Building a Dashboard

Steven R. Bagley

Previously…

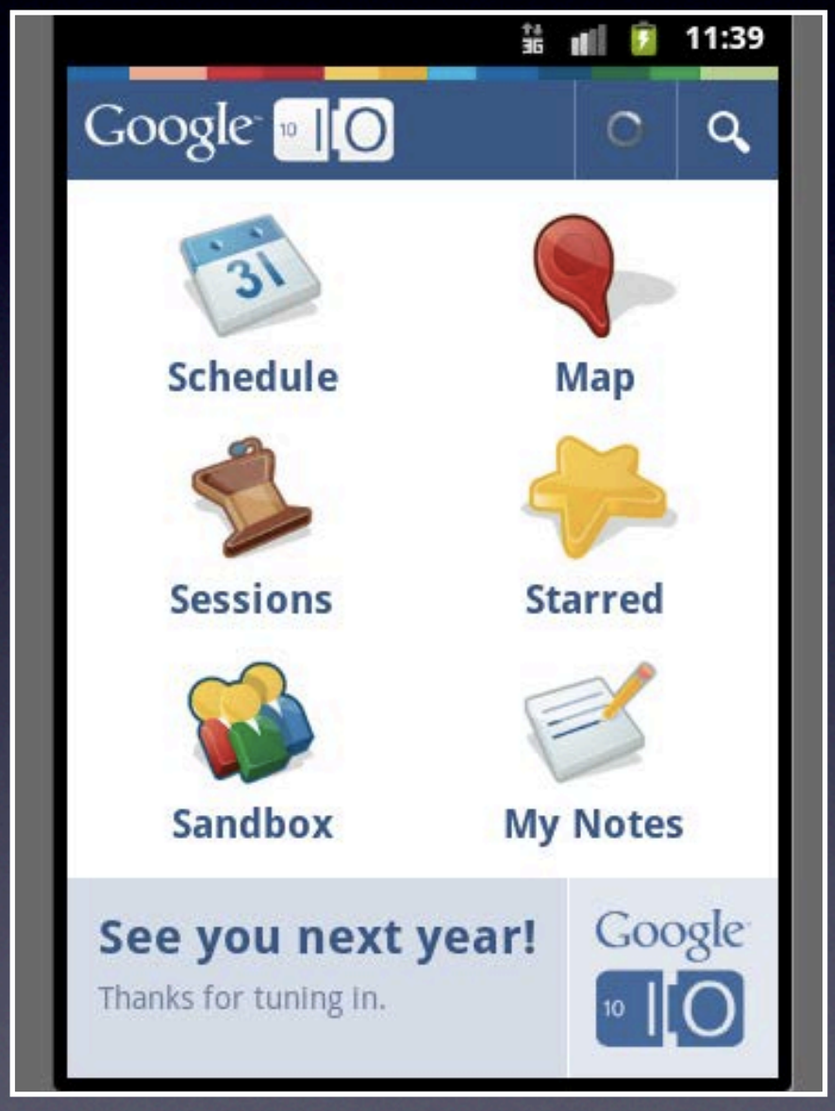
- Seen how to put together an Activity
- Watched video on common UI Design Patterns

Today…

- See how we can implement one of those design patterns
- Look at styling Android UIs in detail…

Dashboard

- Home screen
- Provides a quick way to get to the things you want to do often
- Probably only about 6 options
- How do we implement?



Dashboard

- Relatively simple design
- 6 buttons — one for each option
- Clicking a button takes us to the relevant activity (call `Context.startActivity()`)
- Buttons laid out in a grid…
- Vast majority of the implementation is in the XML, not the code…

Dashboard Layout

- Could use a `TableLayout`
- Or we could form it using `LinearLayout`s
- One vertically
- Containing three horizontal ones
- Each horizontal one contains two `Button`s
- Customize the `Button`s to show icons

Layout Weights

- `layout_weight` attribute is a method to specify how much space a widget takes up
- If all elements have the same weight then they all get the same amount of space
- Otherwise they set the ratio of size between components
- With smaller numbers getting the most space

Styling widgets

- Android lets us style the `Button`s
- By placing attributes on the elements in the XML
- `android:drawableTop` lets us specify the icon for instance
- Can read about the various attributes in the SDK

Styles

- But all our buttons will have a lot of style in common…
- Android has a CSS-like function that lets us define a style once to use multiple times…
- These are specified in the `styles.xml` resource
- Referenced using the `style` attribute

styles.xml

```
<resources>

    <style name="HomeButton">
        <item name="android:layout_width">fill_parent</item>
        <item name="android:layout_height">wrap_content</item>
        <item name="android:textColor">#ff355689</item>
        <item name="android:background">@null</item>
        <item name="android:gravity">center_horizontal</item>
        <item name="android:layout_gravity">center_vertical</item>
        <item name="android:layout_weight">1</item>
    </style>

</resources>
```

Relatively simple syntax, each attribute is defined as an item element with the atts value as the element content
also possible to inherit styles using @parent on the style element

Gravity

- The `layout_gravity` and and `gravity` attributes set the position of a widget
- In relation to its parent
- See the SDK for details…

Lets add the style to our buttons…

Icons

- Button background has vanished due to `android:background` attribute
- Need to specify the icon
- Image stored in `drawable` folder (as a `.PNG`)
- Use `android:drawableTop` to draw it…
- But it doesn't quite work…

Button State

- Actually it does exactly what we've told it
- Which is to display this image to represent the button
- Ideally, though the image should change as we interact with it (click it)
- Need to specify different images depending on the `Button`'s state…

State List

- This is done by using a State List
- This specifies the image to use for the different states the button can be in
- Pressed, has Focus, etc.

- Again specified in an XML file
- Point the `Button`'s `drawableTop` at the XML not directly at the image

State List

```
<selector
    xmlns:android="http://schemas.android.com/apk/res/android">
   <item android:state_pressed="true"
       android:drawable="@drawable/mail_pressed" />
   <item android:state_pressed="false"
       android:drawable="@drawable/mail_norm" />
</selector>
```

Again a simple enough syntax, just specify the states you are interested in and the icon to display
Can combine them (e.g. when state_pressed is true and state_focused is false

Dashboard

- There we have it — a working dashboard
- All that's left is to
- Neaten up the design…
- Link the `Button`s to methods that `startActivity()`s

Button clicking

- Previously done this by `setOnClickListener` on the `Button`
- And implementing `View.OnClickListener`
- There's a simpler way…

- Can specify an `android:onClick` attribute
- That points at the method in the `Activity` to execute

Go give example

Fitting the Width

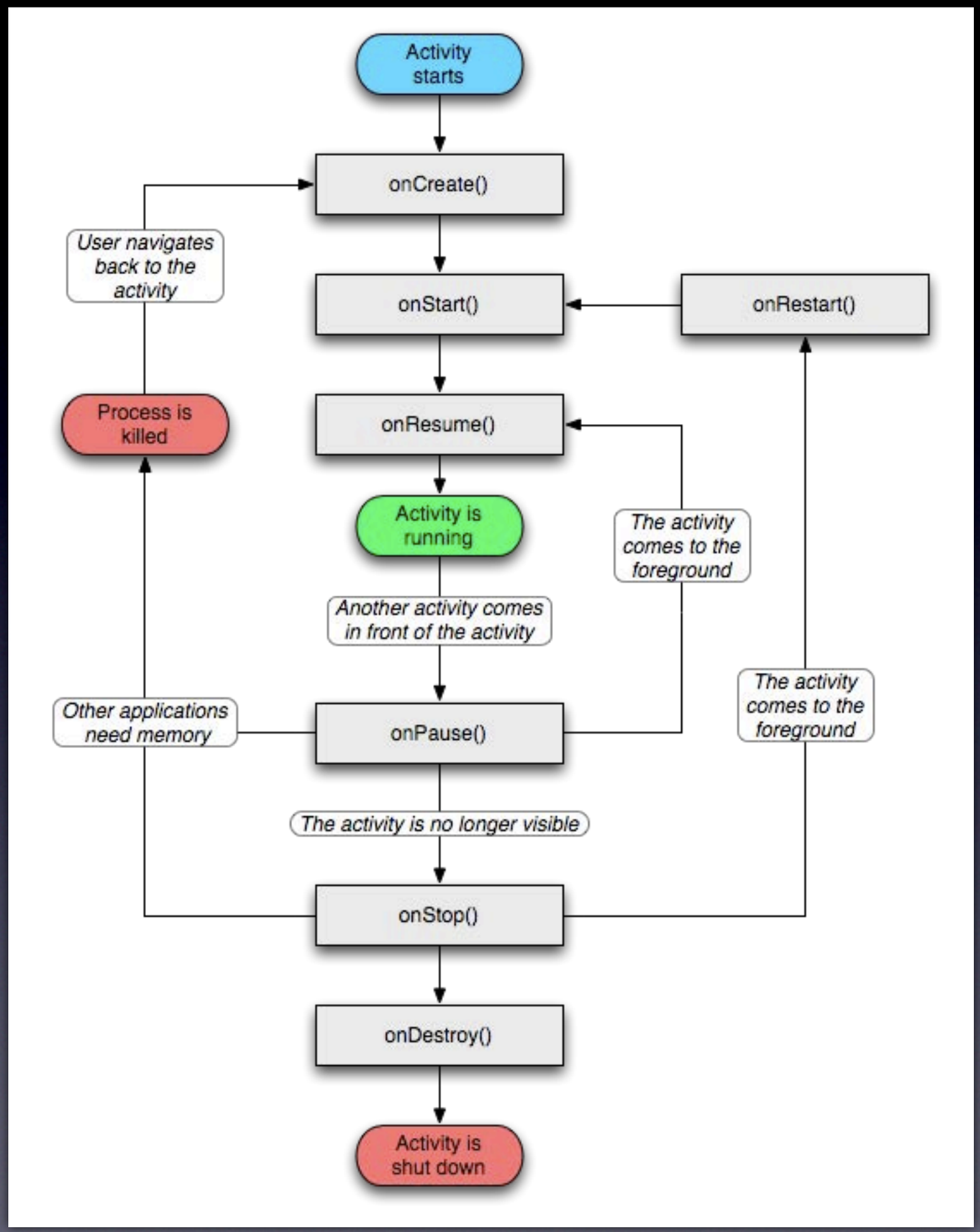
- Our ‘app’ doesn’t fill the display
- Need to tell Android that our app supports multiple screen sizes
- Done in the manifest file with `<support-screens>`
- See `http://developer.android.com/guide/topics/manifest/supports-screens-element.html`

MVC

- Very easy to see how Android UI utilises the Model-View Controller pattern
- View — defined in XML or programatically
- Controller — the role of the `Activity`
- Model — the data the app acts on…

Starting Activities

- Seen how we can call `startActivity()` or `startActivityForResult()` to start a new `Activity`
- This causes various methods in the `Activity` lifecycle to be called
- But what is the exact order they are called?
- Can test this experimentally…



Can do this by logging when the various methods are called from the two Activities are called then seeing the overlap in the logfile

Experimental Activity

- Implement all the methods
- Use Android's logging (`Log.i(…)`facilities to log when they are called
- Then look at the device logs to see the order…