Mobile Application Programming: Android

Android Resources
Activities

- Apps are composed of activities
- Activities are self-contained tasks made up of one screen-full of information
- Activities start one another and are destroyed commonly
- Apps can use activities belonging to another app
Resources

- Files in the res folder
- Accessed from code using integer identifiers maintained in the class R
- Multiple versions of a resource can be automatically chosen by configuration qualifiers

```
MyProject/
  src/
  MyActivity.java
  res/
  drawable/
  icon.png
  layout/
  main.xml
  info.xml
  values/
  strings.xml
```
## Resources

<table>
<thead>
<tr>
<th>Directory</th>
<th>Resource Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>animator/</td>
<td>XML files that define property animations.</td>
</tr>
<tr>
<td>anim/</td>
<td>XML files that define tween animations. (Property animations can also be saved in this directory, but the animator directory is preferred for property animations to distinguish between the two types.)</td>
</tr>
<tr>
<td>color/</td>
<td>XML files that define a state list of colors. See Color State List Resource</td>
</tr>
<tr>
<td>drawable/</td>
<td>Bitmap files (.png, .9.png, .jpg, .gif) or XML files that are compiled into the following drawable resource subtypes:</td>
</tr>
<tr>
<td></td>
<td>- Bitmap files</td>
</tr>
<tr>
<td></td>
<td>- Nine-Patches (re-sizeable bitmaps)</td>
</tr>
<tr>
<td></td>
<td>- State lists</td>
</tr>
<tr>
<td></td>
<td>- Shapes</td>
</tr>
<tr>
<td></td>
<td>- Animation drawables</td>
</tr>
<tr>
<td></td>
<td>- Other drawables</td>
</tr>
<tr>
<td></td>
<td>See Drawable Resources.</td>
</tr>
<tr>
<td>layout/</td>
<td>XML files that define a user interface layout. See Layout Resource.</td>
</tr>
<tr>
<td>menu/</td>
<td>XML files that define application menus, such as an Options Menu, Context Menu, or Sub Menu. See Menu Resource.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Directory</th>
<th>Resource Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>raw/</td>
<td>Arbitrary files to save in their raw form. To open these resources with a raw InputStream, call Resources.openRawResource() with the resource ID, which is R.raw.filename.</td>
</tr>
<tr>
<td></td>
<td>However, if you need access to original file names and file hierarchy, you might consider saving some resources in the assets directory (instead of res/raw/). Files in assets/ are not given a resource ID, so you can read them only using AssetManager.</td>
</tr>
<tr>
<td>values/</td>
<td>XML files that contain simple values, such as strings, integers, and colors.</td>
</tr>
<tr>
<td></td>
<td>Whereas XML resource files in other res/ subdirectories define a single resource based on the XML filename, files in the values directory describe multiple resources. For a file in this directory, each child of the &lt;resources&gt; element defines a single resource. For example, a &lt;string&gt; element creates an R.string resource and a &lt;color&gt; element creates an R.color resource.</td>
</tr>
<tr>
<td></td>
<td>Because each resource is defined with its own XML element, you can name the file whatever you want and place different resource types in one file. However, for clarity, you might want to place unique resource types in different files. For example, here are some filename conventions for resources you can create in this directory:</td>
</tr>
<tr>
<td></td>
<td>- arrays.xml for resource arrays (typed arrays).</td>
</tr>
<tr>
<td></td>
<td>- colors.xml for color values</td>
</tr>
<tr>
<td></td>
<td>- dimens.xml for dimension values.</td>
</tr>
<tr>
<td></td>
<td>- strings.xml for string values.</td>
</tr>
<tr>
<td></td>
<td>- styles.xml for styles.</td>
</tr>
<tr>
<td>xml/</td>
<td>Arbitrary XML files that can be read at runtime by calling Resources.getXML(). Various XML configuration files must be saved here, such as a searchable configuration.</td>
</tr>
</tbody>
</table>
Resources

Directory name: values-en
Resource type: values

Available qualifiers:
- Country Code
- Network Code
- Region
  - Smallest Screen Width
  - Screen Width
  - Screen Height
  - Size
- Ratio
- Orientation
- UI Mode
- Night Mode
- Density
- Touch Screen
- Keyboard
- Text Input
- Navigation State
- Navigation Method
- Dimension
- Android Version

Chosen qualifiers:
- en: English

Language:
- de: German
- dv: Divehi
- dz: Dzongkha
- ee: Ewe
- el: Greek
- eo: Esperanto
- es: Spanish
- et: Estonian
- eu: Basque
- fa: Persian
- ff: Fulah
- fi: Finnish
- fj: Fijian
- fo: Faroese
- fr: French
- fy: Western Frisian
- ga: Irish
- gd: Gaelic

Tip: Type in list to filter

Cancel  OK
Resources

```xml
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string name="app_name">Sprites</string>
    <string name="hello_world">Hello world!</string>
    <string name="action_settings">Settings</string>
</resources>

<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string name="app_name">Sprites</string>
    <string name="hello_world">Olá Mundo!</string>
    <string name="action_settings">Ajustes</string>
</resources>

<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string name="app_name">スプライト</string>
    <string name="hello_world">こんにちは世界</string>
    <string name="action_settings">調整</string>
</resources>

String helloWorldLabelTitle =
    getString(R.string.hello_world);
```
Resources

```
Bitmap icon =
    BitmapFactory.decodeResource(getResources(), R.drawable.ic_launcher);
```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/container"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical">
    <TextView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:gravity="center"
        android:text="@string/title_label"/>
    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="center"
        android:text="@string/add_button_title"/>
</LinearLayout>
XML Layouts (Custom)

```xml
    <TextView android:layout_width="match_parent" android:layout_height="wrap_content" android:gravity="center" android:text="@string/title_label"/>
    <Button android:layout_width="wrap_content" android:layout_height="wrap_content" android:layout_gravity="center" android:text="@string/add_button_title"/>
    <edu.utah.cs4962.shapes.ShapeView
        android:layout_width="match_parent" android:layout_height="0dp" android:layout_weight="1"
        custom:sideCount="5" custom:shapeColor="#80FF80"/>
</LinearLayout>
```

```xml
<resources>
    <declare-styleable name="ShapeView">
        <attr name="sideCount" format="integer"/>
        <attr name="shapeColor" format="color"/>
    </declare-styleable>
</resources>
```
XML Layouts (Custom)

```java
public ShapeView(Context context, AttributeSet attributeSet)
{
    super(context, attributeSet);

    TypedArray attributes = context.obtainStyledAttributes(attributeSet, R.styleable.ShapeView);
    int sideCount = attributes.getInt(R.styleable.ShapeView_sideCount, SIDE_COUNT_MIN);
    int shapeColor = attributes.getColor(R.styleable.ShapeView_shapeColor, SHAPE_COLOR_DEFAULT);
    attributes.recycle();

    setSideCount(sideCount);
    setShapeColor(shapeColor);
}
```