LẬP TRÌNH DI ĐỘNG

View & Layout

Nội dung

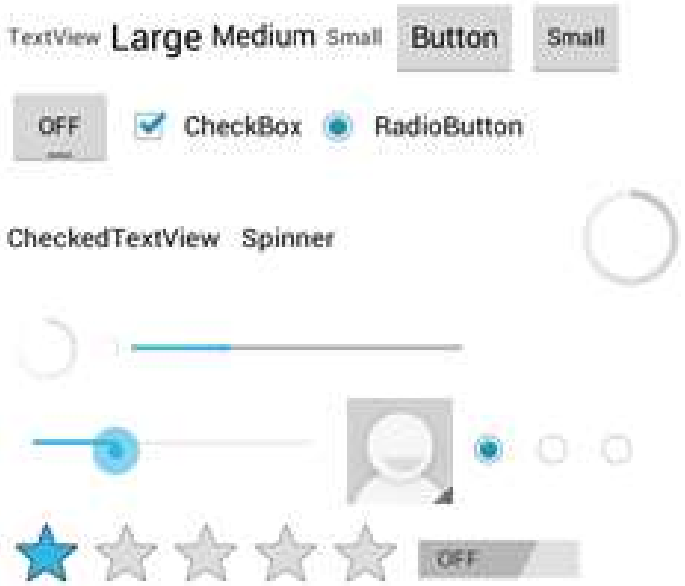
- View
- Layouts
- Các widgets cơ bản
- Các kiểu xử lý sự kiện
- Notification & Dialogs
- Menus

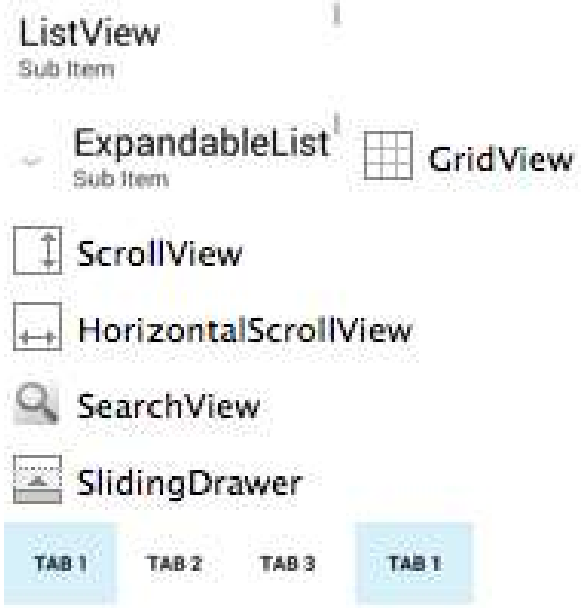
View

- View là đối tượng cơ bản để xây dựng các thành phần giao diện
- Hầu hết các thành phần giao diện đều kế thừa từ lớp View: TextView, Button, Spinner, ToggleButton, RadioButton,…
- Nằm trong gói android.widget (nên thường gọi là widget)
- Custom view: ta có thể tự tạo widget của riêng mình bằng cách tùy biến view để hoạt động theo cách riêng

View

- View bao gồm hai dạng:
  - View: các điều khiển đơn lẻ
  - ViewGroup: tập hợp nhiều điều khiển đơn lẻ





ViewGroup

- ViewGroup là các view đặc biệt, có thể chứa bên trong nó các view khác
  - VD1: thông tin về ngày tháng gồm một số text
  - VD2: danh sách các ngày trong tháng gồm các button
- ViewGroup là cửa sổ cha của các view con
- Một view nằm trong ViewGroup cần phải có thông tin về vị trí của nó trong cửa sổ cha
- ViewGroup = các view con + cách bố trí các view con đó bên trong
- ViewGroup lồng nhau quá sâu làm chậm ứng dụng

View

- Thể hiện
  - Các View thể hiện trên giao diện như một hình chữ nhật tuỳ thuộc vị trí, kích thước và nhận vào cũng như xử lý các tương tác có liên quan.
  - Một số thể hiện của lớp View: TextView, ImageView, SurfaceView…
  - ViewGroup cũng là một thể hiện của View.
- Sử dụng
  - Kéo thả vào layout và tuỳ chỉnh thuộc tính.
  - Thiết lập thông số và truy xuất trong Java Code.

View

- **Thao tác**: có các thao tác chính sau:
  - Thiết lập:
    - Dùng phương thức: set<thuộc tính>(tham số)
    - Vd: textView1.setText(“Hello”);
  - Truy xuất
    - Dùng phương thức: get<thuộc tính>(tham số)
    - Vd: String st=textView1.getText();
  - Xây dựng phương thức “lắng nghe” sự kiện

View

- **Thuộc tính**
  - **Id:**
    - Khai báo kiểu số nguyên int, đánh dấu vùng nhớ của đối tượng View.
    - Phương thức thiết lập: setId
    - Phương thức truy xuất: getId
  - Thuộc tính Id đi kèm với đối tượng View khi khai báo trong XML cho phép truy xuất trong Java Code.

View

- Ví dụ:
  - Khai báo id trong XML

```
<Button
  android:id="@+id/my_button"
  android:layout_width="wrap_content"
  android:layout_height="wrap_content"
  android:text="my_button_text"/>
```

  - Truy xuất trong JavaCode

```
Button myBtn =
  (Button)findViewById(R.id.my_button);
```

View

- Kích thước của View:
  - Có thể thiết lập qua các thông số:
    - WRAP_CONTENT
    - MATCH_PARENT (API 8 trở lên)
    - FILL_PARENT
    - Một con số bất kỳ (tính theo dp/px/dip).
  - Thuộc tính thiết lập trong XML:
    - layout_width
    - layout_height
  - Phương thức truy xuất:
    - getWidth
    - getHeight
    - getMeasuredWidth
    - getMeasureHeight

View

- Canh lề nội dung trong JavaCode:
  - Phương thức thiết lập:
    - setPadding
  - Phương thức truy xuất:
    - getPaddingTop
    - getPaddingLeft
    - getPaddingRight
    - getPaddingBottom

Layout

- Layout là ViewGroup đặc biệt
  - Gồm các view con bên trong nó
  - Quy cách bố cục các view con nhất quán
  - Mỗi loại layout có quy tắc bố cục riêng
- Có thể tạo layout theo 2 cách:
  - XML: soạn thông tin ở dạng XML, nạp layout từ XML bằng cách đọc từng dòng XML và tạo các thành phần phù hợp
  - Code: tạo biến layout, tạo từng biến view, đặt view vào trong layout

Layout by XML

- Là phương pháp tạo giao diện phổ biến nhất
  - XML có cấu trúc dễ hiểu, phân cấp, giống HTML
  - Tên của thành phần XML tương ứng với lớp java trong code
- Dễ chỉnh sửa trên bằng design hoặc sửa file XML
- Tách rời giữa thiết kế và viết mã
- Thực hiện: thiết kế file layout XML sau đó dùng bộ nạp LayoutInflater để tạo biến kiểu Layout

```
LayoutInflater.from(context)
.inflate(R.layout.filename, null);
```

Layout by XML

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://.../res/android"
              android:layout_width="fill_parent"
              android:layout_height="fill_parent"
              android:orientation="vertical" >
    <Button android:id="@+id/button"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Hello, I am a Button" />
    <TextView android:id="@+id/text"
              android:layout_width="wrap_content"
              android:layout_height="wrap_content"
              android:text="Hello, I am a TextView" />
</LinearLayout>
```

Layout by Code

```
Button myButton = new Button(this);
myButton.setText("Press me");
myButton.setBackgroundColor(Color.YELLOW);

RelativeLayout myLayout = new RelativeLayout(this);
RelativeLayout.LayoutParams buttonParams =
    new RelativeLayout.LayoutParams(
        RelativeLayout.LayoutParams.WRAP_CONTENT,
        RelativeLayout.LayoutParams.WRAP_CONTENT);
buttonParams.addRule(RelativeLayout.CENTER_HORIZONTAL);
buttonParams.addRule(RelativeLayout.CENTER_VERTICAL);
myLayout.addView(myButton, buttonParams);
setContentView(myLayout);
```

Layout Parameter

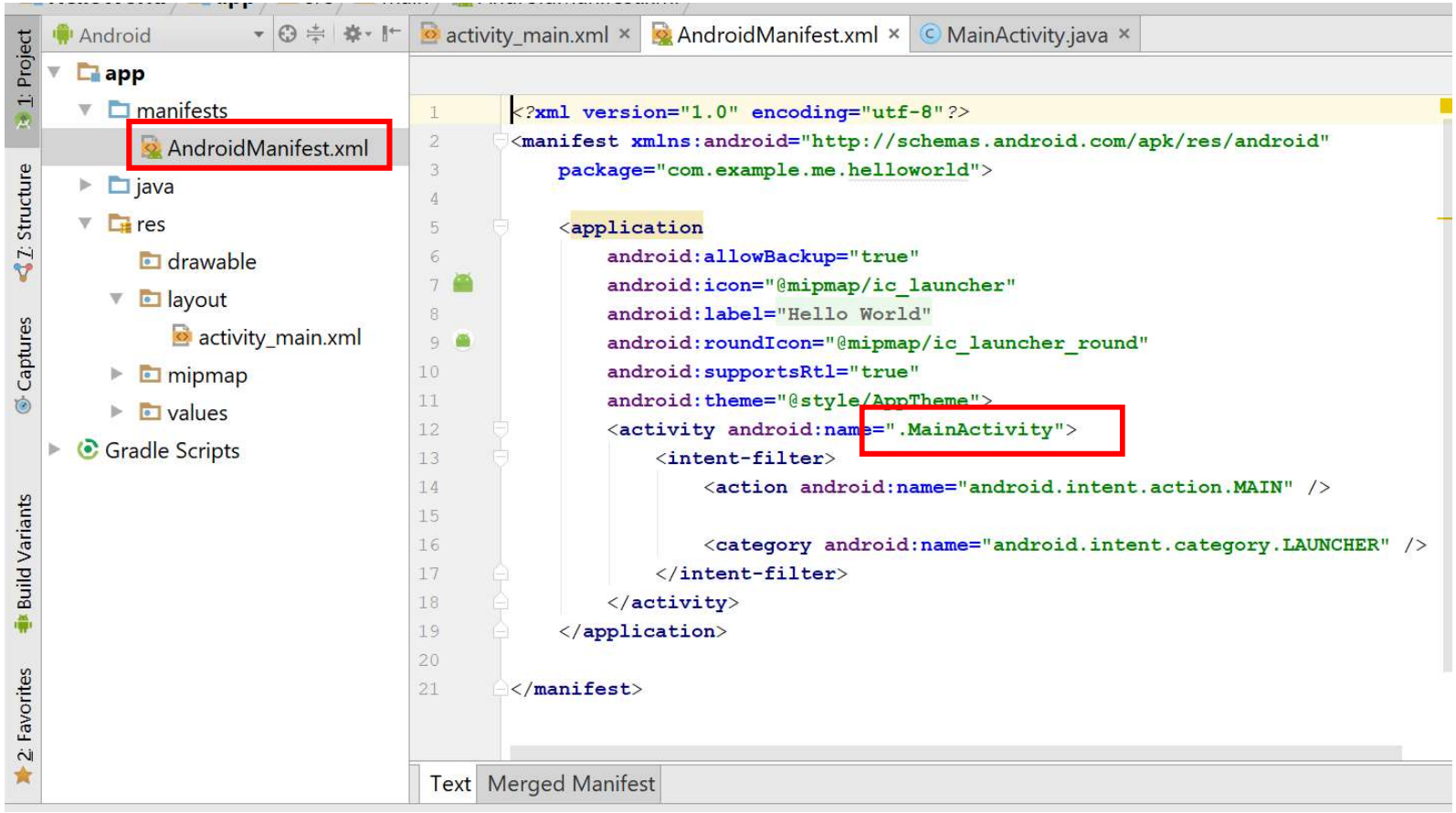
- Quy định cách đặt để của view trong layout
- Mỗi view cần đính kèm LayoutParams khi đặt vào trong Layout

Layout Parameter

- Bản thân layout và view cũng có các tham số của nó khi được đặt vào view cha
  - Vị trí (position): cặp tọa độ Left/Top
  - Kích thước (size): cặp giá trị Width/Height
  - Lề (margin): tham số trong LayoutParams (kiểu MarginLayoutParams), quy định khoảng cách của view với các thành phần xung quanh
  - Đệm (padding): vùng trống từ nội dung của view ra các viền, sử dụng phương thức setPadding(int,int,int,int) để điều chỉnh, đơn vị đo thường là dp

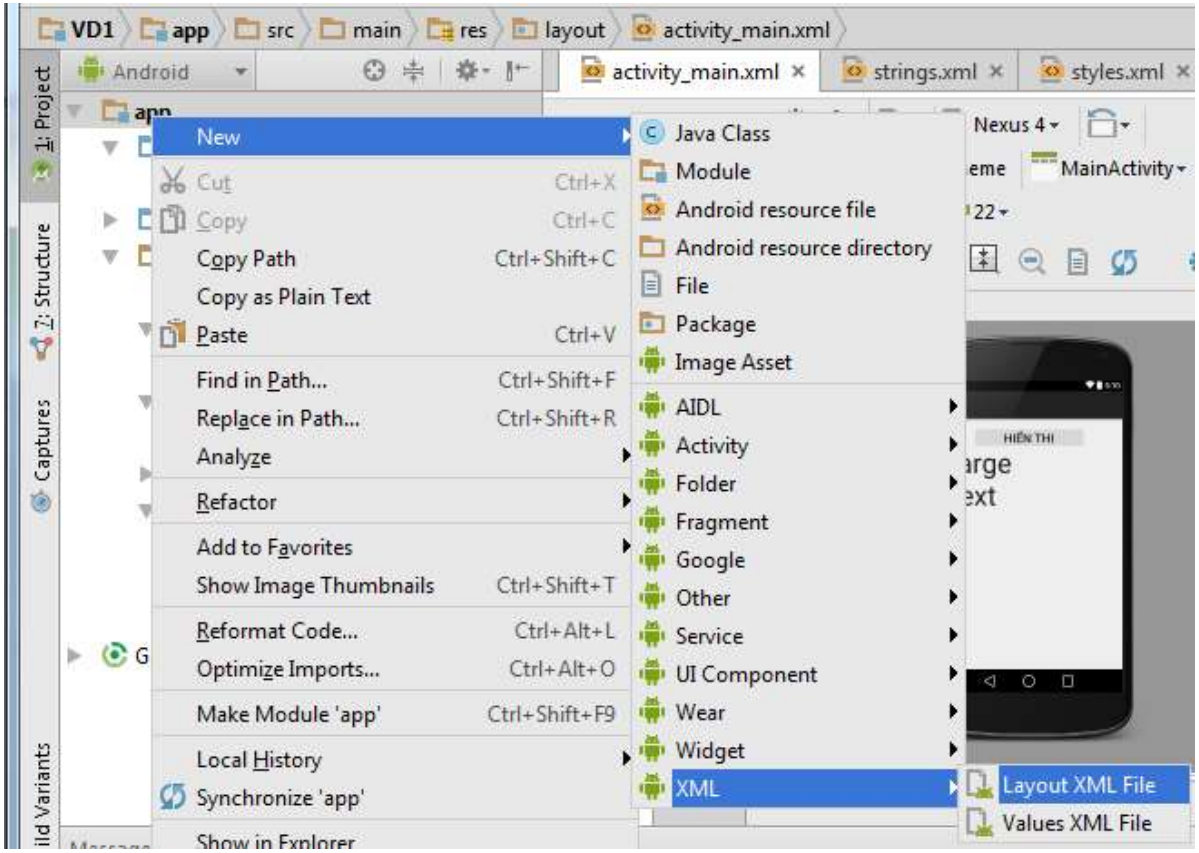
Tạo Layout và kết nối Layout vào Activity

- Khi tạo một Project thì mặc nhiên sẽ có 1 Activity được chỉ định chạy đầu tiên khi thực thi ứng dụng



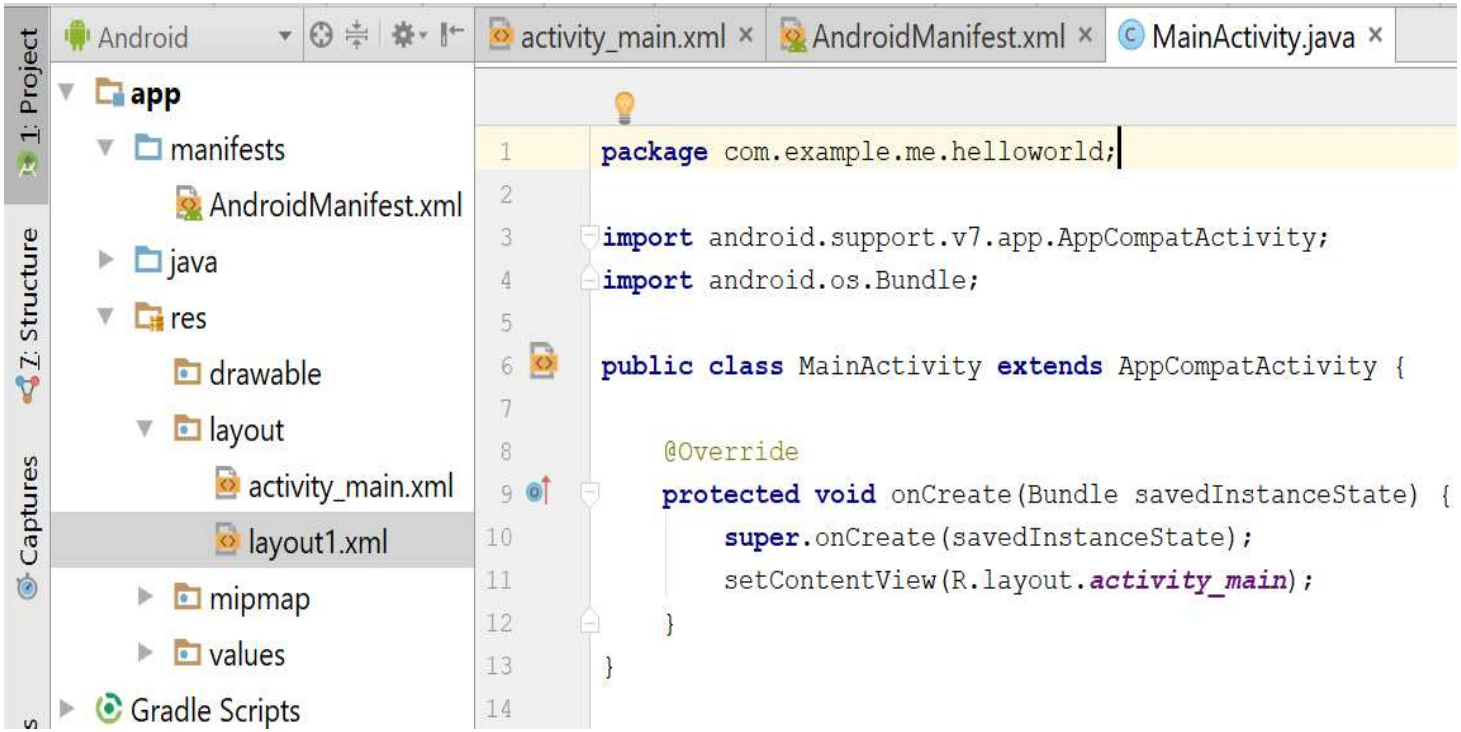
Tạo Layout và kết nối Layout vào Activity

- Đổi Layout mặc định bằng một Layout khác
  - Bước 1: Tạo một Layout: click chuột phải vào **app**/ chọn **New**/ chọn **XML**/ chọn **Layout XML File**



Tạo Layout và kết nối Layout vào Activity

- Bước 2:
  - Mở tập tin **MainActivity.java**
  - Dòng lệnh: `setContentView(R.layout.activity_main);` dùng để kết nối Layout vào Activity.



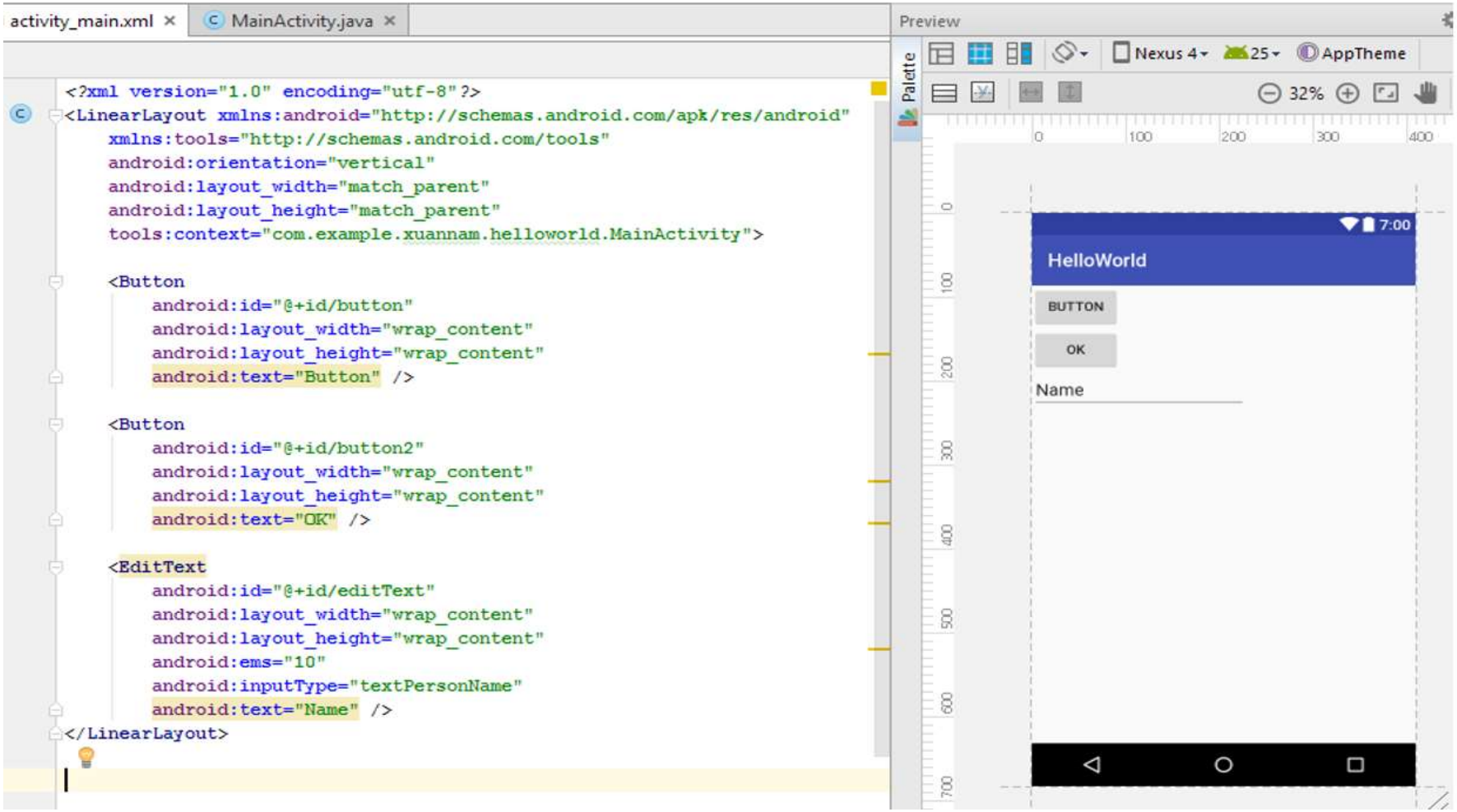
Một số layout thông dụng

- LinearLayout
- RelativeLayout
- ConstraitLayout
- FrameLayout
- TableLayout
- AbsoluteLayout

LinearLayout

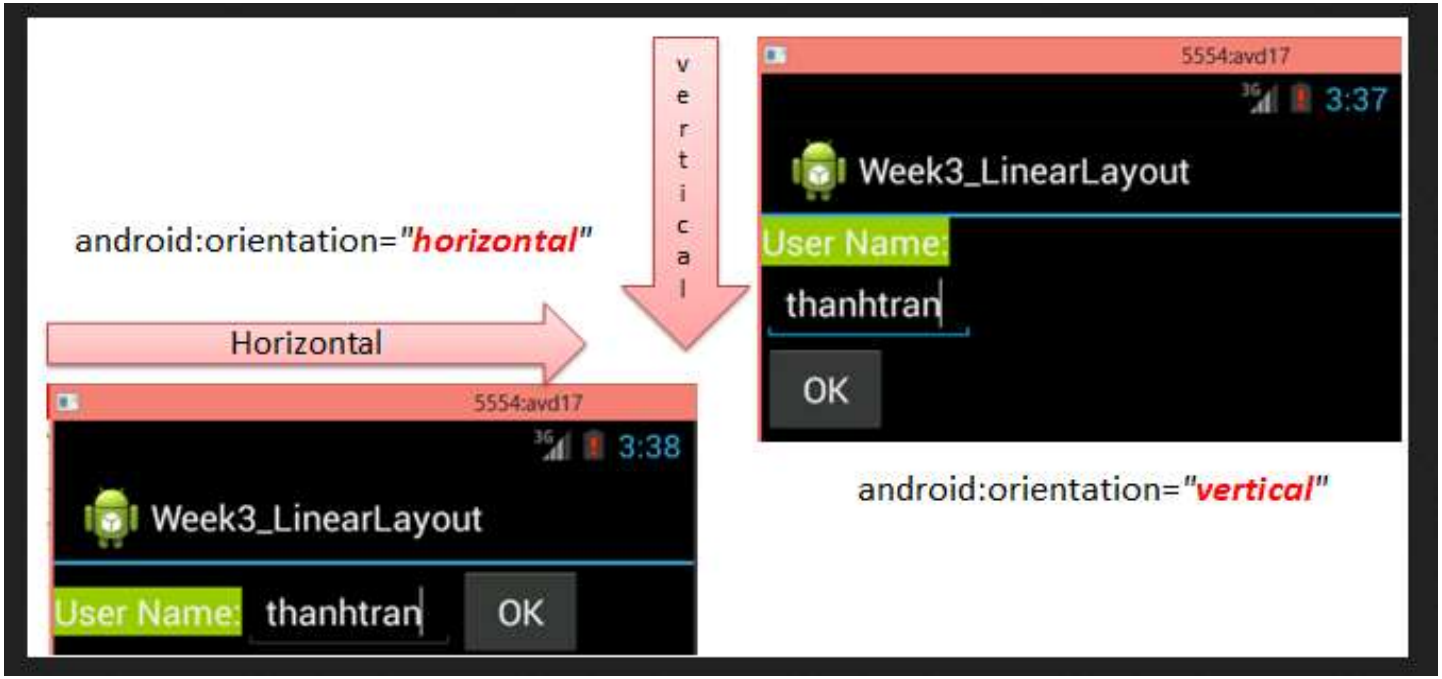
- Các view bên trong được xếp liên tiếp thành một hàng hoặc một cột
- Thuộc tính android:orientation quy định cách bố cục theo hàng hay cột:
  - "vertical": các view bên trong sắp xếp theo hàng
  - "horizontal": các view bên trong sắp xếp theo cột
- LinearLayout không thay đổi kích thước các view con, chỉ điều chỉnh vị trí của chúng

LinearLayout



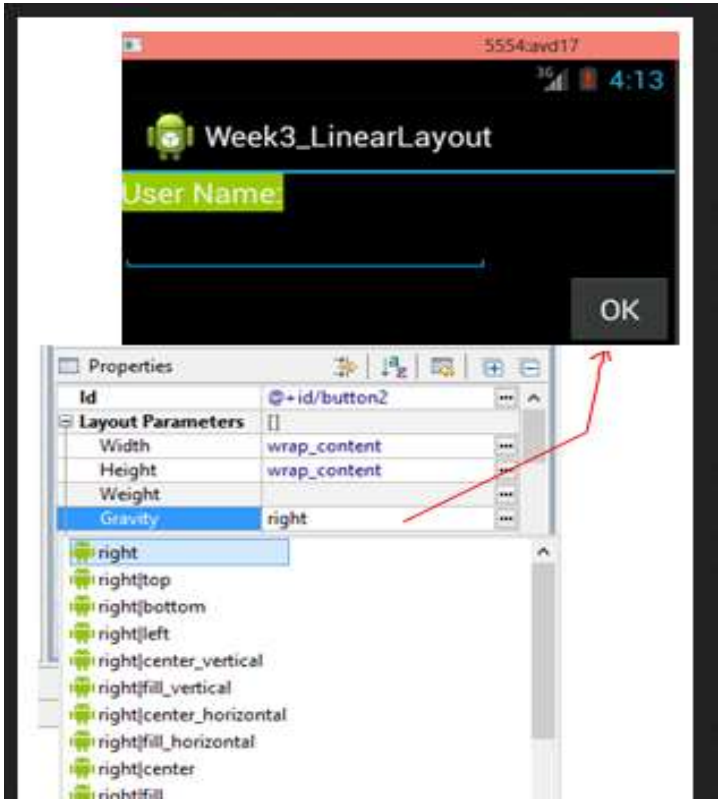
LinearLayout

- Hướng của layout được chỉ định bởi thuộc tính: **android:orientation**, nhận giá trị vertical hoặc horizontal.



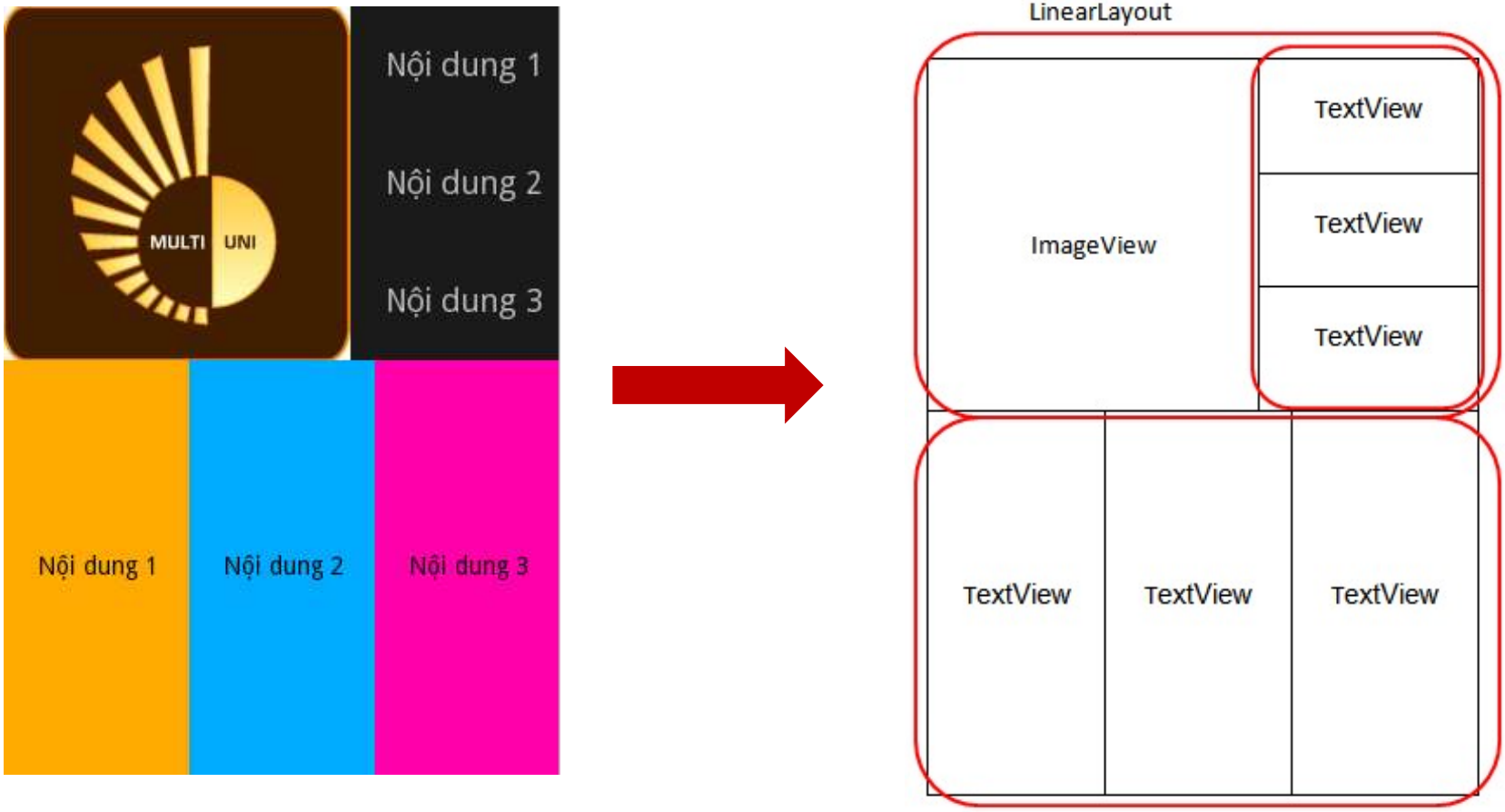
LinearLayout

- Có thể dùng **margin, gravity, weight** để hỗ trợ cho việc thiết kế.
- Dùng Properties để thiết lập các thuộc tính cho control
- Vd: để căn lề các control trên giao diện ta dùng **layout_gravity**



LinearLayout

- Giả sử cần thiết kế một màn hình như sau:



RelativeLayout

- Là loại layout phổ biến nhất trong thiết kế giao diện
- Các view con trong layout xác định vị trí và kích thước dựa trên quan hệ với view cha hoặc các view con khác
- Dùng trong trường hợp đặt trọng tâm vào mối quan hệ giữa các thành phần
- Ý tưởng của RelativeLayout được phát triển và nâng cấp thành ConstraintLayout, hiện là loại layout mặc định khi thiết kế giao diện

RelativeLayout

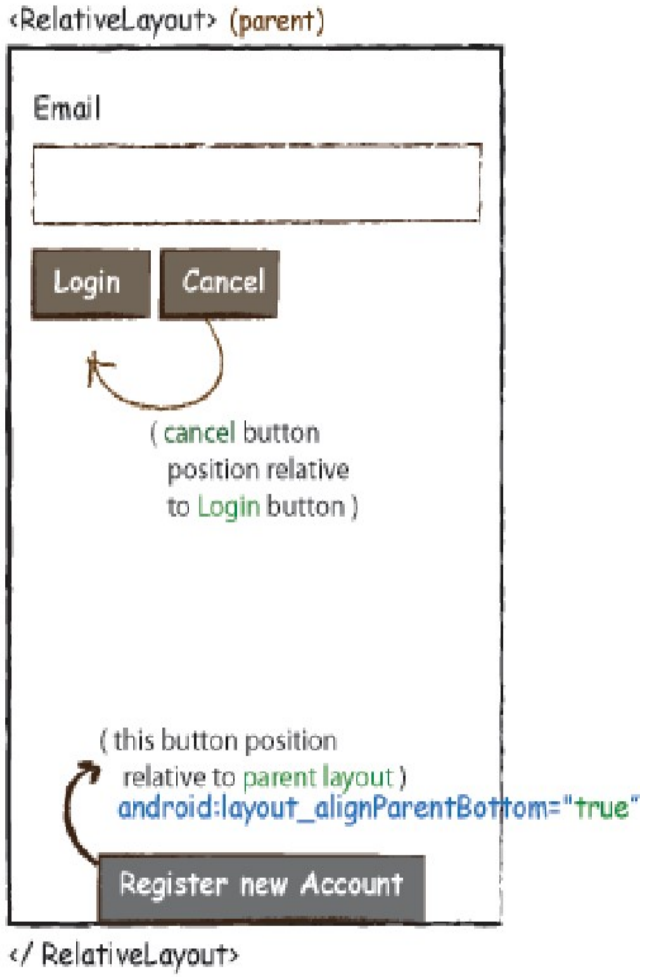
- Một số thuộc tính cho phép định vị View:
  - android:layout_alignTop: cạnh trên của view này kết nối với cạnh trên của view được tham chiếu đến
  - android:layout_alignBottom: cạnh dưới của view này kết nối với cạnh dưới của view được tham chiếu đến
  - android:layout_alignLeft: cạnh trái của view này kết nối với cạnh trái của view được tham chiếu đến
  - android:layout_alignRight: cạnh phải của view này kết nối với cạnh phải của view được tham chiếu đến
  - android:layout_above: cạnh dưới của view này ở trên view được chỉ định trong thuộc tính.
  - android:layout_below: view này nằm bên dưới view được chỉ định trong thuộc tính.

RelativeLayout

- Một số thuộc tính cho phép định vị View:
  - android:layout_alignParentTop: Nếu bằng “true”, view này canh lề trên với view cha của nó, nghĩa là cạnh trên của view này trùng với cạnh trên của view cha.
  - android:layout_alignParentBottom: Nếu true, làm cạnh dưới của view này kết nối với cạnh dưới của view cha. Phải là một giá trị Boolean, hoặc "true" hoặc "false"
  - android:layout_centerVertical: Nếu bằng “true”, view này được canh ở giữa theo chiều dọc so với view cha của nó.
  - android:layout_toRightOf: định vị view này nằm bên phải so với view được chỉ định trong thuộc tính.
  - android:layout_toLeftOf: định vị view này nằm bên trái so với view được chỉ định trong thuộc tính

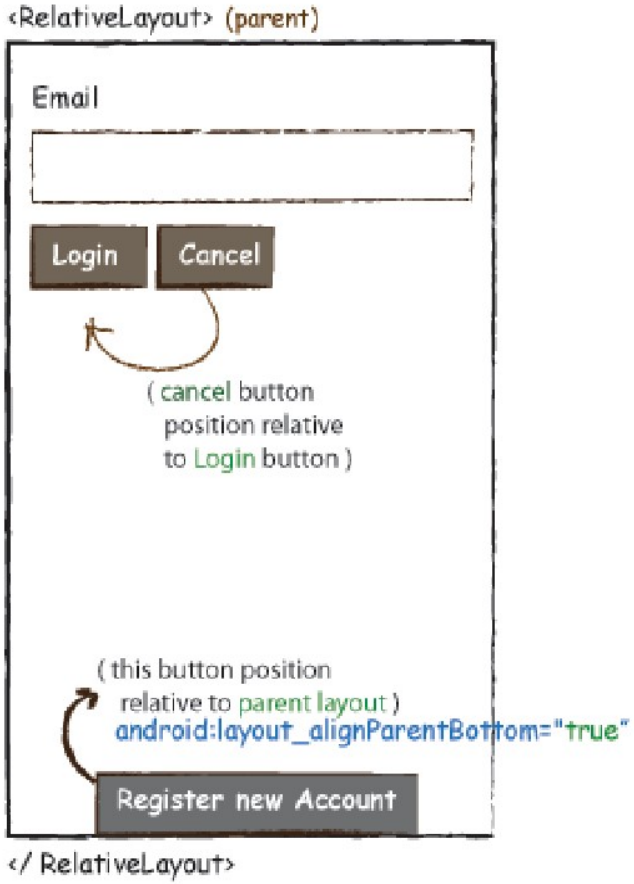
RelativeLayout

```
<TextView android:id="@+id/label"
android:layout_width="fill_parent"
android:layout_height="wrap_content"
android:text="Email" />
<EditText android:id="@+id/inputEmail"
android:layout_width="fill_parent"
android:layout_height="wrap_content"
android:layout_below="@id/label" />
<Button
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:layout_alignParentBottom="true"
android:text="Register new Account"
android:layout_centerHorizontal="true"/>
```



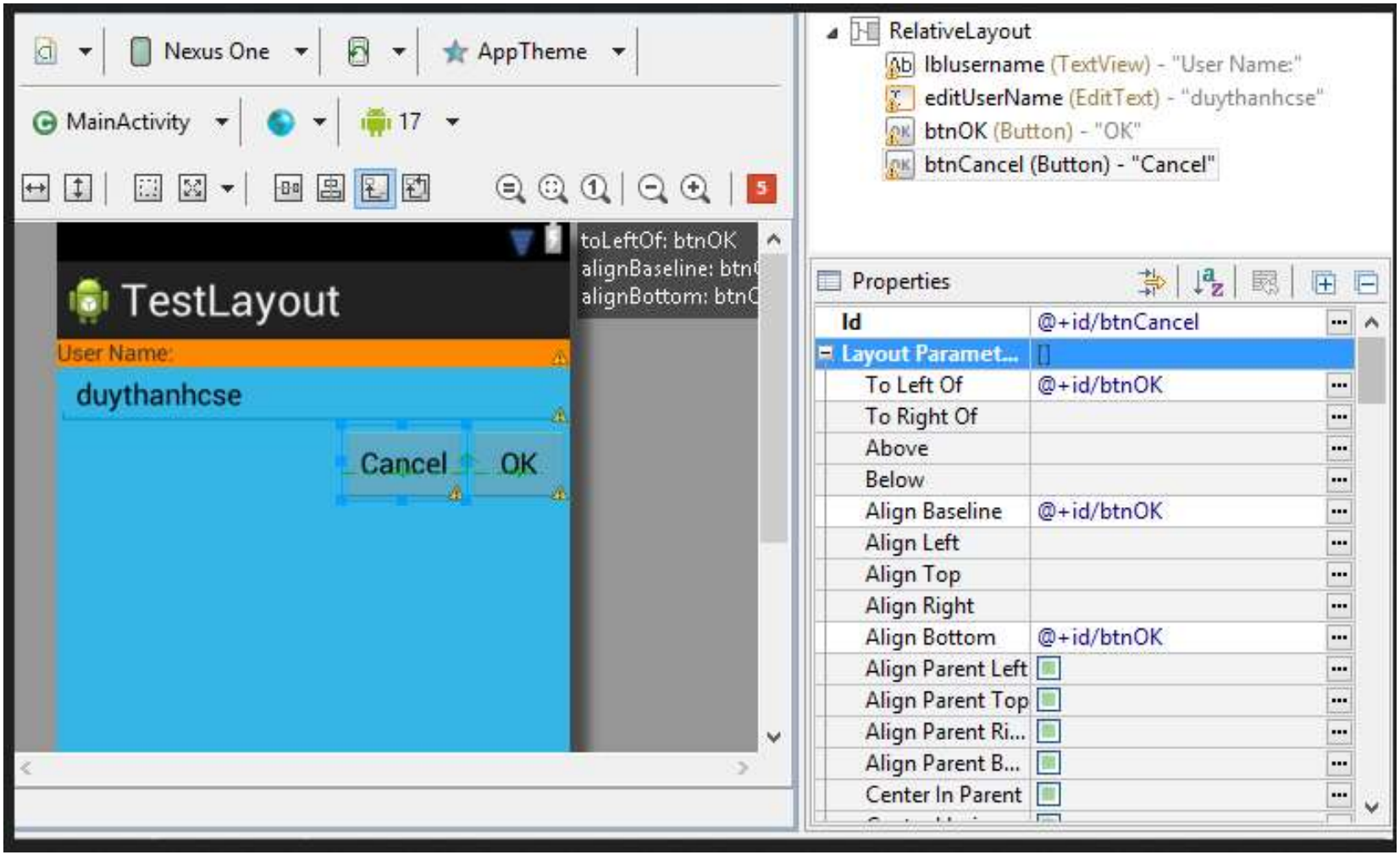
RelativeLayout

```
<Button android:id="@+id/btnLogin"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:layout_below="@id/inputEmail"
android:layout_alignParentLeft="true"
android:layout_marginRight="10px"
android:text="Login" />
<Button
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:layout_toRightOf="@id/btnLogin"
android:layout_alignTop="@id/btnLogin"
android:text="Cancel" />
```



RelativeLayout

- Có thể sử dụng công cụ để thiết kế



ConstraintLayout

- Dùng ConstraintLayout để xây dựng các layout với tính năng ràng buộc các phần tử, thiết lập các xích đa dạng
- **ConstraintLayout** là một layout mạnh, nó giúp tạo ra các giao diện phức tạp, mềm dẻo (hạn chế tối đa sử dụng các layout lồng nhau).
- Việc định vị, sắp xếp các View con dựa trên sự ràng buộc của các View con với View cha và sự liên hệ ràng buộc giữa các View con với nhau, với cơ chế tạo xích các View, gán trọng số hay sử dụng trợ giúp giao diện với Guideline.

ConstraintLayout

- Sự ràng buộc
  - Để định vị mỗi view trong **ConstraintLayout** cần tối thiểu 2 ràng buộc, một theo phương ngang (X) và một theo phương đứng (Y)
  - Khái niệm ràng buộc giữa các phần tử ở đây ám chỉ sự liên kết với nhau của các phần tử kể cả với phần tử cha ConstraintLayout.

ConstraintLayout

- Bảng các thuộc tính

Ràng buộc	Ý nghĩa ràng buộc
layout_constraintLeft_toLeftOf	Rang buộc cạnh trái của phần tử tới phần tử chỉ ra trong giá trị (gán ID)
layout_constraintLeft_toRightOf	Bên trái với bên phải của phần tử chỉ ra
layout_constraintRight_toLeftOf	Bên phải với bên trái
layout_constraintRight_toRightOf	Phải với phải
layout_constraintTop_toTopOf	Cạnh trên với cạnh trên
layout_constraintTop_toBottomOf	Cạnh trên nối với cạnh dưới

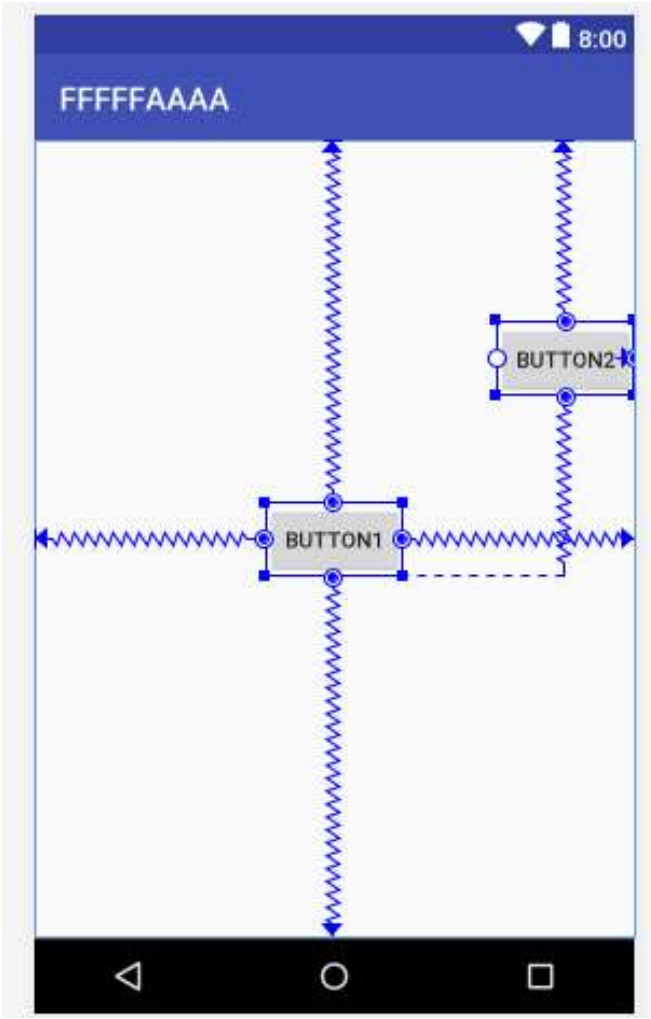
ConstraintLayout

- Bảng các thuộc tính

layout_constraintBottom_toTopOf	Dưới với trên
layout_constraintBottom_toBottomOf	Dưới với dưới
layout_constraintBaseline_toBaselineOf	Trùng Baseline
layout_constraintStart_toEndOf	Bắt đầu - Kết húc
layout_constraintStart_toStartOf	Bắt đầu - Bắt đầu
layout_constraintEnd_toStartOf	Cuối với bắt đầu
layout_constraintEnd_toEndOf	Cuối với cuối

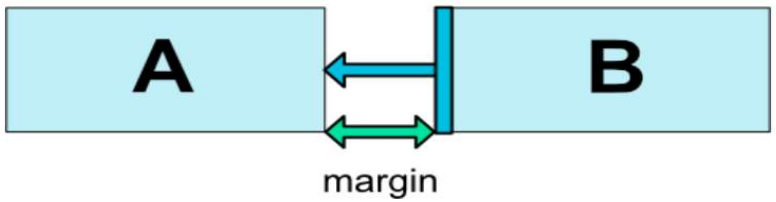
ConstraintLayout

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.constraint.ConstraintLayout xmlns:androi
    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"
    tools:context=".MainActivity">

    <Button
        android:id="@+id/button"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Button1"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

    <Button
        android:id="@+id/button2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="BUTTON2"
        app:layout_constraintBottom_toBottomOf="@+id/butt
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

</android.support.constraint.ConstraintLayout>
```



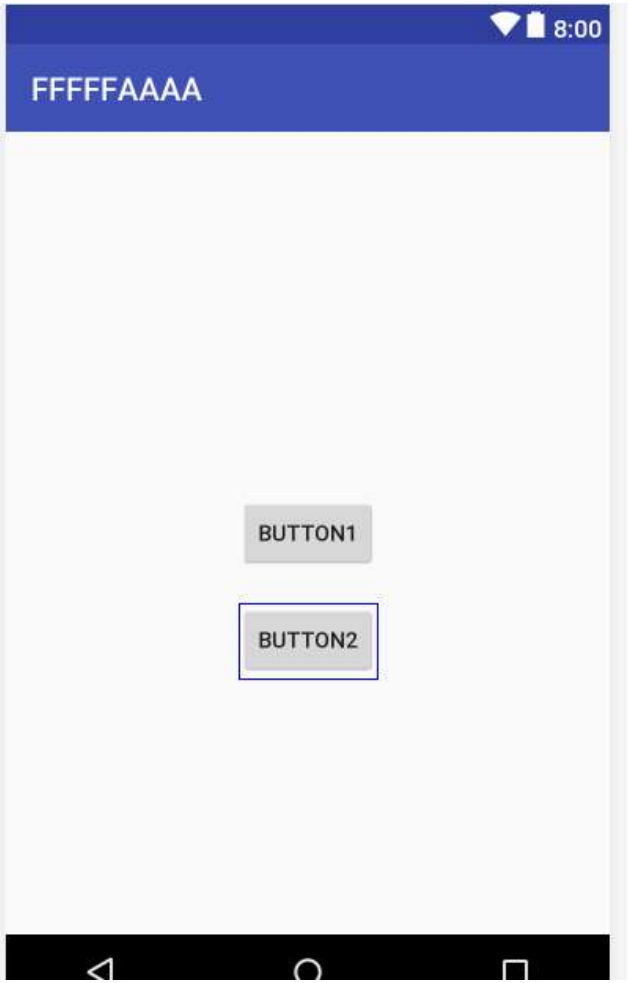
ConstraintLayout

- Thuộc tính Margin trong các phần tử con
  - Cạnh nào của View con có ràng buộc thì có thể thiết lập thêm thuộc tính Margin để điều chỉnh thêm khoảng cách các cạnh tới điểm nối ràng buộc
  - Các thuộc tính margin theo các cạnh:
    - *android:layout_marginStart,*
    - *android:layout_marginEnd,*
    - *android:layout_marginLeft,*
    - *android:layout_marginTop,*
    - *android:layout_marginRight,*
    - *android:layout_marginBottom*



ConstraintLayout

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.constraint.ConstraintLayout xmlns: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"
    tools:context=".MainActivity">

    <Button
        android:id="@+id/button"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Button1"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

    <Button
        android:id="@+id/button2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginTop="20dp"
        android:text="BUTTON2"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/button" />

</android.support.constraint.ConstraintLayout>
```



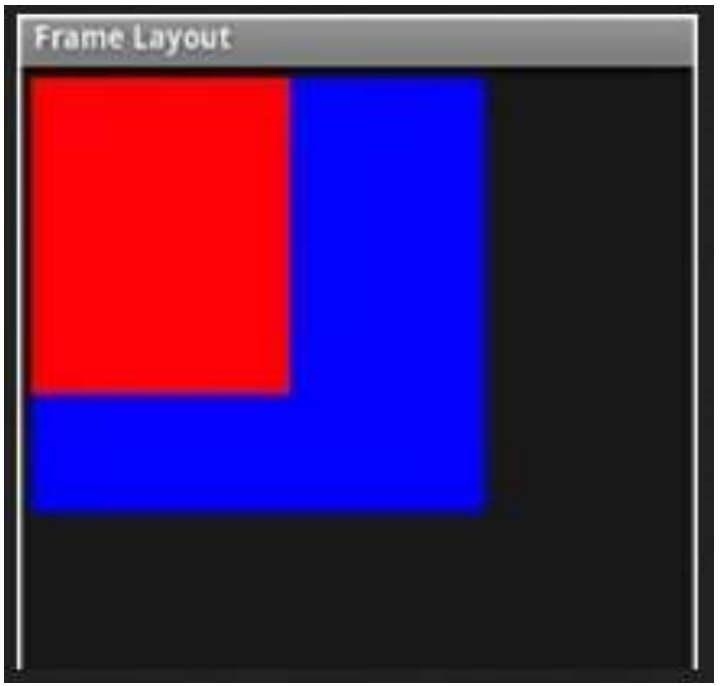
ConstraintLayout

- Phần tử Guideline
  - Tạo một đường kẻ ẩn trong ConstraintLayout nằm ngang hoặc đứng như là một View con để các View khác ràng buộc đến.
  - Thêm vào bằng cách:

```
<android.support.constraint.Guideline
    android:id="@+id/guideline_1“
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:orientation="vertical"
    app:layout_constraintGuide_percent="0.3" />
```

FrameLayout

- Các view con được đặt liên tiếp chồng lên nhau, view sau đặt lên trên view trước
- Có thể chuyển view con lên trên bằng code:
  - parent.bringChildToFront(child);
  - parent.invalidate();



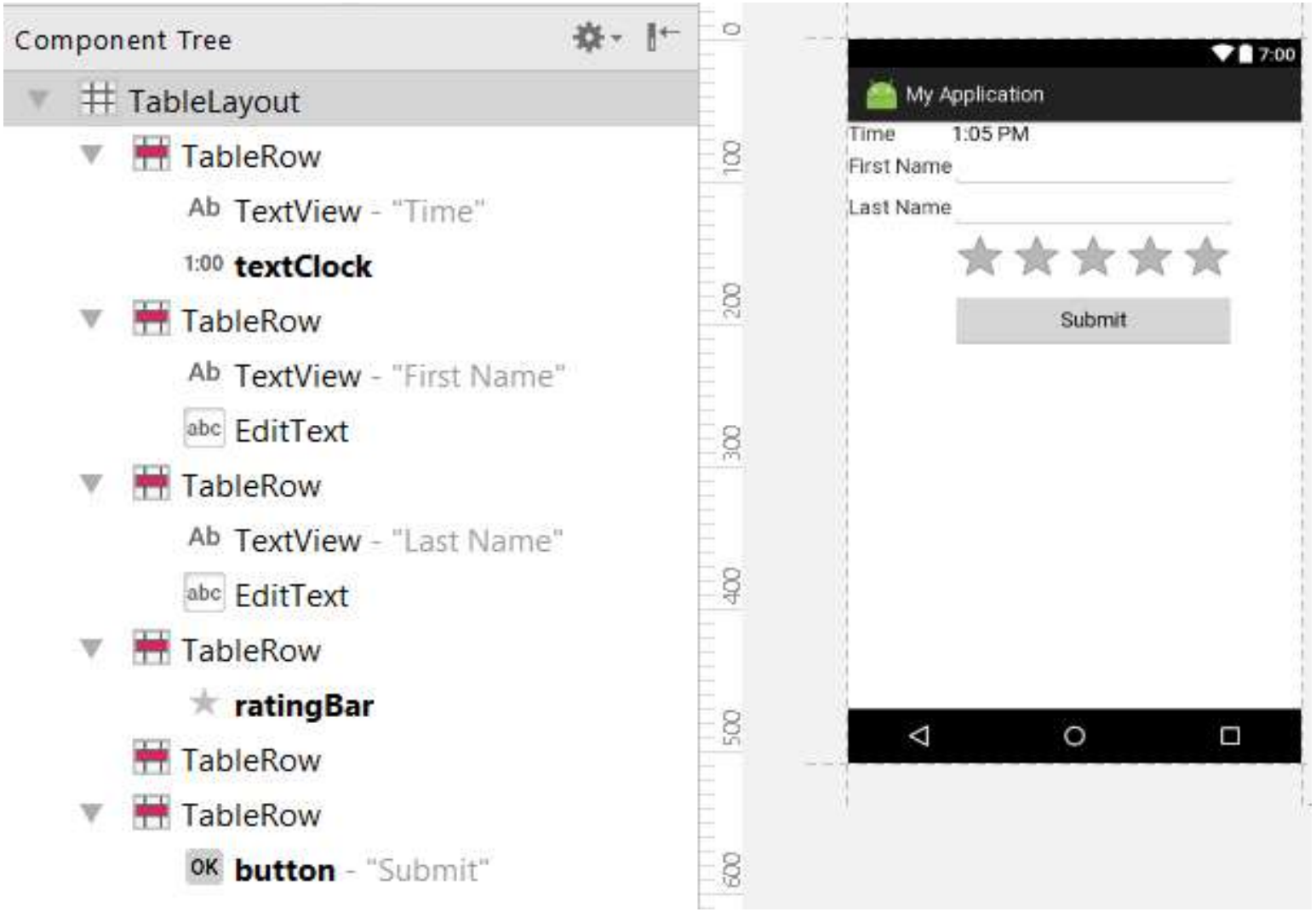
ScrollView & HorizontalScrollView

- ScrollView và HorizontalScrollView là trường hợp đặc biệt của FrameLayout
- Cho phép view con có thể có kích thước lớn hơn view cha
- Trong trường hợp view con nhỏ hơn view cha, người dùng chỉ nhìn và tương tác với view con
- Trường hợp view con có kích thước lớn hơn view cha, ScrollView và HorizontalScrollView sẽ tự động xuất hiện các thanh cuộn phù hợp

TableLayout

- TableLayout dùng để tổ chức các đối tượng view dưới dạng một bảng gồm nhiều dòng và cột
- Mỗi dòng nằm trong một thẻ <TableRow>
- Mỗi đối tượng view đặt trên một dòng sẽ tạo thành một ô trong giao diện lưới do TableLayout tạo ra
- Chiều rộng của mỗi cột được xác định bằng chiều rộng lớn nhất của các ô nằm trên cùng một cột
- Kích thước của mỗi dòng cột không nhất thiết phải bằng nhau

TableLayout



TableLayout

- Thông thường mỗi view sẽ chiếm một ô trên lưới
- Trường hợp muốn để trống ô ta có thể đặt vào đó một textview trống (bằng thẻ <TextView />)
- Tuy nhiên ta cũng có thể chỉ định kích thước và vị trí của view thông qua các thuộc tính ẩn:
  - "android:layout_span": cỡ của view (bao nhiêu cột)
  - "android:layout_column": vị trí cột đặt view

AbsoluteLayout

- Cho phép thiết lập các view theo vị trí tùy thích

```
<?xml version="1.0" encoding="utf-8"?>
<AbsoluteLayout
        android:id="@+id/myAbsoluteLayout"
        android:layout_width="fill_parent"
        android:layout_height="fill_parent"
        xmlns:android="http://schemas.android.

<Button
         android:id="@+id/myButton"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Button"
        android:layout_x="120px"
        android:layout_y="32px"
        >
</Button>
</AbsoluteLayout>
```