विशाल जाधव सरांचे

VJTech Academy

Inspiring Your Success...

UNIT-IV Designing User Interface with View

Fundamentals of UI Design(Android View):

- 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. Some of the important and are often used to build up a complete meaningful screen design

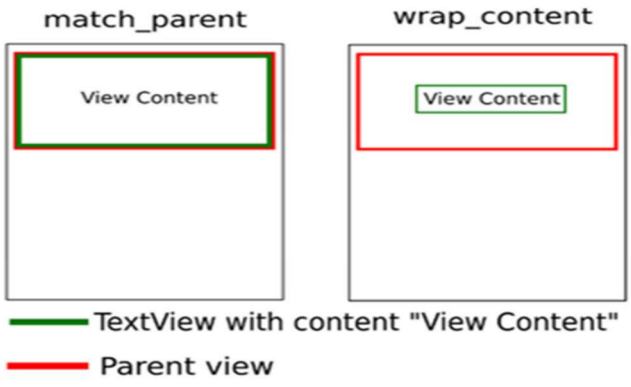
  1. id
  2. width
  3. height
  4. margin
  5. padding

“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).

“width” and “height”

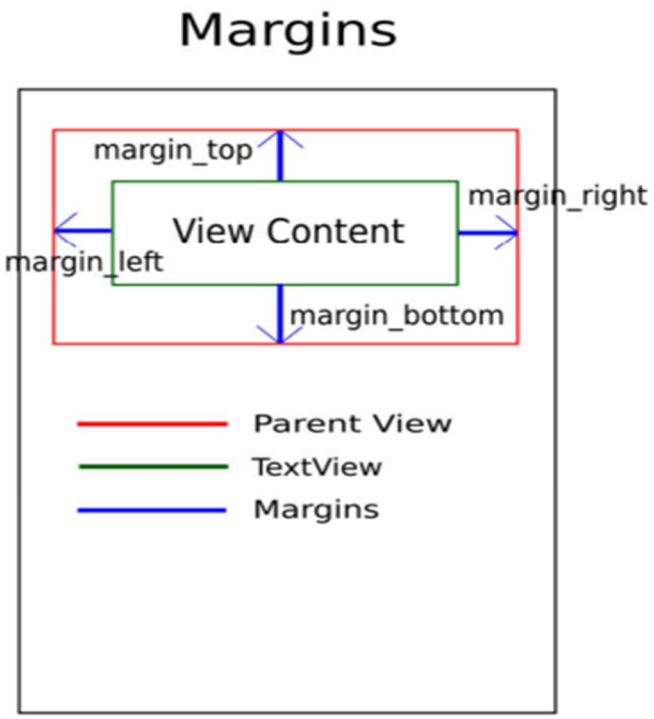
- As the name of these properties suggest, these are the dimensions of that particular view. Now these dimensions can be set using hard-coded values and it will adopt to them in most layouts, but its not a very good design as the content inside them might get cropped or will have unwanted spaces.
- Android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



- Setting the dimensions to “match_parent” will make them equal to those of its parent’s dimensions. If there is no parent to that particular view, it merely sets to the screen’s dimensions (parent of a view is the U.I element in which it is housed in). And setting it to “wrap_content” will force the view to adopt its dimensions according to its content.

“margin”

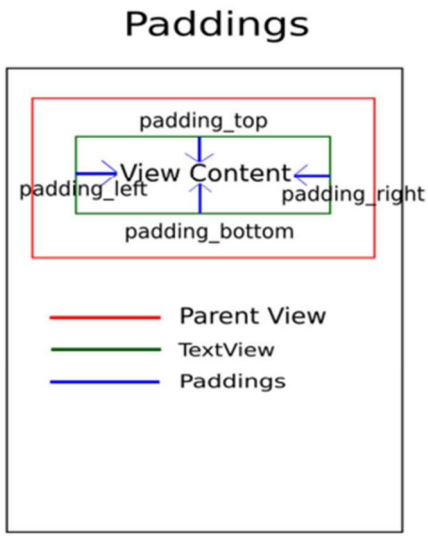
- This is the minimum distance that a view has to maintain from its neighbouring views. That’s it. Since there are four sides to any view, the four margins corresponding to them are “margin_left”, “margin_right”, “margin_top” and “margin_bottom”. If the same margin is needed to be set on all sides, it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



Margins

“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.



Padding

TextView In Android Studio

- In Android, TextView displays text to the user and optionally allows them to edit it programmatically. TextView is a complete text editor, however basic class is configured to not allow editing but we can edit it.

- View is the parent class of TextView. Being a subclass of view the text view component can be used in your app's GUI inside a ViewGroup, or as the content view of an activity. We can create a TextView instance by declaring it inside a layout(XML file) or by instantiating it programmatically(Java Class).

Attributes of TextView:

1. **id:** id is an attribute used to uniquely identify a text view.
2. **gravity:** The gravity attribute is an optional attribute which is used to control the alignment of the text like left, right, center, top, bottom, center_vertical, center_horizontal etc
3. **text:** text attribute is used to set the text in a text view. We can set the text in xml as well as in the java class.
4. **textColor:** textColor attribute is used to set the text color of a text view. Color value is in the form of "#argb", "#rgb", "#rrggbb", or "#aarrggbb".
5. **textSize**: textSize attribute is used to set the size of text of a text view. We can set the text size in sp(scale independent pixel) or dp(density pixel).

6. **textStyle:** textStyle attribute is used to set the text style of a text view. The possible text styles are bold, italic and normal. If we need to use two or more styles for a text view then "|" operator is used for that.
7. **background:** background attribute is used to set the background of a text view. We can set a color or a drawable in the background of a text view.
8. **padding:** padding attribute is used to set the padding from left, right, top or bottom. In above example code of background we also set the 10dp padding from all the side's of text view

Example:

```
<TextView
android:id="@+id/simpleTextView"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:text="AbhiAndroid"
android:layout_centerInParent="true"
android:textSize="40sp"
android:padding="10dp"
android:textColor="#fff"
android:background="#000"/> <!--red color for background of text view-->
```

EditText in Android

- In Android development, the EditText view is a UI element used for user input of text.
- In Android, EditText is a standard entry widget in android apps.
- It is an overlay over TextView that configures itself to be editable.
- EditText is a subclass of TextView with text editing operations.
- We often use EditText in our applications in order to provide an input or text field, especially in forms. The most simple example of EditText is Login or Sign-in form.

Attributes of EditText:

1. **id:** id is an attribute used to uniquely identify a text EditText.
2. **gravity:** The gravity attribute is an optional attribute which is used to control the alignment of the text like left, right, center, top, bottom, center_vertical, center_horizontal etc.
3. **text:** text attribute is used to set the text in a EditText. We can set the text in xml as well as in the java class.
4. **hint:** hint is an attribute used to set the hint i.e., what you want user to enter in this edit text. Whenever user start to type in edit text the hint will automatically disappear
5. **textColor:** textColor attribute is used to set the text color of a text edit text. Color value is in the form of “#argb”, “#rgb”, “#rrggbb”, or “#aarrggbb”.
6. **textSize:** textSize attribute is used to set the size of text of an edit text. We can set the text size in sp(scale independent pixel) or dp(density pixel)
7. **textStyle:** textStyle attribute is used to set the text style of an edit text. The possible text styles are bold, italic and normal. If we need to use two or more styles for a edit text then “|” operator is used for that.
8. **background:** background attribute is used to set the background of an edit text. We can set a color or a drawable in the background of a edit text.
9. **padding:** padding attribute is used to set the padding from left, right, top or bottom. In above example code of background we also set the 10dp padding from all the sides of edit text.
10. **inputType**: It is used to mention what type of value should pass in EditText, like plain text, email, password, etc

Example:

```
<EditText
  android:id = "@+id/editText"
  android:layout_width = "wrap_content"
  android:layout_height = "wrap_content"
  android:inputType = "text" />
```

Button In Android Studio

- In Android, Button represents a push button.
- A Push buttons can be clicked, or pressed by the user to perform an action.
- There are different types of buttons used in android such as CompoundButton, ToggleButton, RadioButton.
- AndroidButton is a subclass of TextView class and compound button is the subclass of Button class. On a button we can perform different actions or events like click event, pressed event, touch event etc.

Attributes of Button in Android:

1. **id:** id is an attribute used to uniquely identify a text Button. Below is the example code in which we set the id of a Button.
2. **gravity:** The gravity attribute is an optional attribute which is used to control the alignment of the text like left, right, center, top, bottom, center_vertical, center_horizontal etc.
3. **text:** text attribute is used to set the text in a Button. We can set the text in xml as well as in the java class.
4. **textColor:** textColor attribute is used to set the text color of a Button. Color value is in the form of “#argb”, “#rgb”, “#rrggbb”, or “#aarrggbb”.
5. **textSize:** textSize attribute is used to set the size of the text on Button. We can set the text size in sp(scale independent pixel) or dp(density pixel).
6. **textStyle:** textStyle attribute is used to set the text style of a Button. The possible text styles are bold, italic and normal. If we need to use two or more styles for a Button then “|” operator is used for that.
7. **background:** background attribute is used to set the background of a Button. We can set a color or a drawable in the background of a Button.
8. **padding:** padding attribute is used to set the padding from left, right, top or bottom. In above example code of background, we also set the 10dp padding from all the sides of button.

Example:

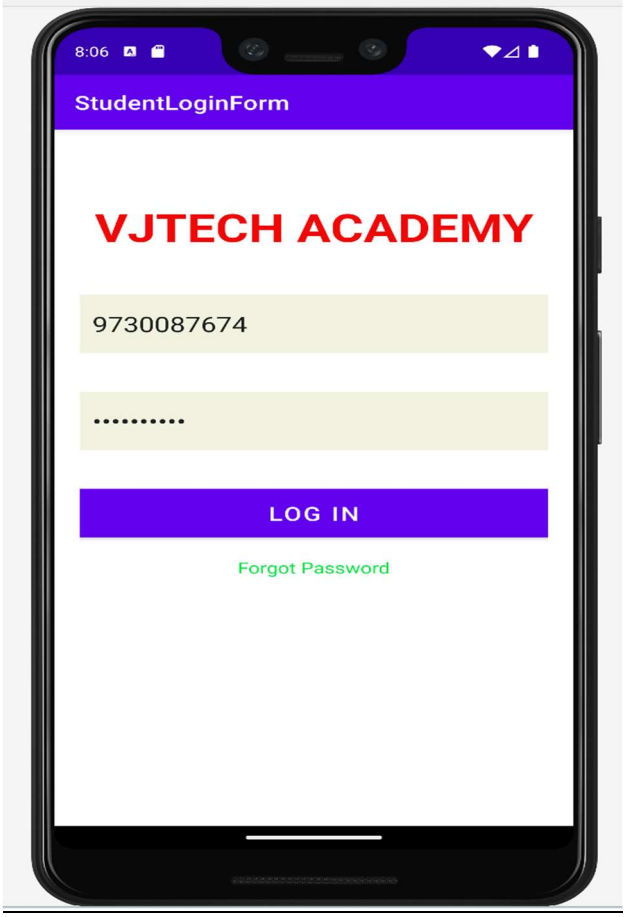
```
<Button
    android:id="@+id/button"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:background="#4CAF50"
    android:text="Submit"
    android:textColor="#fff "
    android:textSize="24sp"/>
```

Program Practice-1(TextView,EditText and Button) : Develop an android application to create a login form for student registration.

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/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"
    android:padding="20dp"
    tools:context=".MainActivity">
    <TextView
        android:id="@+id/tv1"
        android:layout_width="match_parent"
        android:layout_height="80dp"
        android:text="VJTECH ACADEMY"
        android:gravity="center"
        android:textSize="40sp"
        android:textStyle="bold"
        android:layout_marginTop="40dp"
        android:textColor="#EF0A0A"/>
    <EditText
        android:id="@+id/et1"
        android:layout_width="match_parent"
        android:layout_height="60dp"
        android:layout_below="@+id/tv1"
        android:background="#F1F3E0"
        android:hint="Mobile Number"
        android:padding="10dp"
        android:textSize="22sp"
        android:layout_marginTop="30dp"/>
    <EditText
        android:id="@+id/et2"
        android:layout_width="match_parent"
        android:layout_height="60dp"
        android:layout_below="@+id/et1"
        android:background="#F1F3E0"
        android:hint="Password"
        android:padding="10dp"
        android:textSize="22sp"
        android:layout_marginTop="40dp"
```

```
        android:inputType="textPassword"/>
    <Button
        android:id="@+id/b1"
        android:layout_width="match_parent"
        android:layout_height="50dp"
        android:layout_below="@+id/et2"
        android:layout_marginTop="40dp"
        android:textSize="20sp"
        android:text="Log In"
        android:background="#5CCC6E"/>
    <TextView
        android:id="@+id/tv2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Forgot Password"
        android:layout_below="@+id/b1"
        android:layout_marginTop="20dp"
        android:textColor="#0EE447"
        android:textSize="16sp"
        android:layout_centerHorizontal="true"/>
</RelativeLayout>
```

OUTPUT:



ImageButton In Android Studio

- In Android, ImageButton is used to display a normal button with a custom image in a button.
- In simple words we can say, ImageButton is a button with an image that can be pressed or clicked by the users.
- By default, it looks like a normal button with the standard button background that changes the color during different button states.
- An image on the surface of a button is defined within a xml (i.e. layout) by using src attribute or within java class by using setImageResource() method. We can also set an image or custom drawable in the background of the image button.

Attributes of ImageButton:

1. **android: id:** id is an attribute used to uniquely identify a image button.
2. **android:src:** This attribute sets the image that will be displayed on the button.
3. **android:background:** This attribute sets the background color or drawable of the button.
4. **android:scaleType:** This attribute specifies how the image should be scaled to fit within the bounds of the button.
5. **android:padding:** This attribute sets the padding within the button, which is the space between the button's content (the image) and its edges.
6. **android:onClick:** This attribute specifies the method to be called when the button is clicked.
7. **android:contentDescription:** This attribute provides a description of the button's content for accessibility purposes.
8. **android:enabled:** This attribute specifies whether the button is enabled or disabled.
9. **android:layout_gravity:** This attribute specifies how the button should be positioned within its parent layout.
10. **android:layout_width and android:layout_height:** These attributes specify the width and height of the button.

Example:

```
<ImageButton
   android:id="@+id/imageButton"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:src="@drawable/abc"/>
```

Toggle Button In Android Studio

- A toggle button in Android is a UI element that has two states: on and off.
- It's similar to a checkbox, but it has a different appearance and behavior.
- When a toggle button is on, it displays a different color or icon than when it's off, and it typically triggers an action when the user toggles it.
- The most simple example of ToggleButton is doing on/off in sound, Bluetooth, wifi, hotspot etc. It is a subclass of compoundButt
- To check current state of a toggle button programmatically we use isChecked() method. This method returns a Boolean value either true or false. If a toggle button is checked then it returns true otherwise it returns false

Attributes of ToggleButton:

1. **checked:** checked is an attribute of toggle button used to set the current state of a toggle button. The value should be true or false where true shows the checked state and false shows unchecked state of a toggle button. The default value of checked attribute is false.
2. **textOn And textOff:** textOn attribute is used to set the text when toggle button is in checked/on state. We can set the textOn in XML textOn attribute is used to set the text when toggle button is in checked/on state. We can set the textOn in XML as well as in the java class.
3. **textColor:** textColor attribute is used to set the text color of a toggle button. Color value is in the form of “#argb”, “#rgb”, “#rrggbb”, or “#aarrggbb”
4. **textSize:** textSize attribute set the size of the text of a toggle button. We can set the text size in sp(scale independent pixel) or dp(density pixel).
5. **textStyle:** textStyle attribute is used to set the text style of the text of a Toggle button. You can set bold, italic and normal. If you need to use two or more styles for a text view then “|” operator is used for that.
6. **background:** background attribute is used to set the background of a toggle button. We can set a color or a drawable in the background of a toggle button.
7. **padding:** padding attribute is used to set the padding from left, right, top or bottom.
8. **drawableBottom, drawableTop, drawableRight And drawableLeft:** These attribute draw the drawable below, top, right and left of the text of ToggleButton

Example:

```
<ToggleButton
        android:id="@+id/toggle_button"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textOn="On"
         android:textOff="Off" />
```

RadioButton & RadioGroup In Android Studio

- In Android, RadioButtonIn Android, RadioButton are mainly used together in a RadioGroup.
- In RadioGroup RadioGroup checking the one radio button RadioGroup checking the one radio button out of several radio button added in it will automatically unchecked all the others.
- It means at one time we can checked only one radio buttonmeans at one time we can checked only one radio button from a group of radio buttons which belong to same radio group.
- RadioButon is a two state button that can be checked or unchecked.
- If a radio button is unchecked then a user can check it by simply clicking on it.
- Once a RadiaButton is checked by user it can't be unchecked by simply pressing on the same button.
- It will automatically unchecked when you press any other RadioButtonwill automatically unchecked when you press any other RadioButton within same RadioGroup.
- . RadioGroup is a widget used in Android for the grouping of radio buttons and provide the feature of selecting only one radio button from the set.

Attributes:

1. **id:** id is an attribute used to uniquely identify a radio button.
2. **checked:** checked attribute in radio button is used to set the current state of a radio button. We can set it either true or false where true shows the checked state and false shows unchecked state of a radio button. As usual default value of checked attribute is false. We can also set the current state in JAVA
3. **text:** text attribute is used to set the text in a radio button. We can set the text both ways either in XML text attribute is used to set the text in a radio button. We can set the text both ways either in XML or in JAVA class.
4. **gravity:** The gravity attribute is an optional attribute which is used to control the alignment of text like left, right, center, top, bottom, center_vertical, center_horizontal
5. **textColor:** textColor attribute is used to set the text color of a radio button. Color value is in the form of "#argb", "#rgb", "#rrggbb", or "#aarrggbb"
6. **textSize:** textSize attribute set the size of the text of a toggle button. We can set the text size in sp(scale independent pixel) or dp(density pixel).

7. **textStyle:** textStyle attribute is used to set the text style of the text of a Toggle button. You can set bold, italic and normal. If you need to use two or more styles for a text view then "|" operator is used for that.
8. **background:** background attribute is used to set the background of a toggle button. We can set a color or a drawable in the background of a toggle button.
9. **padding:** padding attribute is used to set the padding from left, right, top or bottom.
10. **drawableBottom, drawableTop, drawableRight And drawableLeft:** These attribute draw the drawable below, top, right and left of the text of ToggleButton

Example:

```
<RadioGroup
 android:id="@+id/radioGroup"
 android:layout_width="wrap_content"
 android:layout_height="wrap_content"
 android:orientation="vertical">
    <RadioButton
         android:id="@+id/rb1"
         android:layout_width="wrap_content"
         android:layout_height="wrap_content"
         android:text="Male" />
   <RadioButton
         android:id="@+id/rb2"
         android:layout_width="wrap_content"
         android:layout_height="wrap_content"
         android:text="Female" />
</RadioGroup>
```

Checkbox In Android Studio

- In Android, CheckBoxIn Android, CheckBox is a type of two state button either unchecked or checked in Android
- It is a type of on/off switch that can be toggled by the users.
- You should use checkbox when presenting a group of selectable options to users that are not mutually exclusive.
- CompoundButton is the parent class of CheckBox class.

Attributes:

1. **id:** id is an attribute used to uniquely identify a radio button.
2. **checked** : checked is an attribute of check box used to set the current state of a check box. The value should be true or false where true shows the checked state and false shows unchecked state of a check box. The default value of checked attribute is false. We can also set the current state programmatically.
3. **text:** text attribute is used to set the text in a radio button. We can set the text both ways either in XML text attribute is used to set the text in a radio button. We can set the text both ways either in XML or in JAVA class.
4. **gravity:** The gravity attribute is an optional attribute which is used to control the alignment of text like left, right, center, top, bottom, center_vertical, center_horizontal
5. **textColor:** textColor attribute is used to set the text color of a radio button. Color value is in the form of “#argb”, “#rgb”, “#rrggbb”, or “#aarrggbb”
6. **textSize:** textSize attribute set the size of the text of a toggle button. We can set the text size in sp(scale independent pixel) or dp(density pixel).
7. **textStyle:** textStyle attribute is used to set the text style of the text of a Toggle button. You can set bold, italic and normal. If you need to use two or more styles for a text view then “|” operator is used for that.
8. **background:** background attribute is used to set the background of a toggle button. We can set a color or a drawable in the background of a toggle button.
9. **padding:** padding attribute is used to set the padding from left, right, top or bottom.

Example:

```
<CheckBox
  android:id="@+id/c1"
  android:layout_width="wrap_content"
  android:layout_height="wrap_content"
  android:text="Simple CheckBox"/>
```

Progressbar In Android Studio

- ProgressBar is used to display the status of work being done like analyzing status of work or downloading a file etc.
- In Android, by default a progress bar will be displayed as a spinning wheel but If we want it to be displayed as a horizontal bar then we need to use style attribute as horizontal.
- It mainly use the “android.widget.ProgressBar” class.
- To add a progress bar to a layout (xmlTo add a progress bar to a layout (xml) file, you can use the element.
- By default, a progress bar is a spinning wheel (an indeterminate indicator). To change to a horizontal progress bar, apply the progress bar’s horizontal style.

Attributes:

1. **id:** id is an attribute used to uniquely identify a Progress bar.
2. **max:** max is an attribute used in android to define maximum value of the progress can take. It must be an integer value like 100, 200 etc
3. **progress:** progress is an attribute used in android to define the default progress value between 0 and max. It must be an integer value.
4. **progressDrawable:** progress drawable is an attribute used in Android to set the custom drawable for the progress mode.
5. **background:** background attribute is used to set the background of a Progress bar. We can set a color or a drawable in the background of a Progress bar
6. **padding:** padding attribute is used to set the padding from left, right, top or bottom of ProgressBar.
7. **indeterminate:** indeterminate attribute is used in Android to enable the indeterminate mode. In this mode a progress bar shows a cyclic animation without an indication of progress. This mode is used in application when we don’t know the amount of work to be done. In this mode the actual working will not be shown.

Important Methods Used In ProgressBar

- **getMax() –** returns the maximum value of progress bar We can get the maximum value of the progress bar in java class. This method returns a integer value.
- **getProgress() –** returns current progress value We can get the current progress value from a progress bar in java class. This method also returns a integer value.

Example:

```
<ProgressBar
android:id="@+id/simpleProgressBar"
android:layout_width="fill_parent"
android:layout_height="wrap_content"
```

```
android:max="100"
style="@style/Widget.AppCompat.ProgressBar.Ho
rizontal"
android:progress="50"/>
```

ListView In Android Studio

- List of scrollable items can be displayed in Android using ListView.
- It helps you to displaying the data in the form of a scrollable list. Users can then select any list item by clicking on it.
- ListView is default scrollable so we do not need to use scroll View or anything else with ListView.
- ListView is widely used in android applications.
- Example: A very common example of ListView is your phone contact book, where you have a list of your contacts displayed in a ListView and if you click on it then user information is displayed.
- Adapter: To fill the data in a ListView we simply use adapters. List items are automatically inserted to a list using an Adapter that pulls the content from a source such as an arraylist, array or database.
- In android commonly used adapters are: Array Adapter, Base Adapter, CursorAdapter, SimpleCursorAdapter, SpinnerAdapter WrapperListAdapter.
- Listview is present inside Containers. From there you can drag and drop on virtual mobile screen to create it.
- Alternatively you can also XML code to create.

Attributes:

1. **id:** id is used to uniquely identify a ListView.
2. **divider:** This is a drawable or color to draw between different list items.
3. **dividerHeight:** This specify the height of the divider between list items. This could be in dp(density pixel),sp(scale independent pixel) or px(pixel).
4. **listSelector:** listSelector property is used to set the selector of the listView. It is generally orange or Sky blue color mostly but you can also define your custom color or an image as a list selector as per your desi

Example:

```
// Define an array of items to display in the ListView
String[] items = {"Item 1", "Item 2", "Item 3", "Item 4", "Item 5"};

// Create an ArrayAdapter to provide data to the ListView
```

```
ArrayAdapter<String> adapter = new ArrayAdapter<String>(this,

android.R.layout.simple_list_item_1, items);
```

// Create a ListView object in the XML layout file and set its adapter to the ArrayAdapter

```
ListView listView = findViewById(R.id.list_view);

listView.setAdapter(adapter);
```

GridView In Android Studio

- In android GridView is a view group that display items in two dimensional scrolling grid (rows and columns), the grid items are not necessarily predetermined but they are automatically inserted to the layout using a ListAdapter.
- Users can then select any grid item by clicking on it.
- GridView is default scrollable so we don't need to use ScrollView is default scrollable so we don't need to use ScrollView or anything else with GridVie
- GridView is widely used in android applications. An example of GridView is your default Gallery, where you have number of images displayed using grid.
- Adapter Is Used To Fill Data In Gridview: To fill the data in a GridView we simply use adapter To fill the data in a GridView we simply use adapter and grid items are automatically inserted to a GridView using an Adapter To fill the data in a GridView we simply use adapter and grid items are automatically inserted to a GridView using an Adapter which pulls the content from a source such as an arraylist, array or database
- Important Note: numColumns property has to be specified otherwise GridView behaves like a ListView with just singleChoice.

Attributes:

1. **id:** id is used to uniquely identify a GridView
2. numColumns: numColumn define how many columns to show. It may be a integer value, such as "5" or auto_fit (auto_fit is used to display as many columns as possible to fill the available space on the screen)
3. **horizontalSpacing:** horizontalSpacing property is used to define the default horizontal spacing between columns. This could be in pixel(px),density pixel(dp) or scale independent pixel(sp).
4. **verticalSpacing:** verticalSpacing property used to define the default vertical spacing between rows. This should be in px, dp or sp.
5. **columnWidth:** columnWidth property specifies the fixed width of each column. This could be in px, dp or sp.
6. **listSelector** property which define color for selected item.

Example:

```
<GridView
  android:id="@+id/simpleGridView"
  android:layout_width="fill_parent"
  android:layout_height="wrap_content"
  android:numColumns="3"
  android:columnWidth="80dp"
  android:listSelector="#0f0"/>
```

ImageView In Android Studio

- In Android, ImageView class is used to display an image file in application.
- ImageView comes with different configuration options to support different scale types. Scale type options are used for scaling the bounds of an image to the bounds of the imageview. Some of them scaleTypes configuration properties are center, center_crop, fit_xy, fitStart etc.
-

Attributes:

1. **id:** id is an attribute used to uniquely identify a image view id is an attribute used to uniquely identify a image view in android.
2. **src:** src is an attribute used to set a source file or you can say image in your imageview to make your layout attractive.
3. **background:** background attribute is used to set the background of a ImageView. We can set a color or a drawable in the background of a ImageView.
4. **padding:** padding attribute is used to set the padding from left, right, top or bottom of the Imageview.
   - paddingRight: set the padding from the right side of the image view.
   - paddingLeft: set the padding from the left side of the image view.
   - paddingTop: set the padding from the top side of the image view.
   - paddingBottom: set the padding from the bottom side of the image view.
   - padding: set the padding from the all sides of the image
5. **scaleType:** scaleType is an attribute used to control how the image should be re-sized or moved to match the size of this image view. The value for scale type attribute can be fit_xy, center_crop, fitStart etc.

Example:

```
<ImageView
  android:id="@+id/img"
  android:layout_width="fill_parent"
  android:layout_height="wrap_content"
  android:src="@drawable/lion />
```

ScrollView In Android Studio

- In android ScrollView can hold only one direct child.
- This means that, if you have complex layout with more views(Buttons, TextViews or any other view) then you must enclose them inside another standard layout like Table Layout, Relative Layout or Linear Layout.
- You can specify layout_width and layout_height to adjust width and height of screen.
- You can specify height and width in dp(density pixel) or px(pixel). Then after enclosing them in a standard layout, enclose the whole layout in ScrollView to make all the element or views scrollable.
- We never use a Scroll View with a ListView because List View is default scrollable(i.e. vertical scrollable).
- In android default ScrollView is used to scroll the items in vertical direction and if you want to scroll the items horizontally then you need to implement horizontal ScrollView.
- ScrollView Syntax:

```
<ScrollView
android:id="@+id/scrollView"
android:layout_width="fill_parent"
android:layout_height="fill_parent">
<!-- add child view's here -->
</ScrollView>
```

- **Horizontal ScrollView:** In android, You can scroll the elements or views in both vertical and horizontal directions. To scroll in Vertical we simply use ScrollView as we shown in the previous code and to scroll in horizontal direction we need to use HorizontalScrollview.
- Below is HorizontalScrollView syntax in Android is:

```
<HorizontalScrollView
android:id="@+id/horizontalscrollView"
android:layout_width="fill_parent"
android:layout_height="fill_parent">

<-- add child view's here -->

</HorizontalScrollView >
```

-
- In Android, ImageView class is used to display an image file in application.

Attributes:

ScrollView and HorizontalScrollView has same attributes, the only difference is scrollView scroll the child items in vertical direction while horizontal scroll view scroll the child items in horizontal direction.

1. **id:** In android, id attribute is used to uniquely identify a ScrollView.
2. **scrollbars:** In android, scrollbars attribute is used to show the scrollbars in horizontal or vertical direction. The possible Value of scrollbars is vertical, horizontal or none. By default scrollbars is shown in vertical direction in scrollView and in horizontal direction in HorizontalScrollView.
3. **src:** src is an attribute used to set a source file or you can say image in your imageview to make your layout attractive.

Toast In Android Studio

- In Android, Toast is used to display information for a period of time.
- It contains a message to be displayed quickly and disappears after specified period of time.
- It does not block the user interaction.
- Toast is a subclass of Object class.
- In this we use two constants for setting the duration for the Toast. Toast notification in android always appears near the bottom of the screen.
- In Android, Toast is used when we required to notify user about an operation without expecting any user input. It displays a small popup for message and automatically goes out after timeout.

Important Method:

1. **makeText(Context context, CharSequence text, int duration):** This method is used to initiate the Toast. This method take three parameters First is for the application Context, Second is text message and last one is duration for the Toast.

Constants of Toast: Below is the constants of Toast that are used for setting the duration for the Toast.

- LENGTH_LONG: It is used to display the Toast for a long period of time. When we set this duration the Toast will be displayed for a long duration.
- LENGTH_SHORT: It is used to display the Toast for short period of time. When we set this duration the Toast will be displayed for short duration.

2. **show():** This method is used to display the Toast on the screen. This method is display the text which we create using makeText() method of Toast.
3. **setText(CharSequence s):** This method is used to set the text for the Toast. If we use makeText() method and then we want to change the text value for the Toast then we use this method.

4. **setDuration(int duration):** This method is used to set the duration for the Toast. If we use makeText() method and then we want to change the duration for the Toast then we use this method.

Example:

```
<ImageView
  android:id="@+id/img"
  android:layout_width="fill_parent"
  android:layout_height="wrap_content"
  android:src="@drawable/lion />
```

TimePicker In Android Studio

- In Android, TimePicker is a widget used for selecting the time of the day in either AM/PM mode or 24 hours mode.
- The displayed time consist of hours, minutes and clock format. If we need to show this view as a Dialog then we have to use a TimePickerDialog class.

Important Method:

1. **setCurrentHour(Integer currentHour):** This method is used to set the current hours in a time picker.
2. **setCurrentMinute(Integer currentMinute):** This method is used to set the current minutes in a time picker.
3. **getCurrentHour():** This method is used to get the current hours from a time picker.
4. **getCurrentMinute():** This method is used to get the current minutes from a time picker.
5. **setIs24HourView(Boolean is24HourView):** This method is used to set the mode of the Time picker either 24 hour mode or AM/PM mode. In this method we set a Boolean value either true or false. True value indicate 24 hour mode and false value indicate AM/PM mode.

6. **is24HourView():**This method is used to check the current mode of the time picker. This method returns true if its 24 hour mode or false if AM/PM mode is set.

Attributes:

1. **id:** id is an attribute used to uniquely identify a time picker
2. **timePickerMode:** time picker mode is an attribute of time picker used to set the mode either spinner or clock. Default mode is clock but this mode is no longer used after api level 21, so from api level 21 you have to set the mode to spinner.
3. **background:** background attribute is used to set the background of a time picker. We can set a color or a drawable image in the background. We can also set the background color programmatically means in java class.
4. **padding:** padding attribute is used to set the padding from left, right, top or bottom for a time picker.

Example:

- In your layout XML file, add the following code to include the TimePicker widget:

```
<TimePicker
    android:id="@+id/timePicker"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content" />
```

- In your activity or fragment, find the TimePicker widget by its ID:

```
TimePicker timePicker = findViewById(R.id.timePicker);
```

- You can set an initial time for the TimePicker using the setCurrentHour and setCurrentMinute methods:

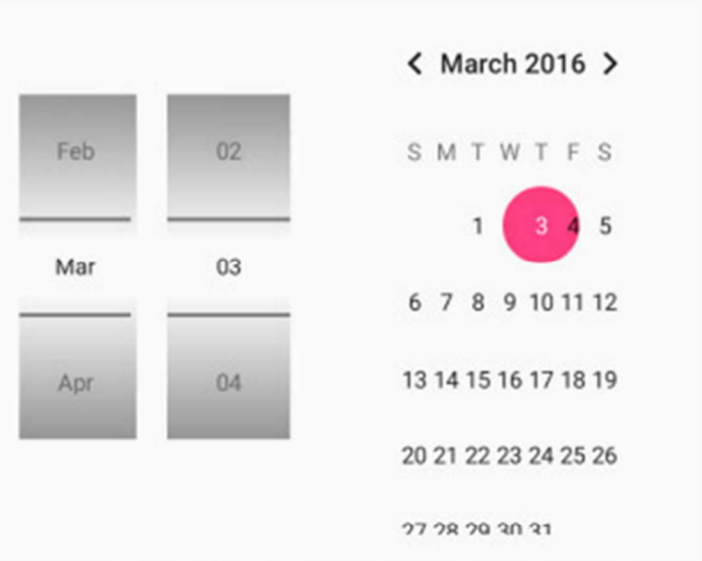
```
timePicker.setCurrentHour(12); // set initial hour to 12
timePicker.setCurrentMinute(0); // set initial minute to 0
```

- To get the selected time from the TimePicker, you can use the getHour and getMinute methods:

```
int hour = timePicker.getHour();
int minute = timePicker.getMinute();
```

DatePicker In Android Studio

- In Android, DatePicker is a widget used to select a date. It allows to select date by day, month and year in your custom UI (user interface).
- If we need to show this view as a dialog then we have to use a DatePickerDialog class. For selecting time Android also provides timepicker to select time.



Important Method:

1. **setSpinnersShown(boolean shown):** This method is used to set whether the spinner of the date picker in shown or not. In this method you have to set a Boolean value either true or false. True indicates spinner is shown, false value indicates spinner is not shown. Default value for this function is true.
2. **getDayOfMonth():** This method is used to get the selected day of the month from a date picker. This method returns an integer value.
3. **getMonth():** This method is used to get the selected month from a date picker. This method returns an integer value.
4. **getYear():** This method is used to get the selected year from a date picker. This method returns an integer value.
5. **getFirstDayOfWeek():** This method is used to get the first day of the week. This method returns an integer value.

Attributes:

1. **id:** id is an attribute used to uniquely identify a date picker
2. **datePickerMode:** This attribute is used to set the Date Picker in mode either spinner or calendar. Default mode is calendar but this mode is not used after api level 21, so from api level 21 you have to set the mode to spinner.
3. **background:** background attribute is used to set the background of a date picker. We can set a color or a drawable image in the background. Below we set the red color for the background of a date picker.
4. **padding:** padding attribute is used to set the padding from left, right, top or bottom for a date picker.

Example:

```
<DatePicker
android:id="@+id/simpleDatePicker"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:datePickerMode="spinner"/>
```