

THE SCREEN LAYOUT AND THE MAIN.XML FILE

The screen configuration is controlled by the main.xml file.

Study the main.xml file here:

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

    <TextView
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="@string/hello"
    />
</LinearLayout>
```

- XML files are composed of units called elements with opening and closing tags.
- The opening tag consists of a tag name surrounded by right and left arrows (less-than and greater-than signs, respectively) such as <some_element>.
- The closing tag is the same name, but with a slash mark / in front of the tag name, such as </some_element>.
- To be usable, the XML file must be perfectly formed; in other words, all the tags have to be perfectly matched.
- A shortcut can also be used where an element is included in one tag, ending with />.

For example:

```
<TextView
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:text="@string/hello"
/>
```

would be functionally equivalent to:

```
<TextView  
android:layout_width="fill_parent"  
android:layout_height="wrap_content"  
android:text="@string/hello">  
</TextView>
```

- The LinearLayout class is indeed a Java class found in the Android software development kit (SDK).
- It is one of many subclasses of the ViewGroup class. The attributes that are set in the XML file, namely orientation, layout_width, and layout_height, can be set in Java code by using methods that belong to the LinearLayout class, one or more of the classes in its hierarchy, or one or more of the inner classes in the hierarchy.
- Using the main.xml file, we can separate form from functionality and design our user interface without writing a line of Java code.
- The outermost LinearLayout object normally represents the whole screen in an application, similar to the way a Frame class represents the application window in a PC Java application.
- Note that the two attributes, layout_width and layout_height, are set to fill_parent. These two settings together cause LinearLayout to occupy the whole screen, filling all the way across and all the way down.
- The orientation attribute set to Vertical means that objects are added top to bottom.

Try this little experiment.

Using copy and paste, take this portion of code and duplicate it:

```
<TextView  
android:layout_width="fill_parent"  
android:layout_height="wrap_content"  
android:text="@string/hello"  
>
```

in such a way that your main.xml looks like this:

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

```

android:layout_height="fill_parent"
>
<TextView
android:layout_width="fill_parent"
android:layout_height="wrap_content"
android:text="@string/hello"
/>
<TextView
android:layout_width="fill_parent"
android:layout_height="wrap_content"
android:text="@string/hello"
/>
</LinearLayout>

```

Your text may be different, but you will see that the two lines come one below the other, or in Vertical orientation. This is what the orientation attribute in the LinearLayout section of main.xml specifies.



Now change the word, vertical to horizontal. As you might guess, the two lines of text should show side by side, rather than top and bottom.

But, Remember that the parameter for the TextView objects' widths is set to fill-parent. This means that each is set to occupy the full width of the screen, hence fill the parent.

If you ran the application at this point, you would **only see one TextView object** on the screen, **because it is set to fill the screen completely from left to right.**

So before we run the application, let's **change the horizontal property for both TextViews to wrapcontent.**

Your main.xml file should look like the following example:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="horizontal"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    >
    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/hello"
    />
    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/hello"
    />
</LinearLayout>
```

Now rerun the application.



- Some of the other Layout classes that are also subclasses of the ViewGroup class are FrameLayout, AbsoluteLayout, TableLayout, and RelativeLayout.
- The AbsoluteLayout allows the programmer to specify the exact x, y coordinates of the components on the screen, but it is of limited use because it doesn't adjust for variations in the screen resolution of multiple target devices.
- On the other hand, if the programmer is writing for only one type of device, say for a specific client, it may be useful.
- The FrameLayout layers multiple controls one on top of the other. This layout might be useful for graphics in an application.
- To get a quick idea of how the FrameLayout responds to text controls such as the Text-View control, modify your main.xml file to look like this:

```
<?xml version="1.0" encoding="utf-8"?>
<FrameLayout xmlns:android="http://schemas.android.com/apk/res/android"
android:orientation="horizontal"
android:layout_width="fill_parent"
android:layout_height="fill_parent"
>
<TextView
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:text="@string/hello"

/>
<TextView
android:layout_width="fill_parent"
android:layout_height="wrap_content"
android:text="Text View number two"
/>
<TextView
android:layout_width="fill_parent"
android:layout_height="wrap_content"
android:text="Text View number three"
/>
</FrameLayout>
```

Then restart your application. You should see three text strings layered on top of one another.

For any doubt contact 9873961590.