Layouts

Dr Tadhg O'Sullivan

Layouts?

- In Android, Layouts define the visual structure for user interfaces
- Layouts can be declared in two ways:
 - Declared UI elements in XML
 - Instantiated at runtime

XML vs Code

In other words:

- with XML you create your UI that directly corresponds to the View classes and subclasses automatically
- or you create View and ViewGroup objects manually in your code
- => the XML way is the currently more popular but sometimes coding the layout makes more sense

Why XML?

- XML usage for layouts is not required, but highly recommended!
- Advantages:
 - GUI-assisted creation
 - Separation of generated vs hand-written code
 - Allows a UI specialist to work on layout without having to worry about underlying code

Implications?

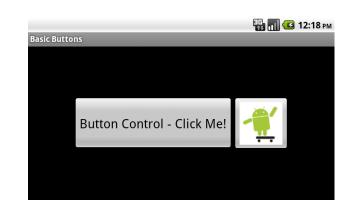
- XML is compiled into very efficient form for runtime, so performance is not a problem
- Your interfaces should be known beforehand.
 Manipulating XML interface at runtime is tricky.

How to Attach the XML layout?

```
In your activity:
setContentView(R.layout.main);
i.e.
@Override
public void onCreate(Bundle icicle) {
super.onCreate(icicle);
setContentView(R.layout.main);
```

Sample Layout (Button)

- <?xml ... ?>
- <Button xmlns:android="..."
- android:id="@+id/button"
- android:text=""
- android:layout_width="fill_parent"
- android:layout_height="fill_parent"
- />



Text View

This is a 'serif' demo!

This is a 'normal' demo!

This is a 'monospace' demo!

```
<?xml version="1.0" encoding="utf8"?>
<TextView xmlns:android="..."
android:layout width="fill parent"
android:layout height="wrap content"
android:text="You were expecting
something profound?"
                            TypographyDemo
/>
                            This is a 'sans' demo!
```

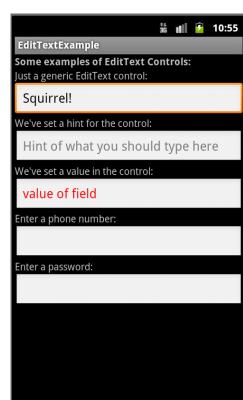
ImageView

```
<?xml version="1.0" encoding="utf8"?>
<ImageView xmlns:android="..."</pre>
android:id="@+id/icon"
android:layout_width="fill parent"
android:layout height="fill parent"
android:adjustViewBounds="true"
android:src="@drawable/molecule"
/>
```



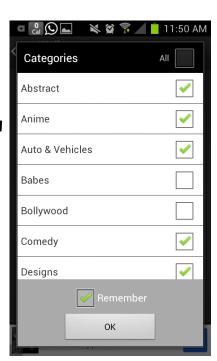
EditView

```
<?xml version="1.0" encoding="utf8"?>
<EditText xmlns:android="..."
android:id="@+id/field"
android:layout width="fill parent" *
android:layout height="fill parent"
android:singleLine="false"
/> * note on fill parent
```



CheckBox

```
<?xml version="1.0" encoding="utf8"?>
<CheckBox xmlns:android="..."</pre>
android:id="@+id/check"
android:layout width="wrap content"
android:layout height="wrap_content"
android:text="This checkbox is:
unchecked" />
```



RadioButton and RadioGroup

< Radio Button

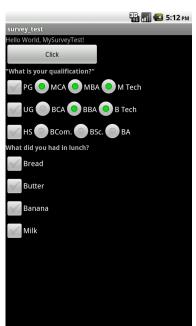
android:id="@+id/radio1"

android:layout_width="wrap_content"

android:layout_height="wrap_content"

android:text="Rock" />

Part of RadioGroup



RadioGroup

```
<?xml version="1.0" encoding="utf8"?>
< Radio Group
xmlns:android="..."
android:orientation="vertical"
android:layout width="fill parent"
android:layout height="fill parent"
> ... add radio buttons here
</RadioGroup>
```

Views

- All of the above widgets extend Views.
- Useful properties:
 - android:nextFocusDown
 - android:nextFocusLeft
 - android:nextFocusRight
 - android:nextFocusUp
 - android:visibility

Views Cont'd

- Useful Methods:
 - getParent(): Finds the parent widget or container.
 - findViewById(): Finds a child widget with a certain
 ID.
 - getRootView(): Gets the root of the tree (e.g., what you provided to the activity via setContentView()).

Containers

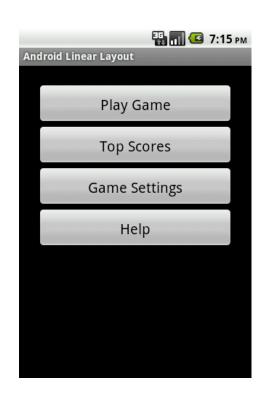
- Container is a collection of widgets and child containers
- Similar to BoxLayout in Swing

LinearLayout

- Resembles Swing's BoxLayout
- Properties
 - Orientation: horizontal, vertical
 - Match model: width, height, wrap, fill
 - Weight: which widget has more of it?
 - Gravity: alignment
 - Padding: top, bottom, left, right...

LinearLayout Example

```
<?xml version="1.0" encoding="utf8"?>
<LinearLayout xmlns:android="..."</pre>
android:orientation="vertical"
android:layout width="fill parent"
android:layout height="fill parent"
> ... your widgets here...
</LinearLayout>
```



RelativeLayout

- Layouts widgets based on their relationship to other widgets in the container and the parent container
- To make your RelativeLayout work, you need ways to reference other widgets within an XML layout file, plus ways to indicate the relative positions of those widgets.

RelativeLayout Example

₩ m @ 8:54 pr

```
<?xml version="1.0" encoding="utf-8"?>
                                               Relative Layout Examples
<RelativeLayout
xmlns:android="..."
android:layout width="fill parent"
android:layout_height="wrap_content"
android:padding="5px">
<TextView android:id="@+id/label"
android:layout width="wrap content"
android:layout height="wrap content"
android:text="URL:"
android:paddingTop="15px"/> ... </RelativeLayout>
```

TableLayout

 Table Layout works in conjunction with TableRow. TableLayout controls the overall behavior of the container, with the widgets themselves poured into one or more TableRow containers, one per row in the grid.

TableRow Example

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

TableLayout Example

3:37 PM

Add row

```
<?xml version="1.0" encoding="utf8"?>
<TableLayout
xmlns:android="..."
android:layout_width="fill_parent"
android:layout_height="fill_parent"
android:stretchColumns="1">
```

<TableRow>...

ScrollView

- ScrollView is a container that provides scrolling for its contents. You can take a layout that might be too big for some screens, wrap it in a ScrollView
- There is HorizontalScrollView as well...

ScrollView Example

```
<?xml version="1.0" encoding="utf8"?>
<ScrollView
xmlns:android="..."
android:layout width="fill parent"
android:layout height="wrap content">
<TableLayout
android:layout ...
```



Welcome to My Application

Lorem ipsum dolor sit amet. Nam et ellt est, sit amet vestibulum turpis. In hac habitasse platea dictumst. Nulla venenatis faucibus mi, in sodales purus consequat vehicula. Nunc sed libero nibh. Etiam felis odio, feuglat a lacinia quis, rutrum in ante. Maecenas elementum, lorem ut malesuada consequat, sem tellus auctor arcu, non volutpat diam libero blandit metus. Quisque fermentum nibh vitae turpis hendrerit a aliquam sapien sodales. Phasellus vulputate enim eget nunc placerat eget laculis nisi tempus. Etiam diam lorem, feuglat ac porttitor a, euismod sed velit. Pellentesque venenatis fermentum pellentesque. Mauris auctor leo sit amet erat aliquam gravida. Nullam tempus tincidunt mi id dignissim. Sed porta sem at risus hendrerit malesuada. Duis porta sem at risus hendrerit malesuada. Duis porttitor mauris ac purus tristique convallis. Mauris accumsan metus a felis tempor adipiscing. Pellentesque rhoncus scelerisque odio vitae ornare. Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nunc non ipsum neque. In condimentum tellus a est suscipit ac

Questions

Please ask in the Student Forum