Lecture 04 (ch 3 of Pro Android 4)

Android Resources

CMSC 4303 Mobile Apps Programming

Hong K. Sung, Ph.D.
Department of Computer Science
University of Central Oklahoma

1

understanding resources

- A resource in Android is a file or a value
  - file
    - a file for music, image, sound, etc
    - a file for window layout
  - value
    - title, name, etc
- These files and values are bound to the executable in such a way that you can change them without recompiling the application.

2

string resources: /res/values

Android allows you to define strings in one or more XML resource files. These XML files containing string-resource definitions reside in the `/res/values` subdirectory. The names of the XML files are arbitrary, although you commonly see the file name as `strings.xml`. Listing 3–1 shows an example of a string-resource file.

Listing 3–1. *Example `strings.xml`*

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

3

layout resources: /res/layout

Listing 3–5. *Example `main.xml` Layout File*

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    >
<TextView    android:id="@+id/text1"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:text="@string/hello"
    />
 <Button    android:id="@+id/b1"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:text="@string/hello"
   />
</LinearLayout>
```

Listing 3–4. *Using a Layout File*

```
public class HelloWorldActivity extends Activity
{
    @Override
    public void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        TextView tv = (TextView)this.findViewById(R.id.text1);
        tv.setText("Try this text instead");
    }
    ...
}
```

4

string arrays: /res/values

Listing 3–10. *Specifying String Arrays*

```
<resources ....>
......Other resources
<string-array name="test_array">
    <item>one</item>
    <item>two</item>
    <item>three</item>
</string-array>
......Other resources
</resources>
```

Listing 3–11. *Specifying String Arrays*

```
//Get access to Resources object from an Activity
Resources res = your-activity.getResources();
String strings[] = res.getStringArray(R.array.test_array);

//Print strings
for (String s: strings)
{
    Log.d("example", s);
}
```

5

plurals: /res/values

```
There is 1 egg.
There are 2 eggs.
There are 0 eggs.
There are 100 eggs.
```

Listing 3–12. *Specifying String Arrays*

```
<resources...>
<plurals name="eggs_in_a_nest_text">
    <item quantity="one">There is 1 egg</item>
    <item quantity="other">There are %d eggs</item>
</plurals>
</resources>
```

Listing 3–13. *Specifying String Arrays*

```
Resources res = your-activity.getResources();
String s1 = res.getQuantityString(R.plurals.eggs_in_a_nest_text, 0,0);
String s2 = res.getQuantityString(R.plurals.eggs_in_a_nest_text, 1,1);
String s3 = res.getQuantityString(R.plurals.eggs_in_a_nest_text, 2,2);
String s4 = res.getQuantityString(R.plurals.eggs_in_a_nest_text, 10,10);
```

6

color resources: /res/values

Listing 3–17. *XML Syntax for Defining Color Resources*

```
<resources>
    <color name="red">#f00</color>
    <color name="blue">#0000ff</color>
    <color name="green">#f0f0</color>
    <color name="main_back_ground_color">#ffffff00</color>
</resources>
```

Listing 3–18. *Color Resources in Java code*

```
int mainBackGroundColor
     =  activity.getResources.getColor(R.color.main_back_ground_color);
```

7

dimension: /res/values

Listing 3–20. *XML Syntax for Defining Dimension Resources*

```
<resources>
    <dimen name="mysize_in_pixels">1px</dimen>
    <dimen name="mysize_in_dp">5dp</dimen>
    <dimen name="medium_size">100sp</dimen>
</resources>
```

px: Pixels

in: Inches

mm: Millimeters

pt: Points

dp: Density-independent pixels based on a 160dpi (pixel density per inch) screen (dimensions adjust to screen density)

sp: Scale-independent pixels (dimensions that allow for user sizing; helpful for use in fonts)

Listing 3–21. *Using Dimension Resources in Java Code*

```
float dimen = activity.getResources().getDimension(R.dimen.mysize_in_pixels);
```

8

image: /res/drawable

Android generates resource IDs for image files placed in the `/res/drawable` subdirectory. The supported image types include `.gif`, `.jpg`, and `.png`. Each image file in this directory generates a unique ID from its base file name. If the image file name is `sample_image.jpg`, for example, then the resource ID generated is `R.drawable.sample_image`.

Listing 3–23. *Using Image Resources in XML*

```
<Button
      android:id="@+id/button1"
      android:layout_width="fill_parent"
      android:layout_height="wrap_content"
      android:text="Dial"
      android:background="@drawable/sample_image"
/>
```

Listing 3–24. *Using Image Resources in Java*

```
//Call getDrawable to get the image
BitmapDrawable d = activity.getResources().getDrawable(R.drawable.sample_image);

//You can use the drawable then to set the background
button.setBackgroundDrawable(d);
```

color-drawable: /res/values

Listing 3–25. *XML Syntax for Defining Color-Drawable Resources*

```
<resources>
    <drawable name="red_rectangle">#f00</drawable>
    <drawable name="blue_rectangle">#0000ff</drawable>
    <drawable name="green_rectangle">#f0f0</drawable>
</resources>
```

Listing 3–26. *Using Color-Drawable Resources in Java Code*

```
// Get a drawable
ColorDrawable redDrawable = (ColorDrawable)
     activity.getResources().getDrawable(R.drawable.red_rectangle);

//Set it as a background to a text view
textView.setBackgroundDrawable(redDrawable);
```

Listing 3–27. *Using Color-Drawable Resources in XML Code*

```
<TextView android:layout_width="fill_parent"
          android:layout_height="wrap_content"
          android:textAlign="center"
          android:background="@drawable/red_rectangle"/>
```

10

raw resources: /res/raw

Listing 3–33. *Reading a Raw Resource*

```
String getStringFromRawFile(Activity activity)
   throws IOException
   {
      Resources r = activity.getResources();
      InputStream is = r.openRawResource(R.raw.test);
      String myText = convertStreamToString(is);
      is.close();
      return myText;
   }

   String convertStreamToString(InputStream is)
   throws IOException
   {
      ByteArrayOutputStream baos = new ByteArrayOutputStream();
      int i = is.read();
      while (i != -1)
      {
         baos.write(i);
         i = is.read();
      }
      return baos.toString();
   }
```

suppose you have placed a text file at: **/res/raw/test.txt**

11

assets resources: /assets

- The files in **/assets** do not generate ID's in R. java. Thus, you must specify the file path to read them – a relative path starting at **/assets**

Listing 3–34. *Reading an Asset*

```
//Note: Exceptions are not shown in the code
String getStringFromAssetFile(Activity activity)
{

     AssetManager am = activity.getAssets();
     InputStream is = am.open("test.txt");
     String s = convertStreamToString(is);
     is.close();
     return s;
 }
```

12

resources directory structure

Listing 3–35. *Resource Directories*

```
/res/values/strings.xml
           /colors.xml
           /dimens.xml
           /attrs.xml
           /styles.xml
    /drawable/*.png
             /*.jpg
             /*.gif
             /*.9.png
    /anim/*.xml
    /layout/*.xml
    /raw/*.*
    /xml/*.xml
/assets/*.*/*.*
```

13

compiled/uncompiled resources

- Resource files are housed in subdirectories based on their type.
  - /**res/anim**: compiled animation files
  - /**res/drawable**: bitmaps
  - /**res/layout**: UI and view definitions
  - /**res/values**: arrays, colors, dimensions, string, styles, etc
  - /**res/xml**: compiled arbitrary XML files
  - /**res/raw**: non-compiled raw files
- All resources except /**res/raw** are compiled and placed into the final .**apk** file

14

types of resources: table 3-1

Resource Type	Location	Description
Colors	`/res/values/any-file`	Represents color identifiers pointing to color codes. These resource IDs are exposed in `R.java` as `R.color.*`. The XML node in the file is `/resources/color`.
Strings	`/res/values/any-file`	Represents string resources. String resources allow Java-formatted strings and raw HTML in addition to simple strings. These resource IDs are exposed in `R.java` as `R.string.*`. The XML node in the file is `/resources/string`.
String arrays	`/res/values/any-file`	Represents a resource that is an array of strings. These resource IDs are exposed in `R.java` as `R.array.*`. The XML node in the file is `/resources/string-array`.

15

Plurals	`/res/values/any-file`	Represents a suitable collection of strings based on the value of a quantity. The quantity is a number. In various languages, the way you write a sentence depends on whether you refer to no objects, one object, few objects, or many objects. The resource IDs are exposed in `R.java` as `R.plural.*`. The XML node in the value file is `/resources/plurals`.
Dimensions	`/res/values/any-file`	Represents dimensions or sizes of various elements or views in Android. Supports pixels, inches, millimeters, density independent pixels, and scale independent pixels. These resource IDs are exposed in `R.java` as `R.dimen.*` . The XML node in the file is `/resources/dimen`.
Images	`/res/drawable/multiple-files`	Represents image resources. Supported images include `.jpg`, `.gif`, `.png`, and so on. Each image is in a separate file and gets its own ID based on the file name. These resource ids are exposed in `R.java` as `R.drawable.*`. The image support also includes an image type called a *stretchable image* that allows portions of an image to stretch while other portions of that image stay static. The stretchable image is also known as a *9-patch file* (`.9.png`).

16

Color drawables	`/res/values/any-file`
	also
	`/res/drawable/multiple-files`
		Represents rectangles of colors to be used as view backgrounds or general drawables like bitmaps. This can be used in lieu of specifying a single-colored bitmap as a background. In Java, this is equivalent to creating a colored rectangle and setting it as a background for a view.
		The `<drawable>` value tag in the values subdirectory supports this. These resource IDs are exposed in `R.java` as `R.drawable.*`. The XML node in the file is `/resources/drawable`.
		Android also supports rounded rectangles and gradient rectangles through XML files placed in `/res/drawable` with the root XML tag of `<shape>`. These resource IDs are also exposed in `R.java` as `R.drawable.*`. Each file name in this case translates to a unique drawable ID.

17

Arbitrary XML files	`/res/xml/*.xml`	Android allows arbitrary XML files as resources. These files are compiled by the AAPT compiler. These resource IDs are exposed in `R.java` as `R.xml.*`.
Arbitrary raw resources	`/res/raw/*.*`	Android allows arbitrary *noncompiled* binary or text files under this directory. Each file gets a unique resource ID. These resource IDs are exposed in `R.java` as `R.raw.*` .
Arbitrary raw assets	`/assets/*.*/*.*`	Android allows arbitrary files in arbitrary subdirectories starting at the `/assets` subdirectory. These are not really resources, just raw files. This directory, unlike the `/res` resources subdirectory, allows an arbitrary depth of subdirectories. These files do not generate any resource IDs. You have to use a relative pathname starting at and excluding `/assets`.

18