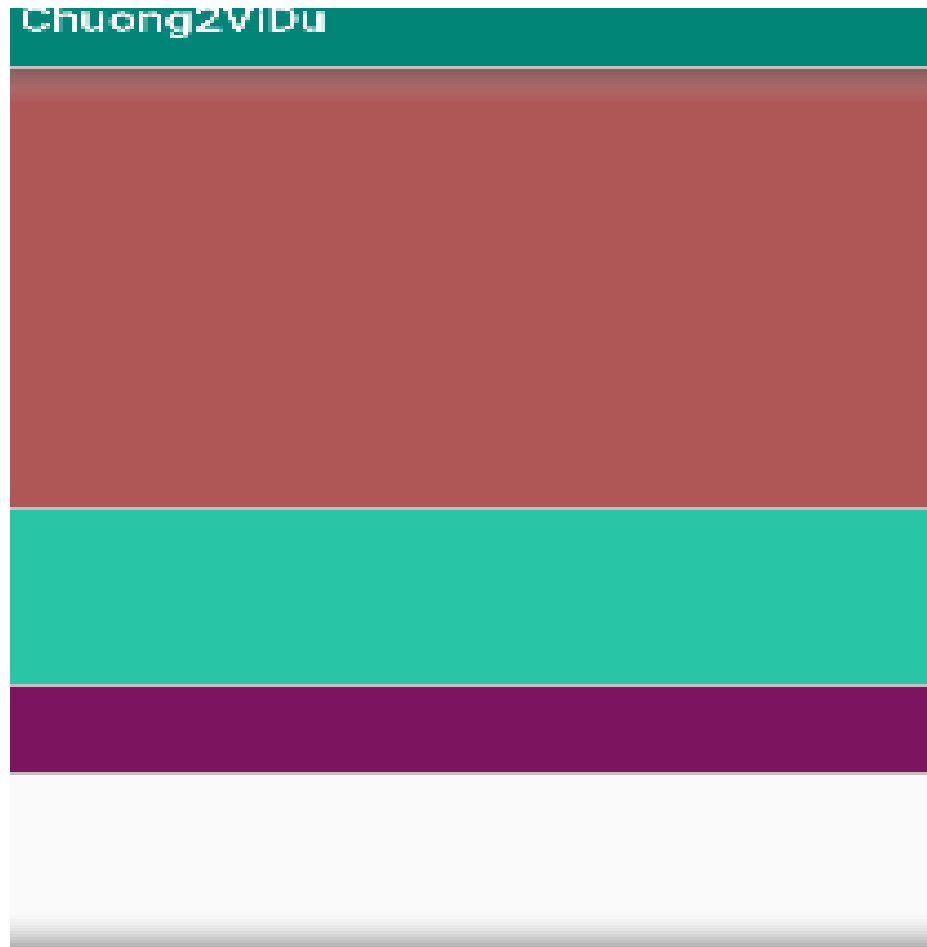


**Android  
LinearLayout**

# | Chia tỉ lệ layout

- **weightSum**: Xác định trọng số của **LinearLayout** hiện tại.
- **layout\_weight**: Trọng số mà view con trong **LinearLayout** chiếm giữ.

# | Ví dụ:



```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
```

```
    android:layout_width="match_parent"
```

```
    android:layout_height="match_parent"
```

```
    android:weightSum="10"
```

```
    android:orientation="vertical">
```

```
<View
```

```
    android:layout_width="match_parent"
```

```
    android:layout_height="0dp"
```

```
    android:layout_weight="5"
```

```
    android:background="#af5656" />
```

```
<View
```

```
    android:layout_width="match_parent"
```

```
    android:layout_height="0dp"
```

```
    android:layout_weight="2"
```

```
    android:background="#28c6a6" />
```

```
<View
```

```
    android:layout_width="match_parent"
```

```
    android:layout_height="0dp"
```

```
    android:layout_weight="1"
```

```
    android:background="#7c145f" />*/LinearLayout>
```

# Ví dụ:

nhập tiêu đề

Nhập nội dung

free for personal use

SEND

```
<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:layout_width="match_parent"
        android:layout_height="0dp"
        android:layout_weight="2"
        android:hint="nhập tiêu đề" />

    <EditText
        android:layout_width="match_parent"
        android:layout_height="0dp"
        android:layout_weight="8"
        android:hint="nhập nội dung" />

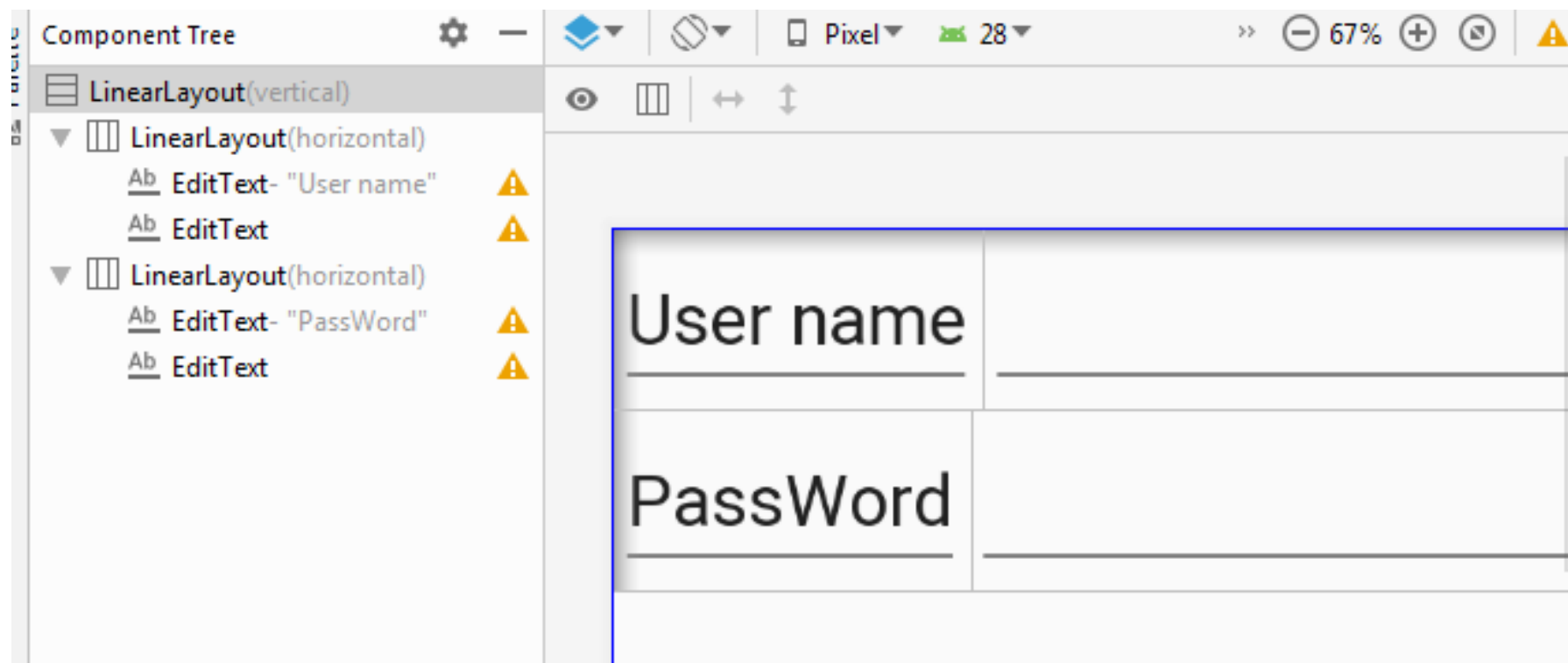
    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="right"
        android:layout_marginRight="10dp"
        android:text="Send" />
</LinearLayout>
```



# Ví dụ:

username

password



```

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

}   <EditText
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
}   android:text="User name" />

}   <EditText
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
}   />
} </LinearLayout>

```

```
<LinearLayout
    android:orientation="horizontal"
    android:layout_width="match_parent"
    android:layout_height="wrap_content">

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

    <EditText
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        />
</LinearLayout>
```

```
</LinearLayout>
```

# | Nhận xét:

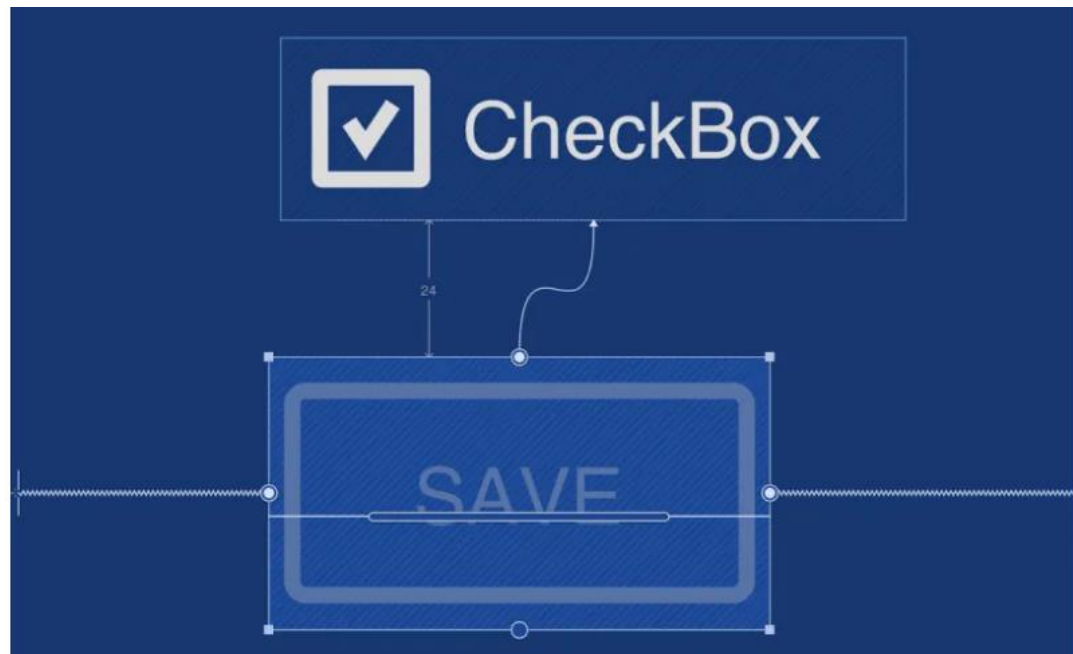
- Ưu điểm
  - Thiết kế được các giao diện phức tạp.
  - Chia tỉ lệ layout, phù hợp với việc phát triển UI trên nhiều device có kích thước màn hình khác nhau.
- Nhược điểm
  - Thời gian tính toán và layout view con tốn chi phí hơn so với `FrameLayout` và `RelativeLayout`



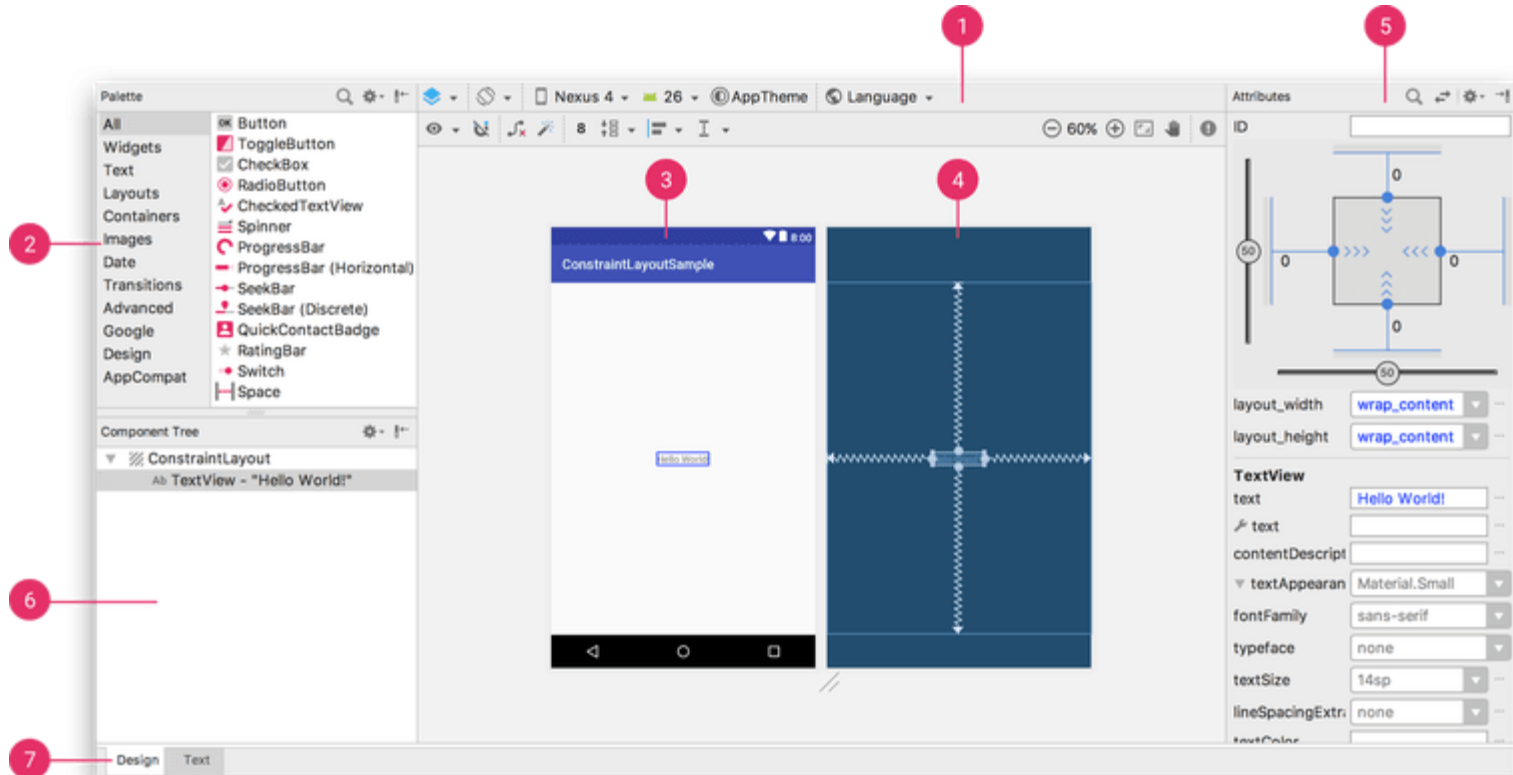
# ConstraintLayout

# | ConstraintLayout

- **ConstraintLayout** là một layout mạnh, vì nó giúp tạo ra các giao diện phức tạp, mềm dẻo (hạn chế tối đa sử dụng các layout lồng nhau).



# Làm Quen Với ConstraintLayout





# Các thành phần:

- **1. Toolbar:** Thanh công cụ, nơi đây chứa một số công cụ quan trọng, lát nữa ở bên dưới chúng ta sẽ tìm hiểu kỹ.
- 2. Palette:** Bảng lựa chọn các view. Bạn có thể thấy các widget, layout và nhiều thành phần giao diện khác ở đây.
- 3. Design view:** Màn hình trực quan. Bạn có thể sử dụng màn hình này để thiết kế và xem luôn kết quả thiết kế như thế nào.
- 4. Blueprint view:** Màn hình xanh đặc biệt dùng cho thiết kế.

# Các thành phần:


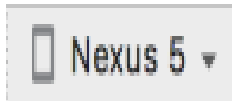
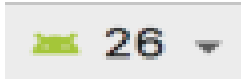
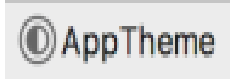
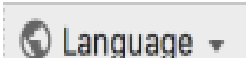
- **5. Attribute:** Nơi chứa đựng các thông số cho việc canh chỉnh giao diện, và một số các thuộc tính khác.
- 6. Component tree:** Các view hiển thị ở *Design view* và *Blueprint view* cũng sẽ được hiển thị ở *Component tree* và được sắp xếp trực quan theo dạng cây để chúng ta dễ dàng theo dõi và quản lý.
- 7. Design & Text:** Hai tab này chắc hẳn quen thuộc với các bạn rồi. Chúng giúp thay đổi cách thức editor hiển thị giao diện ở dạng kéo thả (tab **Design**) hoặc code XML (tab **Text**).

# Toolbar

- Với *toolbar*, bạn có thể chỉ định cho phép hiển thị *Design view*, hoặc *Blueprint view*, hoặc cả hai.

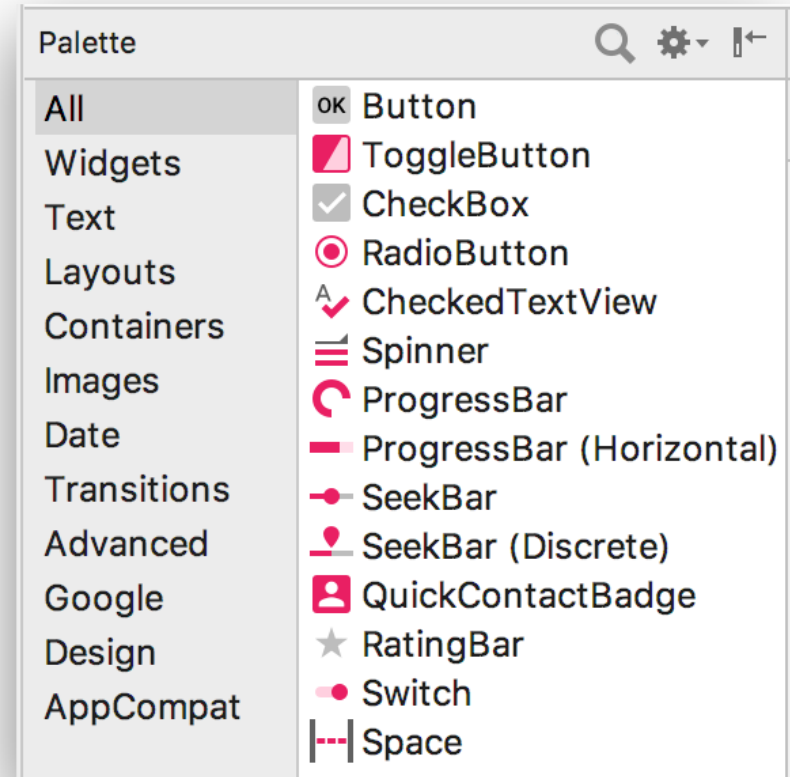


# Toolbar

- Giả lập xoay màn hình ngang/dọc. 
- Chọn loại thiết bị phần cứng giả lập (để xem ngay mà không cần thực thi ứng dụng). 
- Chọn phiên bản hệ điều hành. 
- Chọn theme 
- Chọn ngôn ngữ hiển thị. 

# Palette

- *Palette* là bảng chứa đựng các view, mà bạn có thể kéo thả chúng vào *Design view* hoặc *Blueprint view* một cách thoải mái.

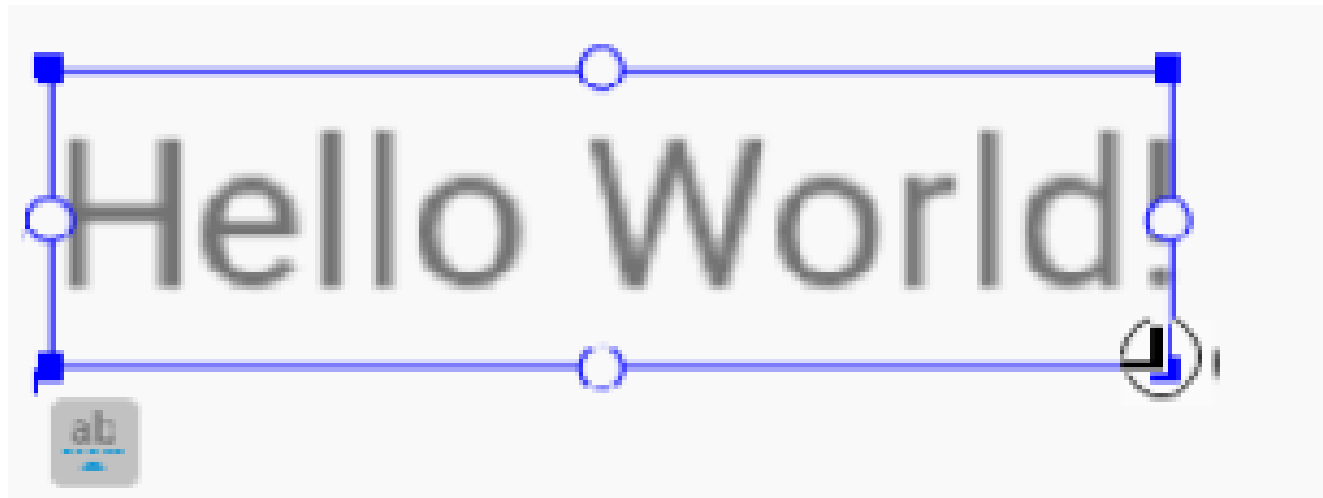


# | Design View & Blueprint View

- *Design view* sẽ cho bạn một kiểu xem trước một cách trực quan. Trong khi *Blueprint view* sẽ giúp loại bỏ hết các giao diện mà chỉ tập trung vào các đường bao của các view, giúp bạn dễ dàng hơn trong việc thiết kế.

# Thay Đổi Kích Thước Cho View

- Ở mỗi góc của các view đều có **các ô vuông**. Nếu đưa trỏ chuột vào một ô vuông nào đó bạn có thể thay đổi kích thước các view.

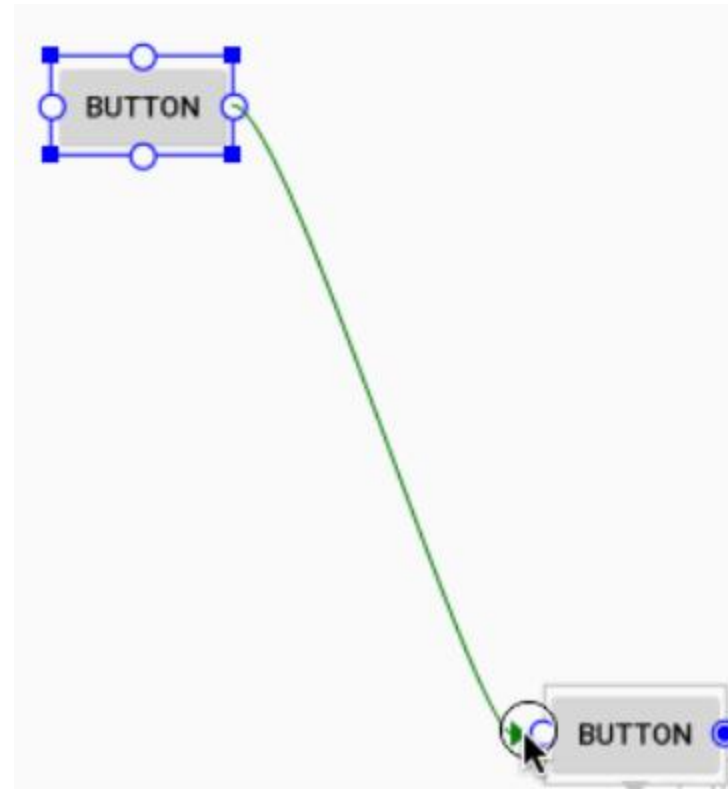


# | Tạo Constraint Cho View

- **Constraint** ở đây được hiểu là các ràng buộc, các nguyên tắc cho các thành phần.
- **Mỗi một view như vậy phải có ít nhất một điểm neo theo chiều ngang và một điểm neo theo chiều dọc**, nếu không đủ các điểm neo tối thiểu, hệ thống sẽ báo lỗi ở cửa sổ *Component tree*.




- Để tạo constraint cho view, thì bạn hãy để ý đến ***các chấm tròn*** ở mỗi cạnh của view.

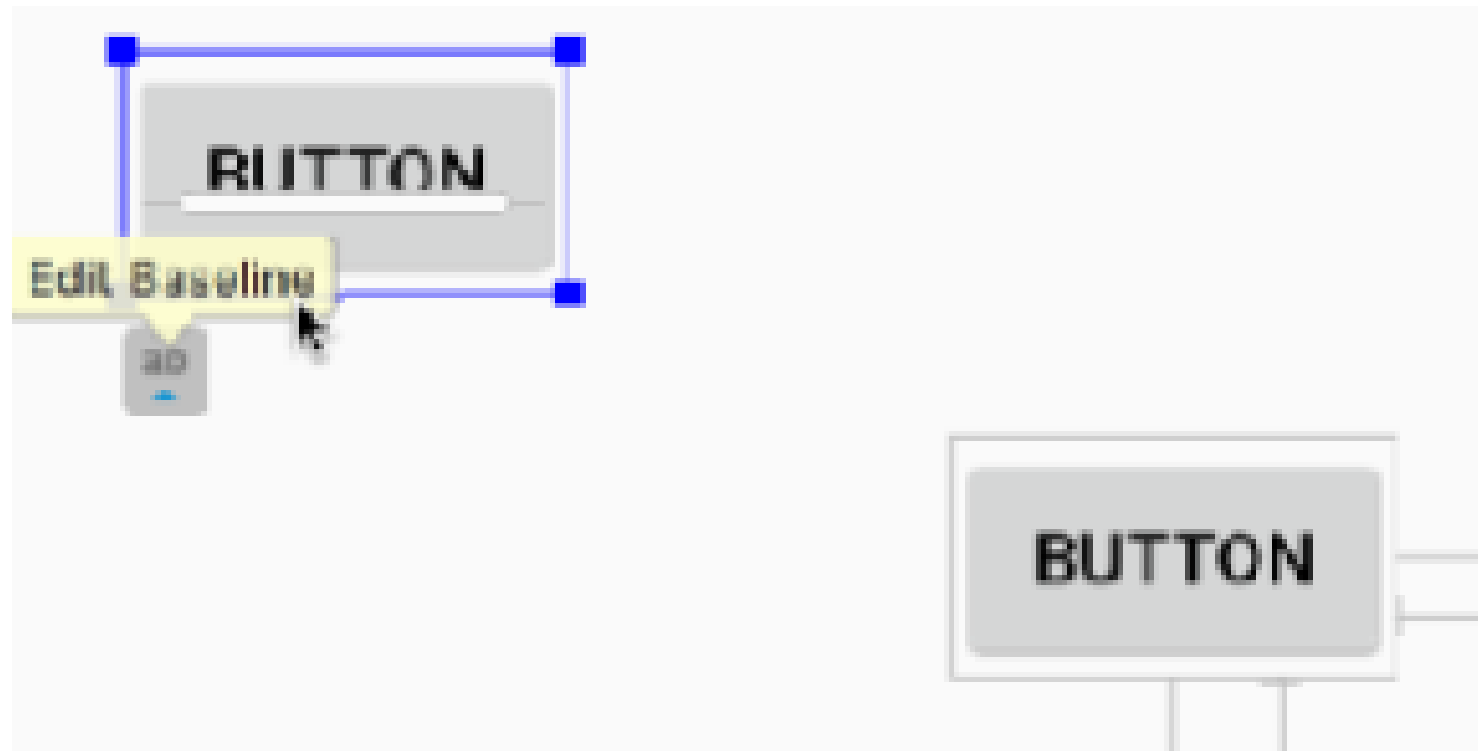


- Để xóa nhiều constraint một lúc.

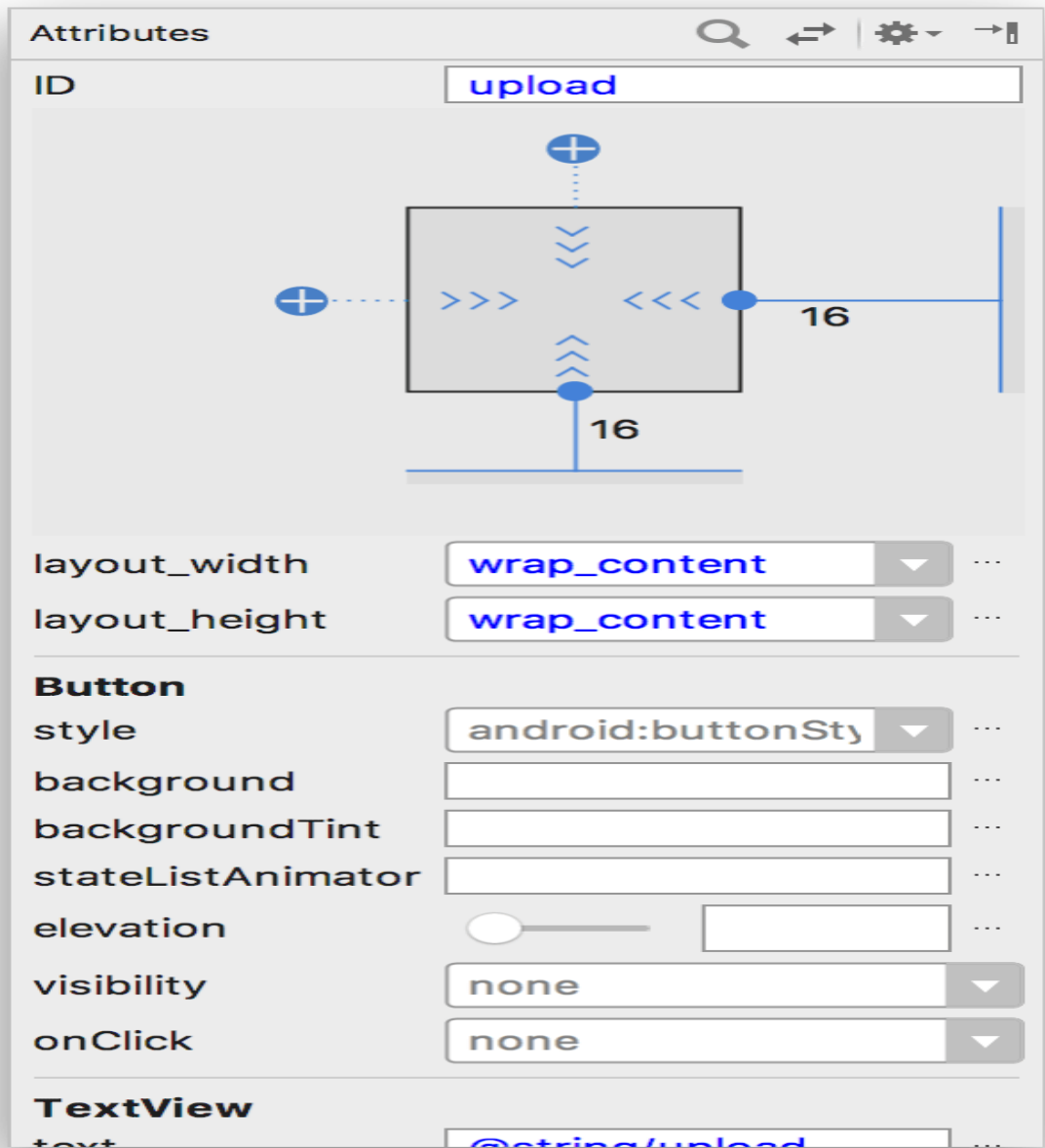


# Tạo Baseline Constraint Cho View

- Nếu như constraint cho view trên kia là các điểm neo giữa view với view. Thì *Baseline constraint* thì lại là sự canh chỉnh các text bên trong một view với nhau. Việc canh chỉnh dựa trên baseline rất thích hợp cho các widget như *TextView*, *EditText* hay *Button*.
- Để tạo ra baseline constraint, bạn hãy nhấn vào một view, rồi nhấn vào nút có ký hiệu 



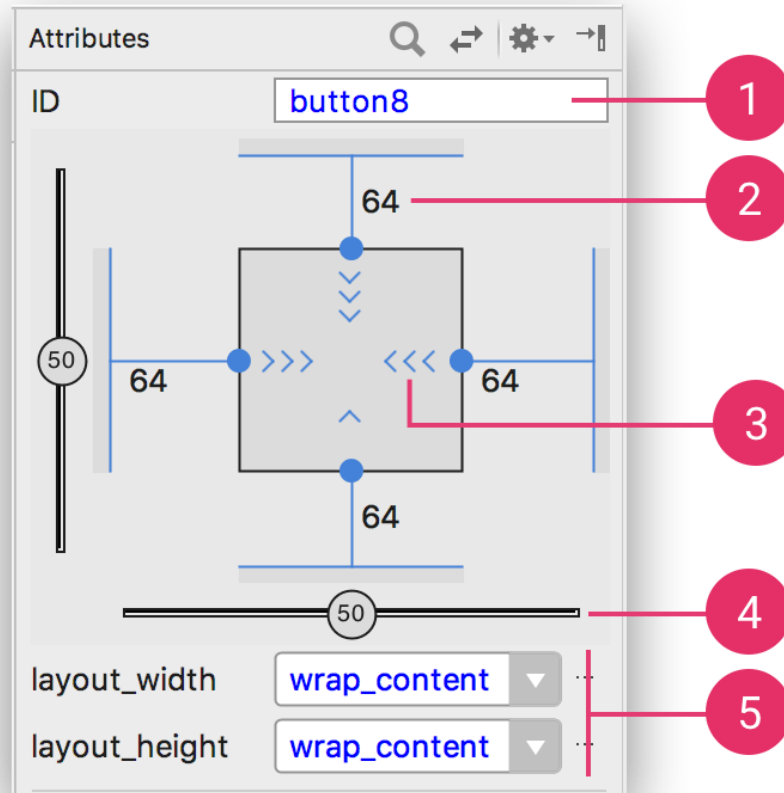
# Attribute



# | Attribute

- *Attribute* của *ConstraintLayout* khá linh hoạt. Được chia ra làm hai phần. Phần bên dưới là các thuộc tính quen thuộc với bạn từ các layout trước. Bên trên các thuộc tính này là một view được gọi là ***View Inspector***.

# Attribute



# 1. ID Của View

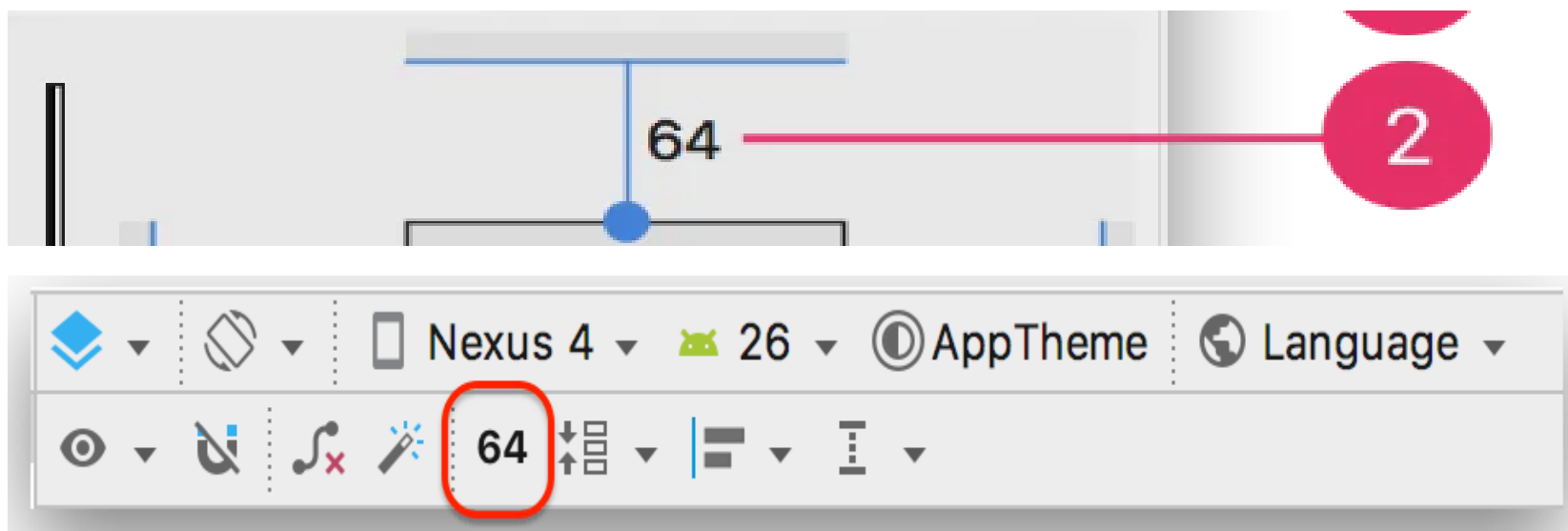
- Khi một view được tạo ra bên trong *ConstraintLayout* sẽ có một *ID* mặc định, nhưng hoàn toàn có thể thay đổi *ID*.








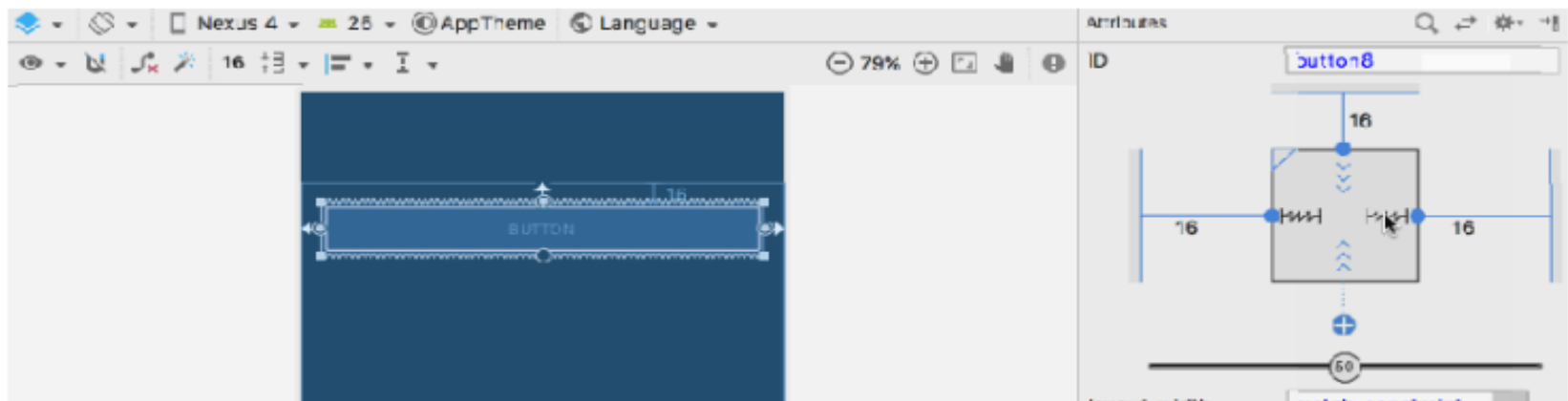
## 2. Margin Của View

- *Margin* chính là khoảng cách của một view đến các view khác. Có thể chỉ định giá trị *margin* mặc định cho tất cả các view bên trong ứng dụng của bạn bằng cách thay đổi con số này ở *toolbar*.



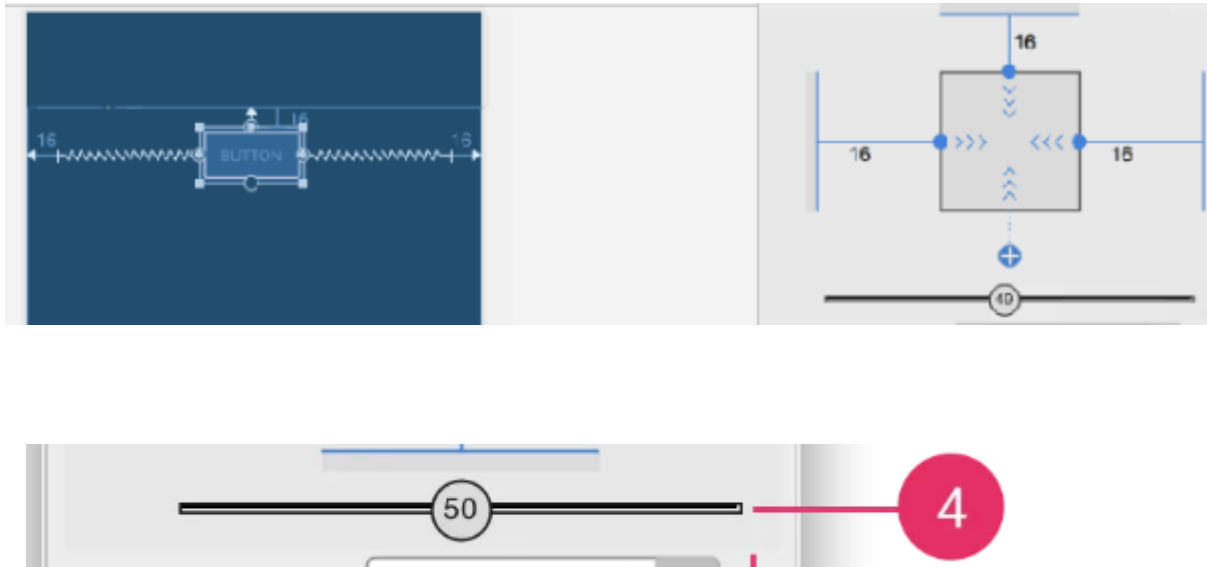
### 3. Các Khoảng Cách Tới Các Thành Viên Bên Trong

- **Fixed:** Nếu bạn chỉ định khoảng cách kiểu này, bạn có thể điền giá trị ở field ***layout\_width*** và ***layout\_height***. 
- **Match Constraint:** kiểu khoảng cách này gần gần giống với ***match\_parent*** quen thuộc ở các layout khác 
- **Wrap Content:** khoảng cách này thì tương tự với ***wrap\_content*** 



## 4. Bias

- Nó mang ý nghĩa giống như trọng số trong *LinearLayout* vậy.





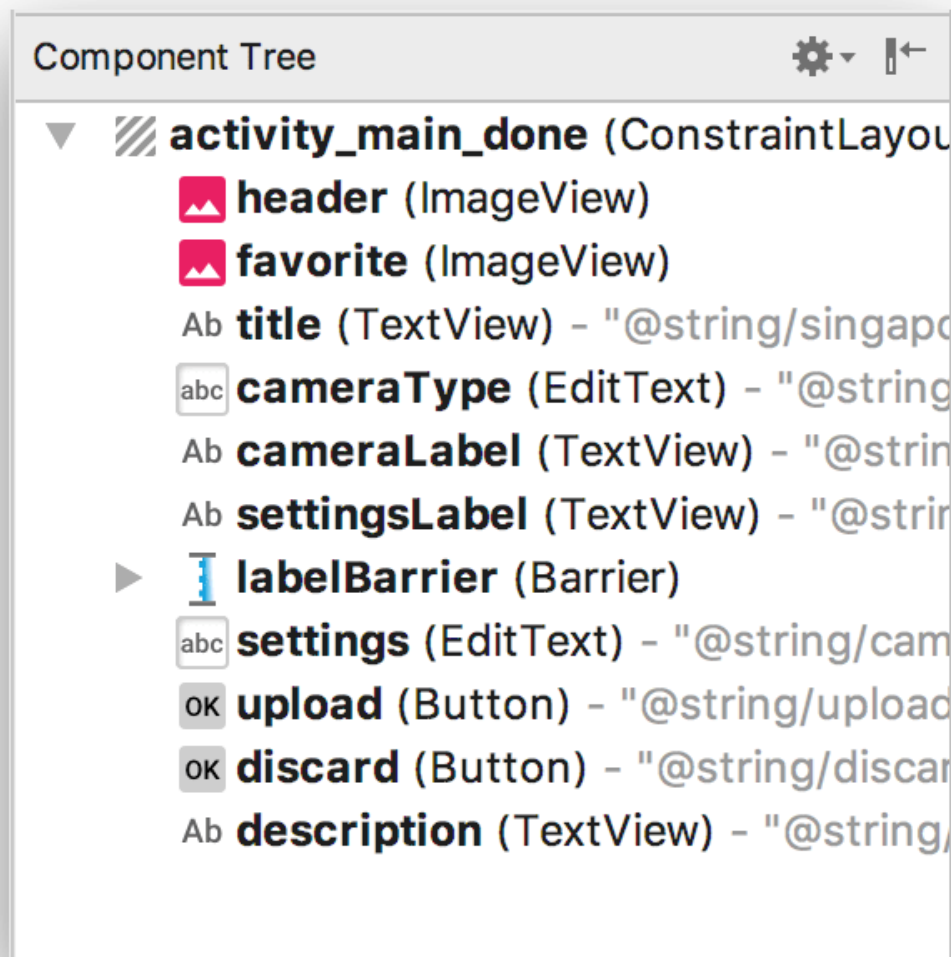
## 5. Chỉ Định layout\_width Và layout\_height

- Phần này chỉ hữu dụng khi bạn chỉ định kích thước cứng (**Fixed**)



# Component Tree

- Cảnh báo 
- Báo lỗi 



# Component Tree

The screenshot displays the Android Studio IDE. On the left, the **Palette** shows various UI components like Button, ToggleButton, CheckBox, etc. Below it, the **Component Tree** shows a **ConstraintLayout** containing a **button8** widget. The main canvas shows a dark blue rectangle with a small button labeled "BUTTON" in the top right corner. At the bottom, a warning message is displayed:

**1 Warning 1 Error** ☐ Show issues on the preview

**Missing Constraints in ConstraintLayout** Correctness button8

This view is not constrained horizontally: at runtime it will jump to the left unless you add a horizontal constraint

The layout editor allows you to place widgets anywhere on the canvas, and it records the current position with designtime attributes (such as `layout_editor_absoluteX`.) These attributes are not applied at runtime, so if you push your layout on a device, the widgets may appear in a different location than shown in the editor. To fix this, make sure a widget has both horizontal and vertical constraints by dragging from the edge connections.

# Chức Năng Autoconnect (Tạo Constraint Tự Động)



Tự tạo các constraint ngay khi thả một view từ *palette* vào *design view* hoặc *blueprint view*.



# Chức Năng Infer Constraints



Chức năng này khi được kích hoạt sẽ tự nó tính toán, suy luận ra các constraint để khớp với bố cục hiện thời của layout. Để sử dụng tính năng này, hãy tìm đến icon toolbar được khoanh đỏ như trên.

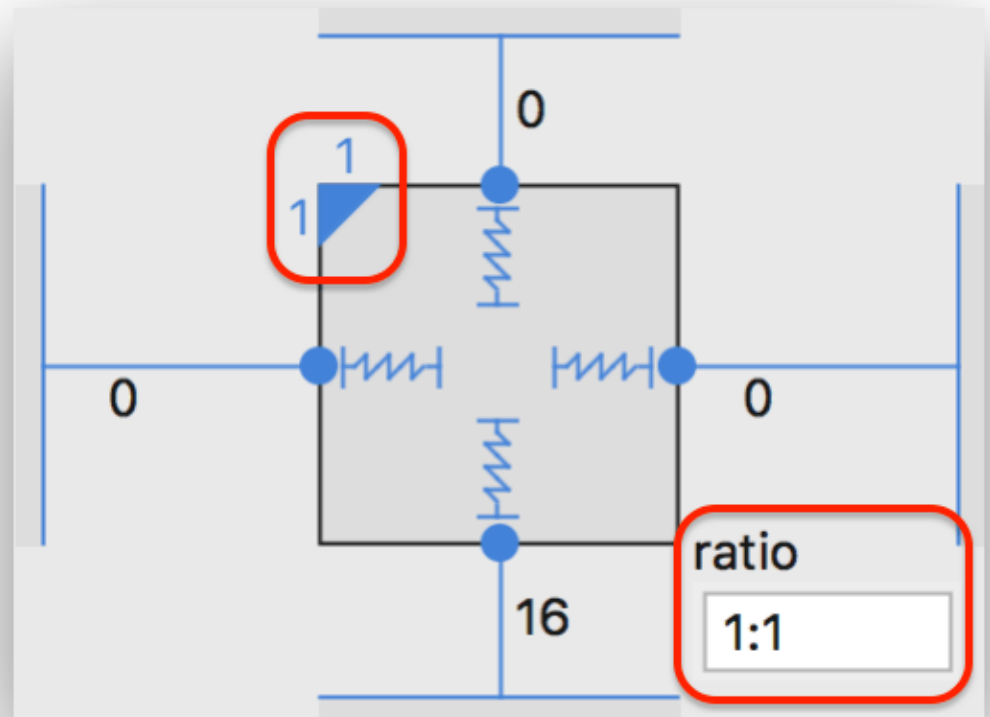
# Chức Năng Pack



Ý nghĩa của chức năng này là *Đóng gói*. Chức năng này thực chất giúp kéo dẫn hết không gian của view theo chiều ngang hay dọc, sự kéo dẫn này không đẩy các view khác đi khỏi vị trí của chúng

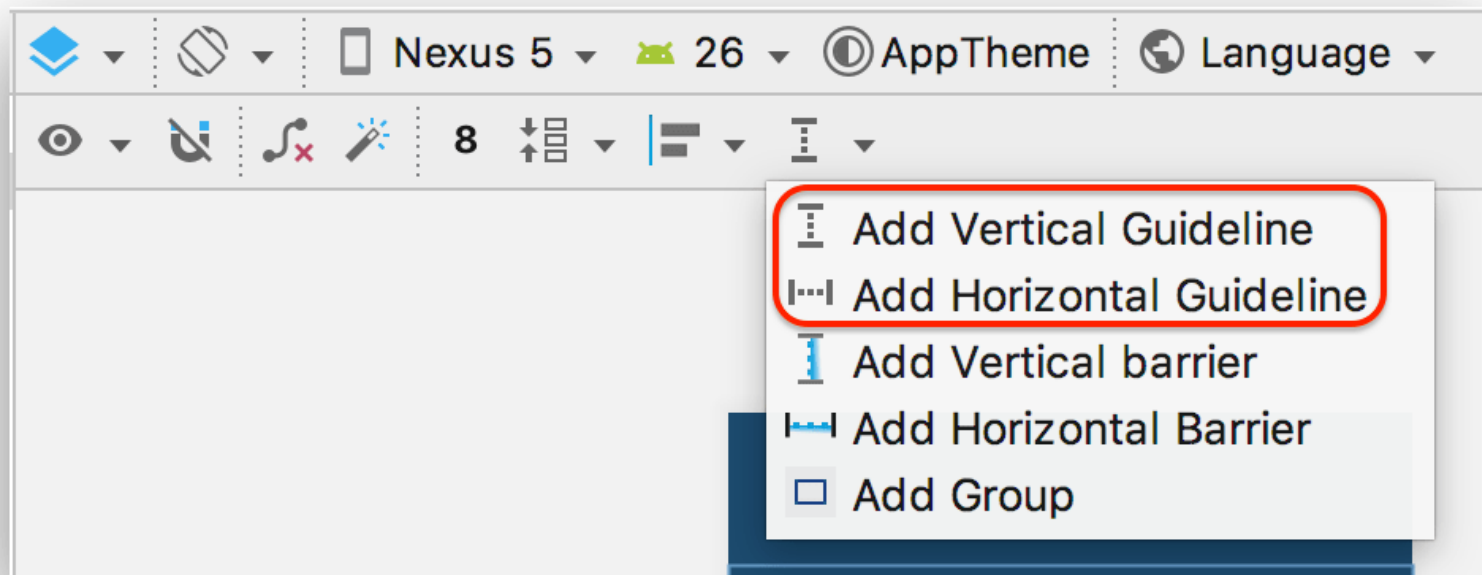
# Chức Năng Ratio

Điều Chỉnh Kích  
Cỡ View Theo  
Tỷ Lệ



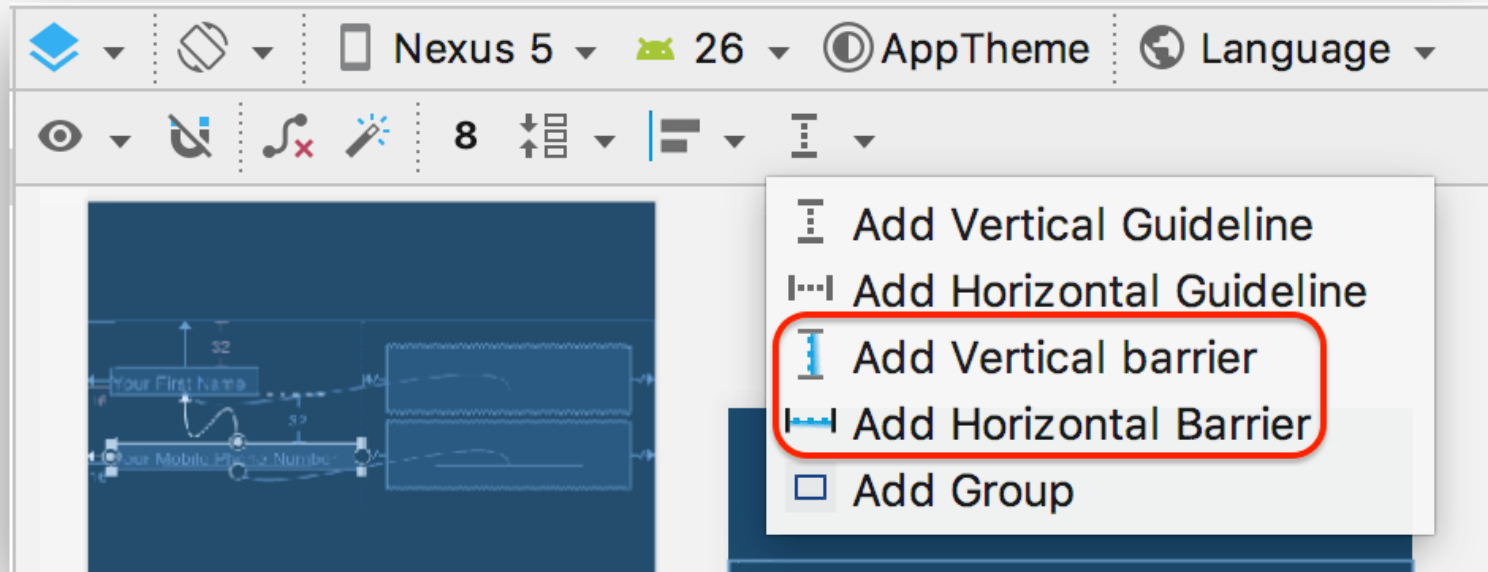
# Chức Năng Guideline

## Neo Constraint Vào Đường Biên



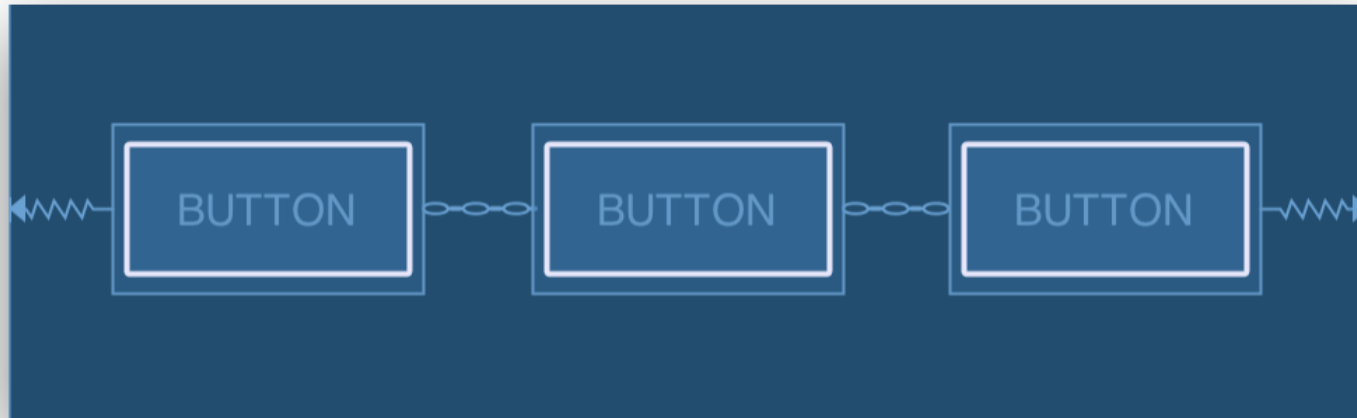
# Chức Năng Barrier

## Neo Constraint Vào Đường Biên Động

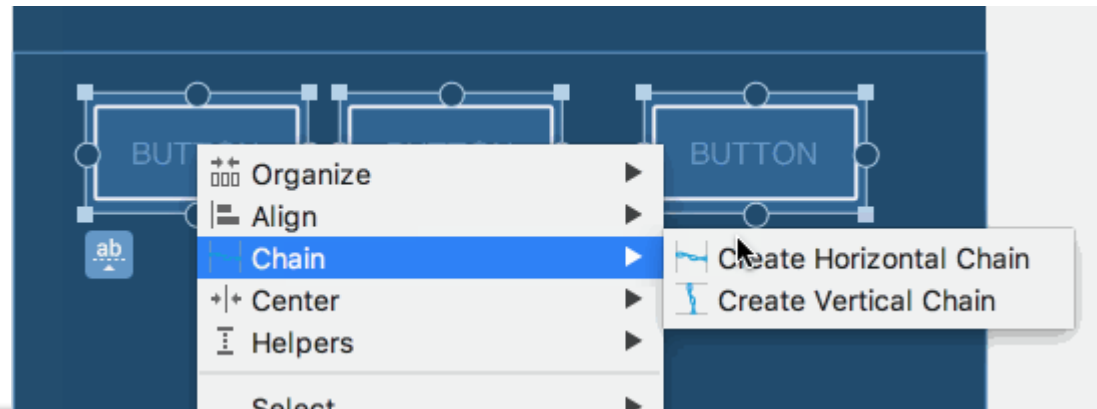


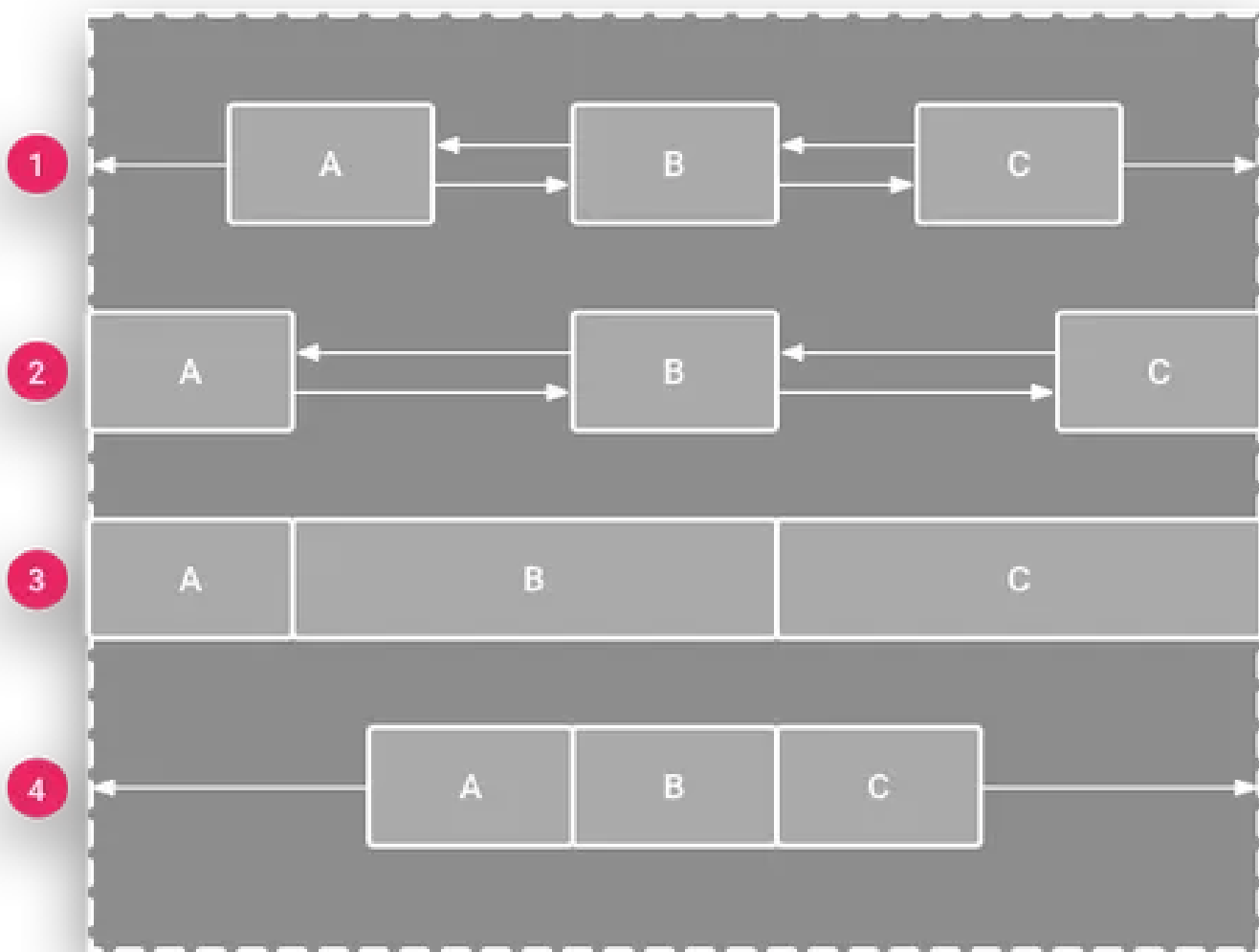
# Chức Năng Chain

- Xâu Chuỗi Các View Lại, Hay Xích Các View Lại



- Để gom các view vào trong một *chain*, chọn hết các view muốn gom bằng cách nhấn giữ phím **Shift** trong lúc click chọn từng view. Rồi click phải lên bất kỳ view nào trong số chúng, và chọn **Chain** > **Create Horizontal Chain** (hay **Create Vertical Chain**).







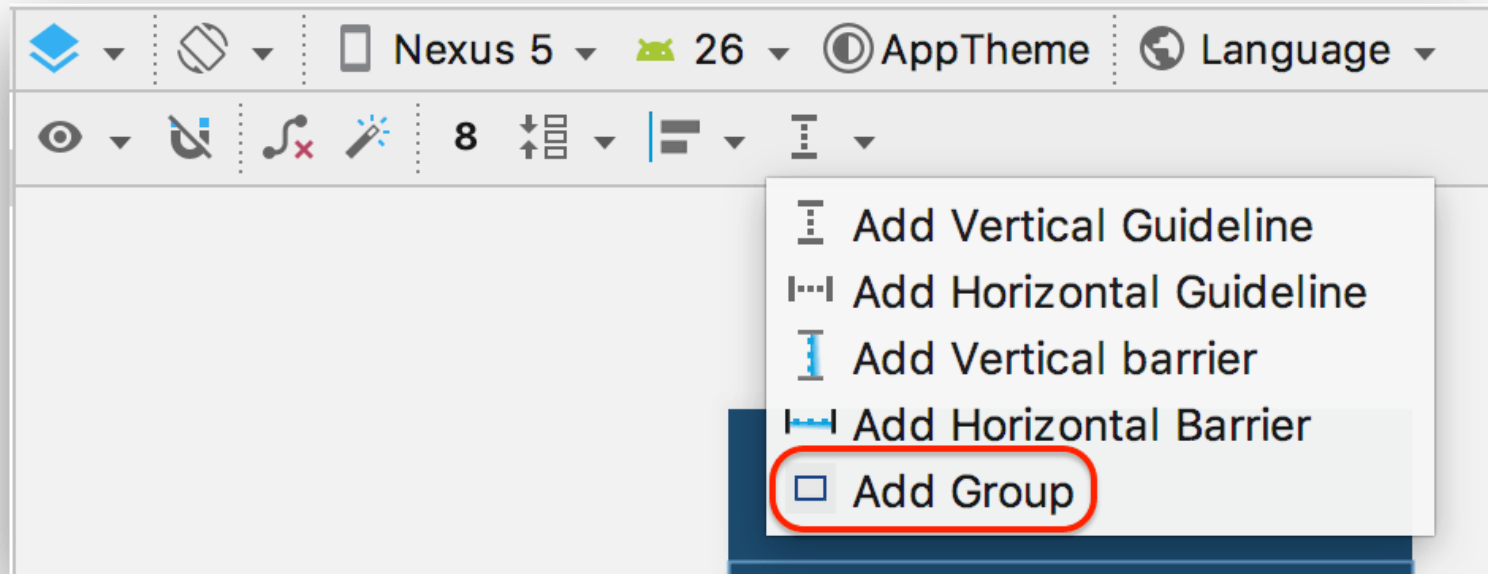
- **1. Spread:** Đây là kiểu dàn đều các view dựa vào không gian của chúng theo phương ngang hoặc dọc. Đây là kiểu sắp xếp mặc định khi bạn tạo mới một *chain*. Bạn có thể xem lại minh họa kiểu *chain* này ở trên kia.

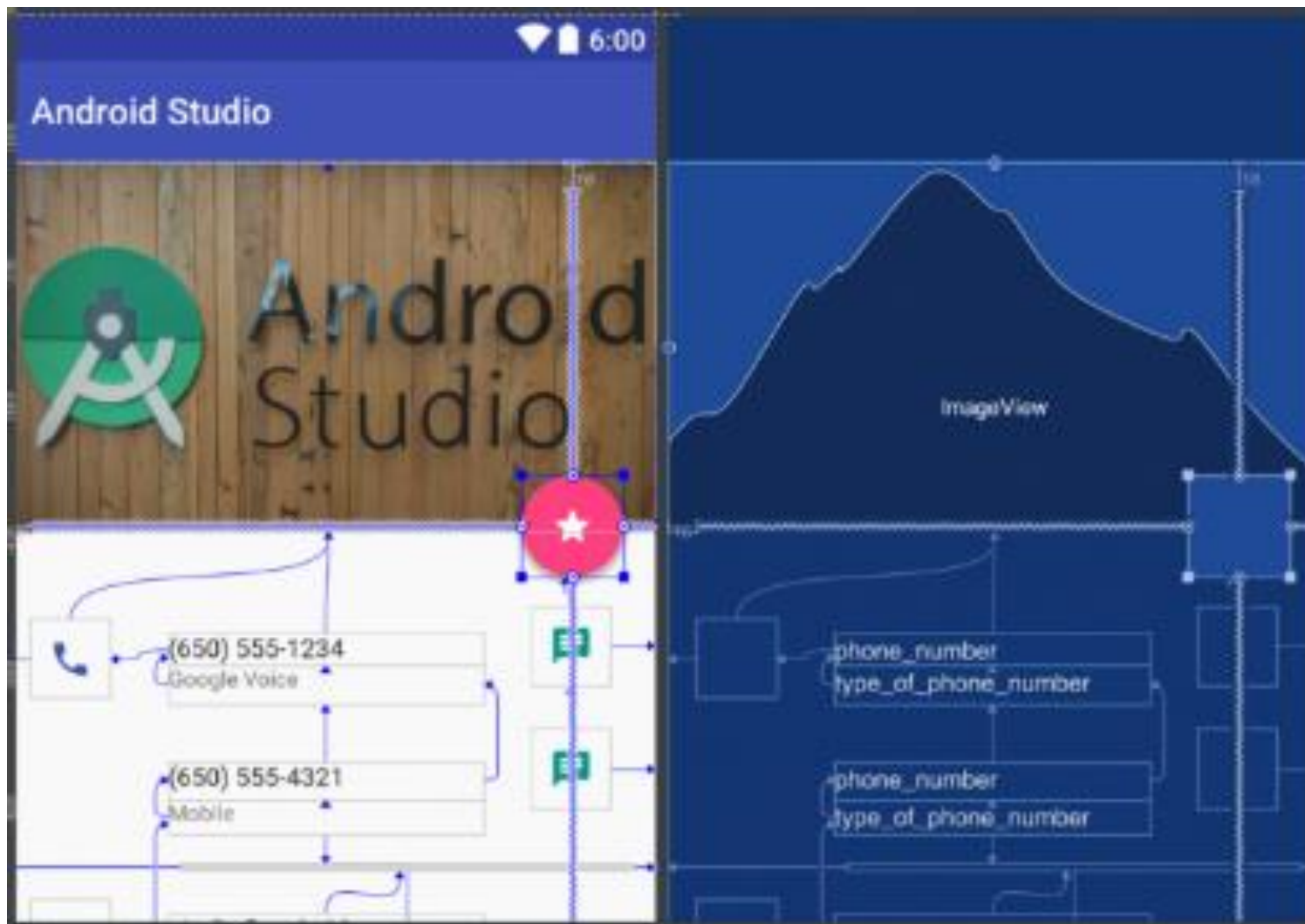
- **2. Spread inside:** Kiểu này cũng sẽ dàn đều các view, nhưng nó sẽ tôn trọng constraint của view đầu và cuối trong một *chain*. Như bạn thấy trên hình, nếu các view **A** và **C** đều set **margin** ở các biên là **0dp** thì chúng sẽ dính chặt vào biên như vậy.

- **3. Weight.** Cách này tương tự như bạn chỉ định trọng số ***layout\_weight*** trong [LinearLayout](#) vậy. Để sử dụng được ***weight*** trong *ConstraintLayout* thì bạn phải chỉ định các view trong chain về [match\\_constraint](#), rồi tìm đến thuộc tính ***horizontal\_weight*** hoặc ***vertical\_weight*** để thiết lập trọng số này cho từng view. Bạn sẽ hiểu rõ hơn ở minh họa phía dưới đây.

- **4. Packed:** Kiểu “đóng gói” các view lại thành một “cục” sát vào nhau. Sau khi đóng gói các view lại xong, bạn có thể sử dụng [bias](#) để thay đổi độ lệch theo chiều ngang hoặc dọc cho các gói này (bạn nhớ để ý xem minh họa cho việc thay đổi *bias* với kiểu *packed* này ở dưới đây).

# Chức Năng Group (Gom Nhóm)







# Các điều khiển cơ bản

# | Các điều khiển cơ bản

Stt	UI Control & Miêu tả
1	<b>TextView</b>  Control này được sử dụng để hiển thị text tới người dùng
2	<b>EditText</b>  EditText là một lớp con được định nghĩa trước của TextView mà bao gồm các khả năng chỉnh sửa đa dạng
3	<b>AutoCompleteTextView</b>  AutoCompleteTextView là một view tương tự như EditText, ngoại trừ rằng nó hiển thị một danh sách các đề nghị tự động trong khi người dùng soạn text



4	<p><b>Button</b></p> <p>Một nút có thể được nhấn, hoặc click bởi người dùng để thực hiện một hành động</p>
5	<p><b>ImageButton</b></p> <p>Là một <code>AbsoluteLayout</code> cho khả năng xác định vị trí chính xác của các view con</p>
6	<p><b>CheckBox</b></p> <p>On/Off có thể được chuyển đổi bởi người dùng. nên sử dụng nó khi biểu diễn cho người dùng với một nhóm các tùy chọn có thể chọn mà không loại trừ lẫn nhau</p>

7	<p><b>ToggleButton</b></p> <p>Hiển thị trạng thái checked/unchecked giống một nút on/off với một light indicator</p>
8	<p><b>ProgressBar</b></p> <p>ProgressBar view cung cấp một phản hồi có thể nhìn thấy về một số tác vụ, như khi chúng ta thực hiện tác vụ ra ngoài trong background</p>
9	<p><b>TimePicker</b></p> <p>TimePicker view cho phép người sử dụng lựa chọn thời gian của một ngày: hoặc chế độ 24 h hoặc chế độ AM/PM</p>
10	<p><b>DatePicker</b></p> <p>DatePicker view cho phép người dùng lựa chọn một date</p>



# Quy Trình thiết kế ứng dụng

# 1. Xây dựng giao diện

```
<TextView
    android:textColor="#ff00"
    android:textSize="30sp"
    android:id="@+id/textViewNoiDung"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignParentTop="true"
    android:layout_centerHorizontal="true"
    android:layout_marginTop="84dp"
    android:text="TextView" />
```

## 2. Ánh Xạ

```
TextView txtNoiDung; // toàn cục

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

    // ánh xạ

    txtNoiDung = (TextView) findViewById(R.id.textViewNoiDung);
}
```

# 3. Viết Code

```
TextView txtNoiDung; // toàn cục

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

    // ánh xạ
    txtNoiDung = (TextView) findViewById(R.id.textViewNoiDung);

    // viết code
    txtNoiDung.setText("Lập trình Android");
}
```



TextView

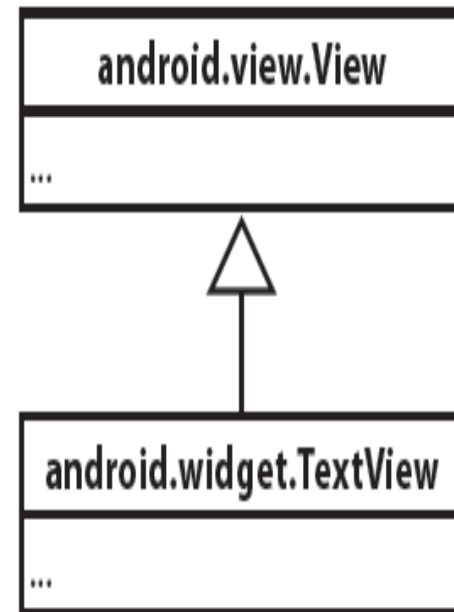
# | TextView

- **TextView** dùng để hiển thị các đoạn văn bản mà không muốn người dùng có thể chỉnh sửa được nội dung, có thể khai báo **TextView** trong file layout **XML** hoặc trong đoạn code Java.
  - Normal: : dạng văn bản kích thước font chữ mặc định.
  - SmallText: dạng văn bản kích thước font chữ nhỏ.
  - MediumText: dạng văn bản kích thước font chữ vừa.
  - LargeText: dạng văn bản kích thước font chữ to.



# | TextView

This is a text view



# | Code TextView trong XML

*<TextView*

*android:id="@+id/txtTdc"*

*android:layout\_width="wrap\_content"*

*android:layout\_height="wrap\_content"/*

*android:text="tdc it" />*

# Code TextView trong JAVA

```
TextView textView = (TextView)findViewById(R.id.txtTdc);  
textView.setText("tdc it");
```

# | Một số phương thức quan trọng khác

- ***setTextColor***: android:textColor: thiết lập màu chữ.
- ***setTextSize***: android:textSize: thiết lập kích cỡ chữ.
- ***setTypeface***: android:typeface: thiết lập các tùy chọn khác về font hay áp dụng một định dạng font ngoài cho TextView.

# | Thuộc tính thường dùng của TextView

- **android:id**: Là thuộc tính duy nhất của **TextView**.
- **android:gravity**: Thuộc tính này thường sử dụng để canh nội dung trên TextView: **left, right, center, top, bottom, center\_vertical, center\_horizontal**
- **android:text**: Thuộc tính này dùng xuất chuỗi văn bản lên **TextView**, Chúng ta có thể khai báo trong XML hoặc code Java
- **android:textColor**: Thuộc tính này dùng xác định màu chữ, dạng màu chữ: “#argb”, “#rgb”, “#rrggbb”, hoặc “#aarrggbb”.

# | Thuộc tính thường dùng của TextView

- **android:textSize:** Thuộc tính textSize xác định kích thước văn bản của **TextView**. Chúng ta có thể đặt kích thước văn bản theo **sp**(scale independent pixel) hoặc **dp**(density pixel).
- **android: textStyle:** Thuộc tính xác định loại văn bản của **TextView**, thông thường có các loại văn bản: **bold**, **italic** và **normal**. Nếu chúng ta muốn sử dụng nhiều hơn một loại văn bản thì phải thêm phép toán hoặc "|" vào giữa các loại văn bản:

# | Thuộc tính thường dùng của TextView

- **android:background:** Thuộc tính này xác định màu nền cho TextView.
- **android:padding:** Thuộc tính này xác định khoảng cách từ đường viền của TextView với nội dung nó chứa: left, right, top or bottom.

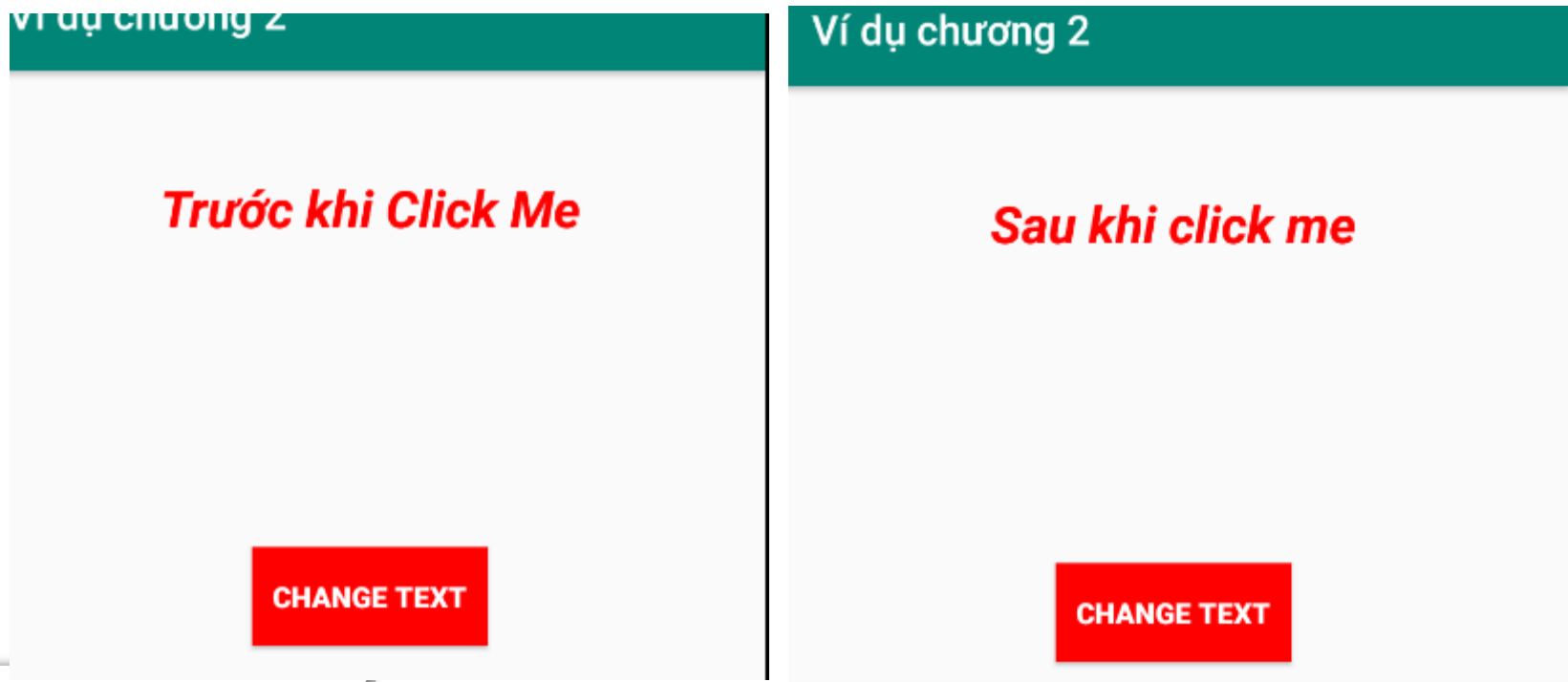
# Ví dụ:

```
<TextView  
    android:id="@+id/txtTDC"  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content"  
    android:layout_centerInParent="true"  
    android:background="#000"  
    android:gravity="center_horizontal"  
    android:padding="10dp"  
    android:text="TDC IT"  
    android:textColor="#fff"  
    android:textSize="40sp"  
    android:textStyle="bold|italic" />
```



# Ví dụ:

- Tạo một **TextView** trong XML, sau đó thay đổi nội dung của nó thông qua một **button** được lập trình xử lý sự kiện trong Java Class

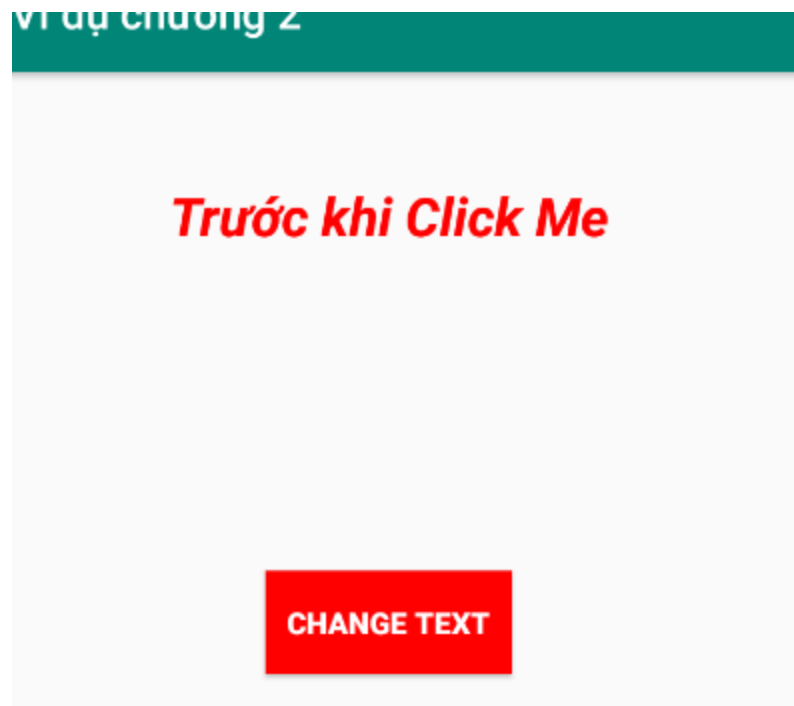


# | Bước 1:

- Tạo một project tên là DemoTextView: **File → New → Android Application Project** điền các thông tin  
**→Next →Finish**

## Bước 2:

- Mở **res→layout→xml** (hoặc) **activity\_main.xml** và thêm code, chúng ta sẽ tạo một **TextView** và **Button**.



```
<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"
    tools:context=".MainActivity">
    <TextView
        android:id="@+id/simpleTextView"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_centerHorizontal="true"
        android:text="Trước khi Click Me"
        android:textColor="#f00"
        android:textSize="25sp"
        android:textStyle="bold|italic"
        android:layout_marginTop="50dp"/>
```

**<Button**

**android:id="@+id/btnChangeText"**

**android:layout\_width="wrap\_content"**

**android:layout\_height="wrap\_content"**

**android:layout\_centerInParent="true"**

**android:background="#f00"**

**android:padding="10dp"**

**android:text="Change Text"**

**android:textColor="#fff"**

**android:textStyle="bold" />**

**</RelativeLayout>**

## Bước 3:

- Mở app → src → MainActivity.java và thêm code Nội dung của **TextView** sẽ thay đổi khi click vào **Button**.

```
public class MainActivity extends AppCompatActivity {  
    TextView txtmsg;  
    Button changeText;  
    @Override  
    protected void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        setContentView(R.layout.activity_main);  
        setControl();  
        setEvent();  
    }
```

```
private void setControl() {  
    txtmsg = findViewById(R.id.simpleTextView);  
    changeText = findViewById(R.id.btnChangeText);  
}  
private void setEvent() {  
    changeText.setOnClickListener(new View.OnClickListener() {  
        @Override  
        public void onClick(View view) {  
            txtmsg.setText("Sau khi click me");  
        }  
    });  
}  
}
```



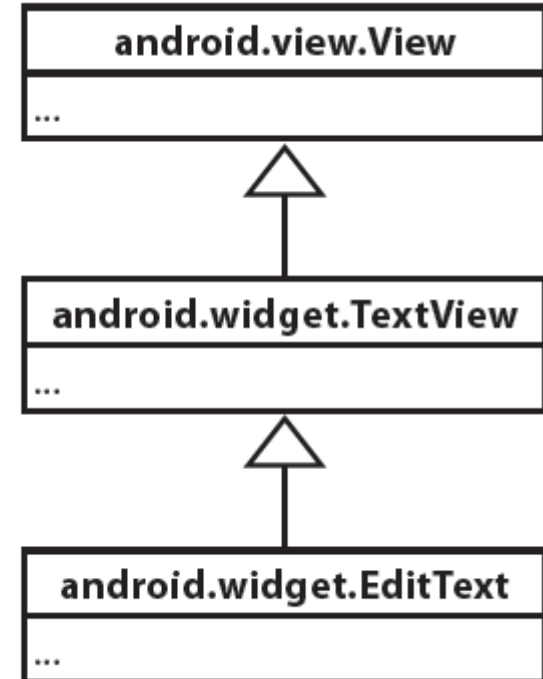
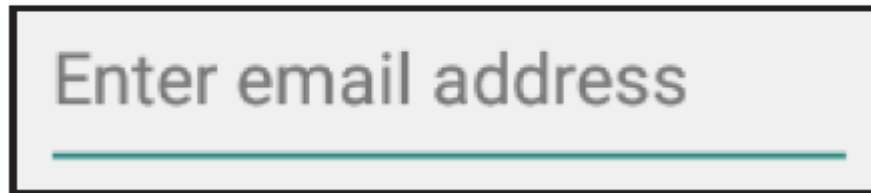
EditText



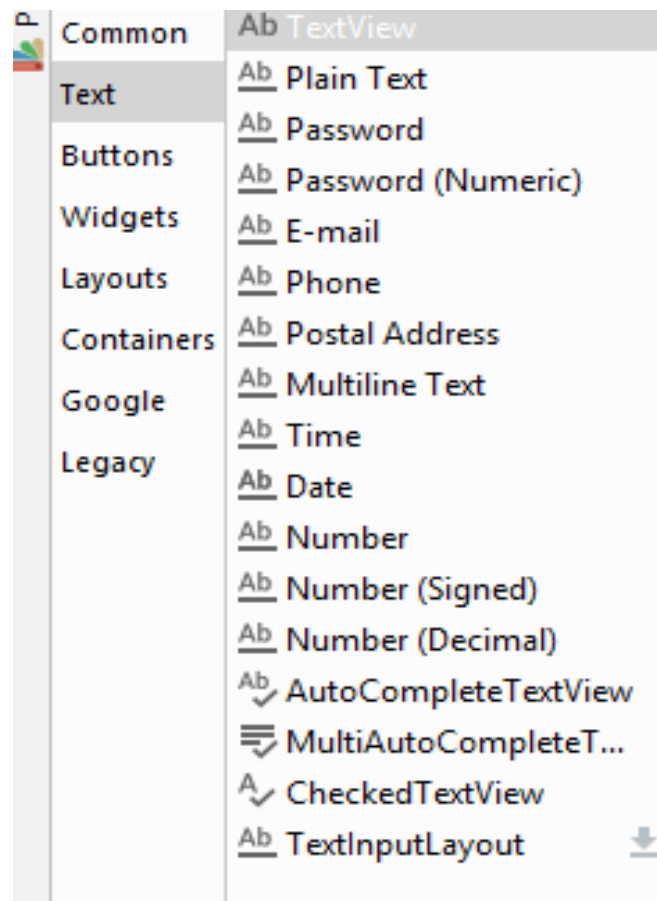
# | EditText

- **EditText** sử dụng cho phép người dùng nhập thông tin để ứng dụng xử lý dưới dạng các text box. EditText là lớp kế thừa từ TextView, được dùng thay đổi nội dung text, chứa tất cả thuộc tính của TextView.

# | EditText



# | EditText



# | Thuộc tính thường dùng của EditText

- **android:id:** Là thuộc tính duy nhất của EditText.
- **android:gravity:** Thuộc tính này thường sử dụng để canh nội dung trên EditText: left, right, center, top, bottom, center\_vertical, center\_horizonta
- **android:text:** Thuộc tính này dùng xuất chuỗi văn bản lên EditText, Chúng ta có thể khai báo trong XML hoặc code Java

# | Thuộc tính thường dùng của EditText

- **android:hint**: Thuộc tính hint để hiển thị thông tin gợi ý trong vùng nhập dữ liệu khi bạn chưa nhập bất kỳ dữ liệu nào vào, chỉ cần có dữ liệu là phần hint sẽ tự động mất đi
- **android:textColor**: Thuộc tính này dùng xác định màu chữ, dạng màu chữ: `#argb`, `"#rgb"`, `"#rrggbb"`, hoặc `#aarrggbb`.

# | Thuộc tính thường dùng của EditText

- **android:textColorHint**: là thuộc tính set màu cho hint
- **android:textSize**: Thuộc tính textSize xác định kích thước văn bản của EditText. Chúng ta có thể đặt kích thước văn bản theo sp(scale independent pixel) hoặc **dp(density pixel)**.

# | Thuộc tính thường dùng của EditText

- **android:textStyle:** Thuộc tính xác định loại văn bản của EditText, thông thường có các loại văn bản: bold, italic và normal. Nếu chúng ta muốn sử dụng nhiều hơn một loại văn bản thì phải thêm phép toán hoặc "|" vào giữa các loại văn bản
- **android:background:** Thuộc tính này xác định màu nền cho EditText.

# | Thuộc tính thường dùng của EditText

- **android:padding**: Thuộc tính này xác định khoảng cách từ đường viền của EditText với nội dung nó chứa: left, right, top or bottom.
- **android:inputType**: Định dạng kiểu văn bản khi người dùng nhập vào(kiểu mật khẩu, kiểu số, kiểu email, phone, ...).Trong ví dụ sau EditText chỉ được nhập số. Chúng ta thêm thuộc tính `android:inputType="number"`
- **android:maxLines**: Cho phép người dùng nhập tối đa bao nhiêu dòng



## *~~EditText~~*

```
android:id="@+id/simpleEditText"  
android:layout_width="fill_parent"  
android:layout_height="wrap_content"  
android:layout_centerInParent="true"  
android:hint="Enter Your Name Here"  
android:textColorHint="#fff"  
android:textStyle="bold|italic"  
android:background="#000"  
android:ems="10"  
android:maxLines="1"  
android:inputType="number"  
android:padding="15dp"
```



# Ví dụ:

Tài khoản \_\_\_\_\_

Mật khẩu \_\_\_\_\_

Ngày sinh \_\_\_\_\_

Email \_\_\_\_\_

ĐĂNG KÝ

Tài khoản **tdc** \_\_\_\_\_

Mật khẩu **...** \_\_\_\_\_

Ngày sinh **15/01/1980** \_\_\_\_\_

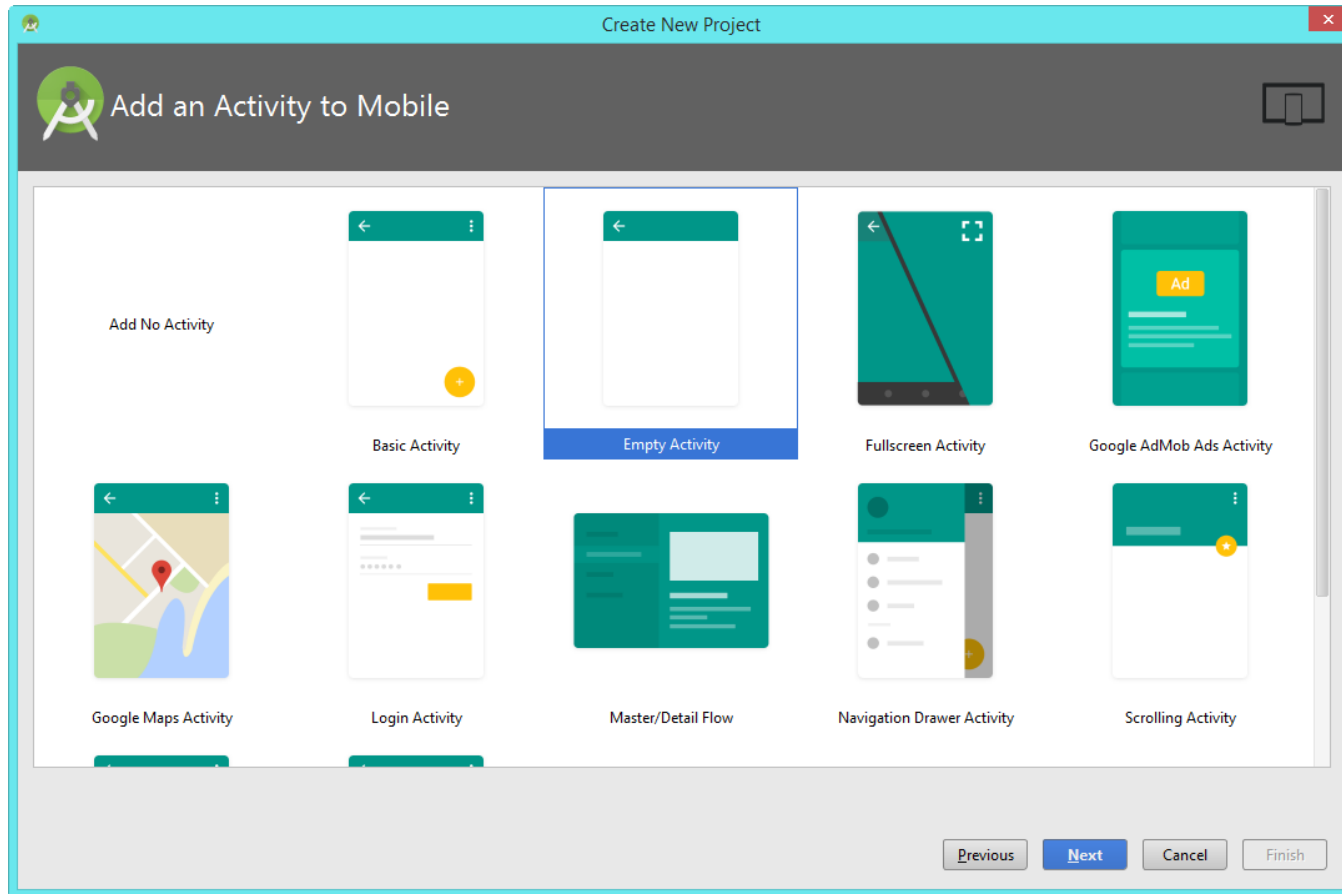
Email **fit@tdc.edu.vn** \_\_\_\_\_

ĐĂNG KÝ

Thông tin tài khoản  
Tài Khoản:tdc  
Mật khẩu:123  
Ngày sinh: 15/01/1980  
Email:fit@tdc.edu.vn

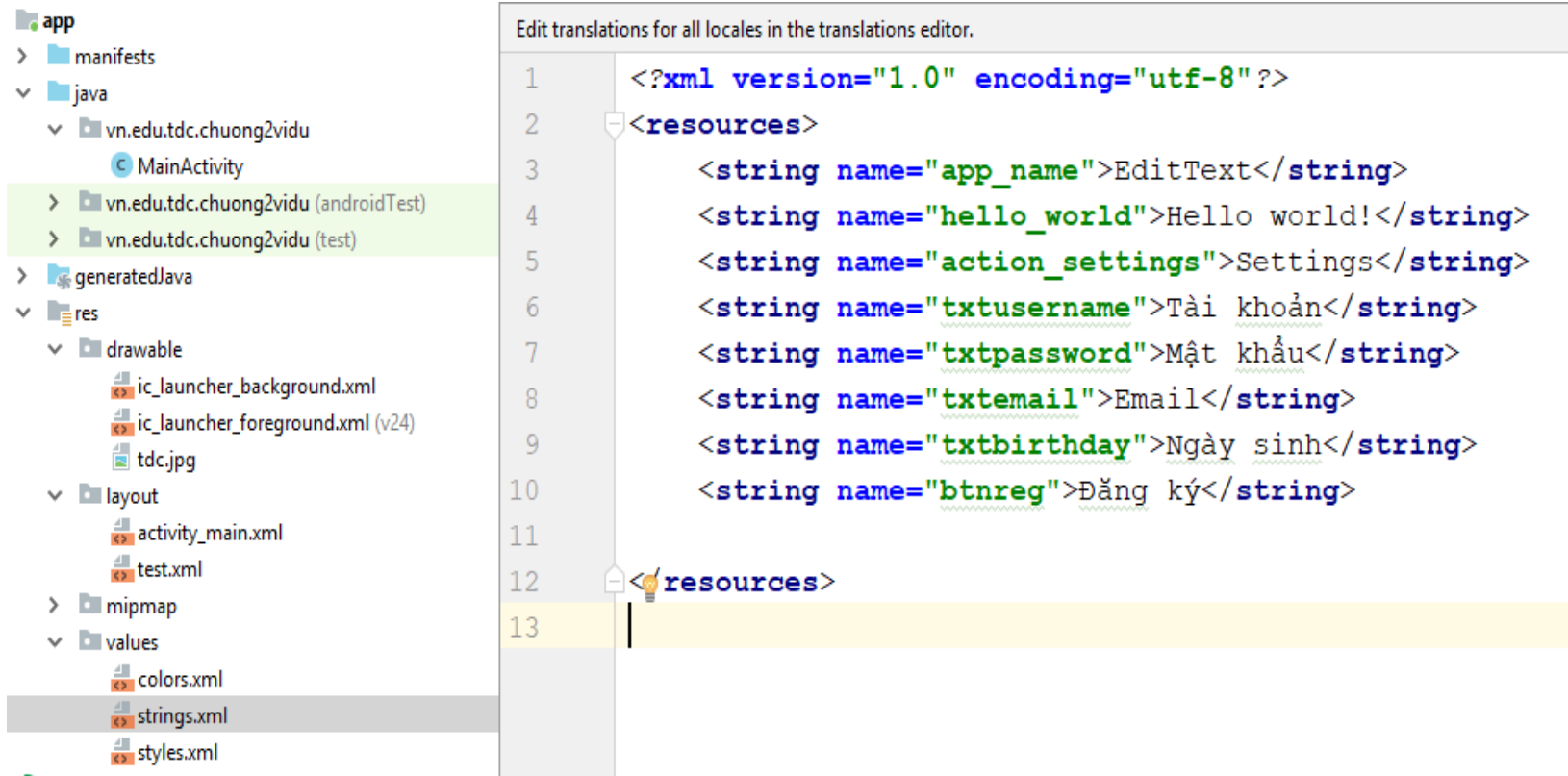
# Bước 1:

- Tiến hành tạo project



## Bước 2:

- Vào thư mục **res/values** bổ sung **string.xml**



## | Bước 3:

- vào thư mục **res /layout**→**activity\_main.xml**.  
Thiết kế giao diện sau

## | Bước 4:

Mở **app** → **src** → **MainActivity.java** và thêm code.  
Khi click vào **Button** sẽ lấy các giá trị của **EditText**,  
sau đó hiển thị lên **TextView**.

# Code

```
public class MainActivity extends AppCompatActivity {  
    //Khai báo các widget  
    private Button btnReg;  
    private TextView txtShowMessage;  
    private EditText edtName, edtPassword, edtBirthday, edtEmail;  
  
@Override  
    protected void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        setContentView(R.layout.activity_main);  
        setControl();  
        setEvent();  
    }
```

# Code

```
private void setControl() {  
    edtName = (EditText) findViewById(R.id.edtName);  
    edtPassword = (EditText) findViewById(R.id.edtPassword);  
    edtBirthday = (EditText) findViewById(R.id.edtBirthday);  
    edtEmail = (EditText) findViewById(R.id.edtEmail);  
    txtShowMessage = (TextView) findViewById(R.id.txtShowMessage);  
    btnReg = (Button) findViewById(R.id.btnReg);  
}
```



```

private void setEvent() {
    btnReg.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View view) {
            String str = "Thông tin tài khoản \n";
            str += "Tài Khoản:" + edtName.getText().toString() + "\n Mật khẩu:"
+ edtPassword.getText().toString();
            str += "\nNgày sinh: " + edtBirthday.getText().toString() + "\nEmail:"
+ edtEmail.getText().toString();
            txtShowMessage.setText(str);
            txtShowMessage.setBackgroundColor(Color.GREEN);
        }
    });
}
}

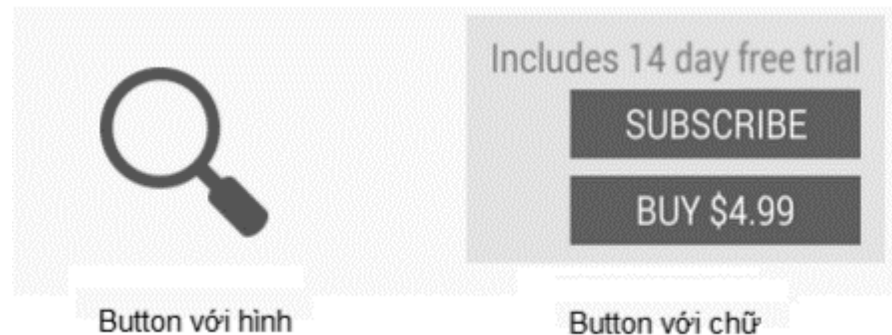
```



Button

# | Button

- **Button** là một loại View, nó hiển thị nút bấm để chờ người dùng bấm vào. Button kế thừa từ TextView nên các thuộc tính, thiết lập cho TextView ở phần trước là có hiệu quả như đối với Button.



# Bắt sự kiện khi người dùng ấn vào Buton

```
view.setOnClickListener(new View.OnClickListener() {  
    @Override  
    public void onClick(View v) {  
        Log.i( tag: "view", msg: "Click");  
    }  
});
```

```
public class MainActivity extends ActionBarActivity implements OnClickListener
{
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        Button btnHello = (Button) findViewById(R.id.btnHello);
        btnHello.setOnClickListener(this);
    }
    @Override
    public void onClick(View v) {
        switch (v.getId()) {
            case R.id.btnHello:
                //do something
                break;
            default:
                break;
        }
    }
}
```

```
<Button
    android:id="@+id/btnHello"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:onClick="doHello"
    android:text="Hello" />
```

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
}
public void doHello(View v){
    switch (v.getId()) {
        case R.id.btnHello:
            //do something
            break;

        default:
            break;
    }
}
```

# | Thuộc tính thường dùng của Button

- **android:id:** Là thuộc tính duy nhất của Button.
- **android:gravity:** Thuộc tính này thường sử dụng để canh nội dung trên EditText: left, right, center, top, bottom, center\_vertical, center\_horizontal.
- **android:text:** Thuộc tính này dùng xuất chuỗi văn bản lên Button, Chúng ta có thể khai báo trong XML hoặc code Java.

# | Thuộc tính thường dùng của Button

- **android:textColor:** Thuộc tính này dùng xác định màu chữ, dạng màu chữ: “#argb”, “#rgb”, “#rrggbb”, hoặc “#aarrggbb”.
- **android:textSize:** Thuộc tính textSize xác định kích thước văn bản của Button. Chúng ta có thể đặt kích thước văn bản theo: sp(scale independent pixel) hoặc dp(density pixel).



# | Thuộc tính thường dùng của Button

- **android:background:** Thuộc tính này xác định màu nền cho Button.
- **android:padding:** Thuộc tính này xác định khoảng cách từ đường viền của Button với nội dung nó chứa: left, right, top or bottom.

# | Thuộc tính thường dùng của Button

- **android:drawableBottom**: drawableBottom hiển thị icon sau chuỗi văn bản: android:drawableTop, android:drawableRight và android:drawableLeft: Hiển thị Icon bên trái, bên phải hoặc phía trên của chuỗi văn bản.



# Ví dụ:

*<Button*

```
android:id="@+id/btnSimple"  
android:layout_width="match_parent"  
android:layout_height="wrap_content"  
android:layout_alignParentLeft="true"  
android:layout_alignParentTop="true"  
android:layout_marginTop="28dp"  
android:textSize="40sp"  
android:text="@string/btnreg"  
android:padding="15dp"  
android:textStyle="bold|italic"  
android:background="#FFCC"  
android:drawableBottom="@drawable/tdc"/>
```



# Ví dụ: button có hình

**Tổng hai số**

Nhập số a:

Nhập số b:

**Tổng hai số**

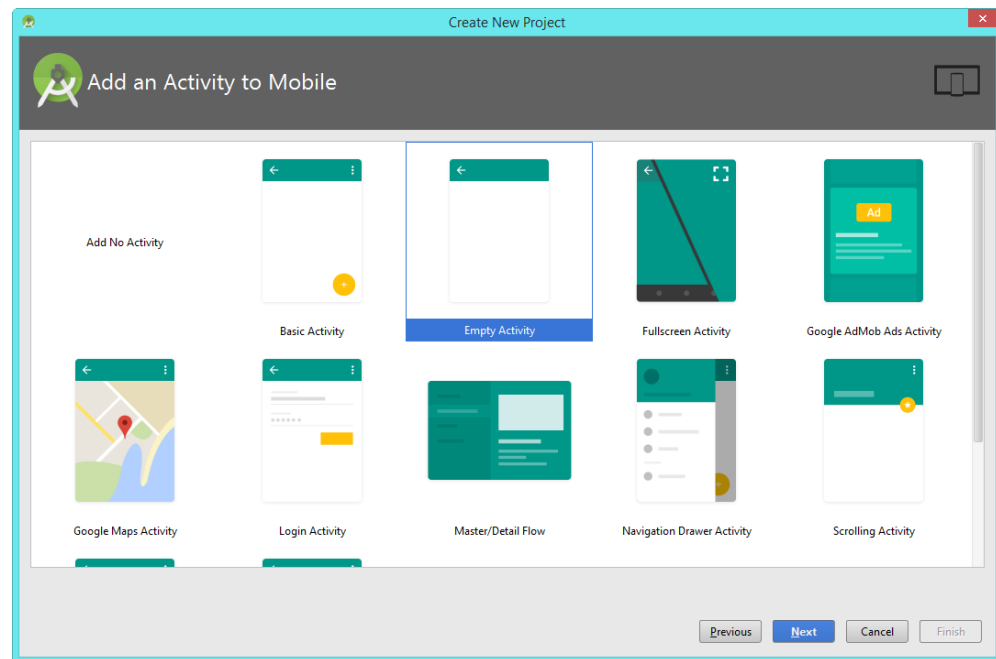
Nhập số a:

Nhập số b:

Tổng 2 số là: 7.0

# Bước 1:

- Tạo một project tên là DemoTextView: File → New → Android Application Project điền các thông tin → Next → Finish



## Bước 2:

- Vào thư mục **res/values** bổ sung **string.xml**

```
<resources>
```

```
<string name="app_name">Button</string>
```

```
<string name="txtlabel">Tổng hai số</string>
```

```
<string name="txtnuma">Nhập số a:</string>
```

```
<string name="txtnumb">Nhập số b:</string>
```

```
<string name="btnsum">Tổng</string>
```

```
<string name="btnclear">Xóa trắng</string>
```

```
</resources>
```

## | Bước 3:

- Mở res → layout → xml (hoặc) activity\_main.xml và thêm code, chúng ta sẽ tạo các TextView, EditText và Button

## Bước 4:

- Mở **app** → **src** → **MainActivity.java** và thêm code.  
Khi click vào **Button** gọi phương thức **sum(a,b)**, trong phương thức này chúng ta truyền 2 tham số lấy từ 2 **EditText**, sau đó hiển thị lên **TextView**.



```
public class MainActivity extends AppCompatActivity {  
//Khai báo các widget  
Button btnSum, btnClear;  
EditText edtNumA, edtNumB;  
TextView txtResult;  
  
@Override  
protected void onCreate(Bundle savedInstanceState) {  
    super.onCreate(savedInstanceState);  
    setContentView(R.layout.activity_main);  
    setControl();  
    setEvent();  
}
```

```
private void setControl() {  
    edtNumA = (EditText)findViewById(R.id.edtNumA);  
    edtNumB = (EditText)findViewById(R.id.edtNumB);  
    txtResult= (TextView)findViewById(R.id.txtResult);  
    btnSum = (Button)findViewById(R.id.btnSum);  
    btnClear= (Button)findViewById(R.id.btnClear);  
}
```

```
private void setEvent() {
```

```
    btnSum.setOnClickListener(new View.OnClickListener() {
```

```
        @Override
```

```
        public void onClick(View v) {
```

```
            // TODO Auto-generated method stub
```

```
            float a, b, c;
```

```
            a = Float.parseFloat(edtNumA.getText().toString());
```

```
            b = Float.parseFloat(edtNumB.getText().toString());
```

```
            c = a + b;
```

```
            txtResult.setText("Tổng 2 số là: "+c);
```

```
            txtResult.setBackgroundColor(Color.GREEN);
```

```
        }
```

```
    });
```

```
btnClear.setOnClickListener(new View.OnClickListener() {
```

```
    @Override
```

```
    public void onClick(View v) {
```

```
        // TODO Auto-generated method stub
```

```
        edtNumA.setText("");
```

```
        edtNumB.setText("");
```

```
    }
```

```
});
```

```
}
```

```
}
```



# Bài tập

# 1. Sử dụng Linear Layout thiết kế giao diện như sau:

Linear Layout - Horizontal

Tên:

Linear Layout - Vertical

Tên:

Tên đăng nhập:

Mật khẩu:

# Xây dựng màn hình đăng nhập

The image shows a wireframe of a login form with the following elements and annotations:

- 1**: Points to the title bar "Đăng nhập vào hệ thống".
- 2**: Points to the logo "CSCOM" on a purple background.
- 3**: Points to the username input field with the placeholder text "TenDangNhap".
- 4**: Points to the password input field with the placeholder text "\*\*\*\*\*".
- 5**: Points to the "Đăng nhập" (Login) button.
- 6**: Points to the checkbox labeled "Lưu thông tin đăng nhập" (Remember login information).



**CẢM ƠN TẤT CẢ  
ĐÃ LẮNG NGHE**