Layout Managers

Introduction

- The android platform includes a collection of view classes that behave as containers for other views.
- These containers are also known as layout managers.
- Each one of the available layout managers implements a specific strategy for managing the size and the position of the views it holds.

The `LinearLayout` Layout Manager

- This layout manager organizes the views it holds either horizontally or vertically.
- The value of the 'orientation' property defines whether the views will be displayed horizontally or vertically.

The `LinearLayout` Layout Manager

```
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="horizontal"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content">

    <TextView android:layout_width="wrap_content"
    android:layout_height="wrap_content"
        android:text="First Name:" />

    <TextView android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Haim" />

</LinearLayout>
```

The `layout_weight` Property

- The `layout_weight` influences the relatively allocated space each view get from the extra available space. The `layout_weight` is the factor the view will be stretched, when the parent is larger than the sum of its child views.

The `gravity` Property

- The `gravity` each view element has means the alignment that will govern its content.
- While the `gravity` of a specific view controls the content of that specific view, the `gravity` of a specific layout controls the alignment of the views within that layout.

The `gravity` Property

```
<?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"
    >

        <Button android:text="Button01" android:id="@+id/Button01"
            android:layout_height="wrap_content" android:gravity="top|left"
            android:layout_weight="1" android:layout_width="fill_parent">
        </Button>

        <Button android:text="Button02" android:id="@+id/Button02"
            android:layout_height="wrap_content" android:gravity="top|right"
            android:layout_weight="1" android:layout_width="fill_parent">
        </Button>

        <Button android:text="Button03" android:id="@+id/Button03"
            android:layout_height="wrap_content" android:gravity="center"
            android:layout_weight="1" android:layout_width="fill_parent">
        </Button>

</LinearLayout>
```

The `gravity` Property

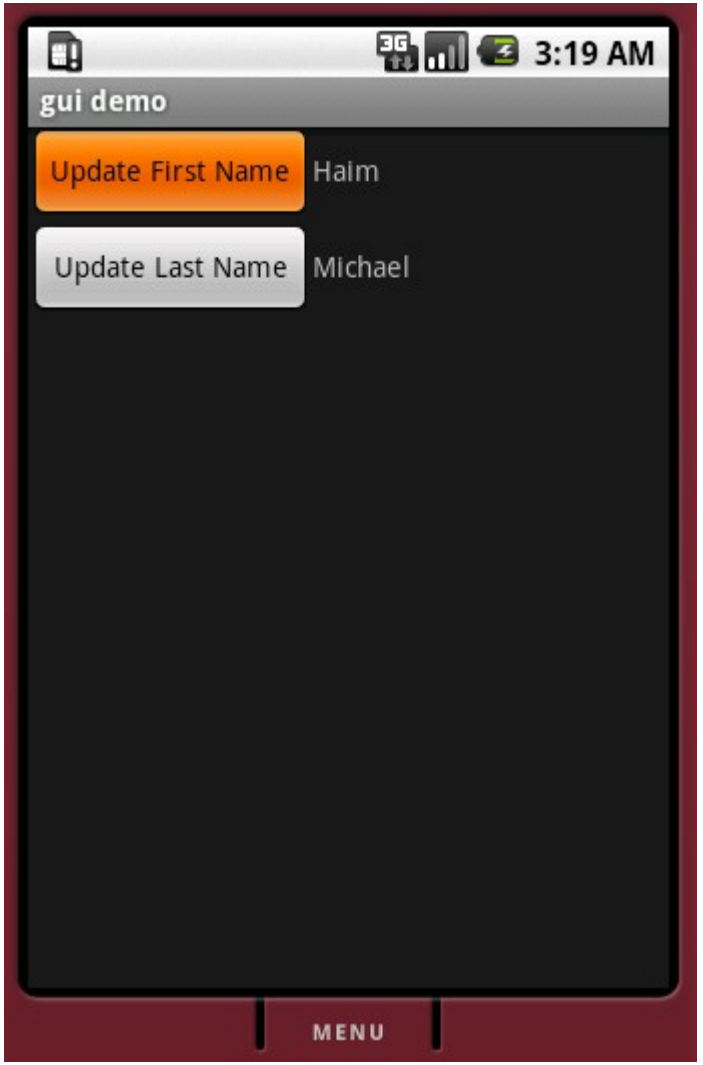
The `TableLayout` Layout Manager

- This layout manager organizes the views it holds into rows and columns.
- The number of columns is set in accordance with the number of columns that should fit the row with the biggest number of cells.
- The `TableLayout` layout manager can have views of a type other than `TableRow` views. Children of a `TableLayout` span the entire row.

The `TableLayout` Layout Manager

```
<?xml version="1.0" encoding="utf-8"?>

<TableLayout android:id="@+id/TableLayout01" android:layout_width="wrap_content"
android:layout_height="wrap_content"
xmlns:android="http://schemas.android.com/apk/res/android">

<TableRow android:id="@+id/TableRow01" android:layout_width="wrap_content"
android:layout_height="wrap_content">
    <Button android:id="@+id/Button01" android:layout_width="wrap_content"
    android:layout_height="wrap_content" android:text="Update First Name">
    </Button>
    <TextView android:id="@+id/TextView01" android:layout_width="wrap_content"
    android:layout_height="wrap_content" android:text="Haim">
    </TextView>
</TableRow>

<TableRow android:id="@+id/TableRow02" android:layout_width="wrap_content"
android:layout_height="wrap_content">
    <Button android:id="@+id/Button02" android:layout_width="wrap_content"
    android:layout_height="wrap_content" android:text="Update Last Name">
    </Button>
    <TextView android:id="@+id/TextView02" android:layout_height="wrap_content"
android:text="Michael" android:layout_width="fill_parent">
    </TextView>
</TableRow>

</TableLayout>
```

The `TableLayout` Layout Manager



The `TableLayout` Layout Manager

- The `android:stretchColumns` property we can add to the `TableLayout` layout manager hints which columns can be stretched if required.
- The columns are identified with a zero based indexing scheme.

The `TableLayout` Layout Manager

- The `android:shrinkColumns` property we can add to the `TableLayout` layout manager hints which columns can be shrink if required.
- The columns are identified with a zero based indexing scheme.

The `Padding` Properties

- Each view supports the padding properties that allow us to set the exact space between the view content and its inner borders.
- The available padding properties include the following:

```
Padding
PaddingBottom
PaddingLeft
PaddingRight
PaddingTop
```

The `Padding` Properties

- The values we can assign the padding properties as well as any other property that defines a dimension value can be any of the following units:

  `px` – pixels

  `in` – inches

  `mm` – milimetters

  `dip` or `dp` – device independent pixels (relatively to 160 dpi screen)

The `Padding` Properties

```
<?xml version="1.0" encoding="utf-8"?>

<TableLayout android:id="@+id/TableLayout01"
android:layout_width="wrap_content" android:layout_height="wrap_content"
xmlns:android="http://schemas.android.com/apk/res/android">

<TableRow android:id="@+id/TableRow01" android:layout_width="wrap_content"
android:layout_height="wrap_content">
<Button android:id="@+id/Button01" android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:text="Update First Name"></Button>
<TextView android:id="@+id/TextView01" android:layout_width="wrap_content"
android:layout_height="wrap_content" android:text="Haim"></TextView>
</TableRow>

<TableRow android:id="@+id/TableRow02" android:layout_width="wrap_content"
android:layout_height="wrap_content">
<Button android:id="@+id/Button02" android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:text="Update Last Name" android:paddingBottom="50px"></Button>
<TextView android:id="@+id/TextView02" android:layout_height="wrap_content"
android:text="Michael" android:layout_width="fill_parent"></TextView>
</TableRow>

</TableLayout>
```

The `Padding` Properties

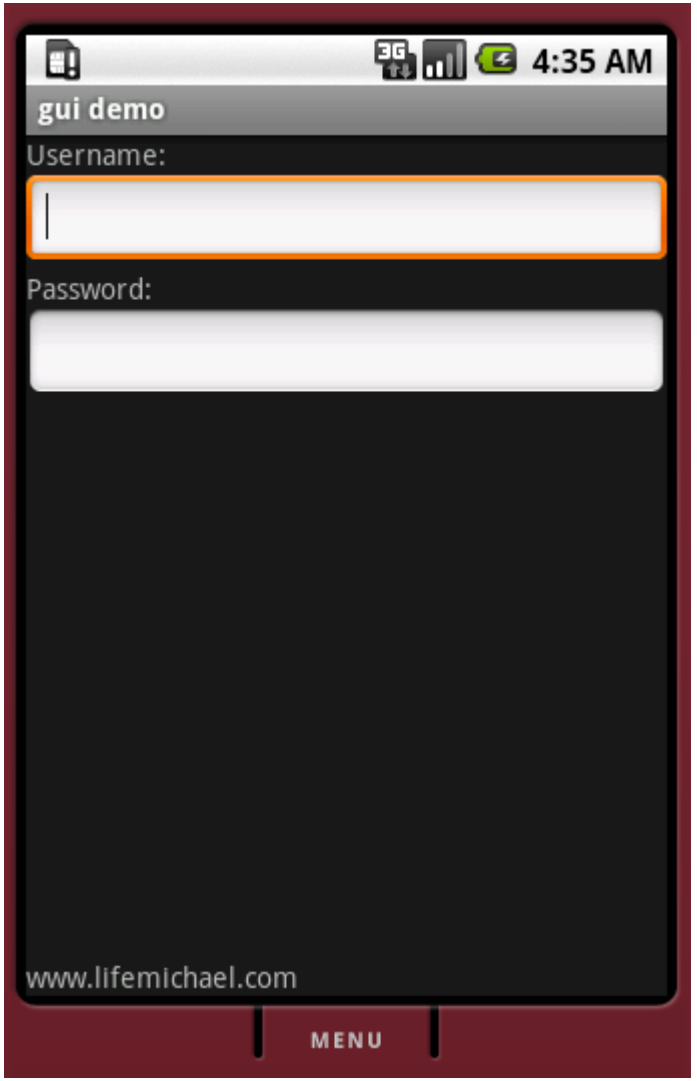
The `RelativeLayout` Layout Manager

- This layout manager places the views relatively to the layout manager itself or relatively to another specific view the layout manager includes.

The `RelativeLayout` Layout Manager

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
android:layout_width="fill_parent"
android:layout_height="wrap_content">

<TextView android:id="@+id/userNameLbl" android:layout_width="fill_parent"
android:layout_height="wrap_content" android:text="Username: "
android:layout_alignParentTop="true" />

<EditText android:id="@+id/userNameText" android:layout_width="fill_parent"
android:layout_height="wrap_content" android:layout_below="@id/userNameLbl" />

<TextView android:id="@+id/pwdLbl" android:layout_width="fill_parent"
android:layout_height="wrap_content" android:layout_below="@id/userNameText"
android:text="Password: " />

<EditText android:id="@+id/pwdText" android:layout_width="fill_parent"
android:layout_height="wrap_content" android:layout_below="@id/pwdLbl"/>

<TextView android:layout_width="fill_parent"
android:layout_height="wrap_content" android:layout_alignParentBottom="true"
android:id="@+id/footer" android:text="www.lifemichael.com"/>

</RelativeLayout>
```

The `RelativeLayout` Layout Manager



The `RelativeLayout` Layout Manager

- Using the `RelativeLayout` we can specify the exact relative position of each view. We can specify the location of each view relatively to its parent or to any of the other views. Each view we refer should be specified using its ID.
- We can align two views either according to their right\left border or make one view below the other, centered to the center of the screen, centered to the left etc.

The `RelativeLayout` Layout Manager

- The elements should be listed in the XML layout document in the same order we use when specifying the relative position of each one of them.

The `RelativeLayout` Layout Manager

```
<?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"
android:background="@drawable/yellow" android:padding="6px" >

<TextView android:id="@+id/labeltext" android:layout_width="fill_parent"
android:layout_height="wrap_content" android:text="Enter Your Name:" />

<EditText android:id="@+id/enteredtext" android:layout_width="fill_parent"
android:layout_height="wrap_content"android:layout_below="@id/labeltext"
android:layout_marginTop="10px"/>

<Button android:id="@+id/submit_bt" android:layout_width="wrap_content"
android:layout_height="wrap_content" android:layout_below="@id/enteredtext"
android:layout_alignParentRight="true"  android:layout_marginLeft="12px"
android:layout_marginTop="8px" android:text="Submit" />

<Button android:layout_width="wrap_content" android:id="@+id/cancel_bt"
android:layout_height="wrap_content" android:layout_toLeftOf="@id/submit_bt"
android:layout_alignTop="@id/submit_bt" android:text="Cancel" />

</RelativeLayout>
```

The `RelativeLayout` Layout Manager

The `AbsoluteLayout` Layout Manager

- Using this layout manager we will be able to specify the exact position separately for each view.
- Using this layout manager isn't recommended. It is a deprecated one.

The `AbsoluteLayout` Layout Manager

```
<AbsoluteLayout
android:layout_width="fill_parent" android:layout_height="fill_parent"
xmlns:android="http://schemas.android.com/apk/res/android">

<TextView android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="First Name"
    android:layout_x="50px"
    android:layout_y="50px" />

<EditText
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_x="160px"
    android:layout_y="50px" />

</AbsoluteLayout>
```

The `FrameLayout` Layout Manager

- The `FrameLayout` blocks out an area on the screen allowing to display a single item only.
- Using this layout manager we can manage two (or more) interchangeable views we can switch between as a response to UI events.
- The multiple children we add are drawn in a stack. The displayed one is the one on top of the stack, the most recently added.

The `FrameLayout` Layout Manager

- We can call on each one of the views (the stack includes) the `setVisiblity` method. Passing over `View.VISIBLE` will cause that view to be visible. Passing over `View.GONE` will cause that view to be invisible.
- We can call these methods on our views in order to replace the currently displayed one with another.

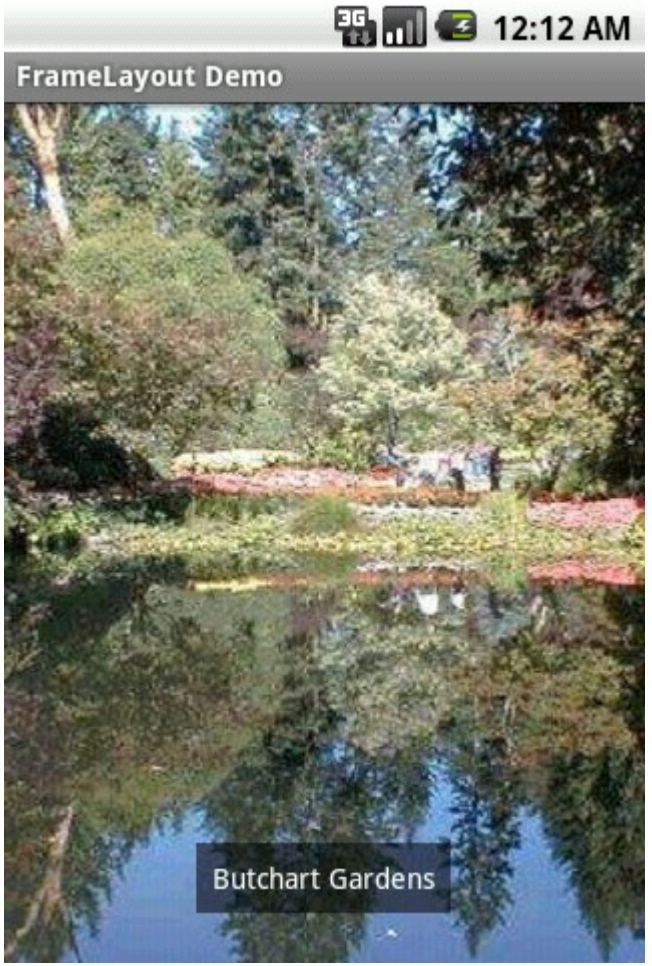
The `FrameLayout` Layout Manager

- All child elements of the `FrameLayout` are pinned to the top left corner of the screen.
- Subsequent child views will be drawn above the previous ones, either partially or totally unless the subsequent child is transparent.

The `FrameLayout` Layout Manager

```
<FrameLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent">

    <ImageView
        android:layout_width="fill_parent"
        android:layout_height="fill_parent"

        android:scaleType="center"
        android:src="@drawable/butchart" />

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginBottom="25dip"
        android:layout_gravity="center_horizontal|bottom"
        android:padding="8dip"
        android:background="#88000000"
        android:textColor="#ffffffff"
        android:text="Butchart Gardens" />

</FrameLayout>
```

The `FrameLayout` Layout Manager



The `TabsHost` Layout Manager

- The `TabHost` layout manager allows us to display tabs for showing different screens of the application.
- When a tab is clicked the displayed screen is swapped and replaced with another.
- The `TabHost` layout manager is the overarching container that holds the tab buttons and the tab contents.

The `TabsHost` Layout Manager

- The `FrameLayout` element is the container for the tab contents. Each tab content is a child of the `FrameLayout` element.
- The `TabWidget` element is responsible for showing the row of the tab buttons, that contain texts and, optionally, icons.

The `TabsHost` Layout Manager

- The `FrameLayout` must have the `android:id` attribute with the `@android:id/tabcontent` value.
- The `TabWidget` must have the `android:id` attribute with the `@android:id/tabs` value.

The `TabsHost` Layout Manager

```
<?xml version="1.0" encoding="utf-8"?>

<TabHost xmlns:android="http://schemas.android.com/apk/res/android"
android:id="@+id/tabs_host" android:layout_width="fill_parent"
android:layout_height="fill_parent">
<LinearLayout android:orientation="vertical"
android:layout_width="fill_parent" android:layout_height="fill_parent">

<TabWidget android:id="@android:id/tabs"
android:layout_width="fill_parent" android:layout_height="wrap_content" />

<FrameLayout android:id="@android:id/tabcontent"
android:layout_width="fill_parent" android:layout_height="fill_parent">
<AnalogClock android:id="@+id/tab1" android:layout_width="fill_parent"
android:layout_height="fill_parent" android:layout_centerHorizontal="true" />
<Button android:id="@+id/tab2" android:layout_width="fill_parent"
android:layout_height="fill_parent" android:text="Press Me" />
<DatePicker android:id="@+id/tab3" android:layout_width="wrap_content"
android:layout_height="wrap_content">
</DatePicker>
</FrameLayout>

</LinearLayout>
</TabHost>
```

The `TabsHost` Layout Manager

```
package com.abelski.samples;

import android.app.Activity;
import android.os.Bundle;
import android.widget.TabHost;

public class TabyActivity extends Activity
{
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle bndl)
    {
        super.onCreate(bndl);
        setContentView(R.layout.main);
        //creating tabs host component
        TabHost tabs = (TabHost) findViewById(R.id.tabs_host);
        tabs.setup();
        //creating first tab
        TabHost.TabSpec specA = tabs.newTabSpec("AAA");
        specA.setContent(R.id.tab1);
        specA.setIndicator("clock");
        tabs.addTab(specA);
```

The `TabsHost` Layout Manager

```
        //creating second tab
        TabHost.TabSpec specB = tabs.newTabSpec("BBB");
        specB.setContent(R.id.tab2);
        specB.setIndicator("button");
        tabs.addTab(specB);
        //creating third tab
        TabHost.TabSpec specC = tabs.newTabSpec("CCC");
        specC.setContent(R.id.tab3);
        specC.setIndicator("date picker");
        tabs.addTab(specC);
    }
}
```

The `TabsHost` Layout Manager