**View and ViewGroups:** An activity contains Views and ViewGroups. A View is a widget that has appearance on screen for example buttons, labels, textboxes etc. All View classes are derived from base class android.view.View.

One or more views can be grouped together into a ViewGroup. A ViewGroup is a special type of view and it provides layout in which we can order the appearance and sequence of views for example LinearLayout and FrameLayout.

Views can be divided into three groups:

1. **Basic views:** These are commonly used views such as TextView, EditText and Button views.
2. **Picker views:** These enable us to select from a list, such as TimePicker and DatePicker views.
3. **List views:** These display a list of items, such as the ListView and SpinnerView views.

**Note:** We can create views either in our activity class via java programming or in **res/layout/main.xml** file. However it is recommended to design our interfaces in xml format because we can change our interface with recompiling our code.

**Common Attributes used in Views and ViewGroups:**

|  |  |
| --- | --- |
| **Attribute** | **Description** |
| android:layout\_width | width of View or ViewGroup Values : fill\_parent, wrap\_content, match\_parent |
| android:layout\_height | height of View or ViewGroup Values : fill\_parent, wrap\_content, match\_parent |
| android:layout\_x | x coordinate of View or ViewGroup |
| android:layout\_y | y coordinate of View or ViewGroup |
| android:layout\_marginTop | extra space on the top side of View or ViewGroup |
| android:layout\_marginBottom | extra space on the bottom side of |
| android:layout\_marginLeft | extra space on the left side of |
| android:layout\_marginRight | extra space on the right side of |
| android:layout\_gravity | specifies how child views are positioned Values: top, bottom, left, right |
| android:layout\_weight | specifies how much extra space should be allocated to the view. Total weight of all components should be equal to 1.0. |

**Common Units used with Views and ViewGroups:**

|  |  |
| --- | --- |
| **Unit** | **Description** |
| dp or dip (Density-Independent Pixel) | 160dp is equivalent to 1 inch of physical screen size. It is recommended unit for specifying dimensions of views. |
| sp (Scale-Independent Pixel) | This is similar to dp and recommended for specify font sizes. |
| pt (point) | A point is 1/72 of an inch, it is based on physical screen size. |
| px (Pixel) | Corresponds to actual pixels on screen. Not recommended because UI may differ with different screen sizes. |

**Some Common Views:**

**(1) TextView:** TextView is used to display non editable text to user. In some other platforms it is commonly referred to as Label.  
 <TextView  
 android:layout\_width=”fill\_parent”  
 android:layout\_height=”wrap\_content”  
 android:text=”@string/hello” />

Here **android:text** specifies text for TextView and **@string/** refers to strings.xml in **res/values** folder. In strings.xml we define our text strings in following format.  
 <resources>  
 <string name=”hello”>Hello World, I am learning Android</string>  
 </resources>

**(2) EditText:** It is a subclass of TextView. It creates a text component with editable text.  
 <EditText  
 android:id="@+id/txtName"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:hint="Enter Name...."  
 android:textSize="18sp" />

Here **android:id=”@+id/..”** specifies id for EditText and it is used to refer EditText component in our Activity class.  
**android:hint** specifies hint for EditText component.  
**android:textSize** specifies text size for EditText.

**Attributes used with EditText:**

|  |  |
| --- | --- |
| Attribute | Description |
| android:capitalize | It automatically capitalize what the user types  Values : none, sentences, words, characters |
| android:hint | Sets the hint for edit text |
| android:textSize | Sets the text size |
| android:inputType | Sets the input type for edit text  Values : none, text, textCapCharacters, textCapWords, textCapSentences, textAutoCorrect, textMultiline, textUri, textEmailAddress, textPassword, number, phone, date, time, datetime |
| android:lines | Total no of lines |
| android:minLines | Minimum no of lines |
| android:maxLines | Maximum no of lines |
| android:scrollbars | Sets the scrollbars for text. Values : none, horizontal, vertical |
| android:password | Sets the password mode. Values: true, false |

**(3) Button:** It creates a push button.  
 <Button   
 android:id="@+id/btnSave"  
 android:layout\_width="160dp"  
 android:layout\_height="wrap\_content"  
 android:layout\_gravity="right"  
 android:text="Save"/>

**(4) ImageButton:** It creates a button with an image on it.  
 <ImageButton   
 android:id="@+id/imageBtn"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:src="@drawable/flower" />  
Here **android:src** specify source of image file to be displayed and **@drawable/** specifies image file located in **res/drawable** folder and flower is reffering flower.png image file.

**(5) CheckBox:** It creates a check box.  
 <CheckBox   
 android:id="@+id/chkRemind"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:text="Remind Me Later"  
 android:checked="true" />  
 <CheckBox

style=*"?android:attr/starStyle"*

android:id=*"@+id/chkStar"*

android:layout\_width=*"wrap\_content"*

android:layout\_height=*"wrap\_content"*

android:checked=*"true"*

/>  
Here style=*"?android:attr/starStyle"*specify that check box should be displayed as star.

**(6) RadioGroup and RadioButton:** RadioButton view is used to create a radiobutton and RadioGroup View is used to make radiobuttons mutually exclusive.  
  
 <RadioGroupandroid:id=*"@+id/rdbGp1"*

android:layout\_width=*"match\_parent"*

android:layout\_height=*"wrap\_content"*

android:orientation=*"vertical"*>

<RadioButtonandroid:id=*"@+id/rdb1"*

android:layout\_width=*"fill\_parent"*

android:layout\_height=*"wrap\_content"*

android:text=*"Option 1"*/>

<RadioButtonandroid:id=*"@+id/rdb2"*

android:layout\_width=*"fill\_parent"*

android:layout\_height=*"wrap\_content"*

android:text=*"Option 2"*/>

</RadioGroup>

**(7) ToggleButton:** It displays a checked/unchecked status using a light indicator.  
 <ToggleButtonandroid:id=*"@+id/toggle"*

android:layout\_width=*"wrap\_content"*

android:layout\_height=*"wrap\_content"*/>

**Some Common View Groups:** A ViewGroup is derrived from the base class **android.view.ViewGroup**. Android Supports following ViewGroups:

**(1) LinearLayout:** LinearLayout arranges views either vertically or horizontally. In LinearLayout we can specify layout\_weight and layout\_gravity attributes for a view.

<LinearLayoutxmlns:android=*"http://schemas.android.com/apk/res/android"*

android:layout\_width=*"match\_parent"*

android:layout\_height=*"match\_parent"*

android:orientation=*"vertical"*>

<TextView

android:layout\_width=*"105dp"*

android:layout\_height=*"wrap\_content"*

android:text=*"@string/hello"*

/>

<Button

android:layout\_width=*"160dp"*

android:layout\_height=*"wrap\_content"*

android:layout\_gravity=*"right"*

android:text=*"Button"*

android:weight=*"0.2"* />

<Button

android:layout\_width=*"match\_parent"*

android:layout\_height=*"wrap\_content"*

android:text=*"Button"*

android:weight=*"0.8"* />

</LinearLayout>

Here xmlns:android specify xml name sapce which specify naming conventions and rules.

**(2) AbsoluteLayout:** AbsoluteLaout enables us to specify exact location of its child views.

<AbsoluteLayoutxmlns:android=*"http://schemas.android.com/apk/res/android"*

android:layout\_width=*" match\_parent "*

android:layout\_height=*" match\_parent "*>

<Buttonandroid:layout\_width=*"200dp"*

android:layout\_height=*"wrap\_content"*

android:text=*"Button-1"*

android:layout\_x=*"12px"*

android:layout\_y=*"30px"*/>

<Buttonandroid:layout\_width=*"300dp"*

android:layout\_height=*"wrap\_content"*

android:text=*"Button-2"*

android:layout\_x=*"40px"*

android:layout\_y=*"100px"*/>

</AbsoluteLayout>

**Note:** AbsoluteLayout is not recommended because appearance of application may varry in different resolution devices.

**(3) TableLayout:** Using TableLayout we can arrange views in tabular format (rows and columns). <TableRow> element is used to create a single row. Each view in a row creates a cell in table.

<TableLayoutxmlns:android=*"http://schemas.android.com/apk/res/android"*

android:layout\_width=*"**match\_parent"*

android:layout\_height=*"match\_parent"*>

<TableRow>

<TextViewandroid:text=*"User Name"*android:width=*"120px"*/>

<EditTextandroid:id=*"@+id/txtUserName"*android:width=*"200px"*/>

</TableRow>

<TableRow>

<TextViewandroid:text=*"Password"*/>

<EditTextandroid:id=*"@+id/txtPassword"*android:password=*"true"*/>

</TableRow>

<TableRow>

<TextView/>

<CheckBoxandroid:id=*"@+id/chkRememberPassword"*

android:layout\_width=*"fill\_parent"*

android:layout\_height=*"wrap\_content"*

android:text=*"Remember Password"*/>

</TableRow>

<TableRow>

<Buttonandroid:id=*"@+id/buttonSignIn"*android:text=*"Log In"*/>

</TableRow>

</TableLayout>

**(4) RelativeLayout:** RelativeLayout enables us to specify elements that are positioned relative to each other.

<RelativeLayoutxmlns:android=*"http://schemas.android.com/apk/res/android"*

android:layout\_width=*"match\_parent"*

android:layout\_height=*"match\_parent"*>

<Button

android:id=*"@+id/btnStr"*

android:layout\_height=*"wrap\_content"*

android:text=*"Stretched"*

android:layout\_alignParentLeft=*"true"*

android:layout\_alignParentRight=*"true"*/>

<Button

android:id=*"@+id/btnR"*

android:layout\_width=*"150dp"*

android:layout\_height=*"wrap\_content"*

android:text=*"Right"*

android:layout\_alignRight=*"@+id/btnStr"*

android:layout\_below=*"@+id/btnStr"*/>

<Button

android:id=*"@+id/btnC"*

android:layout\_width=*"wrap\_content"*

android:layout\_height=*"wrap\_content"*

android:text=*"Center"*

android:layout\_alignLeft=*"@+id/btnR"*

android:layout\_below=*"@+id/btnR"*/>

</RelativeLayout>

RelativeLayout has following additional attributes:

|  |  |
| --- | --- |
| android:layout\_alignParentTop | Aligns the view with parent’s top border(bool) |
| android:layout\_alignParentLeft | Aligns the view with parent’s left border(bool) |
| android:layout\_alignParentRight | Aligns the view with parent’s right border(bool) |
| android:layout\_alignParentBottom | Aligns the view with parent’s bottom border(bool) |
| android:layout\_alignTop | Aligns the view with top of another view(id of view) |
| android:layout\_alignLeft | Aligns the view with left of another view(id of view) |
| android:layout\_alignRight | Aligns the view with right of another view(id of view) |
| android:layout\_alignBottom | Aligns the view with bottom of another view(id of view) |
| android:layout\_centerHorizontal | Aligns the view horizontally center with respect to its parent(bool) |
| android:layout\_centerVertical | Aligns the view vertically center with respect to its parent(bool) |

**(5) FrameLayout:** The FrameLayout is a placeholder on screen that we can use to display a single view. If we add multiple views to FrameLayout they will be displayed stacked at each other.

<FrameLayoutxmlns:android=*"http://schemas.android.com/apk/res/android"*

android:layout\_width=*"fill\_parent"*

android:layout\_height=*"fill\_parent"*>

<Buttonandroid:text=*"Button"*android:layout\_width=*"match\_parent"*  
 android:layout\_height=*"wrap\_content"*/>

<Buttonandroid:text=*"Button"*android:layout\_width=*"wrap\_content"*  
 android:layout\_height=*"wrap\_content"*/>

</FrameLayout>

**(6) ScrollView:** A ScrollView is a special type of FrameLayout and it enables users to scroll views that occupy more space than the physical display. ScrollView can contain only one child view or view group.

<ScrollViewxmlns:android=*"http://schemas.android.com/apk/res/android"*

android:fillViewport=*"true"*

android:layout\_width=*"match\_parent"*

android:layout\_height=*"match\_parent"*>

<LinearLayout

android:layout\_width=*"match\_parent"*

android:layout\_height=*"match\_parent"*

android:orientation=*"vertical"*>

<TextView

android:layout\_width=*"105dp"*

android:layout\_height=*"wrap\_content"*

android:text=*"@string/hello"*/>

<Button

android:layout\_width=*"160dp"*

android:layout\_height=*"wrap\_content"*

android:text=*"Button"*/>

<Button

android:layout\_width=*"160dp"*

android:layout\_height=*"wrap\_content"*

android:text=*"Button"*/>

</LinearLayout>

</ScrollView>