Android View and ViewGroup with Examples

In android, **Layout** is used to define the user interface for an app or activity and it will hold the UI elements that will appear to the user.

The user interface in an android app is made with a collection of `View` and `ViewGroup` objects. Generally, the android apps will contain one or more activities and each activity is a one screen of app. The activities will contain a multiple UI components and those UI components are the instances of `View` and `ViewGroup` subclasses.

The user interface in an android app is made with a collection of `View` and `ViewGroup` objects. Generally, the android apps will contain one or more activities and each activity is a one screen of the app. The activities will contain multiple UI components and those UI components are the instances of `View` and `ViewGroup` subclasses.

Android View

The `View` is a base class for all UI components in android. For example, the **EditText** class is used to accept the input from users in android apps, which is a subclass of `View`.

Following are the some of common `View` subclasses that will be used in android applications.

- TextView
- EditText
- Button
- CheckBox
- RadioButton
- ImageButton
- Progress Bar
- Spinner

Like these we have many `View` subclasses available in android.

Android ViewGroup

The `ViewGroup` is a subclass of `View` and it will act as a base class for **layouts** and **layouts parameters**. The `ViewGroup` will provide an invisible containers to hold other **Views** or **ViewGroups** and to define the layout properties.

For example, Linear Layout is the `ViewGroup` that contains a UI controls like button, textview, etc. and other layouts also.

Following are the commonly used `ViewGroup` subclasses in android applications.

- Linear Layout
- Relative Layout
- Table Layout
- Frame Layout
- Web View
- List View
- Grid View

Both `View` and `ViewGroup` subclasses together will play a key role to create a layouts in android applications.

Android UI Layouts (Linear, Relative, Frame, Table, ListView, GridView, WebView)

In android, **Layout** is used to define the user interface for an app or activity and it will hold the UI elements that will appear to the user.

The user interface in the android app is made with a collection of `View` and `ViewGroup` objects. Generally, the android apps will contain one or more activities and each activity is one screen of the app. The activities will contain multiple UI components and those UI components are the instances of `View` and `ViewGroup` subclasses.

The `View` is a base class for all UI components in android and it is used to create an interactive UI components such as TextView, EditText, Checkbox, Radio Button, etc. and it responsible for event handling and drawing.

The `ViewGroup` is a subclass of `View` and it will act as a base class for **layouts** and **layouts parameters**. The `ViewGroup` will provide an invisible containers to hold other Views or ViewGroups and to define the layout properties.

To know more about View and ViewGroup in android applications, check this Android View and ViewGroup.

In android, we can define a layouts in two ways, those are

- Declare UI elements in XML
- Instantiate layout elements at runtime

The android framework will allow us to use either or both of these methods to define our application's UI.

Declare UI Elements in XML

In android, we can create layouts same like web pages in HTML by using default Views and ViewGroups in the XML file. The layout file must contain only one root element, which must be a `View` or `ViewGroup` object. Once we define the root element, then we can add additional layout objects or widgets as child elements to build the View hierarchy that defines our layout.

Following is the example of defining a layout in an XML file (**activity_main.xml**) using LinearLayout to hold a TextView, EditText, and Button.

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

   android:layout_width="match_parent"
   android:layout_height="match_parent">
   <TextView
      android:id="@+id/fstTxt"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:text="Enter Name"
       />
```

```
    <EditText
        android:id="@+id/name"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:ems="10">
    </EditText>
    <Button
        android:id="@+id/getName"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Get Name" />
</LinearLayout>
```

We need to create a layout files in **/res/layout** project directory, then only the layout files will compile properly.

Load XML Layout File from an Activity

Once we are done with the creation of layout, we need to load the XML layout resource from our activity **onCreate()** callback method like as shown below

```
 protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
}
```

If you observe above code we are calling our layout using **setContentView** method in the form of **R.layout.layout_file_name**. Here our xml file name is **activity_main.xml** so we used file name **activity_main**.

Generally, during the launch of our activity, **onCreate()** callback method will be called by android framework to get the required layout for an activity.

Instantiate Layout Elements at Runtime

If we want to instantiate layout elements at runtime, we need to create own custom `View` and `ViewGroup` objects programmatically with required layouts.

Following is the example of creating a layout using LinearLayout to hold a TextView, EditText and Button in an activity using custom `View` and `ViewGroup` objects programmatically.

```
public class MainActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        TextView textView1 = new TextView(this);
        textView1.setText("Name:");
        EditText editText1 = new EditText(this);
        editText1.setText("Enter Name");
        Button button1 = new Button(this);
        button1.setText("Add Name");
        LinearLayout linearLayout = new LinearLayout(this);
        linearLayout.addView(textView1);
```

```
        linearLayout.addView(editText1);
        linearLayout.addView(button1);
        setContentView(linearLayout);
    }
}
```

By creating a custom `View` and `ViewGroup` programmatically, we can define a layouts based on our requirements in android applications.

Width and Height

When we define a layout using XML file we need to set width and height for every `View` and `ViewGroup` element using **layout_width** and **layout_height** attributes.

Following is the example of setting width and height for `View` and `ViewGroup` elements in XML layout file.

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

   android:layout_width="match_parent"
   android:layout_height="match_parent">
   <TextView
      android:id="@+id/fstTxt"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:text="Enter Name" />
</LinearLayout>
```

If you observe above example, we used different values to set **layout_width** and **layout_height**, those are

- match_parent
- wrap_content

If we set value **match_parent**, then the `View` or `ViewGroup` will try to match with parent width or height.

If we set value **wrap_content**, then the `View` or `ViewGroup` will try to adjust its width or height based on the content.

Android Layout Attributes

In android, like **layout_width** and **layout_height** we have a different type of attributes available for `View` and `ViewGroup` objects to define the appearance of layouts based on our requirements.

The following are some of the common layout attributes used in the android application.

Attribute	Description

android:id	It is used to uniquely identify the view and ViewGroups
android:layout_width	It is used to define the width for View and ViewGroup elements in a layout
android:layout_height	It is used to define the height for View and ViewGroup elements in a layout
android:layout_marginLeft	It is used to define the extra space in the left side for View and ViewGroup elements in a layout
android:layout_marginRight	It is used to define the extra space in right side for View and ViewGroup elements in layout
android:layout_marginTop	It is used to define the extra space on top for View and ViewGroup elements in layout
android:layout_marginBottom	It is used to define the extra space in the bottom side for View and ViewGroup elements in a layout
android:paddingLeft	It is used to define the left side padding for View and ViewGroup elements in layout files
android:paddingRight	It is used to define the right side padding for View and ViewGroup elements in layout files
android:paddingTop	It is used to define padding for View and ViewGroup elements in layout files on top side
android:paddingBottom	It is used to define the bottom side padding for View and ViewGroup elements in layout files
android:layout_gravity	It is used to define how child Views are positioned

Android Layout Types

We have a different type of layouts available in android to implement user interface for our android applications with different designs based on our requirements.

Following are the commonly used layouts in android applications to implement required designs.

- Linear Layout
- Relative Layout
- Frame Layout
- Table Layout
- Web View
- List View
- Grid View

Android Linear Layout

In android, LinearLayout is a `ViewGroup` subclass which is used to render all child `View` instances one by one either in a horizontal direction or vertical direction based on the orientation property.

Android Relative Layout

In android, RelativeLayout is a `ViewGroup` which is used to specify the position of child `View` instances relative to each other (Child A to the left of Child B) or relative to the parent (Aligned to the top of a parent).

Android Frame Layout

In android, FrameLayout is a `ViewGroup` subclass which is used to specify the position of `View` instances it contains on the top of each other to display only a single View inside the FrameLayout.

Android Table Layout

In android, TableLayout is a `ViewGroup` subclass which is used to display the child View elements in rows and columns.

Android Web View

In android, WebView is a browser that is used to display the web pages as a part of our activity layout.

Android List View

In android, ListView is a `ViewGroup` which is used to display scrollable single column list of items.

Android Grid View

In android, GridView is a `ViewGroup` which is used to display items in a scrollable grid of columns and rows.