

SOS in Computer Science and Applications Jiwaji University

Class: MBA (E-commerce) IV semester
Subject: Introduction to Linux & Android
O.S.(402E3)

Topic: (i) Fundamental android UI design
(ii) Drawable resources

Fundamental UI design

The “views” are the building blocks of a U.I design and composes of almost every basic U.I element like TextViews, EditTexts, ImageViews etc. This ‘*view*’ however comes along with a few properties bundled to them. I am gonna talk about the ones which are more important and are often used to build up a complete meaningful screen design.

- **id**
- **width**
- **height**
- **margin**
- **padding**

“id”

This is basically the name of the particular view and will be used to refer that particular view through out the project. It has to be unique(*multiple views referencing to same id will confuse the compiler*). Common ethic to name an id of a view is **“descriptionOfView_viewType”** (For ex. if there is a textview that denotes an email, its id will be **“email_textview”**)

Fundamental UI design

“width” and “height”

As the name of these properties suggest, these are the dimensions of that particular view. Now these dimensions can be set using hard-coded values and it will adopt to them in most layouts, but its not a very good design as the content inside them might get cropped or will have unwanted spaces. Android provides two pre-defined options to set these dimensions — ***“match_parent”*** and ***“wrap_content”***.

“margin”

This is the minimum distance that a view has to maintain from its neighbouring views. That’s it. Since there are four sides to any view, the four margins corresponding to them are ***“margin_left”***, ***“margin_right”***, ***“margin_top”*** and ***“margin_bottom”***. If the same margin is needed to be set on all sides, it can be set directly through ***“margin”*** property.

Fundamental UI design

“padding”

The distance between the view's outline and its content. Again similar to the “margin” property, “padding” too has **“padding_left”**, **“padding_right”**, **“padding_top”**, **“padding_bottom”** and the common padding to all sides can be set through **“padding”** property.

Drawable resources

A drawable resource is a general concept for a graphic that can be drawn to the screen and which you can retrieve with APIs such as [getDrawable\(int\)](#) or apply to another XML resource with attributes such as `android:drawable` and `android:icon`. There are several different types of drawables:

- [Bitmap File](#) A bitmap graphic file (.png, .jpg, or .gif). Creates a [BitmapDrawable](#).
- [Nine-Patch File](#) A PNG file with stretchable regions to allow image resizing based on content (.9.png). Creates a [NinePatchDrawable](#).
- [Layer List](#) A Drawable that manages an array of other Drawables. These are drawn in array order, so the element with the largest index is be drawn on top. Creates a [LayerDrawable](#).
- [State List](#) An XML file that references different bitmap graphics for different states (for example, to use a different image when a button is pressed). Creates a [StateListDrawable](#).

Drawable resources

Level List - An XML file that defines a drawable that manages a number of alternate Drawables, each assigned a maximum numerical value. Creates a [LevelListDrawable](#).

Transition Drawable- An XML file that defines a drawable that can cross-fade between two drawable resources. Creates a [TransitionDrawable](#).

Inset Drawable - An XML file that defines a drawable that insets another drawable by a specified distance. This is useful when a View needs a background drawable that is smaller than the View's actual bounds.

Clip Drawable- An XML file that defines a drawable that clips another Drawable based on this Drawable's current level value. Creates a [ClipDrawable](#).

Scale Drawable- An XML file that defines a drawable that changes the size of another Drawable based on its current level value. Creates a [ScaleDrawableShapeDrawable](#)
An XML file that defines a geometric shape, including colors and gradients. Creates a [ShapeDrawable](#).

Drawable resources

Bitmap

A bitmap image. Android supports bitmap files in a three formats: .png (preferred), .jpg (acceptable), .gif (discouraged). You can reference a bitmap file directly, using the filename as the resource ID, or create an alias resource ID in XML.

Bitmap File

- A bitmap file is a .png, .jpg, or .gif file. Android creates a [Drawable](#) resource for any of these files when you save them in the res/drawable/ directory.
- file location: res/drawable/*filename*.png (.png, .jpg, or .gif)
The filename is used as the resource ID.
- compiled resource datatype: Resource pointer to a [BitmapDrawable](#).
- resource reference:
 - In Java: R.drawable.*filename*
 - In XML: @[package:]drawable/*filename*

Drawable resources

- example: With an image saved at `res/drawable/myimage.png`, this layout XML applies the image to a View:
- `<ImageView`
 - `android:layout_height="wrap_content"`
 - `android:layout_width="wrap_content"`
 - `android:src="@drawable/myimage" />`
- The following application code retrieves the image as a [Drawable](#):
- `Resources res = getResources\(\);`
`Drawable drawable = res.getDrawable(R.drawable.myimage);`

Drawable resources

- **BitmapDrawable**
- extends [Drawable](#)
- [java.lang.Object](#)
- ↳ [android.graphics.drawable.Drawable](#)
- ↳ [android.graphics.drawable.BitmapDrawable](#)

- **Class Overview**
- A Drawable that wraps a bitmap and can be tiled, stretched, or aligned. You can create a BitmapDrawable from a file path, an input stream, through XML inflation, or from a [Bitmap](#) object.
- It can be defined in an XML file with the <bitmap> element. For more information, see the guide to [Drawable Resources](#).
- Also see the [Bitmap](#) class, which handles the management and transformation of raw bitmap graphics, and should be used when drawing to a [Canvas](#).

Drawable resources

- **Public Methods**
- void [draw\(Canvas canvas\)](#) - Draw in its bounds (set via setBounds) respecting optional effects such as alpha (set via setAlpha) and color filter (set via setColorFilter).
- final [Bitmap getBitmap\(\)](#) - Returns the bitmap used by this drawable to render.
- [Int getChangingConfigurations\(\)](#) - Return a mask of the configuration parameters for which this drawable may change, requiring that it be re-created.

Note- All the MBA (E-commerce) Students who wants to learn more on drawable resources can access the following link:

<https://stuff.mit.edu/afs/sipb/project/android/docs/guide/topics/resources/drawable-resource.html>