Relative Layout

`RelativeLayout` is a `ViewGroup` that displays child `View` elements in relative positions. The position of a `View` can be specified as relative to sibling elements (such as to the left-of or below a given element) or in positions relative to the `RelativeLayout` area (such as aligned to the bottom, left of center).

A `RelativeLayout` is a very powerful utility for designing a user interface because it can eliminate nested `ViewGroup`s. If you find yourself using several nested `LinearLayout` groups, you may be able to replace them with a single `RelativeLayout`.

1. Start a new project named *HelloRelativeLayout*.
2. Open the `Resources/Layout/Main.axml` file and insert the following:

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent">
    <TextView
        android:id="@+id/label"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="Type here:"/>
    <EditText
        android:id="@+id/entry"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"

android:background="@android:drawable/editbox_background"
        android:layout_below="@id/label"/>
```

```
    <Button
        android:id="@+id/ok"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_below="@id/entry"
        android:layout_alignParentRight="true"
        android:layout_marginLeft="10dip"
        android:text="OK" />
    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_toLeftOf="@id/ok"
        android:layout_alignTop="@id/ok"
        android:text="Cancel" />
</RelativeLayout>
```

Notice each of the `android:layout_*` attributes, such as `layout_below`, `layout_alignParentRight`, and `layout_toLeftOf`. When using a `RelativeLayout`, you can use these attributes to describe how you want to position each `View`. Each one of these attributes define a different kind of relative position. Some attributes use the resource ID of a sibling `View` to define its own relative position. For example, the last `Button` is defined to lie to the left-of and aligned-with-the-top-of the `View` identified by the ID `ok` (which is the previous `Button`).

All of the available layout attributes are defined in `RelativeLayout.LayoutParams`.

3. Make sure you load this layout in the `OnCreate()` method:

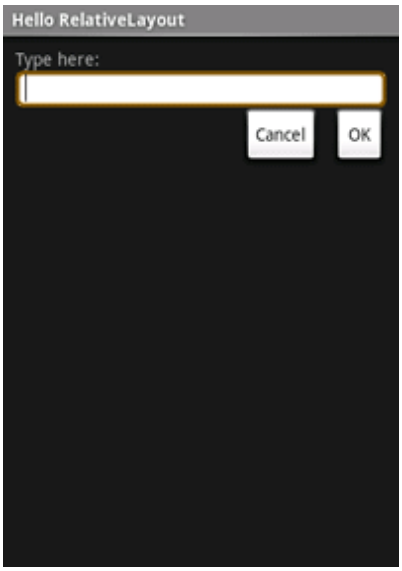
```
protected override void OnCreate (Bundle savedInstanceState)
{
    base.OnCreate (savedInstanceState);
    SetContentView (Resource.Layout.Main);
```

```
}
```

The `SetContentView(int)` method loads the layout file for the `Activity`, specified by the resource ID — `Resource.Layout.Main` refers to the `Resources/Layout/Main.axml` layout file.

4. Run the application.

You should see the following layout:



Resources

- `RelativeLayout`
- `RelativeLayout.LayoutParams`
- `TextView`
- `EditText`
- `Button`

Portions of this page are modifications based on work created and shared by the Android Open Source Project and used according to terms described in the Creative Commons 2.5

Attribution License . *This tutorial is based on the Android Relative Layout tutorial* .