AntTek

User Interface

Le Duc Bao
bao.le@anttek.com
http://anttek.com

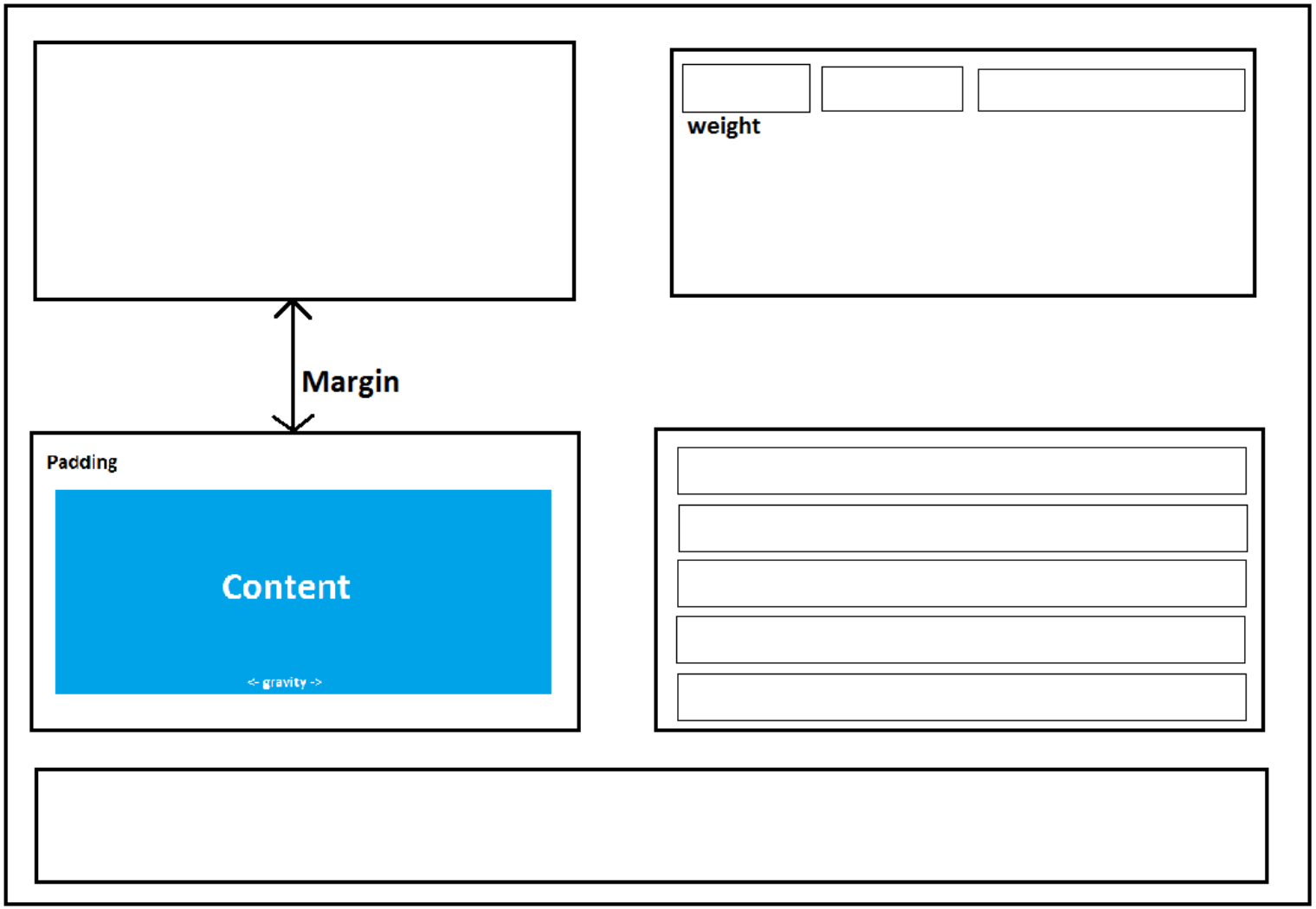
Outline

1. Overview.
2. UI Components
3. Building interface by XML.

Le Duc Bao – bao.le@anttek.com

Overview

Overview



UI Components

- All user interface elements in an Android app are built using **`View`** and **`ViewGroup`** objects.

  - A **`View`** is an object that draws something on the screen that the user can interact with.

    Ex: **`TextView`**, **`Button`**, **`CheckBox`**, **`ListView`**...

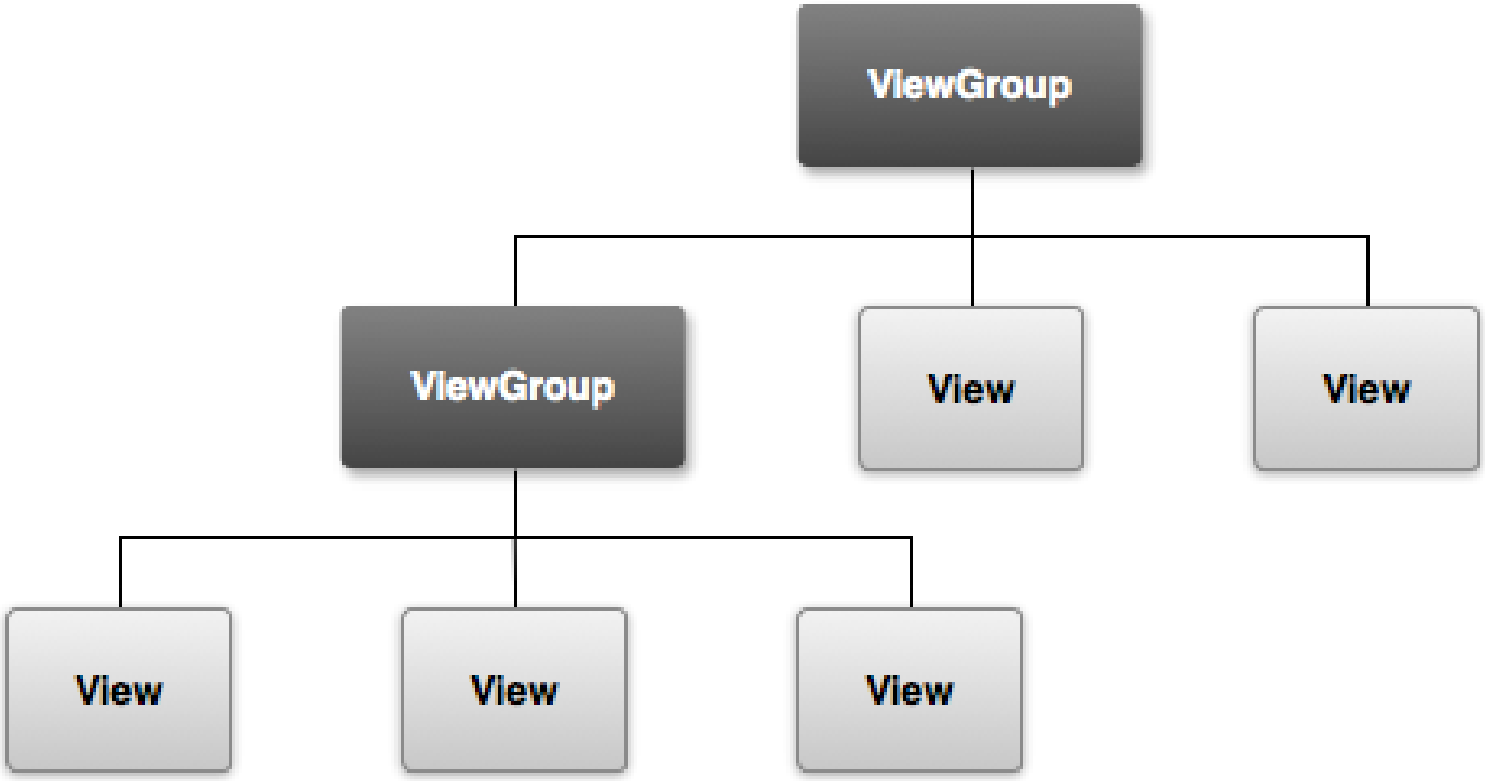
  - A **`ViewGroup`** is an object that holds other **`View`** (and **`ViewGroup`**) objects in order to define the layout of the interface.

    Ex: **`LinearLayout`**, **`FrameLayout`**, **`ListView`**,...

Le Duc Bao – bao.le@anttek.com

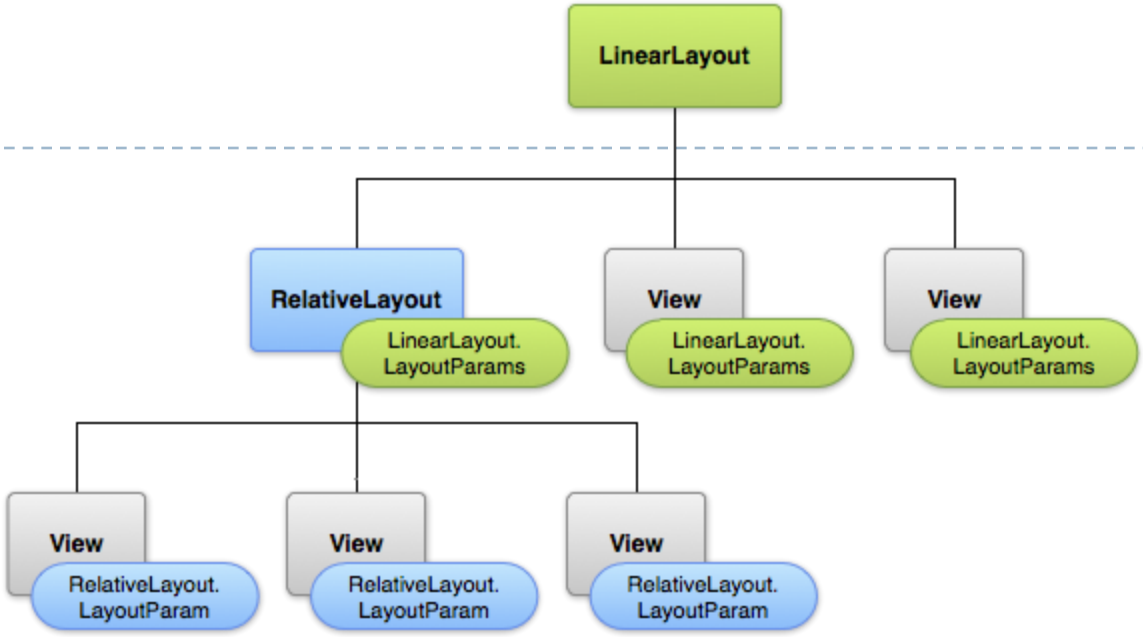
UI Components

- The user interface for each component is defined using a hierarchy of **`View`** and **`ViewGroup`**.

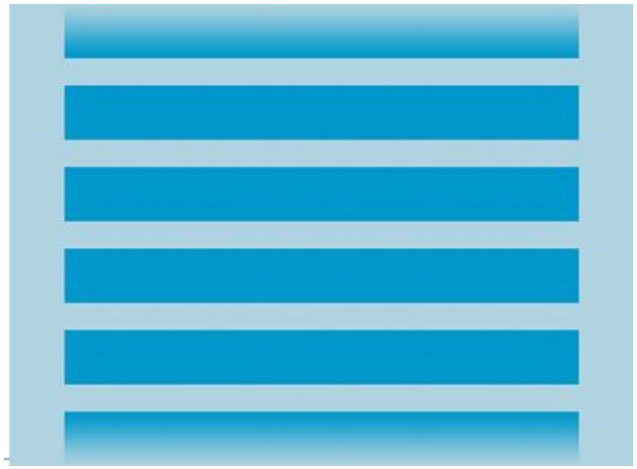


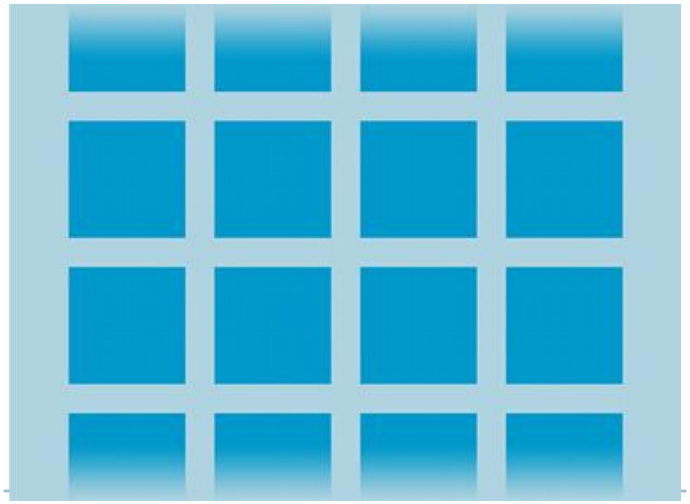
Layout

- Container layout

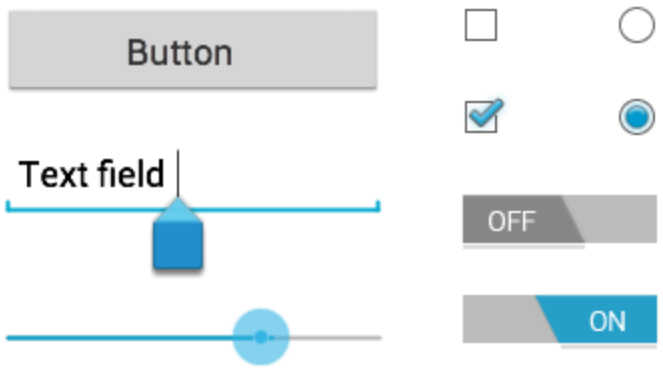


- Listview & GridView





Input Control



Input Events

- onClick()
- onLongClick()
- onTouch()
- onFocusChanged()
- onKey()

Le Duc Bao – bao.le@anttek.com

Layout Parameters

- Every `ViewGroup` class implements a nested class that extends `ViewGroup.LayoutParams`. This subclass contains property types that define the size and position for each child view, as appropriate for the view group.

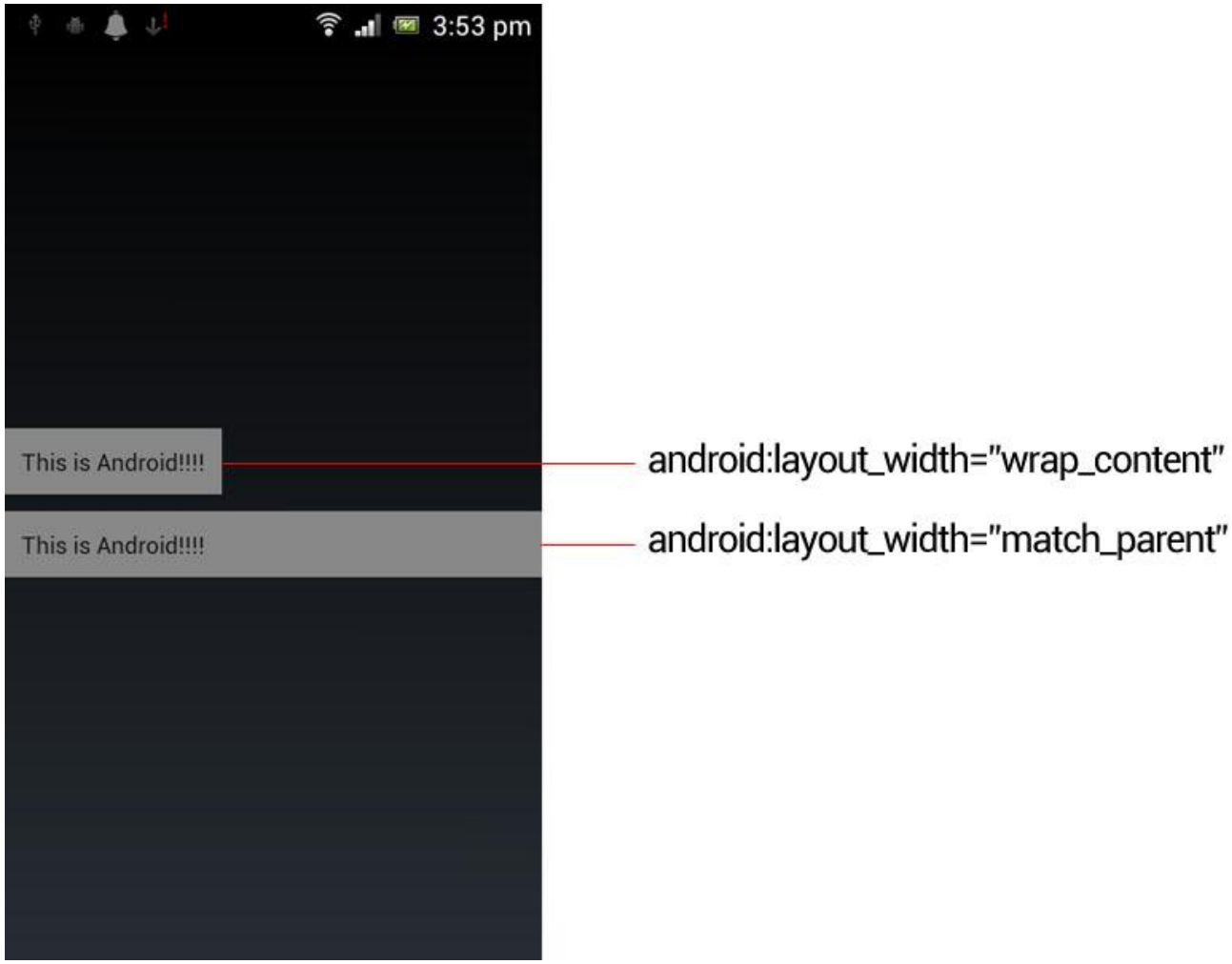
Le Duc Bao – bao.le@anttek.com

Sizing

- The size of a view is expressed with a *width* and a *height*.
- You can specify width and height with exact measurements or use one of these constants to set the width or height:
  - **WRAP_CONTENT** tells the View to size itself to the dimensions required by its content.
  - **MATCH_PARENT** tells the view to become as big as its parent ViewGroup will allow.
- *Weight* indicates how much of the extra space in the LinearLayout.

Le Minh Chanh - chanh.le@anttek.com

Sizing

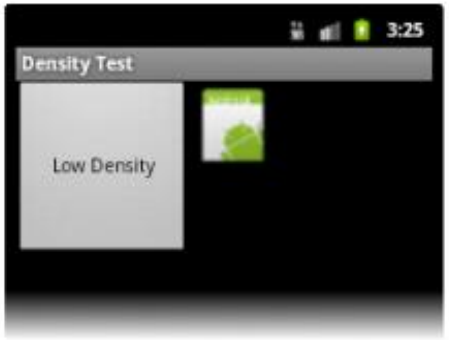

3:53 pm
This is Android!!!!
This is Android!!!!
android:layout_width="wrap_content"
android:layout_width="match_parent"

DP vs PX

- The density-independent pixel (dip or dp) is equivalent to one physical pixel on a 160 dpi screen.
- At runtime, the system transparently handles any scaling of the dp units, as necessary, based on the actual density of the screen in use.
- **Should always use dp units when defining application's UI, to ensure proper display of the UI on screens with different densities.**

Le Duc Bao – bao.le@anttek.com

DP vs PX

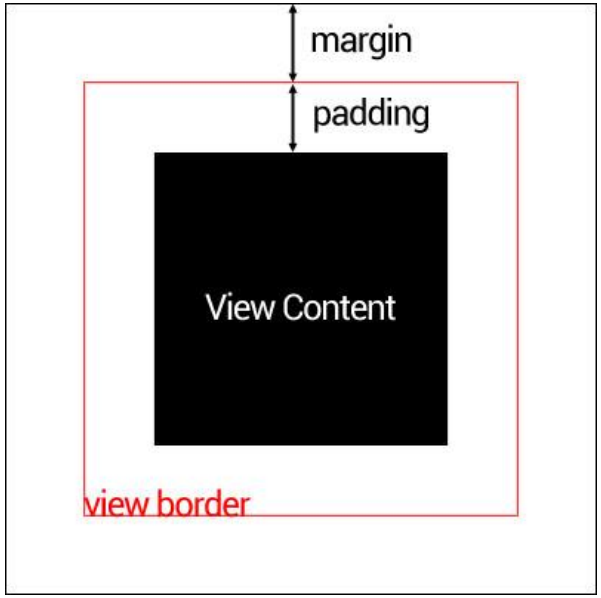


Declare UI using px

Declare UI using dp

Padding & Margin

- Paddings: are the spaces inside the view border, between the border and the actual view contents.
- Margins: are the spaces outside the view border, between the border and the other elements next to this view.



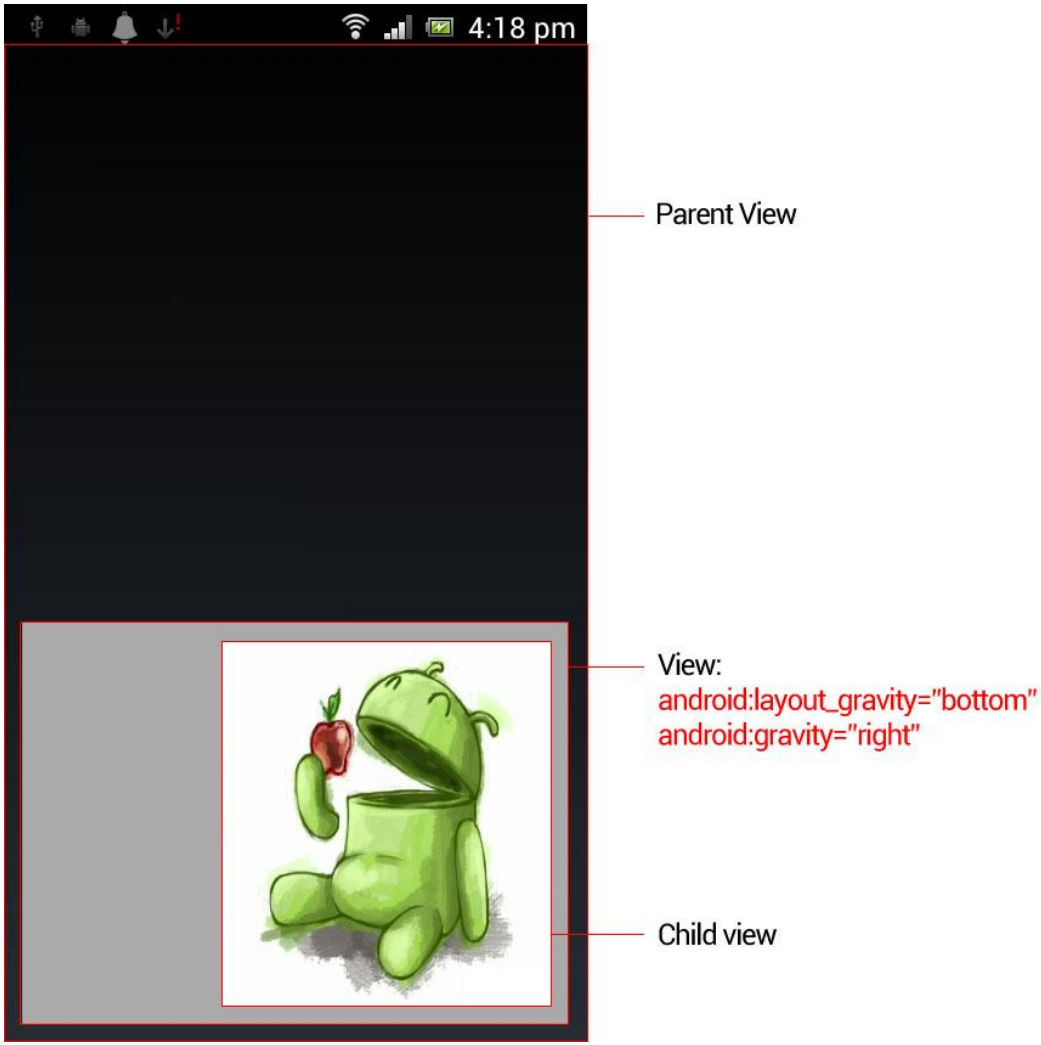
Padding & Margin

Gravity & Layout_Gravity

- Gravity is the gravity of the children inside that view.
- Layout_gravity is the gravity of view inside its parent.

Le Duc Bao – bao.le@anttek.com

Gravity & Layout_Gravity



Building interface by XML

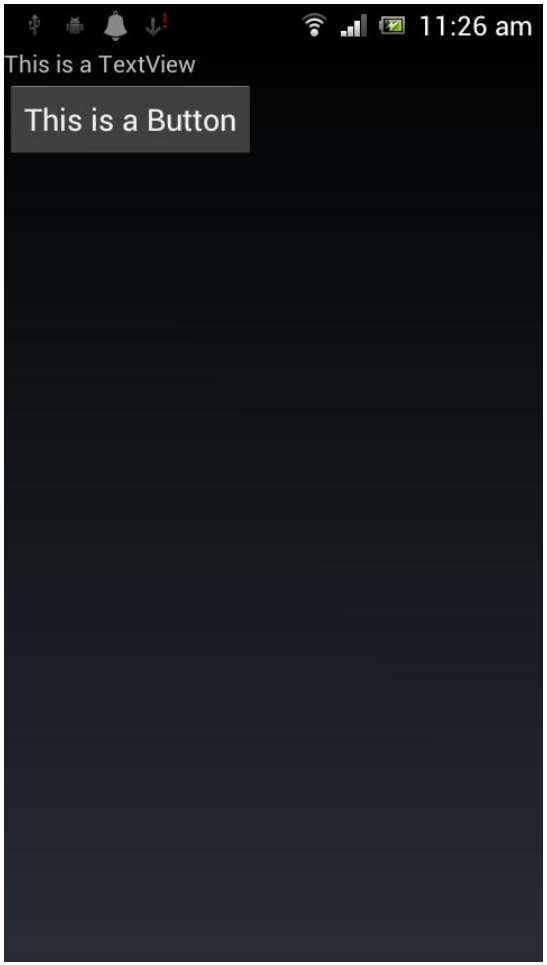
- Android UI can be easily declared by XML.
- Can be modified by code in runtime.
- The advantage to declaring UI in XML is that it enables to separate the presentation of your application from the code that controls its behavior.
- Powerful tool for supporting multiple devices.
- Save layout files in your Android project's `res/layout/` directory.

Building interface by XML

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
  xmlns:android="http://schemas.android.com/apk/res/android"
  android:layout_width="match_parent"
  android:layout_height="match_parent"
  android:orientation="vertical" >

  <TextView android:id="@+id/text"
       android:layout_width="wrap_content"
       android:layout_height="wrap_content"
       android:text="This is a TextView" />

  <Button android:id="@+id/button"
       android:layout_width="wrap_content"
       android:layout_height="wrap_content"
       android:text="This is a Button" />

</LinearLayout>
```


Le Duc Bao – bao.le@anttek.com

Building interface by XML



Building interface by XML

Main Attributes:

- IDs:

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

- Layout Parameters:

```
android:layout_width
android:layout_height
android:layout_margin
android:padding
….
```


Le Duc Bao – bao.le@anttek.com

Building interface by XML

- Define an UI element in XML:

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

- Create an instance of the view object and capture it from the layout:

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

- Handle UI events

```
myButton.setOnClickListener(new OnClickListener() {
    public void onClick(View v) {
      // do something when the button is clicked
    }
};
```


Le Duc Bao – bao.le@anttek.com

Reference

- http://developer.android.com/guide/topics/ui/index.html
- http://developer.android.com/guide/topics/ui/declaring-layout.html
- http://developer.android.com/guide/topics/ui/themes.html
- http://developer.android.com/guide/topics/resources/providing-resources.html
- http://developer.android.com/guide/practices/screens_support.html
- http://mobile.tutsplus.com/series/android-user-interface-design/

Le Duc Bao – bao.le@anttek.com