



THIẾT KẾ GIAO DIỆN ANDROID

BÀI 1: THIẾT KẾ LAYOUT

- ⊙ Kết thúc bài học này bạn có khả năng
 - ⊙ Hiểu được các loại layout trên android
 - ⊙ Biết được cách chọn lựa layout cho phù hợp



Phần I: Giới thiệu về Android UI

 A Giới thiệu về View và ViewGroup

 B Tạo Layout và widget, ánh xạ, bắt sự kiện

 C Đơn vị đo

Phần II: Các loại layout cơ bản

 D ConstraintLayout, RelativeLayout

 E FrameLayout, LinearLayout

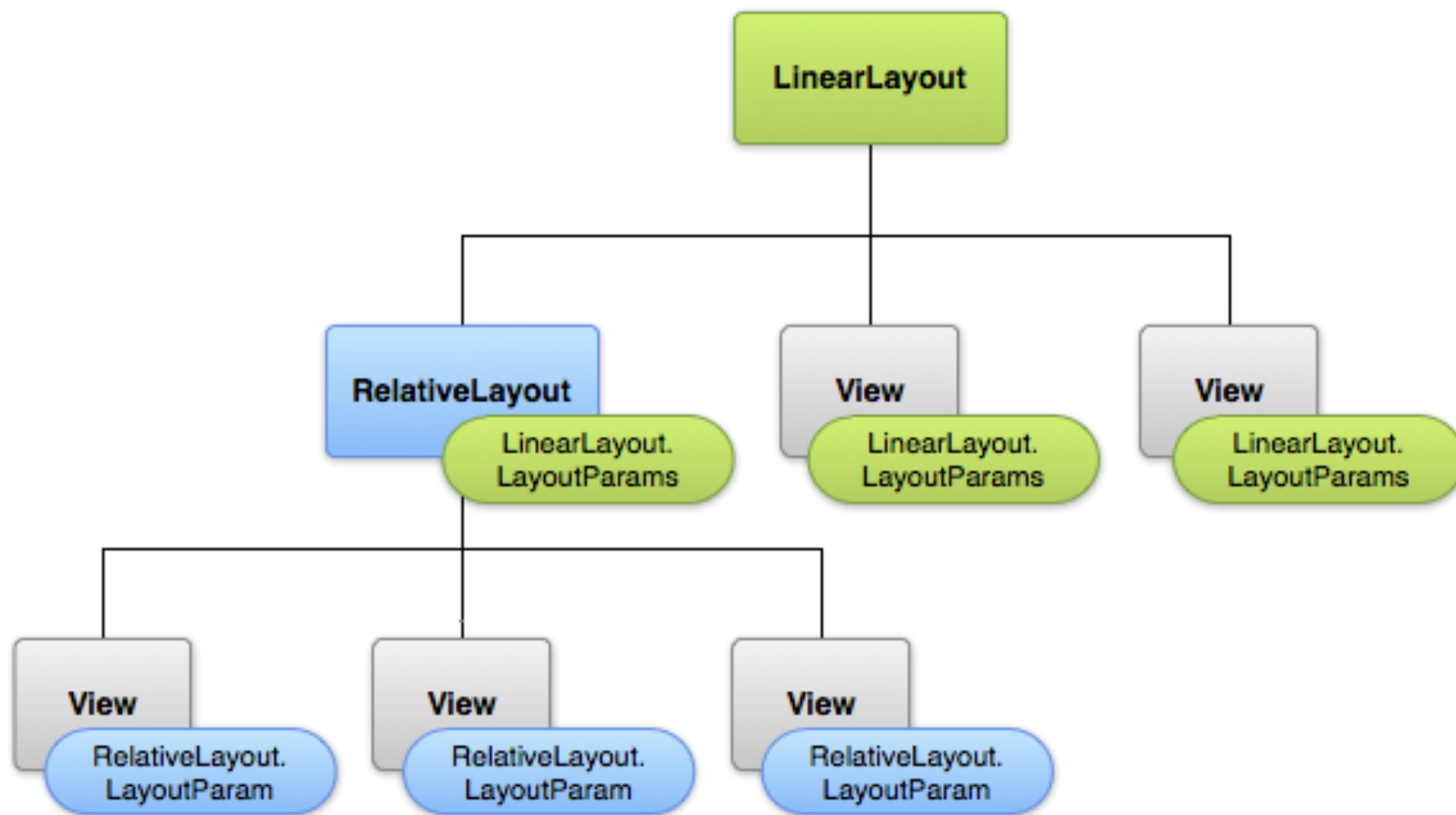




BÀI 1 : THIẾT KẾ LAYOUT

PHẦN 1: GIỚI THIỆU VỀ ANDROID UI

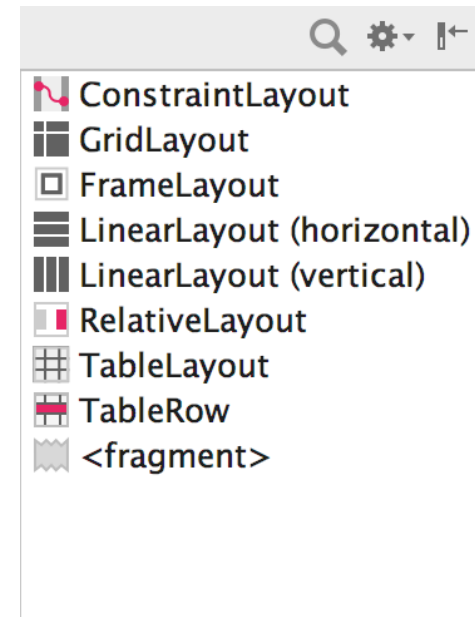
- ❑ Giao diện được tạo từ các View & View Group.



- ❑ Interface được tạo từ các View & View Group. Lớp View đại diện cho khối cơ bản của các thành phần giao diện người dùng.
- ❑ Mỗi View chiếm một vùng hình chữ nhật trên màn hình và chịu trách nhiệm drawing (vẽ) và event handling (xử lý sự kiện).
- ❑ View là lớp cơ sở cho các widget, dùng để tạo các component tương tác của UI (buttons, text fields, ...).
- ❑ Tất cả các view trong một cửa sổ được tổ chức trong một cấu trúc cây.
- ❑ Ta có thể bổ sung các view từ mã nguồn hoặc định nghĩa cấu trúc cây của các view trong một hoặc vài file layout XML.

- ❑ Lớp con **ViewGroup** là lớp cơ sở cho các *layout (bố cục)*, là các container vô hình chứa các View (hoặc các ViewGroup) khác và quy định các đặc điểm bố cục của chúng.
- ❑ Để gán UI, thì các activity phải gọi setContentView() để trỏ đến file layout mô tả UI.
- ❑ Layout được view dễ dàng trong Android Studio, bao gồm tính năng kéo thả, tùy chỉnh nhanh thuộc tính, view source, formatting source

- ❑ ViewGroup về một mặt nào đó, còn gọi là các layout
- ❑ Android hỗ trợ các layout sau:
 - **ConstraintLayout:** ràng buộc các view với các view khác
 - **LinearLayout:** sắp xếp các view theo hàng ngang hoặc hàng dọc, trong một cột hoặc một dòng
 - **GridLayout:** sắp xếp theo kiểu lưới
 - **FrameLayout:** là loại layout luôn lấy tọa độ gốc là top-left. Và nó xếp chồng view sau lên view trước.
 - **TableLayout:** sắp xếp các view thành các cột và dòng. Mỗi view trong 1 tableRow sẽ là 1 cột
 - **RelativeLayout:** cho phép chỉ ra các view bên trong có liên quan đến nhau như thế nào qua các thuộc tính:
 - `android:layout_above`
 - `android:layout_alignBaseline`
 - `android:layout_alignBottom`
 -



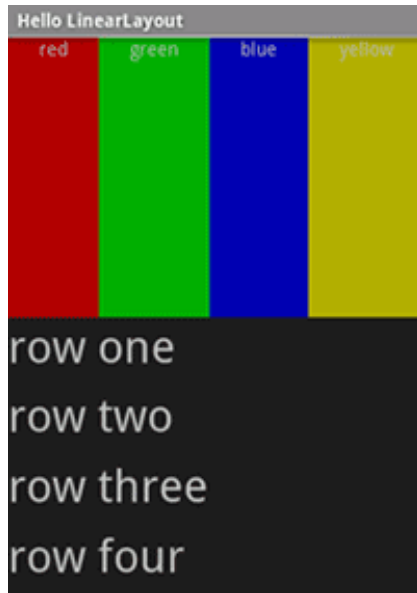
❑ Các loại view hỗ trợ thêm cho layout như:

- **ScrollView**: cho phép cuộn các layout bên trong nó, do đó có thể hiển thị được nội dung lớn hơn màn hình.
 - Chú ý: không sử dụng với ListView.
 - Bắt buộc phải đặt ở bên trong nó 1 child view chứa toàn bộ nội dung khác
- **Include**: Nhúng 1 file layout vào 1 file layout khác.

❑ Một số Layout trong gói support v4,v7:

- ConstraintLayout, CoordinatorLayout, TextInputLayout, AppLayout, TabLayout...

❑ Một số ví dụ về layout



Linear Layout



Relative Layout

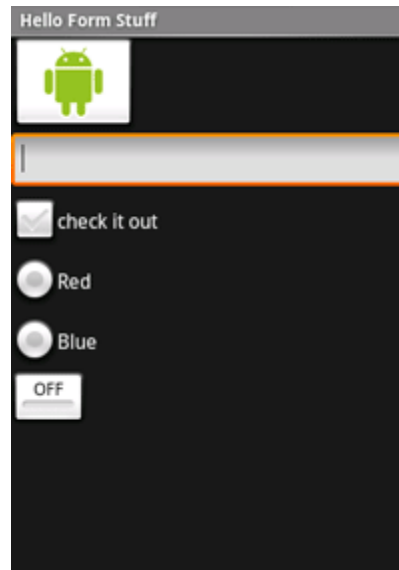


Table Layout

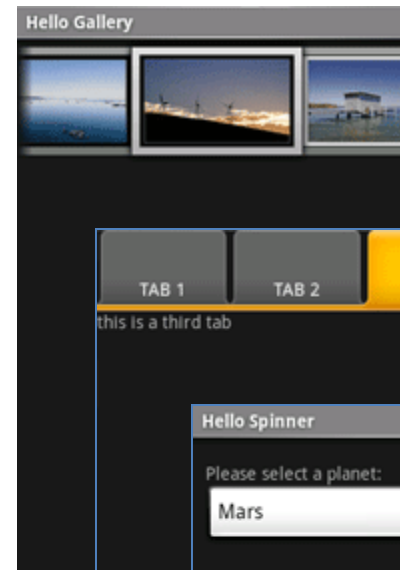
❑ Một số ví dụ về widgets



DatePicker

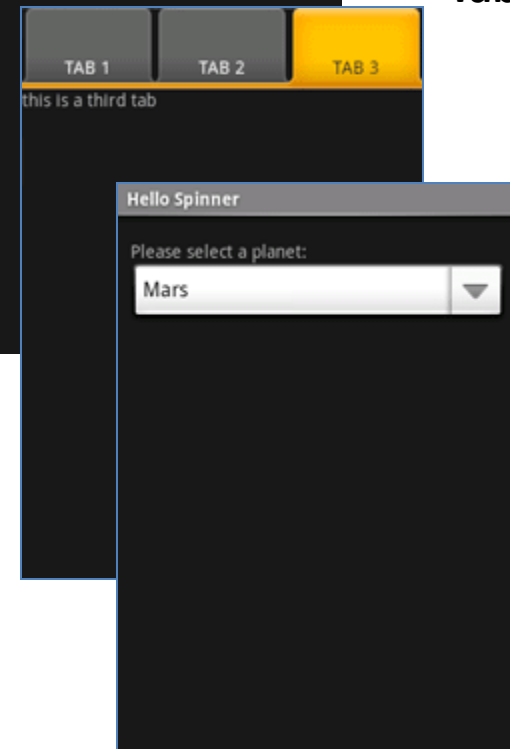


Form Controls
*image buttons,
text fields,
checkboxes and
radio buttons.*



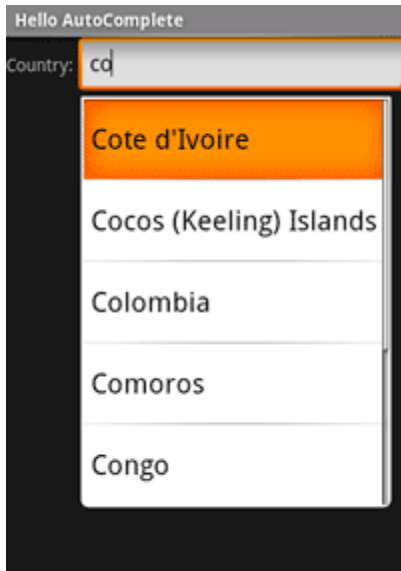
GalleryView

TabWidget

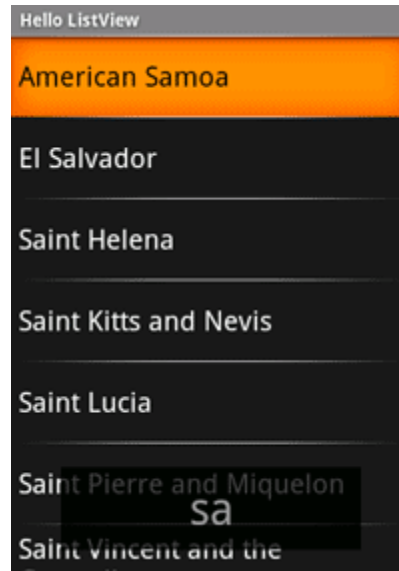


Spinner

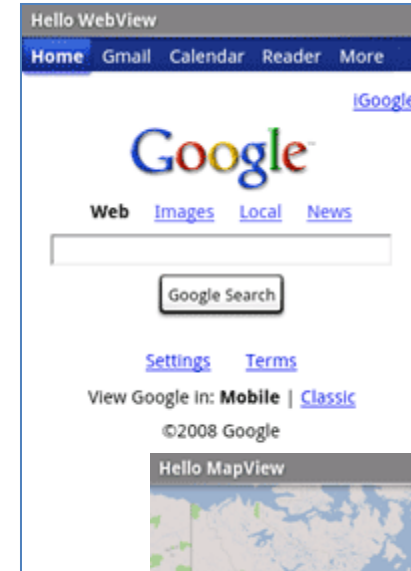
❑ Một số ví dụ về widgets



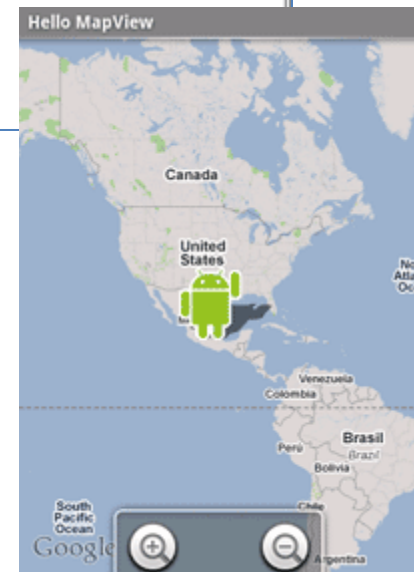
AutoCompleteTextView



ListView



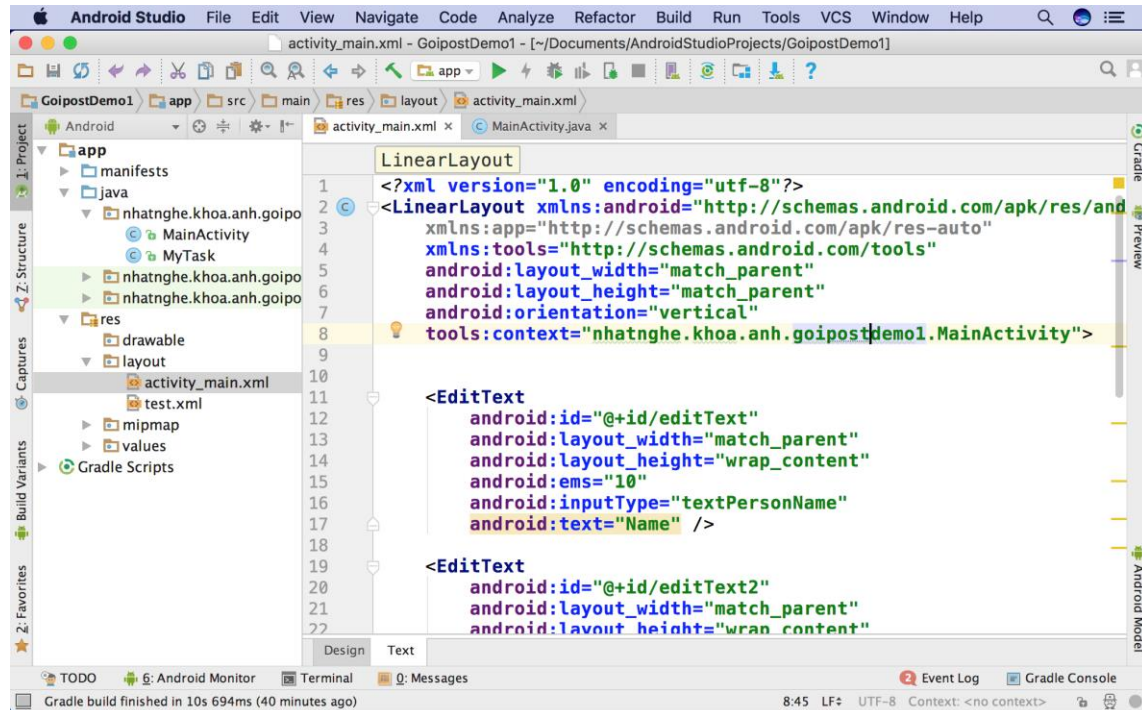
WebView



MapView

- ❑ **XML-based layout** là một đặc tả về các UI component (widget), quan hệ giữa chúng với nhau và với container chứa chúng – tất cả được viết theo định dạng XML.

Android coi các XML-based layout là các **resource (tài nguyên)**, và các file layout được lưu trong thư mục **res/layout** trong project của ta.



- ❑ Mỗi file **XML** chứa một cấu trúc phân cấp dạng cây, đặc tả layout của các widget và các container thành phần của một View.
- ❑ Các thuộc tính của mỗi phần tử XML là các tính chất, mô tả bề ngoài của widget hoặc hoạt động của một container.

Ví dụ:

Nếu một phần tử *Button* có một thuộc tính có giá trị

android:textStyle = "bold"

Nghĩa là phần text hiện trên mặt nút cần được vẽ bằng font chữ đậm (bold).

- ❑ Ta phải nối các phần tử XML với các đối tượng tương đương trong activity. Nhờ đó, ta có thể thao tác với UI từ mã chương trình
- ❑ Giả sử UI đã được tạo tại ***res/layout/main.xml***. Ứng dụng có thể gọi layout này bằng lệnh:
 - ❑ **setContentView(R.layout.*main*);**
 - ❑ Có thể truy nhập các widget, chẳng hạn ***myButton***, bằng lệnh ***findViewById(...)*** như sau
 - ❑ **Button *btn* = (Button) findViewById(R.id.*myButton*);**
 - ❑ Trong đó, **R** là một lớp được sinh tự động để theo dõi các tài nguyên của ứng dụng. Cụ thể, **R.id...** là các widget được định nghĩa trong layout XML.

❑ Gắn Listener cho Widget (event handling)

Button trong ví dụ của ta có thể dùng được sau khi gắn một listener cho sự kiện click:

```
btn.setOnClickListener(new OnClickListener() {  
    @Override  
    public void onClick(View v) {  
        //do some things;  
    }  
});
```


Đơn vị đo

- ☐ Dp/dip (Density- independent pixel): không phụ thuộc mật độ pixel, $160dp=1''$ trên màn hình
- ☐ Sp(Scaled independent pixel): thường dùng để set font size.
- ☐ Pt (Point) = $1/72$ inch, và tùy thuộc vào màn hình vật lý.
- ☐ Px(Pixel): là 1 pixel thực trên màn hình. Không nên dùng đơn vị này vì nó là đơn vị chính xác \rightarrow trên màn hình khác nhau có thể sẽ hiển thị không như ý muốn.



BÀI 1: THIẾT KẾ LAYOUT

PHẦN 2: CÁC LAYOUT CƠ BẢN

- ❑ Đưa vào từ Android Studio 2.2 Preview, nằm trong gói support nên tương thích ngược android 2.3
- ❑ Layout Editor giúp thiết kế nhanh bằng thao tác.
- ❑ Một constraint là phần mô tả làm thế nào để một View nên được định vị trên màn hình tương đối với các phần tử khác trong layout. Bạn có thể xác định một constraint cho một hay nhiều mặt của một view bằng cách chế độ kết nối bất kỳ sau đây :
 - ❖ Điểm neo nằm trên một View khác.
 - ❖ Một cạnh của layout.
 - ❖ Một guideline vô hình.

❑ Các loại của constraint bạn có thể xác định như sau :

❖ **Connection phía bên các cạnh của Layout**

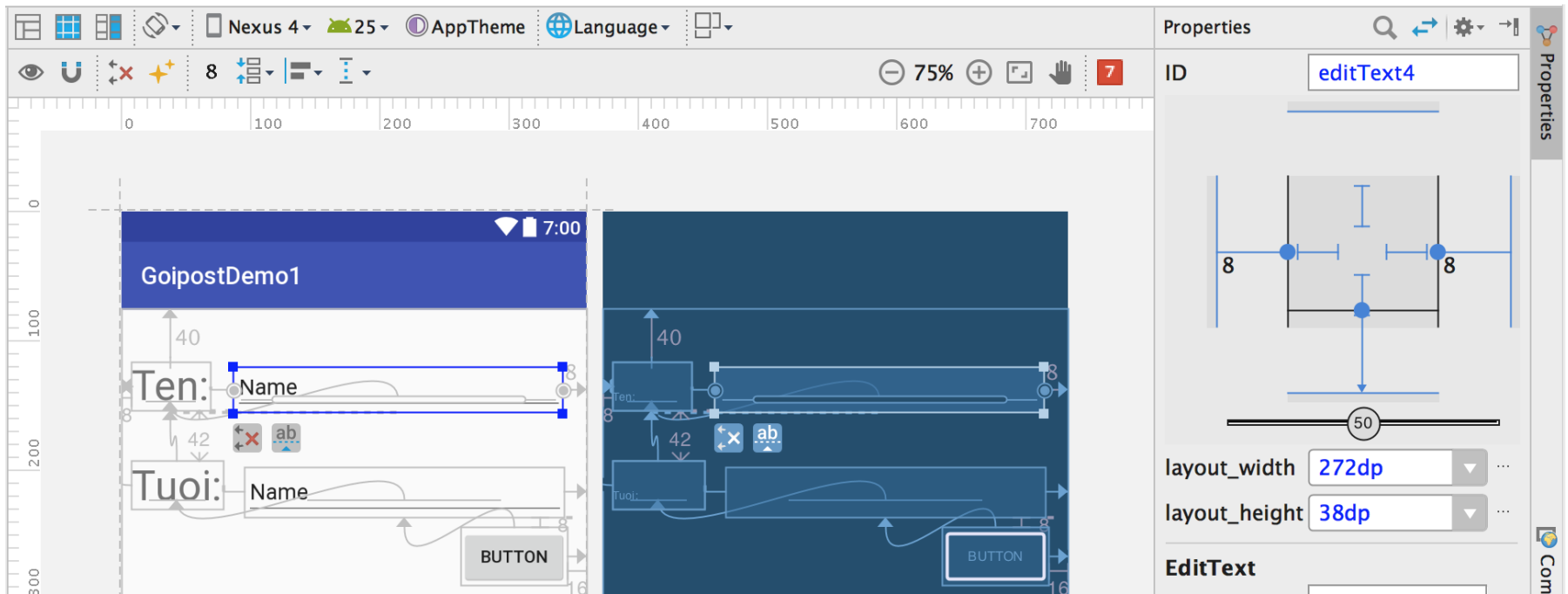
Connection phía bên của một View sang bên tương ứng với Layout. ví dụ Connection phía trên của một View vào mép trên cùng của ConstraintLayout

❖ **Connect phía bên cạnh của một View.**

Connect phía bên cạnh của một View đến phía đối diện của một View khác. Ví dụ : Connect cạnh trên top của View A vào cạnh dưới bottom của View B, như vậy thì A luôn luôn ở bên dưới B

- ❖ **Chỉnh thẳng hàng với một View.** Canh thẳng hàng các cạnh của một view để các cạnh và góc của view khác tương tự. Ví dụ sắp xếp bên trái của View A đến phía bên trái của View B sao cho chúng được xếp chồng theo chiều dọc và canh trái.
- ❖ **Chỉnh cơ bản với một View :** Căn đường cơ bản của văn bản nằm trong view với một đường cơ bản của văn bản trong view khác. Sử dụng điều này để gắn kết các view theo chiều ngang khi nó quan trọng hơn là các văn bản trong view hơn là căn các cạnh trong view.

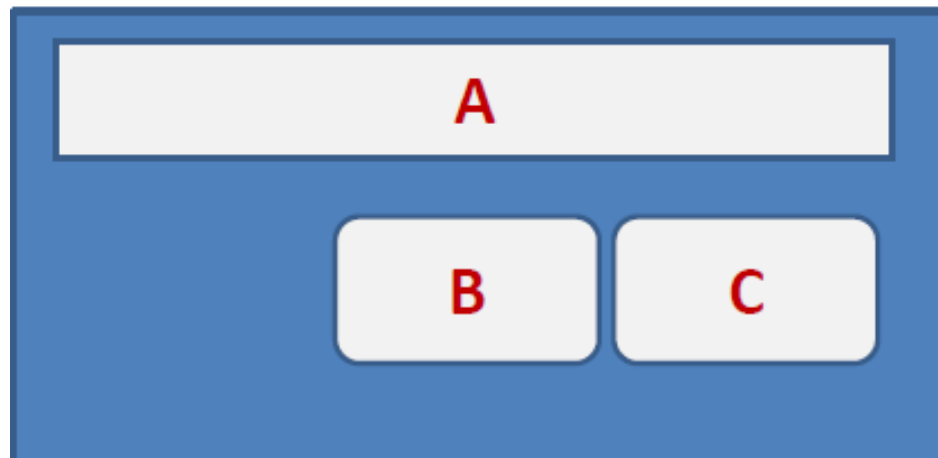
- ❑ Tham khảo các thao tác cho constraintlayout:
 - <https://developer.android.com/training/constraint-layout/index.html>
 - <https://codelabs.developers.google.com/codelabs/s/constraint-layout/#0>



- ❑ RelativeLayout cho phép sắp xếp các control theo vị trí tương đối giữa các control khác trên giao diện (kể cả control chứa nó).
- ❑ Thường ta sẽ dựa vào Id của các control khác để sắp xếp theo vị trí tương đối.
- ❑ Do đó khi làm RelativeLayout phải chú ý là đặt Id control cho chuẩn xác, nếu sau khi Layout xong mà lại đổi Id của các control thì giao diện sẽ bị xáo trộn (do đó nếu đổi ID thì phải đổi luôn các tham chiếu khác sao cho khớp với Id mới đổi).

❑ Ví dụ cho thấy

- A đứng trên đầu ,
- C bên dưới A và ở phía bên phải,
- B bên dưới A và bên trái C



- ❑ Một số thuộc tính sắp xếp widget với layout chứa nó:
 - **android:layout_alignParentTop:** chỉ ra rằng widget phải được đặt ở đầu của layout mà nó nằm.
 - **android:layout_alignParentBottom** đặt ở dưới cùng
 - **android:layout_alignParentLeft** đặt ở bên trái
 - **android:layout_alignParentRight** : đặt ở bên phải
 - **android:layout_centerInParent** : đặt ở trung tâm
 - **android:layout_centerHorizontal:** đặt ở trung tâm theo chiều ngang
 - **android:layout_centerVertical:** đặt ở trung tâm theo chiều dọc

- ❑ Một số thuộc tính sắp xếp widget với các widget hoặc control khác:
 - **android:layout_above** chỉ ra rằng widget phải được đặt ở trên của widget tham chiếu.
 - **android:layout_below** chỉ ra rằng widget phải được đặt ở dưới của widget tham chiếu.
 - **android:layout_toLeftOf** chỉ ra rằng widget phải được đặt ở bên trái của widget tham chiếu.
 - **android:layout_toRightOf** chỉ ra rằng widget phải được đặt ở bên phải của widget tham chiếu.

- **android:layout_alignTop:** làm cho top của widget này căn bằng với top của widget tham chiếu
- **android:layout_alignBottom** làm cho cạnh dưới của widget này căn bằng với cạnh dưới của widget tham chiếu
- **android:layout_alignLeft** làm cho cạnh trái của widget này căn bằng với cạnh trái của widget tham chiếu
- **android:layout_alignRight** làm cho cạnh phải của widget này căn bằng với cạnh phải của widget tham chiếu

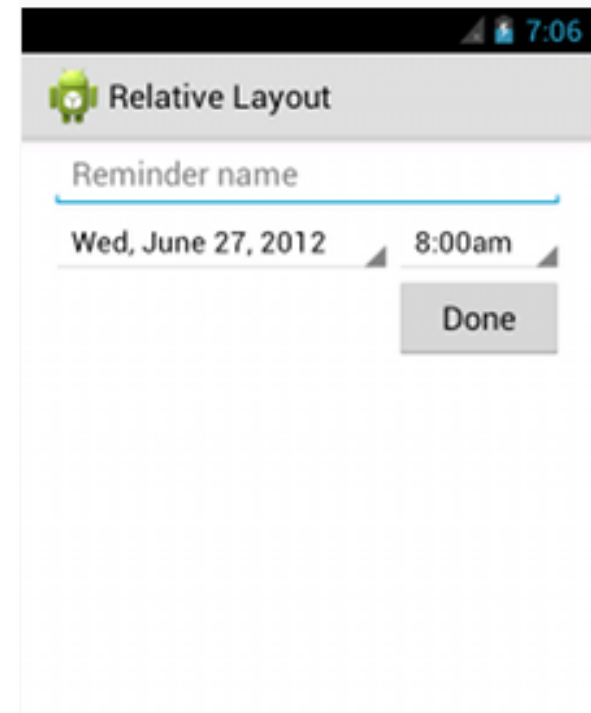
❑ Để sắp xếp các thẻ, ta cần phải làm những bước sau:

- Gán Id cho tất cả các phần tử control (**android:id**)
- Cú pháp: **@+id/...** để đặt id cho từng control, ví dụ cho thẻ EditText là **android:id = "@+id/editUserName"**
- Để bố trí control khác liên quan đến control này, ta cũng sẽ sử dụng giá trị (**@+id/...**). Ví dụ đặt một control dưới hộp EditText ở trên ta có câu lệnh:
❖ **android:layout_below = "@+id/editUserName"**

Ví dụ

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:paddingLeft="16dp"
    android:paddingRight="16dp" >
    <EditText
        android:id="@+id/name"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:hint="@string/reminder" />
    <Spinner
        android:id="@+id/dates"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_below="@id/name"
        android:layout_alignParentLeft="true"
        android:layout_toLeftOf="@+id/times" />
    <Spinner
        android:id="@+id/times"
        android:layout_width="96dp"
        android:layout_height="wrap_content"
        android:layout_below="@id/name"
        android:layout_alignParentRight="true" />
```

```
<Button
    android:layout_width="96dp"
    android:layout_height="wrap_content"
    android:layout_below="@id/times"
    android:layout_alignParentRight="true"
    android:text="@string/done" />
</RelativeLayout>
```



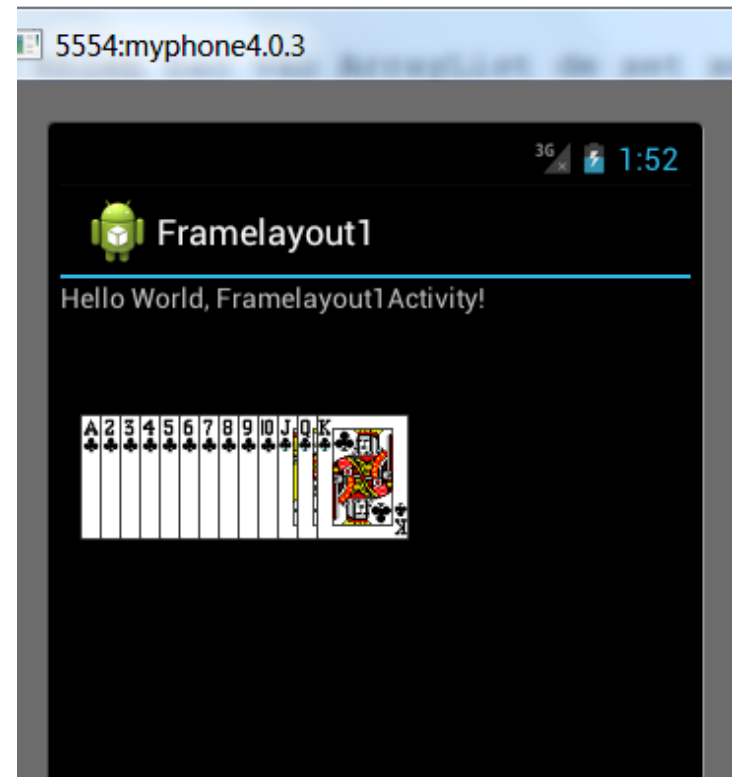


DEMO

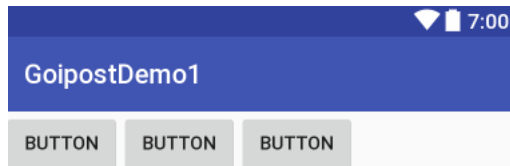
ConstraintLayout
RelativeLayout



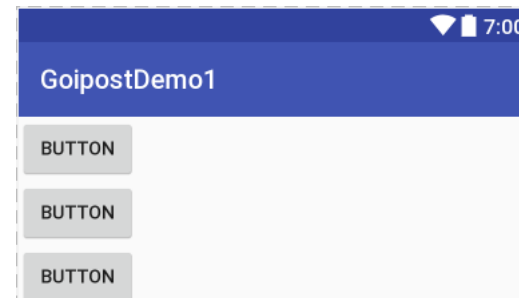
- ❑ Là loại Layout cơ bản nhất, đặc điểm của nó là khi gắn các control lên giao diện thì các control này sẽ luôn được "Neo" ở góc trái trên cùng màn hình.
- ❑ Các control đưa vào sau nó sẽ đè lên trên và che khuất control trước đó.
- ❑ Ta có thể dùng `layout_margin` để canh hờ top left cho các control



- ❑ Layout sắp xếp các control theo từng hàng hoặc từng cột
- ❑ Thuộc tính orientation gán bằng vertical hoặc horizontal

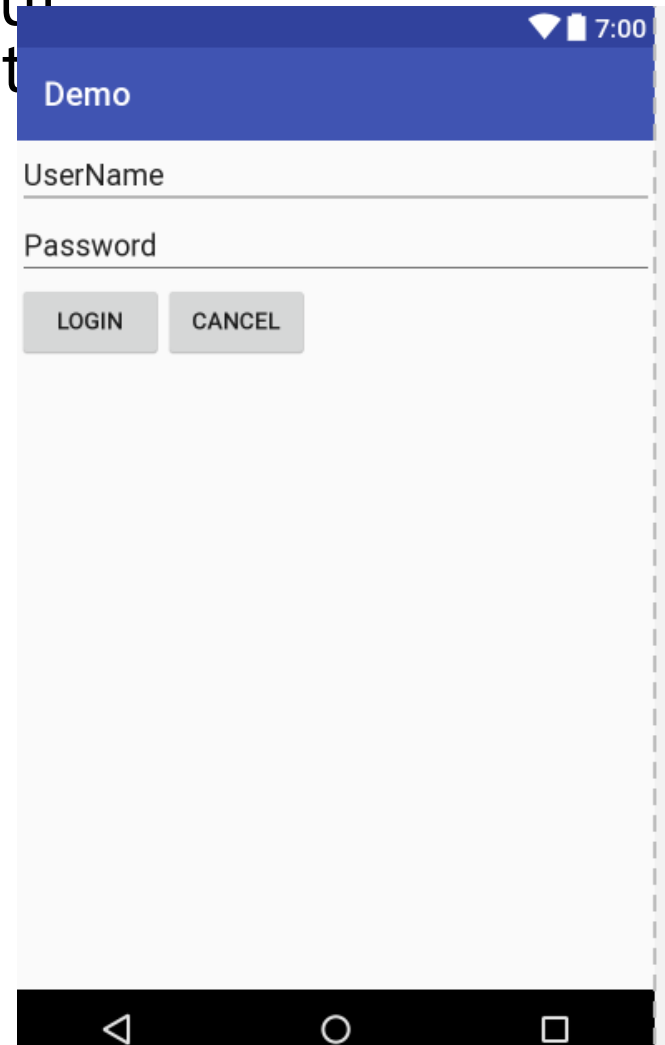
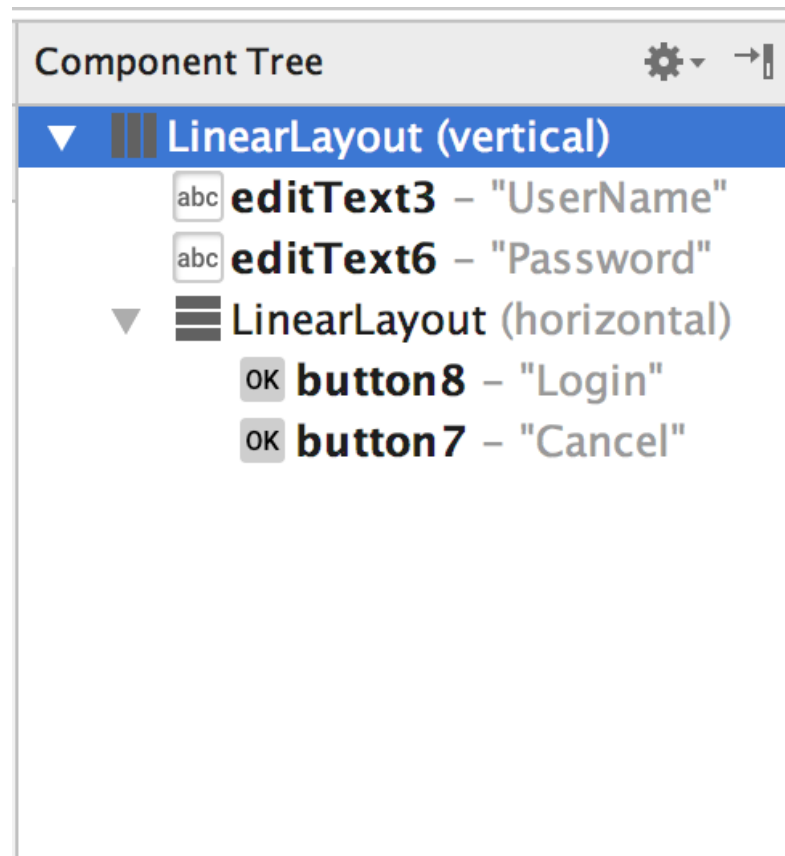


- Hướng từ trái qua phải:
`android:orientation="vertical"`



- Hướng từ trên xuống dưới:
`android:orientation="horizontal"`

- ❑ Có thể dùng Layout lồng nhau để tạo giao diện phức tạp hơn (tất cả các loại layout không phải chỉ riêng LinearLayout)





DEMO

FrameLayout
LinearLayout



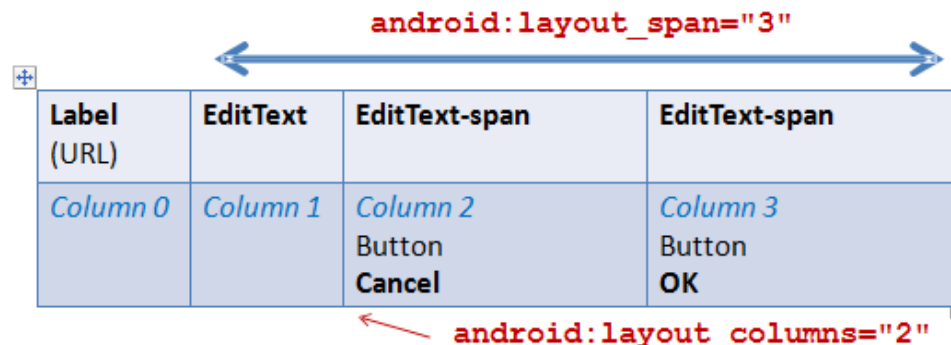
- ❑ Cho phép sắp các control theo dạng lưới (dòng và cột).
- ❑ Các cột có thể thu nhỏ hoặc giãn rộng tùy thuộc vào nội dung chứa.
- ❑ TableLayout làm việc với các TableRow .
- ❑ TableLayout sẽ xem dòng nào có số lượng control nhiều nhất để xác định rằng nó có bao nhiêu cột (lấy dòng có số lượng control nhiều nhất làm số cột chuẩn). Hoặc ta có thể lồng các TableRow để tạo các

| | | | |
|---|---|---|---|
| 0 | | 1 | |
| 0 | | 1 | 2 |
| 0 | 1 | 2 | 3 |

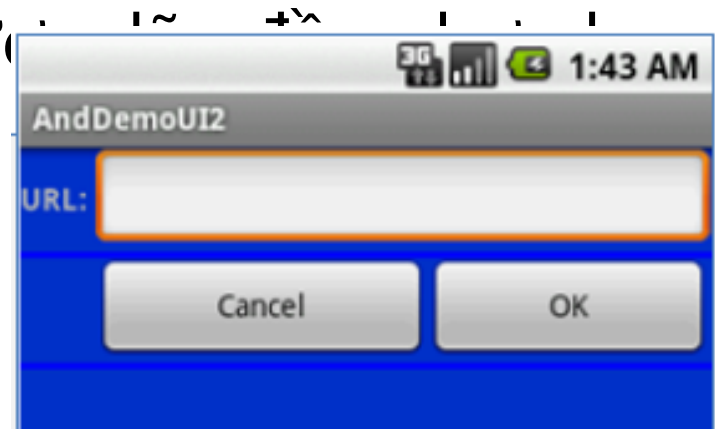
- ❑ Dùng **layout_span** để trộn các cột: thẻ EditText sẽ được trộn bởi 3 cột ở hàng đầu như hình dưới.

```
<TableRow>
    <TextView            android:text="URL:" />
    <EditText
        android:id="@+id/entry"
        android:layout_span="3" />
</TableRow>
```

- ❑ Dùng **layout_column** để di chuyển vị trí của control đến một cột nào đó trên 1 dòng. Các cột sẽ đánh số bắt đầu từ ^



- ❑ Nếu để mặc định mỗi cột sẽ tự động dẫn theo kích cỡ của các control mà nó chứa.
- ❑ Ta có thể dùng thuộc tính **stretchColumns** để dẫn đều các control, các cell (ta thường dùng dấu "*"):
- ❑ Ví dụ dưới: cột 2, 3,4 sẽ được



```
<TableLayout android:id="@+id/myTableLayout"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:background="#ff0033cc"
    android:orientation="vertical"
    android:stretchColumns="2,3,4"
    xmlns:android="http://schemas.android.com/apk/res/android">
```

❏ Ví dụ về TableLayout

Nexus 4
 25
 AppTheme
 Language
 50%

 9+

Demo

Row 1

Row 2 Column 1

Row 2 Column 2

Row 2 Column 3

Row 3 Column 1

Row 3 Column 2

Row 4 Column 1

Row 4 Column 2

Row 5 Column 1

TableLayout

Row 1

Row 2\nColumn 1

Row 2\nColumn 2

Row 2\nColumn 3

Row 3\nColumn 1

Row 3\nColumn 2

Row 4\nColumn 1

Row 4\nColumn 2

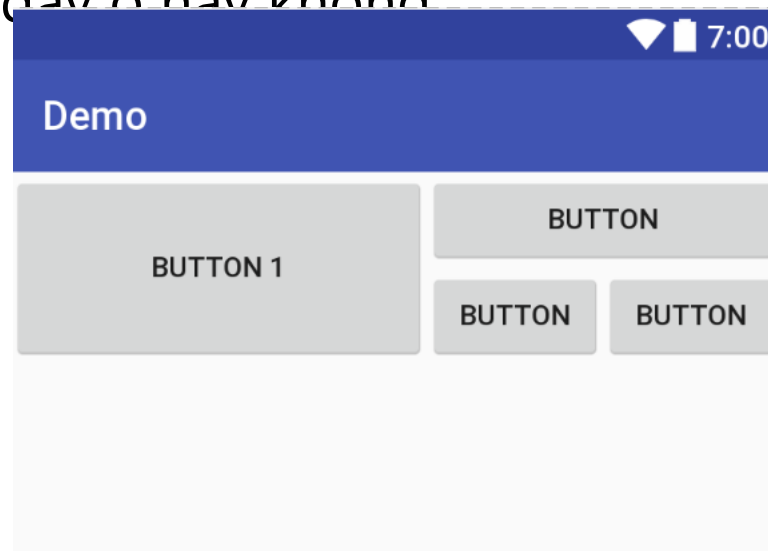
Row 5\nColumn 1

TableLayout

Component Tree

- ▼ **table_layout** (TableLayout)
 - ▼ **row_1** (TableRow)
 - Ab TextView – "Row 1"
 - ▼ **row_2** (TableRow)
 - Ab TextView – "Row 2\nColumn 1"
 - Ab TextView – "Row 2\nColumn 2"
 - Ab TextView – "Row 2\nColumn 3"
 - ▼ **row_3** (TableRow)
 - Ab TextView – "Row 3\nColumn 1"
 - Ab TextView – "Row 3\nColumn 2"
 - ▼ **row_4** (TableRow)
 - Ab TextView – "Row 4\nColumn 1"
 - Ab TextView – "Row 4\nColumn 2"
 - ▼ **row_5** (TableRow)
 - Ab TextView – "Row 5\nColumn 1"
 - ▼ **LinearLayout** (horizontal)
 - Ab TextView – "TableLayout"

- ❑ Yêu cầu android ICS 4.0 (hoặc dùng thư viện v7 để tương thích ngược).
- ❑ Dạng lưới với các hàng và các cột
- ❑ Giống TableLayout nhưng mạnh mẽ hơn. Không cần TableRow.
- ❑ Dùng thuộc tính `columnCount` và `rowCount` để quyết định số hàng và số cột hình thành nên số ô
- ❑ Có thể gom các ô với thuộc tính `layout_rowspan` và `layout_columnspan`
- ❑ Dùng `layout_gravity` để quyết định các view trong ô được gom có được tràn đầy ô hay không



<TextView

```
android:text="Email setup"
android:textSize="32dip"
android:layout_columnSpan="4"
android:layout_gravity="center_horizontal" />
```

<TextView

```
android:text="You can configure email in just a few steps:"
android:textSize="16dip"
android:layout_columnSpan="4"
android:layout_gravity="left" />
```

<TextView

```
android:text="Email address:"
android:layout_gravity="right" />
```

<EditText

```
android:ems="10" />
```

<TextView

```
android:text="Password:"
android:layout_column="0"
android:layout_gravity="right" />
```

<EditText

```
android:ems="8" />
```

<Space

```
android:layout_row="4"
android:layout_column="0"
android:layout_columnSpan="3"
android:layout_gravity="fill" />
```

<GridLayout

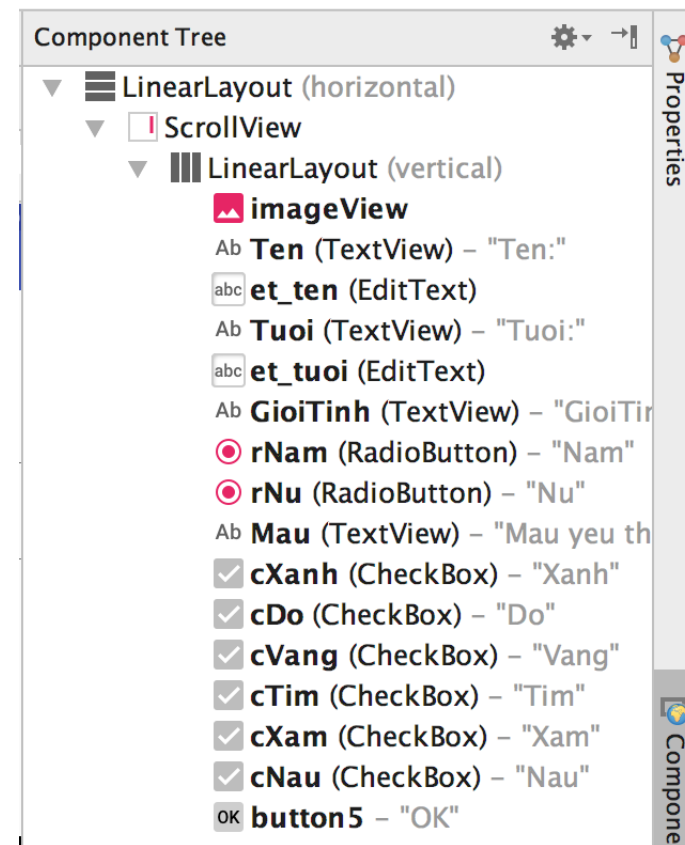
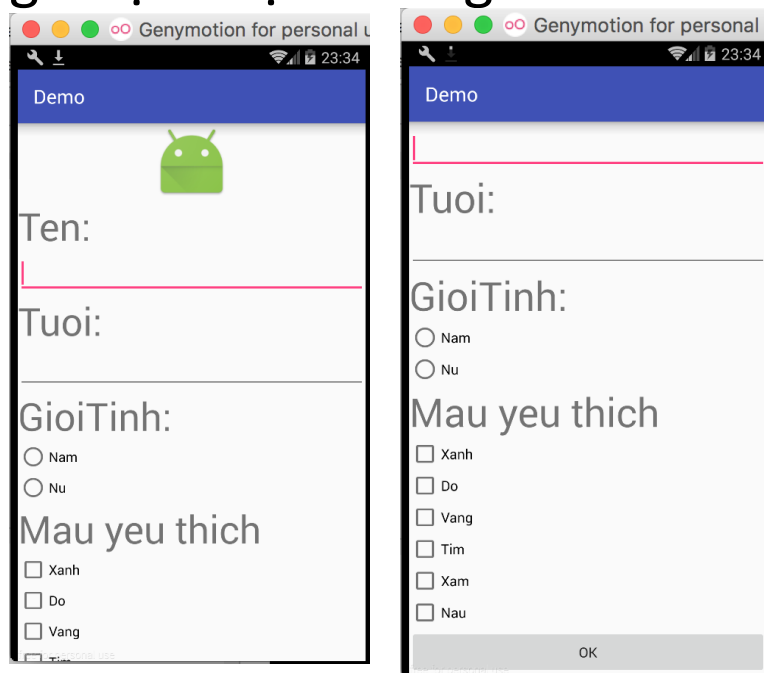
```
android:layout_width="match_parent"
android:layout_height="match_parent"
android:useDefaultMargins="true"
android:alignmentMode="alignBounds"
android:columnOrderPreserved="false"
android:columnCount="4" >
```



<Button

```
android:text="Next"
android:layout_row="5"
android:layout_column="3" />
```


- ❑ Khi cần hiển thị nhiều dữ liệu lên màn hình, ta sử dụng ScrollView.
- ❑ Nhờ có thanh trượt và thanh cuộn, người dùng có thể nhìn thấy phần còn lại của dữ liệu trên màn hình bằng cách trượt sang hoặc cuộn xuống.





DEMO

TableLayout
GridLayout
ScrollView



- ☐ View & ViewGroup
- ☐ ConstraintLayout
- ☐ RelativeLayout
- ☐ FrameLayout
- ☐ Tablelayout
- ☐ ScrollView





Cảm ơn