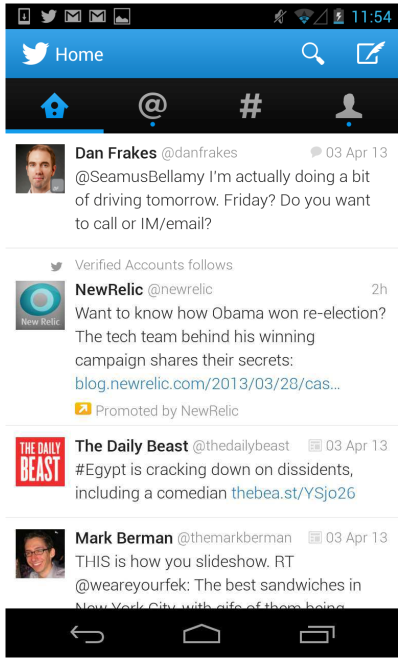
# 13. Generating a Dynamic UI

# Generating a UI at runtime

- Sometimes your app's UI cannot be fully specified in XML.
  - Example: You don't know how many widgets you will need until the user gives input or until a file is downloaded.
- In these cases, your app needs to be able to generate UI widgets dynamically in Java code.


11:54

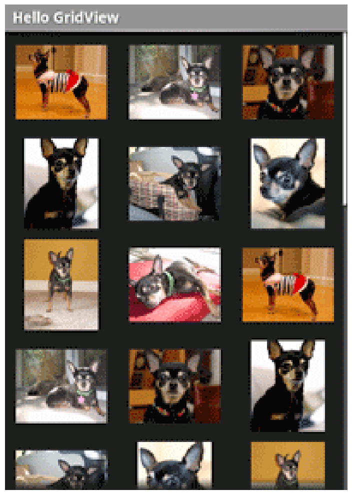
Home

Dan Frakes @danfrakes 03 Apr 13
@SeamusBellamy I'm actually doing a bit of driving tomorrow. Friday? Do you want to call or IM/email?

Verified Accounts follows
NewRelic @newrelic 2h
Want to know how Obama won re-election? The tech team behind his winning campaign shares their secrets: blog.newrelic.com/2013/03/28/cas...
Promoted by NewRelic

The Daily Beast @thedailybeast 03 Apr 13
#Egypt is cracking down on dissidents, including a comedian thebea.st/YSjo26

Mark Berman @themarkberman 03 Apr 13
THIS is how you slideshow. RT @weareyourfek: The best sandwiches in



# UI Widget objects

- Any UI widget class from XML has a corresponding Java class.
- You already used these when you find a view by ID.

```
// inside an activity class
WidgetType name = new WidgetType(this);
```

- Example:

```
TextView tv = new TextView(this);
```

## Adding widget to layout

- You can add a widget to an onscreen container ([ViewGroup](ViewGroup)) such as a layout.
  - Add a widget to a container using the addView method.
  - You must give the container an ID.

```
<!-- activity_main.xml -->
<LinearLayout android:id="@+id/mainlayout" ...>

// MainActivity.java
TextView tv = new TextView(this);
LinearLayout layout = (LinearLayout)
findViewById(R.id.mainlayout); layout.addView(tv);
```

# ViewGroup methods

- `addView(`*`view`*`);`
  `addView(view, `*`index`*`);`
  `addView(view, `*`params`*`);` add a view to this container
- `bringChildToFront(`*`view`*`);` move view to top of Z-order
- `getChildAt(`*`index`*`)` return a view
- `getChildCount()` return number of children
- `removeAllViews();` remove all children
- `removeView(`*`view`*`);` remove a particular child
- `removeViewAt(`*`index`*`);` remove child at given index

## Widget parameters

- What about setting attributes that would have been inside the XML tag?
- Some are just set methods on the widget object itself.

```
<!-- activity_main.xml -->
<TextView
android:id="@+id/mymessage"
android:text="Hello there!"
android:textSize="20dp"
android:textStyle="bold"
android:layout_width="wrap_content"
android:layout_height="wrap_content" />

//MainActivity.java
TextView tv = new TextView(this);
tv.setId(R.id.mymessage); // or use your own
number
tv.setText("Hello there!");
tv.setTextSize(20); ...
```

# Layout parameters

- Attributes that start with layout_ are for the layout.
- These are packaged into an internal LayoutParams object.

```
<!-- activity_main.xml -->
<TextView android:id="@+id/mymessage"
android:text="Hello there!"
android:textSize="20dp"
android:textStyle="bold"
android:layout_width="wrap_content"
android:layout_height="wrap_content" />

//MainActivity.java
TextView tv = new TextView(this);
ViewGroup.LayoutParams params = new
ViewGroup.LayoutParams(
ViewGroup.LayoutParams.WRAP_CONTENT, // width
ViewGroup.LayoutParams.WRAP_CONTENT); // height
tv.setLayoutParams(params);
```

## Layout-specific parameters

- Each layout type has its own LayoutParams inner class.
  - Contains attributes and methods used by that kind of layout.

- Example for LinearLayout:

```
LinearLayout.LayoutParams params = new
LinearLayout.LayoutParams(
ViewGroup.LayoutParams.MATCH_PARENT, // width
ViewGroup.LayoutParams.WRAP_CONTENT); // height
params.weight = 1;
params.gravity = Gravity.TOP | Gravity.CENTER;
```

# Setting widget size

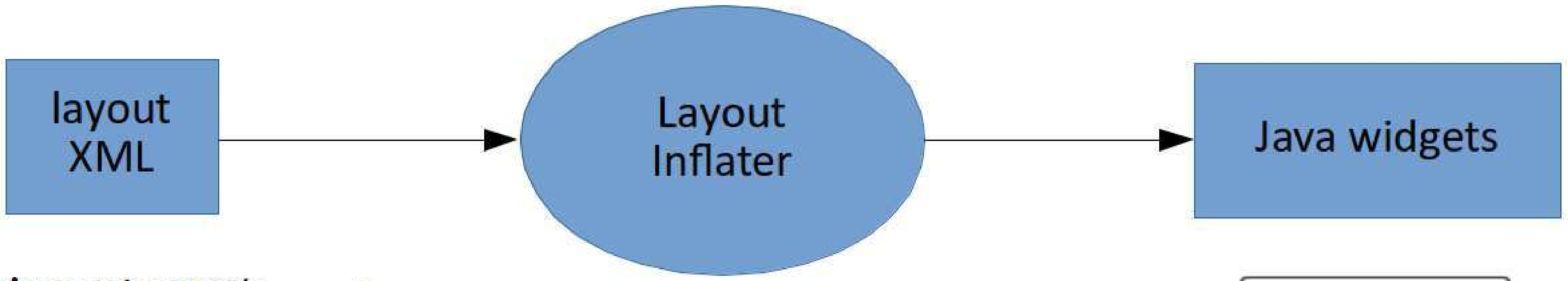
- Most common sizes are wrap_content and match_parent.
  - ViewGroup.LayoutParams.WRAP_CONTENT
  - ViewGroup.LayoutParams.MATCH_PARENT

- If you want to set width that is relative to the screen size:

```
Display display =
getWindowManager().getDefaultDisplay();
Point size = new Point();
display.getSize(size);
int screenWidth = size.x;
int screenHeight = size.y;
LinearLayout.LayoutParams params = new
LinearLayout.LayoutParams( screenWidth / 2,
// width = half of screen
screenHeight / 2); // height = half of screen
```

# Layout inflater

- **layout inflater**: Converts layout XML into Java widget objects.
  - Manual creation of widgets works, but it is pretty painful if you are creating a lot of them, or a complex nested structure of widgets.
  - A layout inflater lets you specify an entire chunk of layout, perhaps a complex subcomponent, as XML and then load it in Java as needed.
  - Similar to a fragment but without its own events and lifecycle.



```
<LinearLayout ...>
    <ImageView ... />
    <TextView ... />
</LinearLayout>
```

# Using the layout inflater

- Inside an activity:

```
View name = getLayoutInflater()
     .inflate(R.layout.name, parent);
```

- When not in an activity:

```
LayoutInflater inflater = (LayoutInflater)
     context.getSystemService(Context.LAYOUT_INFLA
TER_SERVICE);
View name = inflater.inflate(R.layout.name,
parent);
```

- in both cases, parent can be null
- if parent is non-null, new view is automatically added to parent