Resources In Android

Mobile Computing - Android

An Android Project Simplified

```
MyProject/
    java/
        MainActivity.java
    res/
        drawable/
            anImage.jpg
        layout/
            activity_main.xml
        values/
            colors.xml
            strings.xml
```

A Simple Equation

Android App = Code + Resources

Code

- Subclasses of Activity, Service, ContentProvider, BroadcastReceiver + your own specialized subclasses

Resource Directories

Directory	Resource Type
animator	Property Animations
anim	Tween Animations
color	XML files that specify color lists
drawable	Bitmap files (.png, .jpg, .gif, .9.png)
layout	XML files that define user interfaces
menu	application menus (options menus, context menus, etc.)
raw	arbitrary files that you can read in with an InputStream
values	simple values: strings, integers, single colors

Resources

- All the non-code elements of an app: **images**, **strings**, **layout files**, **menus**, etc.
- Grouped in subdirectories of /res, to make maintenance easier -- never have to recompile code
- Each subdirectory stores a specific type of resource (**/res/layout** stores layout files; **/res/drawable** stores images, etc).
- Can specify default and multiple alternative resources (used for specific devices/circumstances)
- Alternative resources are stored in directories with qualifiers to identify the device/circumstance in question
- When you ask for a resource (say, **sun.png**), Android OS searches the appropriate directory depending on the device (if it's hdpi or not, in this project)

```
res/
    drawable/
        sun.png
        background.png
    drawable-hdpi/
        sun.png
        background.png
```

Resource IDs

- Each resource has a resource ID, a unique integer value
- Resource IDs are generated automatically when you add a resource to the project and compile it
- *Many* methods accept resource IDs to gain access to the resources

Access in Java (R class version)

- Up until Lollipop, image resources where held in a class with the name R.
- Each resource type is held in a static inner class.
- Each resource name is a static variable.
- Resource names have global visibility.

```
res/
    drawable/
        icon.png
        background.png
    layout/
        main.xml
        info.xml
    values/
        strings.xml
```

```
public final class R {
        public final static class drawable {
                public static final int icon = 123;
                public static final int background = 124;
        }
        public final static class layout {
                public static final int main = 300;
                public static final int info = 301;
        }
}
```

Access in XML and Java

Example Resource name in XML

@drawable/icon

Example Resource name in Java

R.drawable.icon

```
res/
    drawable/
        icon.png
        background.png
    layout/
        main.xml
        info.xml
```

```
public final class R {
        public final static class drawable {
                public static final int icon = 123;
                public static final int background = 124;
        }
        public final static class layout {
                public static final int main = 300;
                public static final int info = 301;
        }
}
```

Java examples

- The naming usage is the same in the old implementation.
- The current implementation just keeps the resource ids in a flat file. A preprocessor replaces the names with the appropriate integer value.

```
EditText nameET = findViewById(R.id.nameET);
```

Id is the category for all of the views and components in a layout.

```
setContentView(R.layout.main_screen);
```

Set the view for an activity explicitly.

```
ImageView frameIV = findViewById(R.id.frameIV)
frameIV.setImageResource(R.drawable.sun);
```

Use a drawable as the view for the frameIV widget.

Modern Implementation

- The naming usage is the same in the old implementation.
- The current implementation just keeps the resource ids in a flat file. A preprocessor replaces the names with the appropriate integer value.

```
EditText nameET = (EditText)findViewById(R.id.nameET);
```

Old style requires type cast from return type of View.

```
EditText nameET = findViewById(R.id.nameET);
```

New style will generate an error if the assignment does not match the component type from the layout.

XML examples

- A resource may be the entire file (ic_launcher.png)
- Files in /res/values contain multiple resources (indicated by the <resources> tag)
- These too may be accessed in code
  - getText(R.string.greeting)

Name in XML is @string/greeting

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string name="greeting">Hello</string>
    <string name="greeting_fr">Bonjour</string>
    <string name="fortune">Fortune</string>
    <string name="tell_me_my_fortune">Tell Me My Fortune</string>
</resources>
```

Four resources in strings.xml

XML example

- Fragment of a layout with three widgets.

```
<TextView
        android:text="@string/fortune"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/fortuneTV"/>
<EditText
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/nameET"/>
<Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/tell_me_my_fortune"
        android:id="@+id/fetchBTN"
```

@string/fortune is in strings.xml

@+id/fortuneTV creates the resource id if not already created.

Widget Naming advice

- Most layout components should have an ID.
- Add in a short code for widget type at the end of the name. Here are a few:
  - TV for TextView
  - IV for ImageView
  - ET for EditText
  - BTN for Button
  - CL for ConstraintLayout
  - RV for RecyclerView
- While not required, making the Java reference have the same name as the id of your widget can help. (See code on the next slide.)

Widget Properties

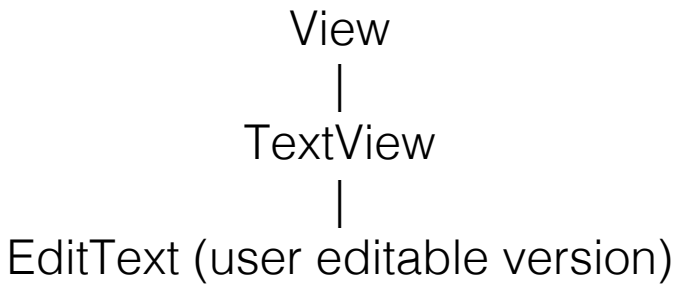
- Widgets have a lot of possible attributes that can be set in XML via the inspector.
- Many of these properties start with default values.
- Many of these properties have get/set methods.
- Examples: getText/setText.

```
EditText ageET = findViewById(R.id.ageET);
String contents = ageET.getText().toString();

int age = Integer.parseInt(contents)

TextView resultTV = findViewById(R.id.resultTV);
resultTV.setText("Age in dog years is " + age*7);
```

getText returns a character sequence which we convert to a String

setText wants a character sequence, but a String will works here.

Relation of View/TextView/EditText



Note: In the Android Studio palette you will not find an EditText widget. A number of the widgets under Text, however, will create an EditText component. For example, choosing Simple Text results in an EditText widget with type "textPersonName"

Button OnClick Property

- We can specify what a button does by setting its onClick property from the inspector. We give it the method that will respond. The method must have the right signature.

```
public void buttonAction(View v){
   TextView displayTV = findViewById(R.id.displayTV);
   displayTV.setText("Replace it");

}
```

Safer to get a reference to the components as you need them since the location may change through the Apps execution.

Button OnClick (Set Method)

- We can also create an anonymous function that we use as a button hander method.
- This would be inside the onCreate() method.

```
Button btn = findViewById(R.id.clickMeBTN);
btn.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        Toast.makeText(getApplicationContext(), "hello world!",
            Toast.LENGTH_LONG).show();
        EditText nameET = findViewById(R.id.nameET);
        nameET.setText("hey, this works!!");
    }
});
```

Android Manifest

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
   xmlns:tools="http://schemas.android.com/tools">
   <application
      android:allowBackup="true"
      android:icon="@mipmap/ic_launcher"
      android:label="@string/app_name"
      android:roundIcon="@mipmap/ic_launcher_round"
      android:supportsRtl="true"
      android:theme="@style/Theme.Demo"
      tools:targetApi="31">
      <activity
         android:name=".MainActivity"
         android:exported="true">
         <intent-filter>
            <action android:name="android.intent.action.MAIN" />
            <category android:name="android.intent.category.LAUNCHER" />
         </intent-filter>
         <meta-data
            android:name="android.app.lib_name"
            android:value="" />
      </activity>
   </application>
</manifest>
```

- Each app must include a manifest
- Specifies all the components
- Intent filter allows other apps to make queries.
- With structure, XML is fairly readable.

References

- Oracle Java tutorial on nested classes
- Android Developer Resources main page
- Android Developer Resources Basic Guide

Questions

1. In what /res folder would you find these items:
    1. xml files specifying a UI
    2. images
    3. .java files
2. What is the purpose of R?
3. An image, my_face.jpg, is stored in drawable. How would you reference it in code? In xml?
4. A layout, my_layout.xml, is stored in layout. How would you reference it in code? In xml?
5. Write the signature for onCreate(). When is it called? What is its purpose?
6. Write an onCreate() method so that it will cause the layout, my_layout.xml, to be used as the activity's View
7. What is a manifest file? What is its full name, and where is it stored in an Android project?
8. What is the difference between EditText and TextView
9. Give the signature of a method that could be invoked when a button is clicked.
10. Write the code necessary, in an onClick listener, to retrieve the contents of an EditText and, assuming that the contents represent a number, replace the contents by that number squared. You may assume that the EditText has an id of numET.
11. Name the android:attributes that are used to
    1. set the id to a particular value
    2. set the onClick listener to a method called fireAction