

Native Mobile Applications

TalTech, Andres Käver, 2020-2021, Fall semester

Skype: akaver Email: akaver@itcollege.ee



Android - Resources

2

- ▶ All in some /res subdirectory
 - ▶ ./anim – Animations
 - ▶ ./color – Colors and Color State lists
 - ▶ ./drawable – Pictures (binary and xml)
 - ▶ ./layout – Layouts
 - ▶ ./menu – Menus
 - ▶ ./mipmap – launcher icons only
 - ▶ ./raw – Misc files (audio, video, etc)
 - ▶ ./values – texts, sizes, styles
 - ▶ ./xml – xml files
 - ▶ ...

Android – Resources - Animation

3

- ▶ Property animation
 - ▶ Modify objects property values over a time with an Animator
- ▶ View animation
 - ▶ Tween animation – series on transformations on single image with Animation
 - ▶ Frame animation – sequence of images with AnimationDrawable

Android – Resources - Color

4

- ▶ File location res/color/filename.xml
- ▶ Resource reference R.color.filename
- ▶ Android:color
 - ▶ 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

```
<?xml version="1.0" encoding="utf-8"?>
<selector xmlns:android="http://schemas.android.com/apk/res/android" >
  <item
    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>
```

Android – Resources - Color

5

res/color/button_text.xml

```
<?xml version="1.0" encoding="utf-8"?>
<selector xmlns:android="http://schemas.android.com/apk/res/android">
    <item android:state_pressed="true"
        android:color="#ffff0000"/> <!-- pressed -->
    <item android:state_focused="true"
        android:color="#ff0000ff"/> <!-- focused -->
    <item android:color="#ff000000"/> <!-- default -->
</selector>
```

<Button

```
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:text="@string/button_text"
    android:textColor="@color/button_text" />
```

Android – Resources - Drawable

6

- ▶ Graphics that can be drawn
 - ▶ Retrieve with APIs such as `getDrawable(...)`
 - ▶ Apply to other XML using `android:drawable` and `android:icon`
- ▶ `BitmapDrawable` – .png, .jpg, .gif
- ▶ `NinePatchDrawable` - .9.png – PNG with stretchable regions
- ▶ `Layer List` – array of other Drawables. Drawn in order, largest index on top
- ▶ `State List` – different bitmap graphics for different states (dif image when button is pressed)
- ▶ `Transition Drawable` – can crossfade between two drawables
- ▶ `Inset`, `Clip`, `Scale`, `Shape`....

Android – Resources – Drawable Bitmap

7

- ▶ .png preferred, .jpg acceptable, .gif discouraged
- ▶ File location
/res/drawable/filename.png
- ▶ XML bitmap –
/res/drawable/filename.xml
- ▶ Nine-patch
 - ▶ PNG image in which you can define stretchable regions
 - ▶ Usually background of a View that has at least one dimension set to "wrap_content"

```
<?xml version="1.0" encoding="utf-8"?>
<bitmap
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:src="@[package:]drawable/drawable_resource"
    android:antialias=["true" | "false"]
    android:dither=["true" | "false"]
    android:filter=["true" | "false"]
    android:gravity=["top" | "bottom" | "left" | "right" | "center_vertical" |
        "fill_vertical" | "center_horizontal" | "fill_horizontal" |
        "center" | "fill" | "clip_vertical" | "clip_horizontal"]
    android:mipMap=["true" | "false"]
    android:tileMode=["disabled" | "clamp" | "repeat" | "mirror"] />
```

```
<?xml version="1.0" encoding="utf-8"?>
<bitmap xmlns:android="http://schemas.android.com/apk/res/android"
    android:src="@drawable/icon"
    android:tileMode="repeat" />
```

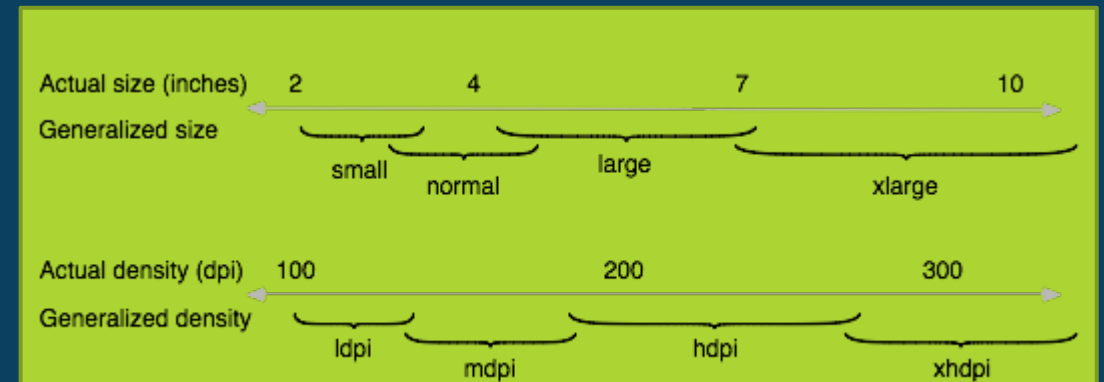
Android – Resources – configuration qualifiers

- ▶ To ensure your resources look their best, you should include alternative versions for different screen possibilities.
- ▶ ldpi (low)
- ▶ mdpi (medium)
- ▶ hdpi (high)
- ▶ xhdpi extra-high
- ▶ xxhdpi (extra-extra-high)
- ▶ xxxhdpi (extra-extra-extra-high)
- ▶ nodpi (no scaling)
- ▶ Folder name example:
 - ▶ res/layout-xlarge-land/my_layout.xml
- ▶ port (portrait)
- ▶ land (landscape)
- ▶ long (long aspect ratio)
- ▶ notlong (normal aspect ratio)
- ▶ small, normal, large, xlarge (screen size)

Android – Resources – screen sizes and densities

9

- ▶ Screen sizes and densities
- ▶ *xlarge* screens are at least 960dp x 720dp
- ▶ *large* screens are at least 640dp x 480dp
- ▶ *normal* screens are at least 470dp x 320dp
- ▶ *small* screens are at least 426dp x 320dp
- ▶ *ldpi* (low) ~120dpi
- ▶ *mdpi* (medium) ~160dpi
- ▶ *hdpi* (high) ~240dpi
- ▶ *xhdpi* (extra-high) ~320dpi
- ▶ *xxhdpi* (extra-extra-high) ~480dpi
- ▶ *xxxhdpi* (extra-extra-extra-high) ~640dpi



Android – Resources – Layer list

10

- ▶ Array of other drawables
- ▶ Last drawable in top
- ▶ /res/drawable/filename.xml



```
<?xml version="1.0" encoding="utf-8"?>
<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>
```

```
<ImageView
  android:layout_height="wrap_content"
  android:layout_width="wrap_content"
  android:src="@drawable/layers" />
```

Android – Resources – State list

11

- ▶ Different images to represent the same graphic, depending on the state of the object
- ▶ /res/drawable/filename.xml

```
<?xml version="1.0" encoding="utf-8"?>
<selector xmlns:android="http://schemas.android.com/apk/res/android">
    <item android:state_pressed="true"
        android:drawable="@drawable/button_pressed" /> <!-- pressed -->
    <item android:state_focused="true"
        android:drawable="@drawable/button_focused" /> <!-- focused -->
    <item android:state_hovered="true"
        android:drawable="@drawable/button_focused" /> <!-- hovered -->
    <item android:drawable="@drawable/button_normal" /> <!-- default -->
</selector>
```

```
<Button
    android:layout_height="wrap_content"
    android:layout_width="wrap_content"
    android:background="@drawable/button" />
```

```
<?xml version="1.0" encoding="utf-8"?>
<selector xmlns:android="http://schemas.android.com/apk/res/android"
    android:constantSize=["true" | "false"]
    android:dither=["true" | "false"]
    android:variablePadding=["true" | "false"] >
    <item
        android:drawable="@[package:]drawable/drawable_resource"
        android:state_pressed=["true" | "false"]
        android:state_focused=["true" | "false"]
        android:state_hovered=["true" | "false"]
        android:state_selected=["true" | "false"]
        android:state_checkable=["true" | "false"]
        android:state_checked=["true" | "false"]
        android:state_enabled=["true" | "false"]
        android:state_activated=["true" | "false"]
        android:state_window_focused=["true" | "false"] />
</selector>
```

Android – Resources – Transition

12

- ▶ Can cross-fade between the two drawable resources.

```
<?xml version="1.0" encoding="utf-8"?>
<transition xmlns:android="http://schemas.android.com/apk/res/android">
    <item android:drawable="@drawable/on" />
    <item android:drawable="@drawable/off" />
</transition>
```

```
<ImageButton
    android:id="@+id/button"
    android:layout_height="wrap_content"
    android:layout_width="wrap_content"
    android:src="@drawable/transition" />
```

```
ImageButton button = (ImageButton) findViewById(R.id.button);
TransitionDrawable drawable = (TransitionDrawable) button.getDrawable();
drawable.startTransition(500);
```

```
<?xml version="1.0" encoding="utf-8"?>
<transition
    xmlns:android="http://schemas.android.com/apk/res/android" >
    <item
        android:drawable="@[package:]drawable/drawable_resource"
        android:id="@+[package:]id/resource_name"
        android:top="dimension"
        android:right="dimension"
        android:bottom="dimension"
        android:left="dimension" />
</transition>
```

Android – Resources – Menu

13

- Options Menu, Context Menu, or submenu

```
<?xml version="1.0" encoding="utf-8"?>
<menu xmlns:android="http://schemas.android.com/apk/res/android">
  <item android:id="@+[package:]id/resource_name"
        android:title="string"
        android:titleCondensed="string"
        android:icon="@[package:]drawable/drawable_resource_name"
        android:onClick="method name"
        android:showAsAction=["ifRoom" | "never" | "withText" | "always" | "collapseActionView"]
        android:actionLayout="@[package:]layout/layout_resource_name"
        android:actionViewClass="class name"
        android:actionProviderClass="class name"
        android:alphabeticShortcut="string"
        android:numericShortcut="string"
        android:checkable=["true" | "false"]
        android:visible=["true" | "false"]
        android:enabled=["true" | "false"]
        android:menuCategory=["container" | "system" | "secondary" | "alternative"]
        android:orderInCategory="integer" />
  <group android:id="@+[package:]id/resource name"
        android:checkableBehavior=["none" | "all" | "single"]
        android:visible=["true" | "false"]
        android:enabled=["true" | "false"]
        android:menuCategory=["container" | "system" | "secondary" | "alternative"]
        android:orderInCategory="integer" >
    <item />
  </group>
  <item >
    <menu>
      <item />
    </menu>
  </item>
</menu>
```

Android – Resources – Menu

14

```
public boolean onCreateOptionsMenu(Menu menu) {
    MenuInflater inflater = getMenuInflater();
    inflater.inflate(R.menu.example_menu, menu);
    return true;
}

public void onGroupItemClick(MenuItem item) {
    // One of the group items (using the onClick attribute) was clicked
    // The item parameter passed here indicates which item it is
    // All other menu item clicks are handled by onOptionsItemSelected()
}
```

```
<menu xmlns:android="http://schemas.android.com/apk/res/android">
    <item android:id="@+id/item1"
        android:title="@string/item1"
        android:icon="@drawable/group_item1_icon"
        android:showAsAction="ifRoom|withText"/>
    <group android:id="@+id/group">
        <item android:id="@+id/group_item1"
            android:onClick="onGroupItemClick"
            android:title="@string/group_item1"
            android:icon="@drawable/group_item1_icon" />
        <item android:id="@+id/group_item2"
            android:onClick="onGroupItemClick"
            android:title="@string/group_item2"
            android:icon="@drawable/group_item2_icon" />
    </group>
    <item android:id="@+id/submenu"
        android:title="@string/submenu_title"
        android:showAsAction="ifRoom|withText" >
        <menu>
            <item android:id="@+id/submenu_item1"
                android:title="@string/submenu_item1" />
        </menu>
    </item>
</menu>
```

Android – Resources – Values

15

- ▶ Typed array
- ▶ res/values/filename.xml

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
  <array
    name="array_name">
    <item>resource</item>
  </array>
</resources>
```

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
  <array name="icons">
    <item>@drawable/home</item>
    <item>@drawable/settings</item>
    <item>@drawable/logout</item>
  </array>
</resources>
```

```
Resources res = getResources();
TypedArray icons = res.obtainTypedArray(R.array.icons);
Drawable drawable = icons.getDrawable(0);
```

Android – Resources – Values

16

- ▶ res/values/colors.xml
- ▶ R.color.color_name
- ▶ *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

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
  <color name="color_name">hex_color</color>
</resources>
```

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
  <color name="opaque_red">#f00</color>
  <color name="translucent_red">#80ff0000</color>
</resources>
```

```
Resources res = getResources();
int color = res.getColor(R.color.opaque_red);
```

```
<TextView
  android:layout_width="fill_parent"
  android:layout_height="wrap_content"
  android:textColor="@color/translucent_red"
  android:text="Hello"/>
```

Android – Resources – Values

17

- ▶ Dimensions
- ▶ `res/values/filename.xml`
- ▶ `R.dimen.dimension_name`
 - ▶ *dp* - Density-independent Pixels
 - ▶ *sp* - Scale-independent Pixels (scaled by the user's font size preference)
 - ▶ *pt* - Points - 1/72 of an inch based on the physical size of the screen.
 - ▶ *px* - Pixels - Corresponds to actual pixels on the screen.
 - ▶ *mm* - Millimeters - Based on the physical size of the screen
 - ▶ *in* - Inches - Based on the physical size of the screen.

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
  <dimen name="dimension_name">dimension</dimen>
</resources>
```

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
  <dimen name="textview_height">25dp</dimen>
  <dimen name="textview_width">150dp</dimen>
  <dimen name="ball_radius">30dp</dimen>
  <dimen name="font_size">16sp</dimen>
</resources>
```

```
Resources res = getResources();
float fontSize = res.getDimension(R.dimen.font_size);
```

```
<TextView
  android:layout_height="@dimen/textview_height"
  android:layout_width="@dimen/textview_width"
  android:textSize="@dimen/font_size"/>
```

Android – Resources – Values

18

- ▶ String
- ▶ res/values/filename.xml
- ▶ R.string.string_name

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
  <string name="string_name">text_string</string>
</resources>
```

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
  <string name="hello">Hello!</string>
</resources>
```

```
<TextView
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:text="@string/hello" />
```

```
String string = getString(R.string.hello);
```

Android – Resources – Values

19

- ▶ String array
- ▶ `res/values/filename.xml`
- ▶ `R.array.string_array_name`

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string-array name="string_array_name">
        <item>text_string</item>
    </string-array>
</resources>
```

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string-array name="planets_array">
        <item>Mercury</item>
        <item>Venus</item>
        <item>Earth</item>
        <item>Mars</item>
    </string-array>
</resources>
```

```
Resources res = getResources();
String[] planets = res.getStringArray(R.array.planets_array);
```

Android – Resources – Values

20

- ▶ Quantity Strings (Plurals)
- ▶ `res/values/filename.xml`
- ▶ `R.plurals.plural_name`

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <plurals name="numberOfSongsAvailable">
        <!--
            As a developer, you should always supply "one" and "other"
            strings. Your translators will know which strings are actually
            needed for their language. Always include %d in "one" because
            translators will need to use %d for languages where "one"
            doesn't mean 1 (as explained above).
        -->
        <item quantity="one">%d song found.</item>
        <item quantity="other">%d songs found.</item>
    </plurals>
</resources>
```

```
int count = getNumberOfSongsAvailable();
Resources res = getResources();
String songsFound = res.getString(R.plurals.numberOfSongsAvailable, count, count);
```

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <plurals
        name="plural_name">
        <item
            quantity=["zero" | "one" | "two" | "few" | "many" | "other"]
            >text_string</item>
        </plurals>
</resources>
```

Value	Description
zero	When the language requires special treatment of the number 0 (as in Arabic).
one	When the language requires special treatment of numbers like one (as with the number 1 in English and most other languages; in Russian, any number ending in 1 but not ending in 11 is in this class).
two	When the language requires special treatment of numbers like two (as with 2 in Welsh, or 102 in Slovenian).
few	When the language requires special treatment of "small" numbers (as with 2, 3, and 4 in Czech; or numbers ending 2, 3, or 4 but not 12, 13, or 14 in Polish).
many	When the language requires special treatment of "large" numbers (as with numbers ending 11-99 in Maltese).
other	When the language does not require special treatment of the given quantity (as with all numbers in Chinese, or 42 in English).

Android – Resources – Style

21

- ▶ Style resource defines the format and look for a UI.
- ▶ `res/values/filename.xml`

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <style name="CustomText" parent="@style/Text">
        <item name="android:textSize">20sp</item>
        <item name="android:textColor">#008</item>
    </style>
</resources>
```

```
<?xml version="1.0" encoding="utf-8"?>
<EditText
    style="@style/CustomText"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:text="Hello, World!" />
```

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <style
        name="style_name"
        parent="@[package:]style/style_to_inherit">
        <item
            name="[package:]style_property_name"
            >style_value</item>
        </style>
</resources>
```

Android

22

► The END!