Fpt University

FPT POLYTECHNIC

THIẾT KẾ GIAO DIỆN TRÊN ANDROID

Bài 1: Tổng quan về UI

www.poly.edu.vn

Nội dung bài học

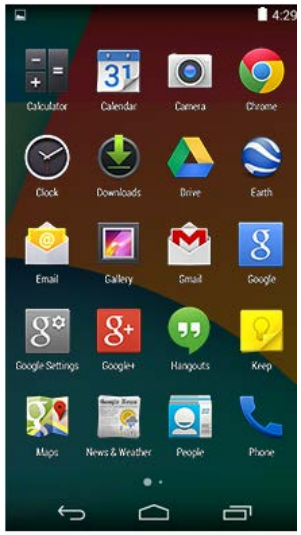
1. Giới thiệu về Android UI
2. Các Layout cơ bản
   - FrameLayout
   - LinearLayout

1. Giới thiệu về Android User Interface

Giới thiệu về screen:

- Home screen: sẽ chứa các folder, widgets hoặc các app shortcuts tùy theo ý người dùng
- All apps screen: hiển thị toàn bộ ứng dụng được cài trên máy, và có thể đưa một ứng dụng bất kỳ ra màn hình Home
- Recents screen: cung cấp cho người dùng truy cập đến các ứng dụng đang chạy gần đây một cách nhanh nhất

Home screen



All apps screen

Recents screen

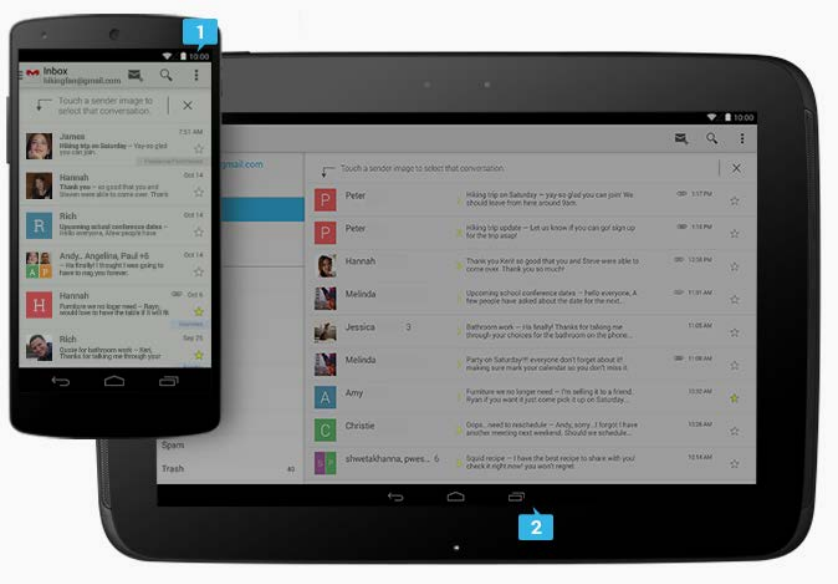
1. Giới thiệu về Android User Interface

Giới thiệu về screen:

- System Bar:

  1. Status Bar: bên trái hiển thị trạng thái như thời gian, mức độ pin, bên phải hiển thị tín hiệu sóng…

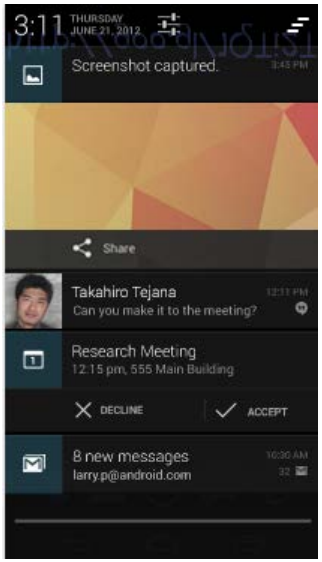
  2. Navigation Bar: là thanh điều khiển chứa các nút Back, Home, Recents, thay cho các phím cứng được thiết kế trên máy.

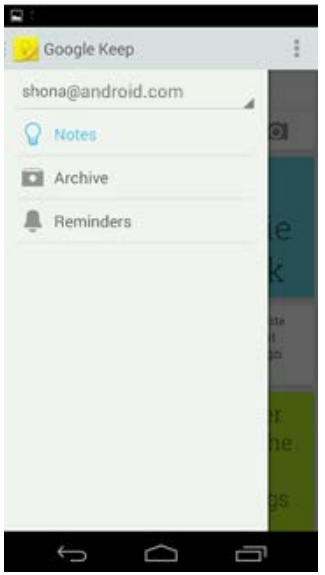


1. Giới thiệu về Android User Interface

Giới thiệu về screen:

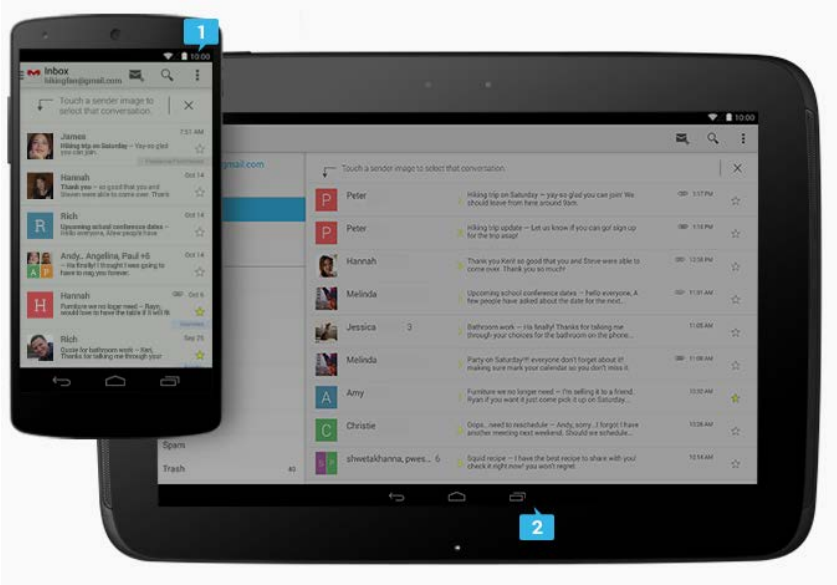
- Notifications: là các thông báo ngắn gọn mà không làm ảnh hưởng tới hoạt động người dùng, có thể truy cập nhanh từ thanh Status Bar
- Common App UI
  - Action Bar
  - Navigation Drawer
  - Content Area





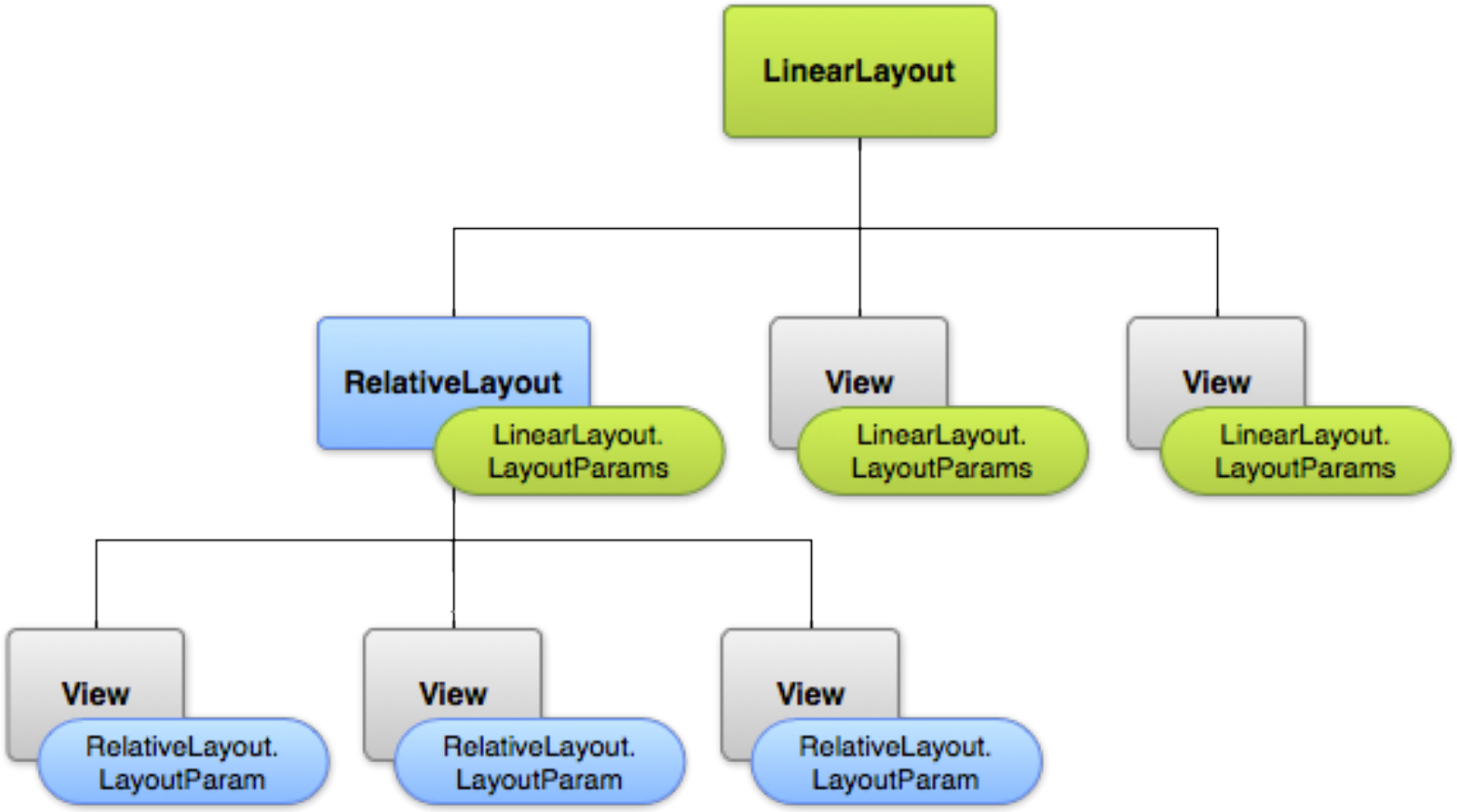
1. Giới thiệu về Android User Interface

- Demo → Giới thiệu màn hình trực tiếp trên thiết bị



1. Giới thiệu về Android User Interface

- Interface được tạo từ các View & View Group.



1. Giới thiệu về Android User Interface

- 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.

1. Giới thiệu về Android User Interface

- 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 Eclipse, bao gồm tính năng kéo thả, tùy chỉnh nhanh thuộc tính, view source, formating source.

1. Giới thiệu về Android User Interface

- Sau khi đã tạo một cây view, có một số thao tác có thể cần thực hiện:
  1. **Set properties:** ví dụ gán sẵn dòng text trong một *TextView*. Các property biết từ trước có thể được đặt sẵn trong các file layout XML.
  2. **Set focus:** cơ chế di chuyển focus để đáp ứng input của người dùng. Để yêu cầu focus cho một view cụ thể, gọi hàm *requestFocus().*
  3. **Set up listeners:** View cho phép đặt các listener, các listener này được gọi khi có sự kiện xảy ra đối với view. Ví dụ, một Button dùng một listener để nghe sự kiện button được click.
  4. **Set visibility:** Ta có thể che hoặc hiện view bằng *setVisibility(int)*.

1. Giới thiệu về Android User Interface

ViewGroup về một mặt nào đó, còn gọi là các layout

Android hỗ trợ các layout sau:

- **LinearLayout**: sắp xếp các view theo hàng ngang hoặc hàng dọc, trong một cột hoặc một dòng
- **AbsoluteLayout**: sắp xếp các view ở chính xác tọa độ mong muốn thông qua các thuộc tính layout_x và layout_y
- **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:

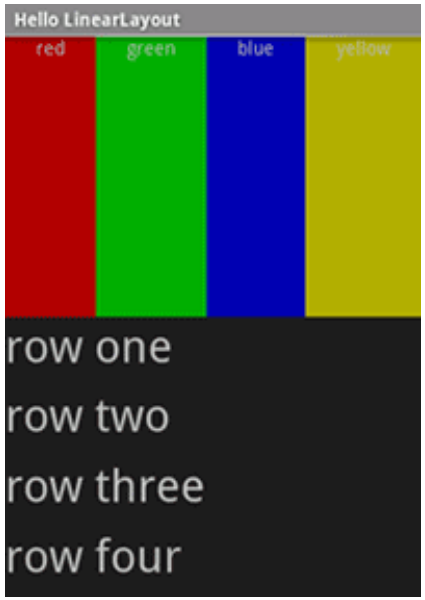
  android:layout_above
  android:layout_alignBaseline
  android:layout_alignBottom
  ……

1. Giới thiệu về Android User Interface

- **FrameLayout**: la loại layout luôn lấy tọa độ gốc là top- left. Và nó xếp chồng view sau lên view trước.
- **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

1. Giới thiệu về Android User Interface

Một số ví dụ về Layouts



Linear Layout

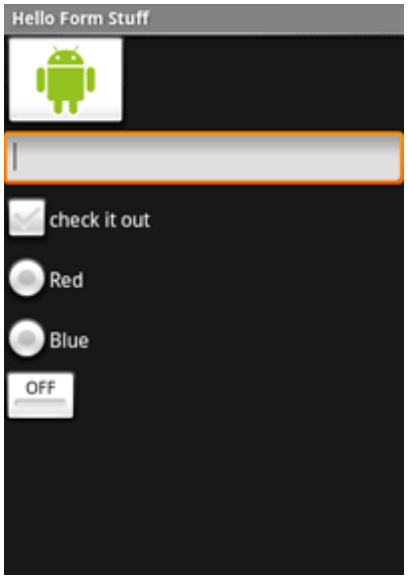
Relative Layout

Table Layout

1. Giới thiệu về Android User Interface

Một số ví dụ về Widgets

DatePicker



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

GalleryView

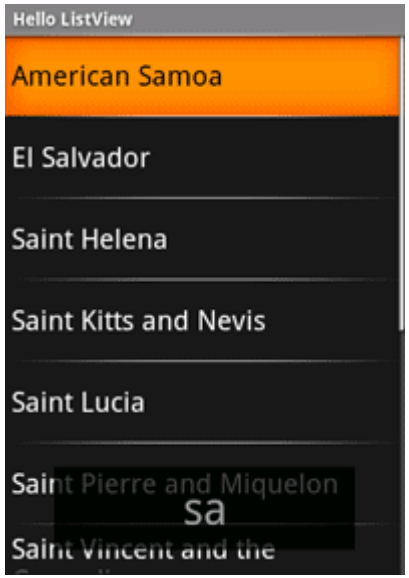
TabWidget

Spinner

1. Giới thiệu về Android User Interface

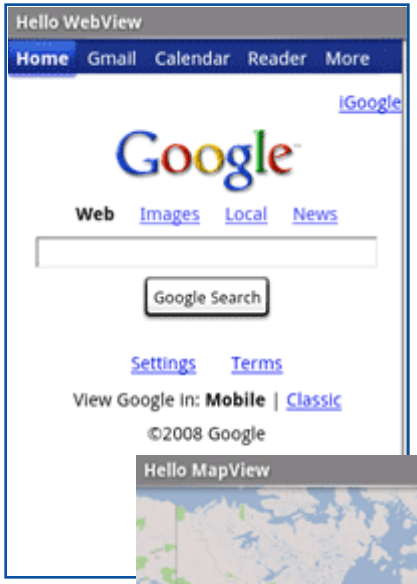
Một số ví dụ về widget khác trong android

AutoCompleteTextView

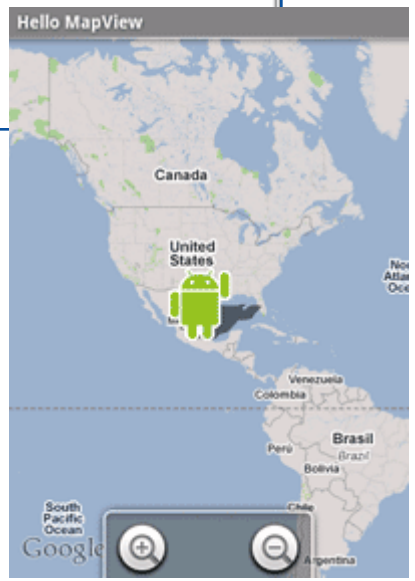


ListView

WebView



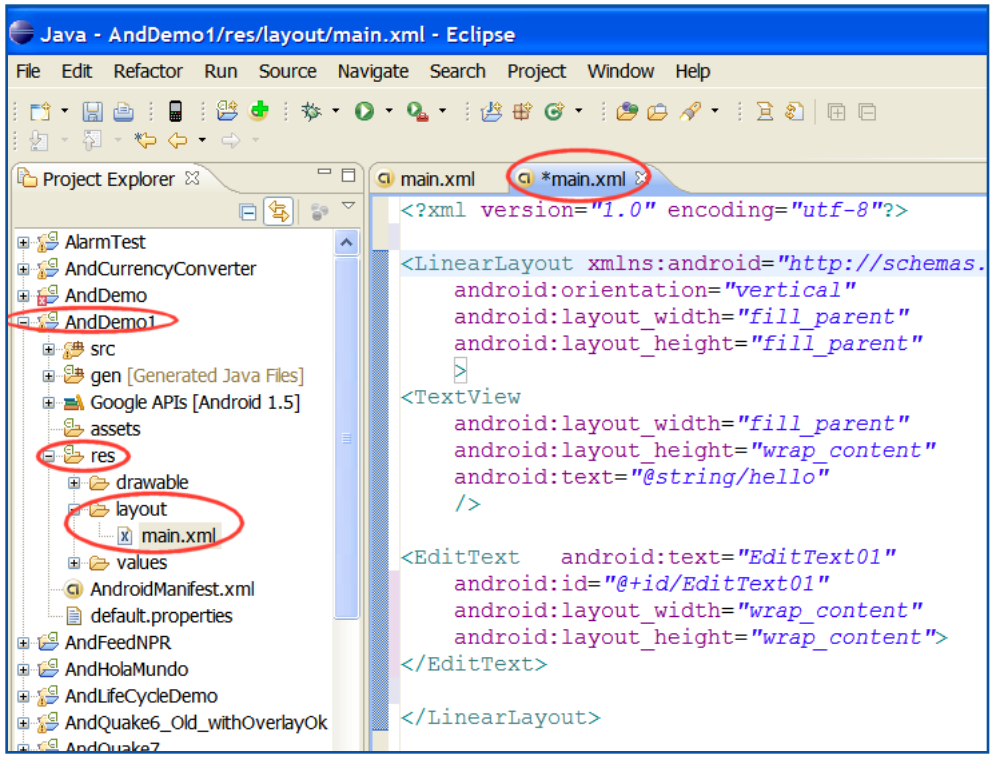
MapView



1. Giới thiệu về Android User Interface

- **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.



1. Giới thiệu về Android User Interface

- 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).

1. Giới thiệu về Android User Interface

- 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.

1. Giới thiệu về Android User Interface

- **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) {
          updateTime();
        }
      });

      private void updateTime() {
        btn.setText(new Date().toString());
      }
```

1. Giới thiệu về Android User Interface

Đơn vị đo

- Db/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→ trên màn hình khác nhau có thể sẽ hiển thị không như ý muốn.

2. Các layout cơ bản

Chi tiết một số layout

- **FrameLayout:**
  - 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, nó không cho phép chúng ta thay đổi vị trí của các control theo một Location nào đó.
  - Các control đưa vào sau nó sẽ đè lên trên và che khuất control trước đó (trừ khi ta thiết lập transparent cho control sau)
  - Xem đoạn cấu trúc XML trang sau: hai hình luôn được "neo" ở góc trái màn hình. Hình pic2 đưa vào sau sẽ đè lên trên hình pic1.

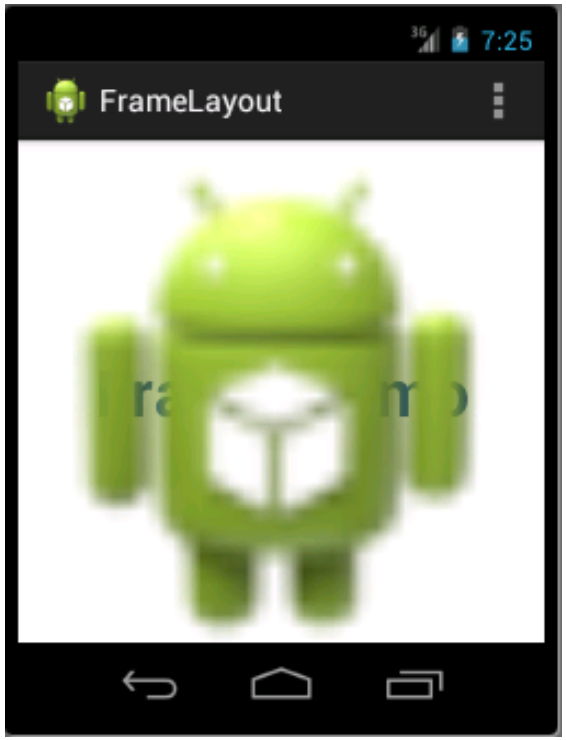
2. Các layout cơ bản

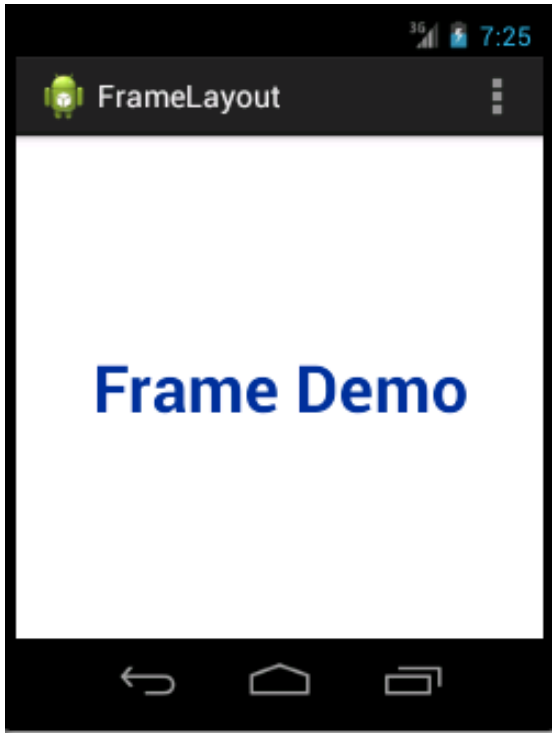
- **FrameLayout (tiếp)**
  - Tệp myframewoklayout.xml

```
<?xml version="1.0" encoding="utf-8"?>
<FrameLayout
        xmlns:android="http://schemas.android.com/apk
                                        /res/android"
         android:layout_width="match_parent"
          android:layout_height="match_parent" >
    <ImageView
        android:id="@+id/imageView1"
         android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:src="@drawable/pic2" />

     <ImageView
         android:id="@+id/imageView2"
        android:layout_width="158dp"
         android:layout_height="258dp"
        android:src="@drawable/pic1" />

</FrameLayout>
```

2. Các layout cơ bản

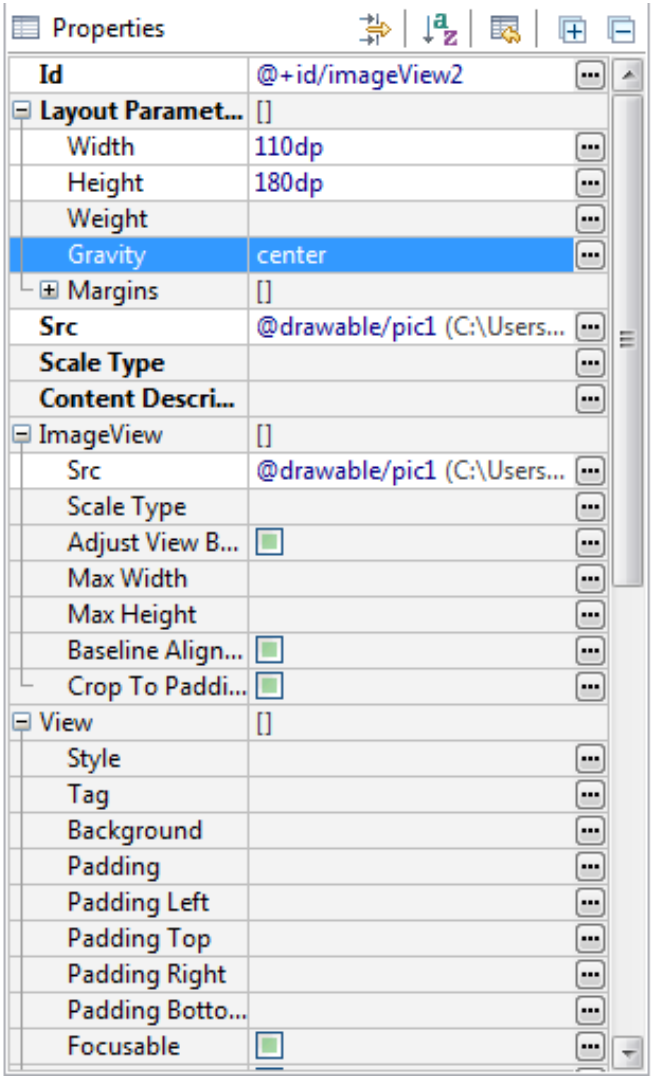
- **Demo → FrameLayout**





2. Các layout cơ bản

- **LinearLayout**
  - LinearLayout cho phép các widgets hoặc containers con sắp xếp theo hàng hoặc cột, theo một tuần tự cái trước cái sau.
  - Có thể dùng: orientation, fill model,padding, margin, gravity, weight để hỗ trợ cho việc thiết kế.
  - Ta có thể dùng Properties hỗ trợ sẵn trong Eclipse để thiết lập các thuộc tính cho control



2. Các layout cơ bản

- **LinearLayout (tiếp)**
- **Orientation:** Trong code (runtime), ta có thể thiết lập orientaion bằng hàm: setOrientaion().
- Trong xml, Layout này cho phép sắp xếp các control theo 2 hướng trên giao diện:
  - Hướng từ trái qua phải: android:orientation="*vertical*“
  - Hướng từ trên xuống dưới: android:orientation="*horizontal*“

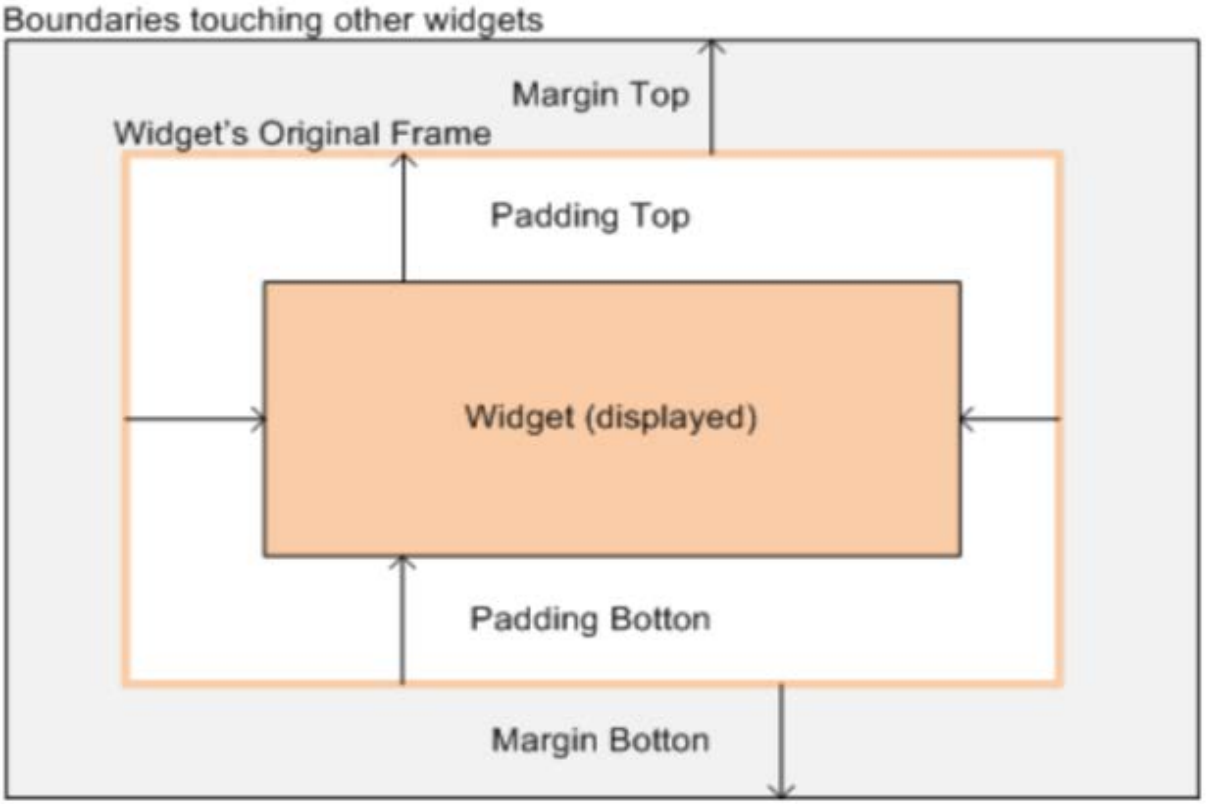
2. Các layout cơ bản

- **LinearLayout (tiếp)**
- **Gravity**
  - Được dùng để bố trí các control trên màn hình.
  - Nếu để mặc định, các widgets sẽ sắp xếp từ trái qua phải, từ trên xuống dưới.
  - Dùng thuộc tính android: layout_gravity =“center” để thiết lập cho control có thể đặt ở các vị trí: left, center, right, top, bottom… trên layou

  - Chú ý phân biệt với android:gravity= “center” : đặt nội dung bên trong control ở giữa

2. Các layout cơ bản

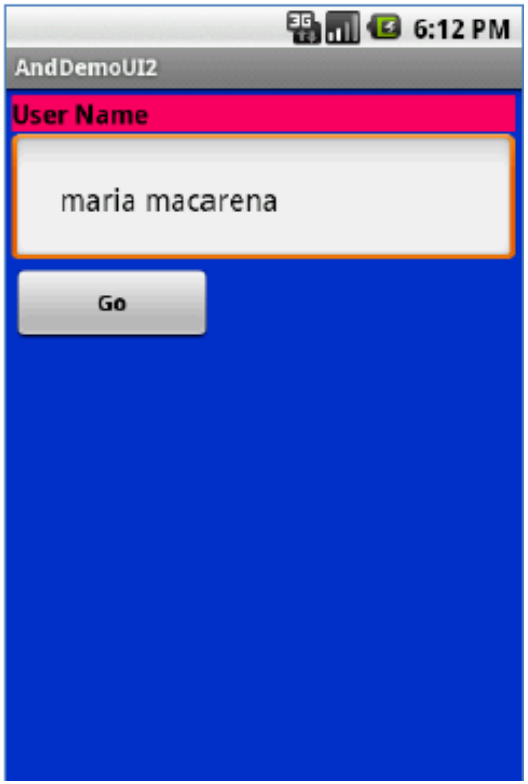
- **LinearLayout (tiếp)**
- Chú ý sự khác biệt giữa **Padding** và **Margin** :



2. Các layout cơ bản

- **LinearLayout (tiếp)**
- Padding (internal spacing – khoảng cách giữa nội dung bên trong so với đường viền của control):

```
<EditText android:id="@+id/ediName"
android:layout_width="fill_parent"
android:layout_height="wrap_content"
android:textSize="18sp"

android:padding="30dip“
 >
</EditText>
...
```



2. Các layout cơ bản

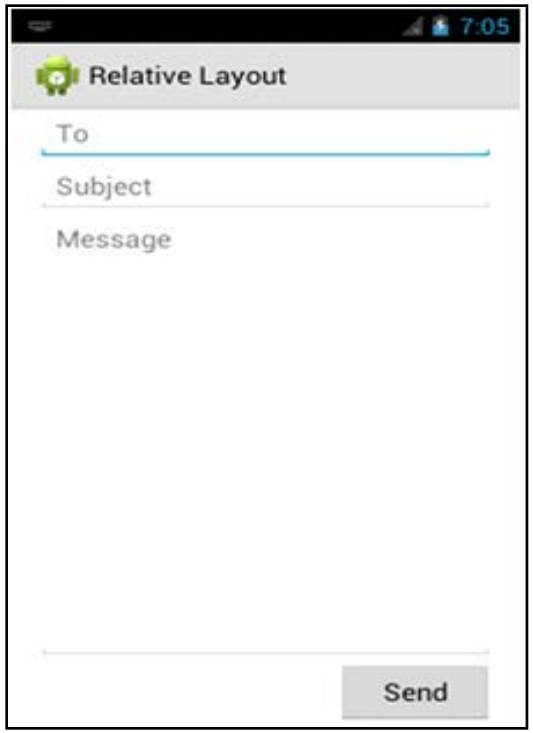
- **LinearLayout (tiếp)**
- Margin (external spacing – khoảng cách giữa control này với control khác):

```
<EditText android:id="@+id/ediName"
      android:layout_width="fill_parent"
      android:layout_height="wrap_content"
      android:textSize="18sp"
       android:layout_margin="6dip"
 >
</EditText>
 ...
```

2. Các layout cơ bản

- **LinearLayout (tiếp)**

```
<LinearLayout
        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"
        android:orientation="vertical" >
        <EditText
                android:layout_width="fill_parent"
                android:layout_height="wrap_content"
                android:hint="@string/to" />
        <EditText
                android:layout_width="fill_parent"
                android:layout_height="wrap_content"
                android:hint="@string/subject" />
        <EditText
                android:layout_width="fill_parent"
                android:layout_height="0dp"
                android:layout_weight="1"
                android:gravity="top"
                android:hint="@string/message" />
        <Button
                android:layout_width="100dp"
                android:layout_height="wrap_content"
                android:layout_gravity="right"
                android:text="@string/send" />
</LinearLayout>
```



2. Các layout cơ bản

- **Demo → LinearLayout**

Tổng kết buổi học

- Giới thiệu về Android UI
- Các Layout cơ bản
  - FrameLayout
  - LinearLayout


Kết thúc!