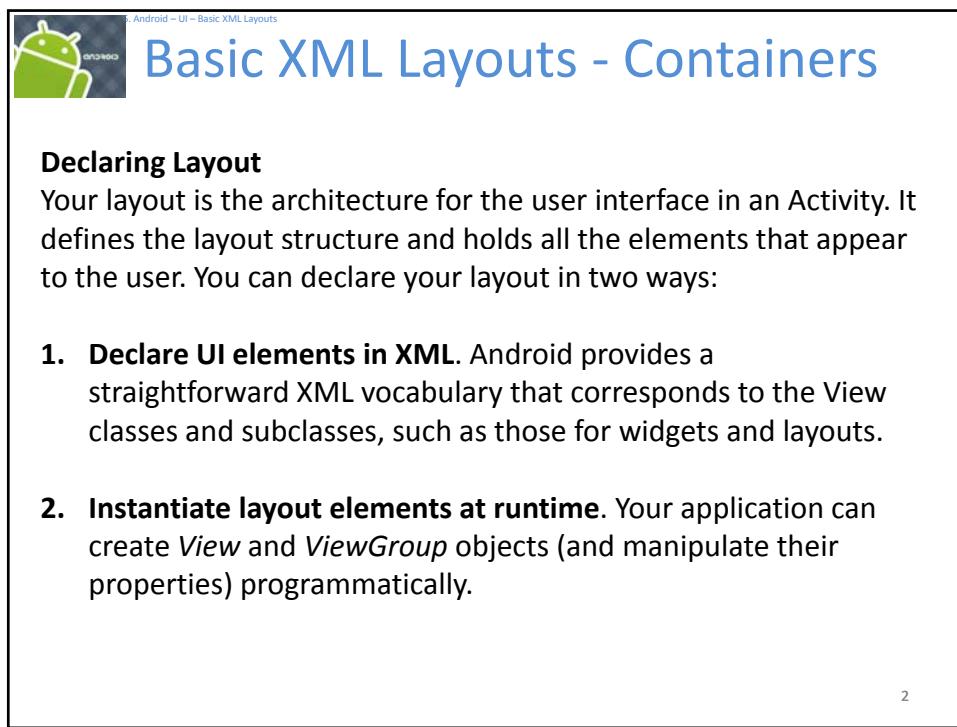


A presentation slide titled "Android Basic XML Layouts". The slide features a blue number "5" in the top right corner. On the left side, there is a vertical decorative bar with a repeating pattern of small green Android icons. At the bottom left, there is a large green Android robot icon. The title "Android Basic XML Layouts" is centered in large blue font. Below it, the author's name "Victor Matos" and affiliation "Cleveland State University" are listed in smaller blue font. A note at the bottom left states: "Notes are based on: The Busy Coder's Guide to Android Development by Mark L. Murphy Copyright © 2008-2009 CommonsWare, LLC. ISBN: 978-0-9816780-0-9 & Android Developers <http://developer.android.com/index.html>".



A slide titled "Basic XML Layouts - Containers". The slide features a small green Android robot icon on the left. The title "Basic XML Layouts - Containers" is centered in large blue font. Below the title, a section titled "Declaring Layout" is described. It states: "Your layout is the architecture for the user interface in an Activity. It defines the layout structure and holds all the elements that appear to the user. You can declare your layout in two ways:"

- 1. Declare UI elements in XML.** Android provides a straightforward XML vocabulary that corresponds to the View classes and subclasses, such as those for widgets and layouts.
- 2. Instantiate layout elements at runtime.** Your application can create *View* and *ViewGroup* objects (and manipulate their properties) programmatically.



s. Android – UI – Basic XML Layouts

Basic XML Layouts - Containers

- Android's **LinearLayout** offers a "box" model similar to the Java-Swing *Box-Layout*.
- The general (and proven) strategy is to obtain the desired UI structure through the right combination of *nested* boxes.
- In addition Android supports a range of containers providing different layout organizations.

Commonly-used Android containers are:

1. **LinearLayout** (the box model),
2. **RelativeLayout** (a rule-based model), and
3. **TableLayout** (the grid model), along with
4. **ScrollView**, a container designed to assist with implementing scrolling containers.

3



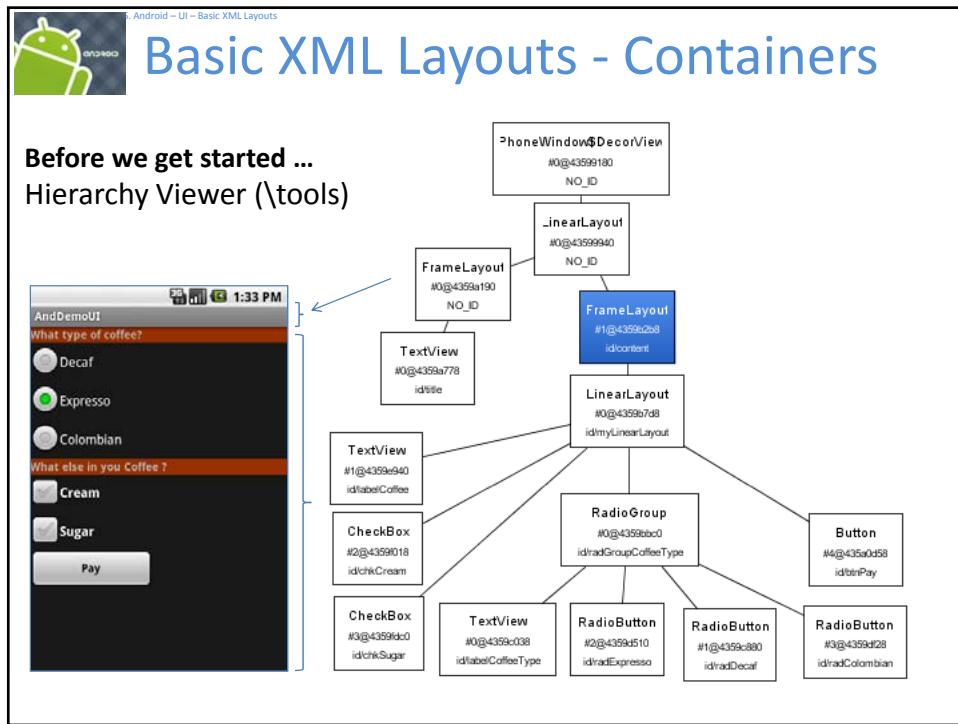
s. Android – UI – Basic XML Layouts

Basic XML Layouts - Containers

Before we get started ...

1. Android's simplest layout manager is called: **Frame Layout**.
2. A Frame Layout is a rectangular container that pins *each child* to its upper left corner.
3. Adding multiple views to a frame layout just stacks one on top of the other (overlapping the views)

4



The screenshot shows the Android Hierarchy Viewer interface with the title "Basic XML Layouts - Containers". On the left, there is a preview of an Android application titled "AndDemoUI" with the time "1:33 PM". The app's UI includes a list of coffee types ("Decaf", "Expresso", "Colombian") and toppings ("Cream", "Sugar"). A "Pay" button is at the bottom. On the right, a tree diagram shows the XML layout hierarchy:

```

graph TD
    PhoneWindow$DecorView["PhoneWindow$DecorView  
#0@43599180  
NO_ID"] --> FrameLayout1["FrameLayout  
#0@4359a190  
NO_ID"]
    FrameLayout1 --> TextView1["TextView  
#0@4359a778  
id:title"]
    FrameLayout1 --> FrameLayout2["FrameLayout  
#1@4359a2b8  
id:content"]
    FrameLayout2 --> LinearLayout1["LinearLayout  
#0@4359b7d8  
id:myLinearLayout"]
    LinearLayout1 --> TextView2["TextView  
#1@4359a940  
id:labelCoffee"]
    LinearLayout1 --> CheckBox1["CheckBox  
#2@4359f018  
id:chkCream"]
    LinearLayout1 --> CheckBox2["CheckBox  
#3@4359fd0  
id:chkSugar"]
    LinearLayout1 --> RadioGroup1["RadioGroup  
#0@4359bb0  
id:radGroupCoffeeType"]
    RadioGroup1 --> RadioButton1["RadioButton  
#2@4359d510  
id:radExpresso"]
    RadioGroup1 --> RadioButton2["RadioButton  
#1@4359c880  
id:radDecaf"]
    RadioGroup1 --> RadioButton3["RadioButton  
#3@4359d28  
id:radColombian"]
    LinearLayout1 --> Button1["Button  
#4@4359d58  
id:lnPay"]
  
```

The "FrameLayout" node has two children: "TextView" (with ID #0@4359a778) and another "FrameLayout" node (with ID #1@4359a2b8). This second "FrameLayout" node contains a "LinearLayout" node (with ID #0@4359b7d8). This "LinearLayout" node contains several child views: "TextView" (#1@4359a940), "CheckBox" (#2@4359f018), "CheckBox" (#3@4359fd0), a "RadioGroup" node (#0@4359bb0), and a "Button" node (#4@4359d58). The "RadioGroup" node has three "RadioButton" children: "#2@4359d510", "#1@4359c880", and "#3@4359d28".

1. Linear Layout

LinearLayout is a *box model* – widgets or child containers are lined up in a *column* or *row*, one after the next.

To configure a LinearLayout, you have five main areas of control besides the container's contents:

- orientation,
- fill model,
- weight,
- gravity, and
- padding

6



Basic XML Layouts - Containers

1. Linear Layout

Orientation

indicates whether the LinearLayout represents a *row* or a *column*.

Add the `android:orientation` property to your LinearLayout element in your XML layout, setting the value to be **horizontal** for a row or **vertical** for a column.

The orientation can be modified at runtime by invoking `setOrientation()`

7

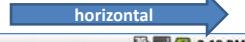


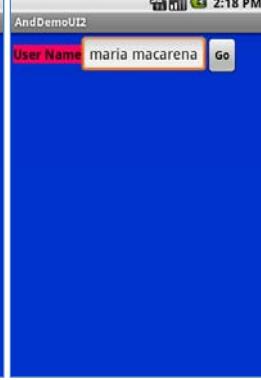
Basic XML Layouts - Containers

1.1 Linear Layout: Orientation

indicates whether the LinearLayout represents a *row* (HORIZONTAL) or a *column* (VERTICAL).

vertical

horizontal


```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    android:id="@+id/my_linearlayout"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:background="#ff033cc"
    android:padding="4px"
    android:orientation="horizontal" >
<TextView
    android:id="@+id/labelUserName"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:background="#ffff0000"
    android:text="User Name"
    android:textSize="16px"
    android:textStyle="bold"
    android:textColor="#ff000000" />
<EditText
    android:id="@+id/editName"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:textSize="18px" />
</EditText>
<Button
    android:id="@+id/btnGo"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Go"
    android:textStyle="bold" />
</Button>
</LinearLayout>
```



Basic XML Layouts - Containers

1.2 Linear Layout: Fill Model

- Widgets have a "natural" size based on their accompanying text.
- When their combined sizes does not *exactly* match the width of the Android device's screen, we may have the issue of what to do with the remaining space.

9



Basic XML Layouts - Containers

1.2 Linear Layout: Fill Model

All widgets inside a LinearLayout must supply dimensional attributes `android:layout_width` and `android:layout_height` to help address the issue of empty space.

Values used in defining height and width are:

1. Specific a *particular dimension*, such as **125px** to indicate the widget should take up exactly 125 pixels.
2. Provide **wrap_content**, which means the widget should fill up its natural space, unless that is too big, in which case Android can use **word-wrap** as needed to make it fit.
3. Provide **fill_parent**, which means the widget should fill up all available space in its enclosing container, after all other widgets are taken care of.

10

Basic XML Layouts - Containers

1.2 Linear Layout: Fill Model

G1 phone resolution is: 240 x 320 px.

```

<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    android:id="@+id/myLinearLayout"
    android:layout_width="fill_parent"
    android:background="#ff0033cc"
    android:padding="4px"
    android:orientation="vertical"
    xmlns:android="http://schemas.android.com/apk/res/android"
>
<TextView
    android:id="@+id/labelUserName"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:background="#ffff0066"
    android:text="User Name"
    android:textSize="16sp"
    android:textStyle="bold"
    android:textColor="#ff000000"
    >
</TextView>
<EditText
    android:id="@+id/ediName"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:textSize="18sp"
    >
</EditText>
<Button
    android:id="@+id/btnGo"
    android:layout_width="125px"
    android:layout_height="wrap_content"
    android:text="Go"
    android:textStyle="bold"
    >
</Button>
</LinearLayout>

```

11

Basic XML Layouts - Containers

1.2 Linear Layout: Weight

It is used to proportionally assign space to widgets in a view.

You set **android:layout_weight** to a value (1, 2, 3, ...) to indicates what proportion of the free space should go to that widget.

Example

Both the *TextView* and the *Button* widgets have been set as in the previous example. Both have the additional property **android:layout_weight = "1"** whereas the *EditText* control has **android:layout_weight = "2"**

Takes: 2 / (1+1+2)
of the free space

12

5. Android – UI – Basic XML Layouts

Basic XML Layouts - Containers

1.3 Linear Layout: Gravity

- It is used to indicate how a control will align on the screen.
- By default, widgets are left- and top-aligned.
- You may use the XML property
`android:layout_gravity="..."`

to set other possible arrangements:
left, center, right, top, bottom, etc.

13

5. Android – UI – Basic XML Layouts

Basic XML Layouts - Containers

1.4 Linear Layout: Padding

- By default, widgets are tightly packed next to each other.
- If you want to increase the whitespace between widgets, you will want to use the `android:padding` property (or by calling `setPadding()` at runtime on the widget's Java object).
- The padding specifies how much space there is between the boundaries of the widget's "cell" and the actual widget contents.

Note: Padding is analogous to the margins on a word processing document.

14



5. Android – UI – Basic XML Layouts

Basic XML Layouts - Containers

1.4 Linear Layout: Padding

- By default, widgets are tightly packed next to each other.
- If you want to increase the whitespace between widgets, you will want to use the `android:padding` property (or by calling `setPadding()` at runtime on the widget's Java object).
- The padding specifies how much space there is between the boundaries of the widget's "cell" and the actual widget contents.

Note: Padding is analogous to the margins on a word processing document.

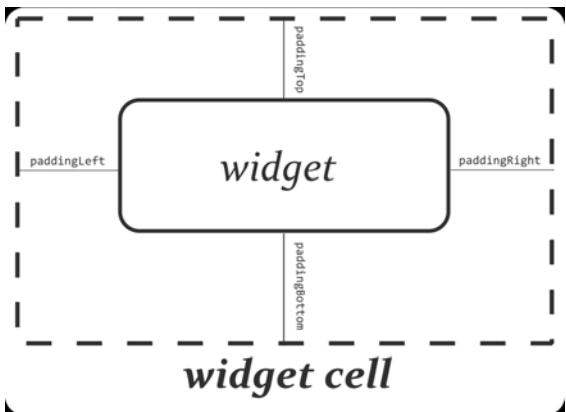
15



5. Android – UI – Basic XML Layouts

Basic XML Layouts - Containers

1.3 Linear Layout: Padding



16

 5. Android – UI – Basic XML Layouts

Basic XML Layouts - Containers

1.3 Linear Layout: Padding

Example:
The EditText box has been changed to display 30px of padding all around



```
<EditText  
    android:id="@+id/ediName"  
    android:layout_width="fill_parent"  
    android:layout_height="wrap_content"  
    android:textSize="18sp"  
  
    android:padding="30px"  
  
>  
</EditText>  
...
```

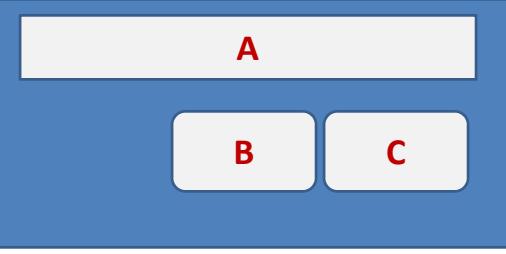
17

 5. Android – UI – Basic XML Layouts

Basic XML Layouts - Containers

2. Relative Layout

RelativeLayout places widgets based on their relationship to other widgets in the container and the parent container.



Example:
A is by the parent's top
C is below A, to its right
B is below A, to the left of C

18



Basic XML Layouts - Containers

2. Relative Layout - Referring to the container

Some positioning XML (boolean) properties mapping a widget according to its location respect to the parent's place are:

- **android:layout_alignParentTop** says the widget's top should align with the top of the container
- **android:layout_alignParentBottom** the widget's bottom should align with the bottom of the container
- **android:layout_alignParentLeft** the widget's left side should align with the left side of the container
- **android:layout_alignParentRight** the widget's right side should align with the right side of the container
- **android:layout_centerHorizontal** the widget should be positioned horizontally at the center of the container
- **android:layout_centerVertical** the widget should be positioned vertically at the center of the container
- **android:layout_centerInParent** the widget should be positioned both horizontally and vertically at the center of the container

19



Basic XML Layouts - Containers

2. Relative Layout – Referring to other widgets

The following properties manage positioning of a widget respect to other widgets:

- **android:layout_above** indicates that the widget should be placed above the widget referenced in the property
- **android:layout_below** indicates that the widget should be placed below the widget referenced in the property
- **android:layout_toLeftOf** indicates that the widget should be placed to the left of the widget referenced in the property
- **android:layout_toRightOf** indicates that the widget should be placed to the right of the widget referenced in the property

20



Basic XML Layouts - Containers

2. Relative Layout – Referring to other widgets – cont.

- **android:layout_alignTop** indicates that the widget's top should be aligned with the top of the widget referenced in the property
- **android:layout_alignBottom** indicates that the widget's bottom should be aligned with the bottom of the widget referenced in the property
- **android:layout_alignLeft** indicates that the widget's left should be aligned with the left of the widget referenced in the property
- **android:layout_alignRight** indicates that the widget's right should be aligned with the right of the widget referenced in the property
- **android:layout_alignBaseline** indicates that the baselines of the two widgets should be aligned

21



Basic XML Layouts - Containers

2. Relative Layout – Referring to other widgets

In order to use Relative Notation in Properties you need to consistently:

1. Put identifiers (**android:id** attributes) on *all elements* that you will need to address. Syntax is: **@+id/...** (for instance an EditText box could be XML called: **android:id="@+id/ediUserName"**)
2. Reference other widgets using the same identifier value (**@+id/...**) already given to a widget. For instance a control below the EditText box could say: **android:layout_below="@+id/ediUserName"**

22



5. Android – UI – Basic XML Layouts

Basic XML Layouts - Containers

2. Relative Layout – Referring to other widgets

In order to use Relative Notation in Properties you need to consistently:

1. Put identifiers (`android:id` attributes) on *all elements* that you will need to address. Syntax is: `@+id/...` (for instance an EditText box could be XML called: `android:id="@+id/ediUserName"`)
2. Reference other widgets using the same identifier value (`@+id/...`) already given to a widget. For instance a control below the EditText box could say: `android:layout_below="@+id/ediUserName"`

23



5. Android – UI – Basic XML Layouts

Basic XML Layouts - Containers

2. Relative Layout – Example

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout
    android:id="@+id/myRelativeLayout"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:background="#ffff0099"
    xmlns:android="http://schemas.android.com/apk/res/android">

    <EditText
        android:id="@+id/ediUserName"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:layout_below="@+id/lblUserName"
        android:layout_alignParentLeft="true"
        android:layout_alignLeft="@+id/myRelativeLayout"
        android:padding="20px">
    </EditText>

    <TextView
        android:id="@+id/lblUserName"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:background="#ffff0066"
        android:text="User Name"
        android:textStyle="bold"
        android:textColor="#ff000000"
        android:layout_alignParentTop="true"
        android:layout_alignParentLeft="true">
    </TextView>

    <Button
        android:id="@+id	btnGo"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_below="@+id/ediUserName"
        android:layout_alignRight="@+id/ediUserName"
        android:text="Go"
        android:textStyle="bold">
    </Button>

    <Button
        android:id="@+id btnCancel"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_toLeftOf="@+id	btnGo"
        android:layout_below="@+id/ediUserName"
        android:text="Cancel"
        android:textStyle="bold">
    </Button>
</RelativeLayout>
```



24

2. Relative Layout – Comment (as of Aug. 2009)
 Use the **Eclipse ADT Layout Editor** for laying out *RelativeLayouts*.
DroidDraw is of very little help in this respect.

layout/main.xml - Eclipse

Properties

Property	Value
password	Phone number
single line	
layout_width	wrap_content
layout_height	wrap_content
layout_align_right	@+id/edUserName
layout_align_top	@+id/edUserName
layout_centerHorizontal	
layout_margin	
layout_marginBottom	
layout_marginLeft	
layout_marginRight	
layout_marginTop	
layout_toLeftOf	

25

Basic XML Layouts - Containers

3. Table Layout

1. Android's **TableLayout** allows you to position your widgets in a grid made of identifiable *rows* and *columns*.
2. Columns might *shrink* or *stretch* to accommodate their contents.
3. TableLayout works in conjunction with **TableRow**.
4. TableLayout controls the overall behavior of the container, with the widgets themselves positioned into one or more *TableRow* containers, one per row in the grid.

26



5. Android – UI – Basic XML Layouts

Basic XML Layouts - Containers

3. Table Layout

Rows are declared by you by putting widgets as children of a **TableRow** inside the overall *TableLayout*.

The **number of columns is determined by Android** (you control the number of columns in an indirect way).

So if you have three rows, one with two widgets, one with three widgets, and one with four widgets, there will be at least four columns.

0	1	2	3

27



5. Android – UI – Basic XML Layouts

Basic XML Layouts - Containers

3. Table Layout

However, a single widget can take up more than one column by including the **android:layout_span** property, indicating the number of columns the widget spans (this is similar to the **colspan** attribute one finds in table cells in **HTML**)

```
<TableRow>
    <TextView android:text="URL:" />
    <EditText
        android:id="@+id/entry"
        android:layout_span="3" />
</TableRow>
```

28

 5. Android – UI – Basic XML Layouts

Basic XML Layouts - Containers

3. Table Layout

Ordinarily, widgets are put into the first available column of each row.

In the previous fragment, the label ("URL") would go in the first column (column 0, as columns are counted starting from 0), and the TextField would go into a spanned set of three columns (columns 1 through 3).

				<code>android:layout_span="3"</code>
Label (URL)	EditText	EditText-span	EditText-span	
<i>Column 0</i>	<i>Column 1</i>	<i>Column 2</i> Button Cancel	<i>Column 3</i> Button OK	

← →

← →

`← android:layout_columns="2" →`

29

 5. Android – UI – Basic XML Layouts

Basic XML Layouts - Containers

3. Table Layout – Example



```

<?xml version="1.0" encoding="utf-8"?>
<TableLayout
    android:id="@+id/myTableLayout"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:background="#ff0033cc"
    android:orientation="vertical"
    xmlns:android="http://schemas.android.com/apk/res/android"
    >
    <TableRow>
        <TextView
            android:text="URL:" />
        <EditText
            android:id="@+id/ediUrl"
            android:layout_span="3"/>
    </TableRow>
    <View
        android:layout_height="3px"
        android:background="#0000FF" />
    <TableRow>
        <Button
            android:id="@+id/cancel"
            android:layout_column="2"
            android:text="Cancel" />
        <Button
            android:id="@+id/ok"
            android:text="OK" />
    </TableRow>
    <View
        android:layout_height="3px"
        android:background="#0000FF" />
</TableLayout>

```

← →

← →

← →

 5. Android – UI – Basic XML Layouts

Basic XML Layouts - Containers

3. Table Layout

By default, each column will be sized according to the "natural" size of the widest widget in that column.

If your content is narrower than the available space, you can use the *TableLayout* property:

```
    android:stretchColumns = "..."
```

Its value should be a single column number (0-based) or a comma-delimited list of column numbers. Those columns will be stretched to take up any available space yet on the row.

31

 5. Android – UI – Basic XML Layouts

Basic XML Layouts - Containers

3. Table Layout

In our running example we stretch columns 2, 3, and 4 to fill the rest of the row.

```
...
<TableLayout
    android:id="@+id/myTableLayout"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:background="#ff0033cc"
    android:orientation="vertical"
    android:stretchColumns="2,3,4"
    xmlns:android="http://schemas.android.com/apk/res/android"
>
...
```



TODO: try to stretch one column at the time 1, then 2, and so on.

32



5. Android – UI – Basic XML Layouts

Basic XML Layouts - Containers

4. ScrollView Layout

When we have more data than what can be shown on a single screen you may use the **ScrollView** control.

It provides a sliding or scrolling access to the data. This way the user can only see part of your layout at one time, but the rest is available via scrolling.

This is similar to browsing a large web page that forces the user to scroll up the page to see the bottom part of the form.

33



5. Android – UI – Basic XML Layouts

Basic XML Layouts - Containers

4. Example ScrollView Layout

```
<?xml version="1.0" encoding="utf-8"?>
<ScrollView
    android:id="@+id/widget28"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:background="#ff009999"
    xmlns:android="http://schemas.android.com/apk/res/android"
    >
    <LinearLayout
        android:id="@+id/myLinearLayoutVertical"
        android:layout_width="fill_parent"
        android:layout_height="fill_parent"
        android:orientation="vertical"
        >
        <LinearLayout
            android:id="@+id/myLinearLayoutHorizontal1"
            android:layout_width="fill_parent"
            android:layout_height="fill_parent"
            android:orientation="horizontal"
            >
                <ImageView
                    android:id="@+id/myPicture"
                    android:layout_width="wrap_content"
                    android:layout_height="wrap_content"
                    android:src="@drawable/icon" />
                <TextView
                    android:id="@+id/textView1"
                    android:layout_width="fill_parent"
                    android:layout_height="wrap_content"
                    android:text="Line1"
                    android:textSize="70px" />
            </LinearLayout>
            <TextView
                android:layout_width="fill_parent"
                android:layout_height="6px"
                android:background="#ffccffcc" />
            <TextView
                android:id="@+id/textView3"
                android:layout_width="fill_parent"
                android:layout_height="wrap_content"
                android:text="Line3"
                android:textSize="70px" />
            <TextView
                android:layout_width="fill_parent"
                android:layout_height="6px"
                android:background="#ffccffcc" />
            <TextView
                android:id="@+id/textView5"
                android:layout_width="fill_parent"
                android:layout_height="wrap_content"
                android:text="Line5"
                android:textSize="70px" />
        </LinearLayout>
    </ScrollView>
```

5. Android – UI – Basic XML Layouts

Basic XML Layouts - Containers

4. Example ScrollView Layout

Combining an ImageView & TextView in a horizontal Linear Layout

Scroller

35

5. Android – UI – Basic XML Layouts

Basic XML Layouts - Containers

4. ScrollView Layout - Example

```
<?xml version="1.0" encoding="utf-8"?>
<ScrollView
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content">
    <TableLayout
        android:layout_width="fill_parent"
        android:layout_height="fill_parent"
        android:stretchColumns="0">
        <TableRow>
            <View
                android:layout_height="80px"
                android:background="#000000"/>
            <TextView android:text="#000000"
                android:paddingLeft="4px"
                android:layout_gravity="center_vertical" />
        </TableRow>
        <TableRow>
            <View
                android:layout_height="80px"
                android:background="#440000"/>
            <TextView android:text="#440000"
                android:paddingLeft="4px"
                android:layout_gravity="center_vertical" />
        </TableRow>
        <TableRow>
            <View
                android:layout_height="80px"
                android:background="#884400"/>
            <TextView android:text="#884400"
                android:paddingLeft="4px"
                android:layout_gravity="center_vertical" />
        </TableRow>
        <TableRow>
            <View
                android:layout_height="80px"
                android:background="#aa8844"/>
            <TextView android:text="#aa8844"
                android:paddingLeft="4px"
                android:layout_gravity="center_vertical" />
        </TableRow>
        <TableRow>
            <View
                android:layout_height="80px"
                android:background="#ffaa88"/>
            <TextView android:text="#ffaa88"
                android:paddingLeft="4px"
                android:layout_gravity="center_vertical" />
        </TableRow>
        <TableRow>
            <View
                android:layout_height="80px"
                android:background="#ffffaa"/>
            <TextView android:text="#ffffaa"
                android:paddingLeft="4px"
                android:layout_gravity="center_vertical" />
        </TableRow>
        <TableRow>
            <View
                android:layout_height="80px"
                android:background="#ffffff"/>
            <TextView android:text="#ffffff"
                android:paddingLeft="4px"
                android:layout_gravity="center_vertical" />
        </TableRow>
    </TableLayout>
</ScrollView>
```

5. Android – UI – Basic XML Layouts

Basic XML Layouts - Containers

4. ScrollView Layout - Example



The screenshot shows a vertical stack of five colored rectangular blocks. From top to bottom, the colors are black (#000000), light gray (#eeeeee), yellow (#ffffaa), orange (#ffaa88), and brown (#aa8844). Each block has a color hex code label to its right. The entire stack is contained within a scrollable view, indicated by a vertical scroll bar on the right side of the screen. The title bar at the top says "AndDemoUI2". The status bar shows signal strength, battery level, and the time "2:21 AM".

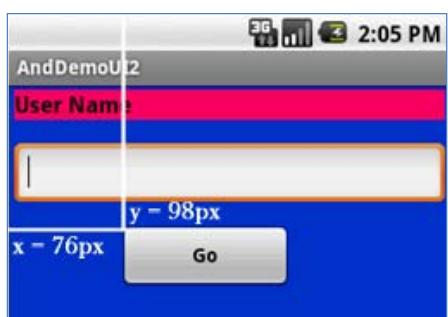
37

5. Android – UI – Basic XML Layouts

Basic XML Layouts - Containers

5. Miscellaneous. Absolute Layout

- A layout that lets you specify exact locations (x/y coordinates) of its children.
- Absolute layouts are *less flexible* and harder to maintain than other types of layouts without absolute positioning.



The screenshot displays a user interface with a red header bar containing the text "User Name". Below it is a white input field with a black border and a small cursor inside. To the right of the input field is a blue rectangular button with the word "Go" in white. The background of the screen is blue. Two coordinate values are overlaid on the screen: "x = 76px" pointing to the left edge of the input field, and "y = 98px" pointing to the top edge of the same field. The title bar at the top says "AndDemoUI2". The status bar shows signal strength, battery level, and the time "2:05 PM".

38



5. Android – UI – Basic XML Layouts

Basic XML Layouts - Containers

5. Miscellaneous Absolute Layout (cont.)

```

<?xml version="1.0" encoding="utf-8"?>
<AbsoluteLayout
    android:id="@+id/myLinearLayout"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:background="#ff0033cc"
    android:padding="4px"
    xmlns:android="http://schemas.android.com/apk/res/android"
    >

    <TextView
        android:id="@+id/labelUserName"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:background="#ffff0066"
        android:text="User Name"
        android:textSize="16sp"
        android:textStyle="bold"
        android:textColor="#ff000000"
        android:layout_x="0px"
        android:layout_y="-1px"
        >
    
```

```

        </TextView>
        <EditText
            android:id="@+id/ediName"
            android:layout_width="fill_parent"
            android:layout_height="wrap_content"
            android:textSize="18sp"
            android:layout_x="0px"
            android:layout_y="38px"
            >
        </EditText>

        <Button
            android:id="@+id/btnGo"
            android:layout_width="125px"
            android:layout_height="wrap_content"
            android:text="Go"
            android:textStyle="bold"
            android:layout_x="76px"
            android:layout_y="98px"
            >
        </Button>
    
```

</AbsoluteLayout>



Button location



5. Android – UI – Basic XML Layouts

Basic XML Layouts - Containers

Questions?

40