**Resources Overview**

You should always externalize resources such as images and strings from your application code, so that you can maintain them independently.

Externalizing your resources also allows you to provide alternative resources that support specific device configurations such as different languages or screen sizes, which becomes increasingly important as more Android-powered devices become available with different configurations.

In order to provide compatibility with different configurations, you must organize resources in your project's **res/directory,** using various sub-directories that group resources by type and configuration.

**String**

A single string that can be referenced from the application or from other resource files (such as an XML layout).

**FILE LOCATION:**

res/values/*filename*.xml  
The filename is arbitrary. The <string> element's name will be used as the resource ID.

**RESOURCE REFERENCE:**

In Java: R.string.*string\_name*  
In XML:@string/*string\_name*

SYNTAX:

<[resources](http://developer.android.com/guide/topics/resources/string-resource.html#string-resources-element)>  
    <[string](http://developer.android.com/guide/topics/resources/string-resource.html#string-element) name="*string\_name*"> *text\_string* </string>  
</resources>

**String Array**

An array of strings that can be referenced from the application.

FILE LOCATION:

res/values/*filename*.xml  
The filename is arbitrary. The <string-array> element's name will be used as the resource ID.

RESOURCE REFERENCE:

In Java: R.array.*string\_array\_name*

SYNTAX:

<[resources](http://developer.android.com/guide/topics/resources/string-resource.html#string-array-resources-element)>  
    <[string-array](http://developer.android.com/guide/topics/resources/string-resource.html#string-array-element) name="*string\_array\_name*">

        <[item](http://developer.android.com/guide/topics/resources/string-resource.html#string-array-item-element)> *text\_string* </item>  
    </string-array>  
</resources>

**Snippet In java**

Resources r = getResources(); *// finding Resources Ref*String items[] = r.getStringArray(R.array.***languages***);

**Quantity Strings (Plurals)**

Different languages have different rules for grammatical agreement with quantity. In English, for example, the quantity 1 is a special case. We write "1 book", but for any other quantity we'd write "*n* books". This distinction between singular and plural is very common, but other languages make finer distinctions. The full set supported by Android is zero,one, two, few, many, and other.

**FILE LOCATION:**

res/values/*filename*.xml  
The filename is arbitrary. The <plurals> element's name will be used as the resource ID.

**RESOURCE REFERENCE:**

In Java: R.plurals.*plural\_name*

**Example**

*<****plurals name="qty"****>  
 <****item quantity="one"****> Book </****item****>  
 <****item quantity="other"****> %d Books</****item****>  
</****plurals****>*

**Java Snippet**

*EditText et = (EditText)findViewById(R.id.****editText****);  
String qty = et.getText().toString().trim();****int*** *no = Integer.parseInt(qty);  
  
Resources r = getResources();  
String value = r.getQuantityString(R.plurals.****qty****,no,no);  
  
Toast.makeText(MainActivity.****this****,* ***""****+value, Toast.****LENGTH\_SHORT****).show();*

**Escaping apostrophes and quotes**

*If you have an apostrophe (') in your string, you must either escape it with a backslash (\') or enclose the string in double-quotes ("").*

*For example, here are some strings that do and don't work:*

*<string name="good\_example">This\'ll work</string>  
<string name="good\_example\_2">"This'll also work"</string>  
<string name="bad\_example">This doesn't work</string>*

*If you have a double-quote in your string, you must escape it (\"). Surrounding the string with single-quotes does not work.*

*<string name="good\_example">This is a \"good string\".</string>  
<string name="bad\_example">This is a "bad string".</string>  
    <!-- Quotes are stripped; displays as: This is a bad string. -->  
<string name="bad\_example\_2">'This is another "bad string".'</string>*

**Styling with HTML markup**

You can add styling to your strings with HTML markup. For example:

<?xml version="1.0" encoding="utf-8"?>  
<resources>  
    <string name="welcome">Welcome to <b>Android</b>!</string>  
</resources>

Supported HTML elements include:

* <b> for **bold** text.
* <i> for *italic* text.
* <u> for underline text.

**Color**

A color value defined in XML.

The color is specified with an RGB value and alpha channel.

The value always begins with a pound (#) character and then followed by the Alpha-Red-Green-Blue information in one of the following formats:

* #*RGB*
* #*ARGB*
* #*RRGGBB*
* #*AARRGGBB*

**FILE LOCATION:**

res/values/colors.xml  
The filename is arbitrary. The <color> element's name will be used as the resource ID.

**RESOURCE REFERENCE:**

In Java: R.color.*color\_name*  
In XML: @[*package*:]color/*color\_name*

**Color State List Resource**

A [ColorStateList](http://developer.android.com/reference/android/content/res/ColorStateList.html) is an object you can define in XML that you can apply as a color, but will actually change colors, depending on the state of the [View](http://developer.android.com/reference/android/view/View.html) object to which it is applied.

For example, a[Button](http://developer.android.com/reference/android/widget/Button.html) widget can exist in one of several different states (pressed, focused, or neither) and, using a color state list, you can provide a different color during each state

FILE LOCATION:

res/color/*filename*.xml  
The filename will be used as the resource ID.

RESOURCE REFERENCE:

In Java: R.color.*filename*  
In XML: @[*package*:]color/*filename*

**SYNTAX:**

<[selector](http://developer.android.com/guide/topics/resources/color-list-resource.html#selector-element) xmlns:android="http://schemas.android.com/apk/res/android" >  
    <[item](http://developer.android.com/guide/topics/resources/color-list-resource.html#item-element)  
        android:color="*hex\_color*"  
        android:state\_pressed=["true" | "false"]  
        android:state\_focused=["true" | "false"]  
        android:state\_selected=["true" | "false"]  
        android:state\_checkable=["true" | "false"]  
        android:state\_checked=["true" | "false"]  
        android:state\_enabled=["true" | "false"]  
        android:state\_window\_focused=["true" | "false"] />  
</selector>

**Style Resource**

A style resource defines the format and look for a UI.

A style can be applied to an individual [View](http://developer.android.com/reference/android/view/View.html) (from within a layout file) or to an entire [Activity](http://developer.android.com/reference/android/app/Activity.html) or application (from within the manifest file).

FILE LOCATION:

res/values/*filename*.xml  
The filename is arbitrary. The element's name will be used as the resource ID.

RESOURCE REFERENCE:

In XML: @[package:]style/*style\_name*

**Drawable Resources**

A drawable resource is a general concept for a graphic that can be drawn to the screen and which you can retrieve with [getDrawable(int)](http://developer.android.com/reference/android/content/res/Resources.html#getDrawable(int)) method or apply to another XML resource with attributes such as **android:drawable** and **android:icon**.

There are several different types of drawables:

1. [Bitmap File](http://developer.android.com/guide/topics/resources/drawable-resource.html#Bitmap)

A bitmap graphic file (.png, .jpg, or .gif). Creates a [BitmapDrawable](http://developer.android.com/reference/android/graphics/drawable/BitmapDrawable.html).

1. [Nine-Patch File](http://developer.android.com/guide/topics/resources/drawable-resource.html#NinePatch)

A PNG file with stretchable regions to allow image resizing based on content (.9.png). Creates a [NinePatchDrawable](http://developer.android.com/reference/android/graphics/drawable/NinePatchDrawable.html).

1. [Layer List](http://developer.android.com/guide/topics/resources/drawable-resource.html#LayerList)

A Drawable that manages an array of other Drawables. These are drawn in array order, so the element with the largest index is be drawn on top. Creates a[LayerDrawable](http://developer.android.com/reference/android/graphics/drawable/LayerDrawable.html).

1. [Transition Drawable](http://developer.android.com/guide/topics/resources/drawable-resource.html#Transition)

An XML file that defines a drawable that can cross-fade between two drawable resources. Creates a [TransitionDrawable](http://developer.android.com/reference/android/graphics/drawable/TransitionDrawable.html).

1. [Clip Drawable](http://developer.android.com/guide/topics/resources/drawable-resource.html#Clip)

An XML file that defines a drawable that clips another Drawable based on this Drawable's current level value. Creates a [ClipDrawable](http://developer.android.com/reference/android/graphics/drawable/ClipDrawable.html).

1. [Shape Drawable](http://developer.android.com/guide/topics/resources/drawable-resource.html#Shape)

An XML file that defines a geometric shape, including colors and gradients. Creates a[ShapeDrawable](http://developer.android.com/reference/android/graphics/drawable/ShapeDrawable.html).

Bitmap File

A bitmap file is a .png, .jpg, or .gif file. Android creates a [Drawable](http://developer.android.com/reference/android/graphics/drawable/Drawable.html) resource for any of these files when you save them in the res/drawable/ directory.

FILE LOCATION:

res/drawable/*filename*.png (.png, .jpg, or .gif)  
The filename is used as the resource ID.

RESOURCE REFERENCE:

In Java: R.drawable.*filename*  
In XML: @[*package*:]drawable/*filename*

EXAMPLE:

With an image saved at res/drawable/myimage.png, this layout XML applies the image to a View:

<ImageView  
    android:layout\_height="wrap\_content"  
    android:layout\_width="wrap\_content"  
    android:src="@drawable/myimage" />

The following application code retrieves the image as a [Drawable](http://developer.android.com/reference/android/graphics/drawable/Drawable.html):

Resources res = [getResources()](http://developer.android.com/reference/android/content/Context.html#getResources());  
Drawable drawable = res.[getDrawable](http://developer.android.com/reference/android/content/res/Resources.html#getDrawable(int))(R.drawable.myimage);

XML Bitmap

An XML bitmap is a resource defined in XML that points to a bitmap file. The effect is an alias for a raw bitmap file. The XML can specify additional properties for the bitmap such as dithering and tiling.

FILE LOCATION:

res/drawable/*filename*.xml  
The filename is used as the resource ID.

RESOURCE REFERENCE:

In Java: R.drawable.*filename*  
In XML: @[*package*:]drawable/*filename*

Example

<**bitmap xmlns:android="http://schemas.android.com/apk/res/android"  
 android:src="@drawable/goofy"  
 android:gravity="center"  
 android:tileMode="mirror"**>  
  
</**bitmap**>

**Nine-Patch**

A [NinePatch](http://developer.android.com/reference/android/graphics/NinePatch.html) is a PNG image in which you can define stretchable regions that Android scales when content within the View exceeds the normal image bounds.

An example use of a Nine-Patch image is the background used by Android's standard [Button](http://developer.android.com/reference/android/widget/Button.html) widget, which must stretch to accommodate the text (or image) inside the button.

Same as with a normal [bitmap](http://developer.android.com/guide/topics/resources/drawable-resource.html#Bitmap), you can reference a Nine-Patch file directly or from a resource defined by XML.

FILE LOCATION:

res/drawable/*filename*.9.png  
The filename is used as the resource ID.

RESOURCE REFERENCE:

In Java: R.drawable.*filename*  
In XML: @[*package*:]drawable/*filename*

**Layer List**

A [LayerDrawable](http://developer.android.com/reference/android/graphics/drawable/LayerDrawable.html) is a drawable object that manages an array of other drawables. Each drawable in the list is drawn in the order of the list—the last drawable in the list is drawn on top.

Each drawable is represented by an <item> element inside a single <layer-list>element.

FILE LOCATION:

res/drawable/*filename*.xml  
The filename is used as the resource ID.

RESOURCE REFERENCE:

In Java: R.drawable.*filename*  
In XML: @[*package*:]drawable/*filename*

EXAMPLE:

XML file saved at res/drawable/layers.xml:

<layer-list xmlns:android="http://schemas.android.com/apk/res/android">  
    <item>  
      <bitmap android:src="@drawable/android\_red" android:gravity="center" />  
    </item>  
    <item android:top="10dp" android:left="10dp">  
      <bitmap android:src="@drawable/android\_green"  
        android:gravity="center" />  
    </item>  
    <item android:top="20dp" android:left="20dp">  
      <bitmap android:src="@drawable/android\_blue"  
        android:gravity="center" />  
    </item>  
</layer-list>

**Transition Drawable**

A [TransitionDrawable](http://developer.android.com/reference/android/graphics/drawable/TransitionDrawable.html) is a drawable object that can cross-fade between the two drawable resources.

Each drawable is represented by an <item> element inside a single <transition>element.

No more than two items are supported.

To transition forward, call [startTransition()](http://developer.android.com/reference/android/graphics/drawable/TransitionDrawable.html#startTransition(int)).

To transition backward, call [reverseTransition()](http://developer.android.com/reference/android/graphics/drawable/TransitionDrawable.html#reverseTransition(int)).

**FILE LOCATION:**

res/drawable/*filename*.xml  
The filename is used as the resource ID.

**RESOURCE REFERENCE:**

In Java: R.drawable.*filename*  
In XML: @[*package*:]drawable/*filename*

**Example**

<**transition xmlns:android="http://schemas.android.com/apk/res/android"**>  
 <**item android:drawable="@drawable/maroon"**/>  
 <**item android:drawable="@drawable/blue"**/>  
</**transition**>

<**ImageButton  
 android:id="@+id/ib1"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:src="@drawable/tran\_images"  
 android:onClick="display"**/>

**public void display(View v)  
{  
 ImageButton ib = (ImageButton)findViewById(R.id.*ib1*);  
 TransitionDrawable td = (TransitionDrawable)ib.getDrawable();  
 td.startTransition(10000);  
}**

**Clip Drawable**

A drawable defined in XML that clips another drawable based on this Drawable's current level. You can control how much the child drawable gets clipped in width and height based on the level, as well as a gravity to control where it is placed in its overall container. Most often used to implement things like progress bars.

**FILE LOCATION:**

res/drawable/*filename*.xml  
The filename is used as the resource ID.

**RESOURCE REFERENCE:**

In Java: R.drawable.*filename*  
In XML: @[*package*:]drawable/*filename*

***ImageView iv = (ImageView)findViewById(R.id.iv1);***

***ClipDrawable cd = (ClipDrawable)iv.getDrawable();  
int level = cd.getLevel();  
cd.setLevel(level+1000);***

**State List**

A [StateListDrawable](https://developer.android.com/reference/android/graphics/drawable/StateListDrawable.html) is a drawable object defined in XML that uses a several different images to represent the same graphic, depending on the state of the object.

Each graphic is represented by an <item> element inside a single <selector>element.

**FILE LOCATION:**

res/drawable/*filename*.xml  
The filename is used as the resource ID.

**RESOURCE REFERENCE:**

In Java: R.drawable.*filename*  
In XML: @[*package*:]drawable/*filename*

**Shape Drawable**

This is a generic shape defined in XML.

**FILE LOCATION:**

res/drawable/*filename*.xml  
The filename is used as the resource ID.

**RESOURCE REFERENCE:**

In Java: R.drawable.*filename*  
In XML: @[*package*:]drawable/*filename*