Android Embedded Resources

Victor Matos

Cleveland State University

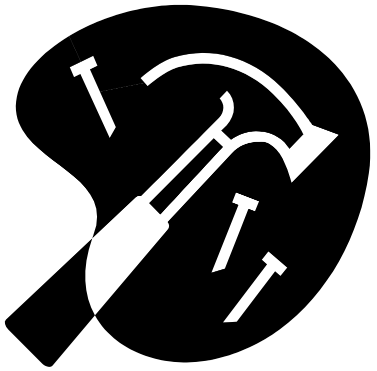
Notes are based on:

Android Developers
http://developer.android.com/index.html

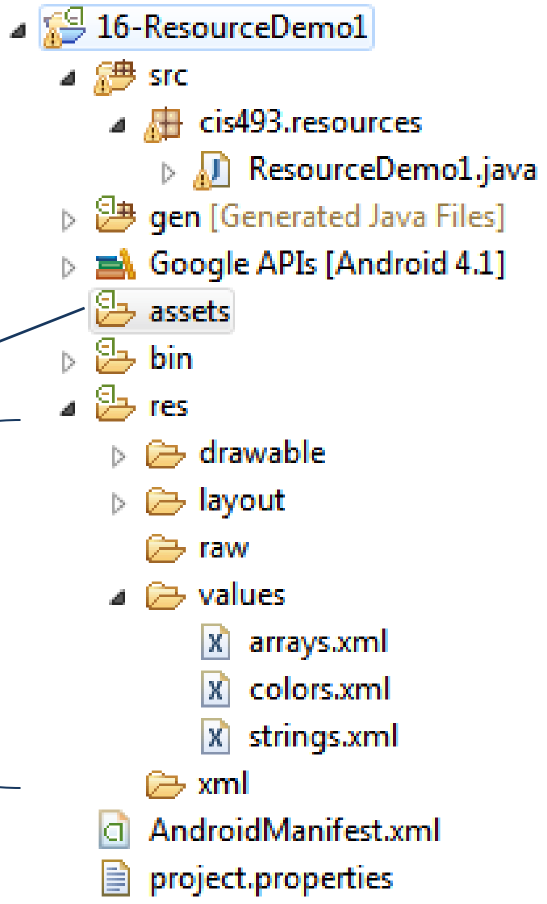
Android Resources

Resources

Resources are external files (images, strings, files, audio, animations, layouts, styles, xml …) which are integrated into the application at compile time for their later use at execution time.



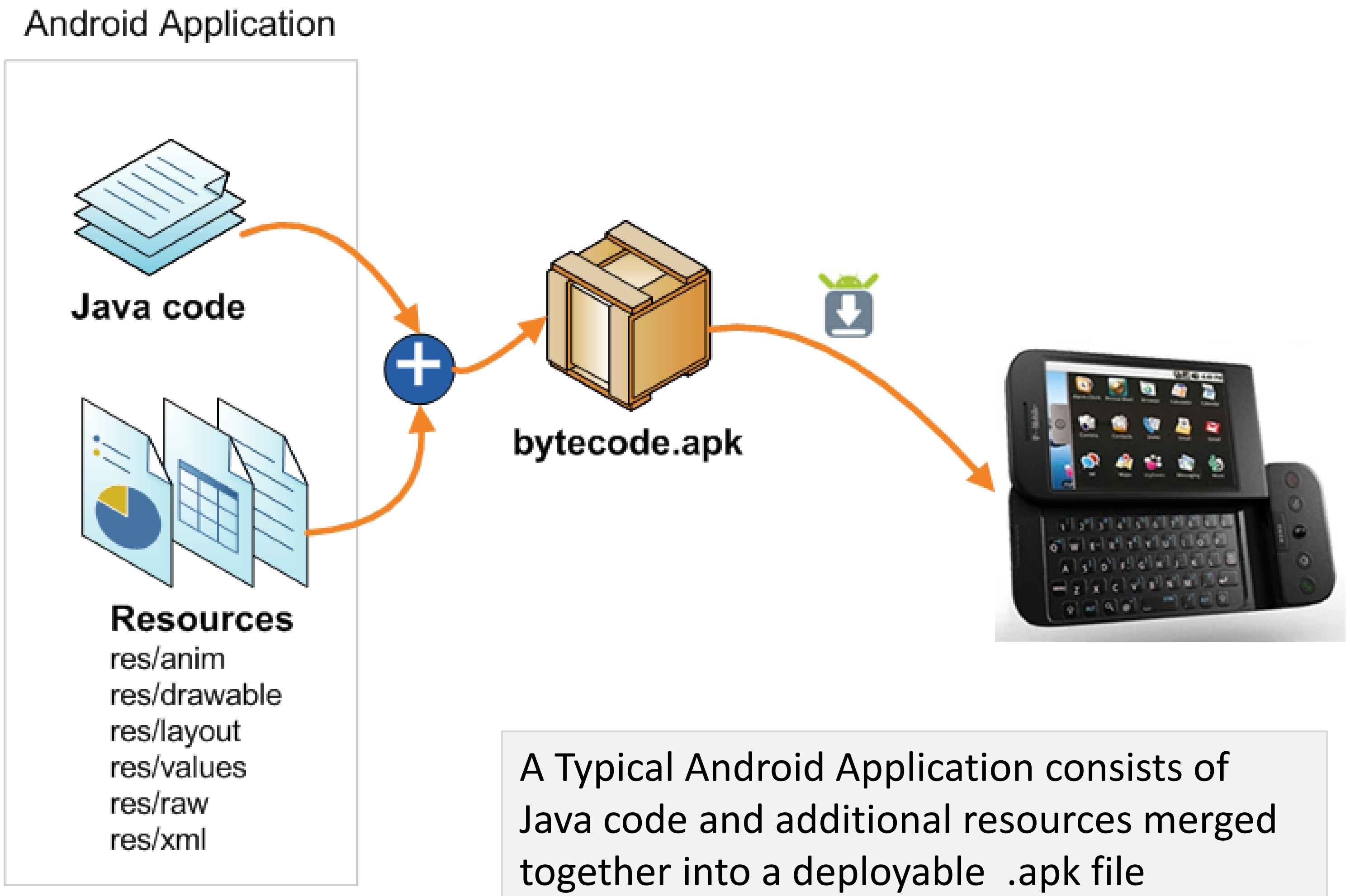
Raw resources are placed here



```
16-ResourceDemo1
  src
    cis493.resources
      ResourceDemo1.java
  gen [Generated Java Files]
  Google APIs [Android 4.1]
  assets
  bin
  res
    drawable
    layout
    raw
    values
      arrays.xml
      colors.xml
      strings.xml
    xml
  AndroidManifest.xml
  project.properties
```

http://developer.android.com/guide/topics/resources/providing-resources.html

Android Resources



A Typical Android Application consists of Java code and additional resources merged together into a deployable .apk file

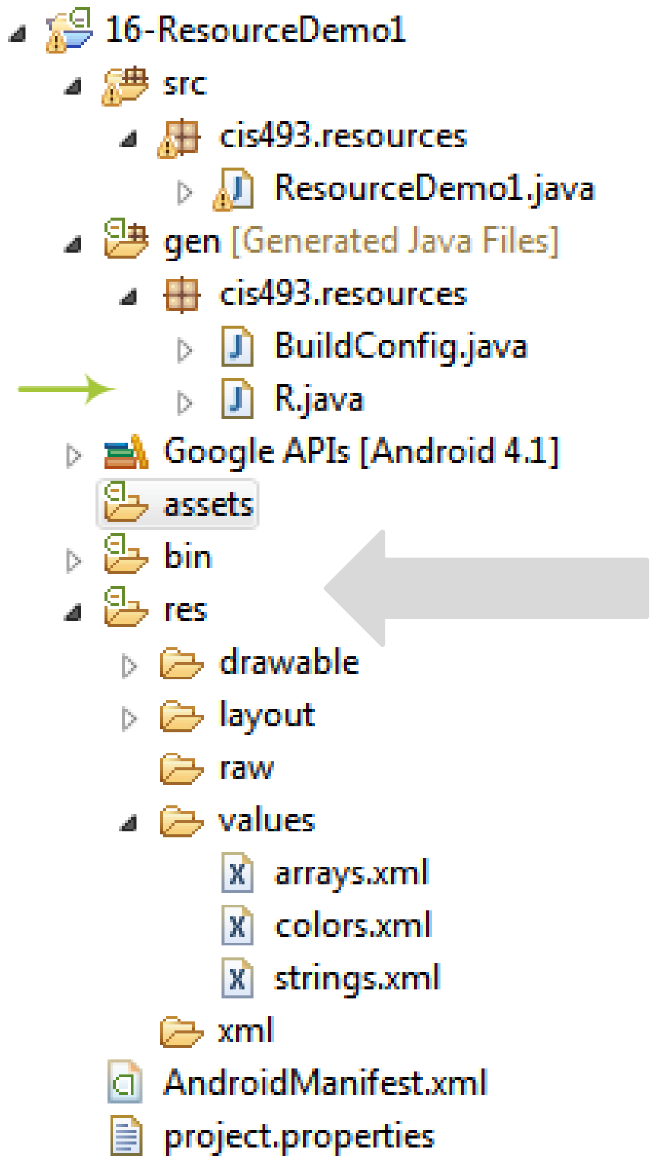
Android Resources

Copy/Paste Resources

You place your resources under the appropriate **/res** or **/assets** **s**ubdirectory in your project's workspace.

Resources are compiled into the final APK file.

Android creates a wrapper class, called **R**, that you can use to refer to these resources in your code.



```
16-ResourceDemo1
  src
    cis493.resources
      ResourceDemo1.java
  gen [Generated Java Files]
    cis493.resources
      BuildConfig.java
      R.java
  Google APIs [Android 4.1]
  assets
  bin
  res
    drawable
    layout
    raw
    values
      arrays.xml
      colors.xml
      strings.xml
    xml
  AndroidManifest.xml
  project.properties
```

Android Resources

Directory	Resource Type
animator/	XML files that define property animations.
anim/	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.)
color/	XML files that define a state list of colors.
drawable/	Bitmap files (.png, .9.png, .jpg, .gif) or XML (shapes, animations) files that are compiled into drawable resources.
layout/	XML files that define a user interface layout.
menu/	XML files that define application menus, such as an Options Menu, Context Menu, or Sub Menu.
raw/	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. 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.
Values/	XML files that contain simple values, such as strings, integers, and colors. For example, a <string> element creates an R.string resource and a <color> element creates an R.color resource. You can name the file whatever you want and place different resource types in one file. For clarity use: arrays.xml, colors.xml, dimens.xml, strings.xml, styles.xml.
xml/	Arbitrary XML files that can be read at runtime by calling Resources.getXML().

http://developer.android.com/guide/topics/resources/providing-resources.html

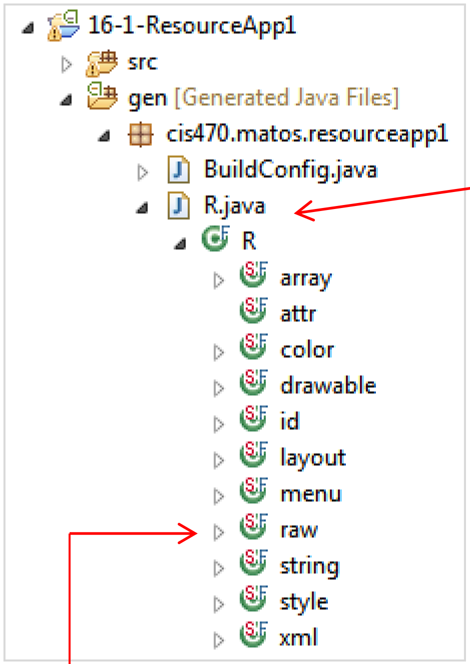
Difference between folders: /res/raw and /assets

Both folders are very similar (they can store arbitrary data items) .

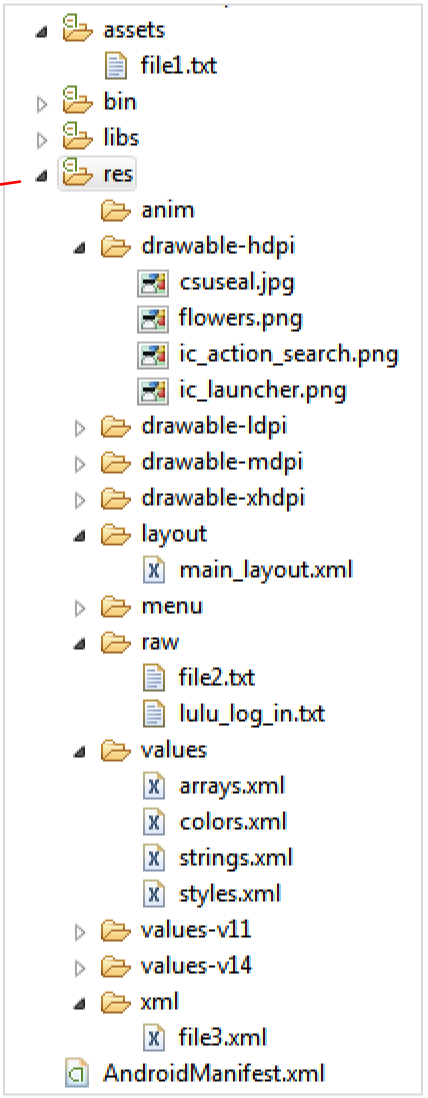
All items saved in the **/res/** folder are indexed by ID. Those entries are stored in the **R.java** file .

Indexed items can be easily retrieve using notation:

R.array . [resource_name]
R.attr. [resource_name]
…
R.id . [resource_name]
…
R.xml . [resource_name]



```
16-1-ResourceApp1
  src
  gen [Generated Java Files]
    cis470.matos.resourceapp1
      BuildConfig.java
      R.java
        R
          array
          attr
          color
          drawable
          id
          layout
          menu
          raw
          string
          style
          xml
```



```
assets
  file1.txt
bin
libs
res
  anim
  drawable-hdpi
    csuseal.jpg
    flowers.png
    ic_action_search.png
    ic_launcher.png
  drawable-ldpi
  drawable-mdpi
  drawable-xhdpi
  layout
    main_layout.xml
  menu
  raw
    file2.txt
    lulu_log_in.txt
  values
    arrays.xml
    colors.xml
    strings.xml
    styles.xml
  values-v11
  values-v14
  xml
    file3.xml
AndroidManifest.xml
```

Fragment taken from R.java

```
public static final class raw {
  public static final int file2=0x7f050000;
  public static final int lulu_log_in=0x7f050001;
}
```

Note: /res/raw/file2 and /res/xml/file3 appear in R.java however file1 does not.

Android Resources

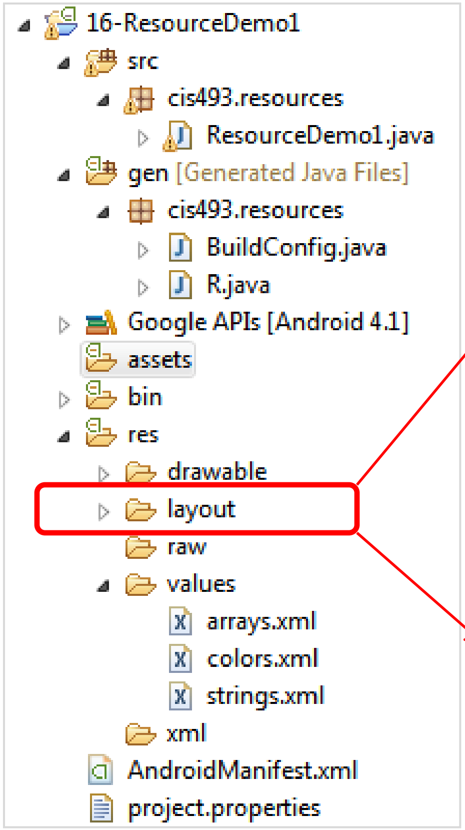
Retrieving Resources witch code

```
getResources().getAnimation(id);

getResources().getAssets().open(fileName)
getResources().getAssets().openXmlResourceParser(fileName)

getResources().getColor(id)
getResources().getDrawable(id)

getResources().getIntArray(id)
getResources().getStringArray(id)
getResources().getTextArray(id)

getResources().getLayout(id)
getResources().getMovie(id)
getResources().getString(id)
getResources().getXml(id)
```

Android Resources

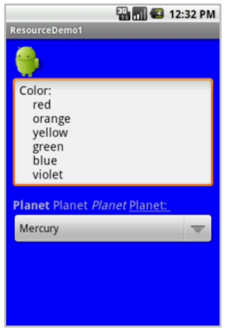
Java Statements for Using Resources

Displaying a **screen layout**:

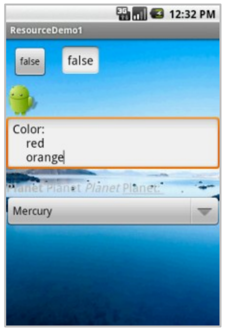


```
16-ResourceDemo1
  src
    cis493.resources
      ResourceDemo1.java
  gen [Generated Java Files]
    cis493.resources
      BuildConfig.java
      R.java
  Google APIs [Android 4.1]
  assets
  bin
  res
    drawable
    layout
    raw
    values
      arrays.xml
      colors.xml
      strings.xml
    xml
  AndroidManifest.xml
  project.properties
```

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



```
setContentView(R.layout.screen2);
```

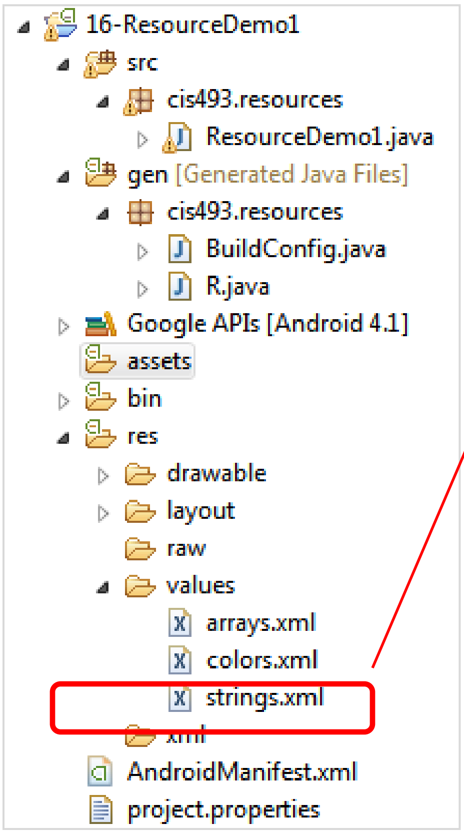


Android Resources

Java Statements for Using Resources

Retrieving **String** Resources from: **res/values/strings.xml**

Convenient strategy for internalization/redeployment.



```xml
<?xml version="1.0" encoding="utf-8"?>
<resources>
  <string name="hello">Hola Mundo!, ResourceDemo1!</string>
  <string name="app_name">ResourceDemo1</string>
  <string name="good_bye">Hasta luego</string>
  <string name="color_caption">Color:</string>
  <string name="color_prompt">Seleccione un Color</string>
  <string name="planet_caption">
              <b>Planeta</b>  Planeta  <i>Planeta </i>
              <u>Planeta</u>  </string>
  <string name="planet_prompt">Seleccione un Planeta</string>
</resources>
```

```java
String msg =
 this.getString(R.string.color_prompt);
```

Android Resources

Java Statements for Using Resources

Enhancing externalized **String** resources from: **res/values/strings.xml**

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string name="hello">Hola Mundo!, ResourceDemo1!</string>
    <string name="app_name">ResourceDemo1</string>
    <string name="good_bye">Hasta luego</string>
    <string name="color_caption">Color:</string>
    <string name="color_prompt">Seleccione un Color</string>
    <string name="planet_caption">
                 <b>Planeta</b>  Planeta  <i>Planeta</i>
                 <u>Planeta</u>  </string>
    <string name="planet_prompt">Seleccione un Planeta</string>
</resources>
```

As in HTML a string using **<b>, <i>, <u>** modifiers will be rendered in: bold, italics, and, underlined modes. In our example:

```
String myColors[] = this.getResources().getStringArray(R.array.colors);
```

Planeta *Planeta* Planeta <u>Planeta</u>

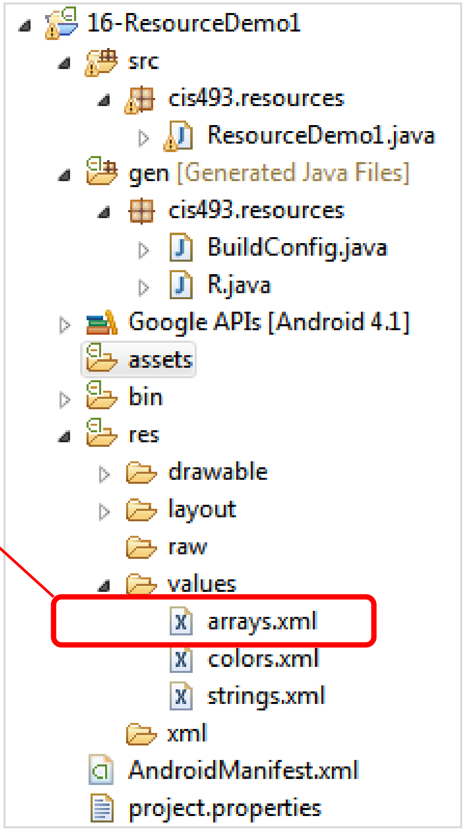
Android Resources

Java Statements for Using Resources

Retrieving **Array** Resources from: **res/values/arrays.xml**

```
String myColors[] = this.getResources().getStringArray(R.array.colors);
```

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string-array name="colors">
        <item>red</item>
        <item>orange</item>
        <item>yellow</item>
        <item>green</item>
        <item>blue</item>
        <item>violet</item>
    </string-array>

    <string-array name="planets">
        <item>Mercury</item>
        <item>Venus</item>
        <item>Earth</item>
        <item>Mars</item>
        <item>Jupiter</item>
        <item>Saturn</item>
        <item>Uranus</item>
        <item>Neptune</item>
        <item>Pluto</item>
    </string-array>
  </resources>
```

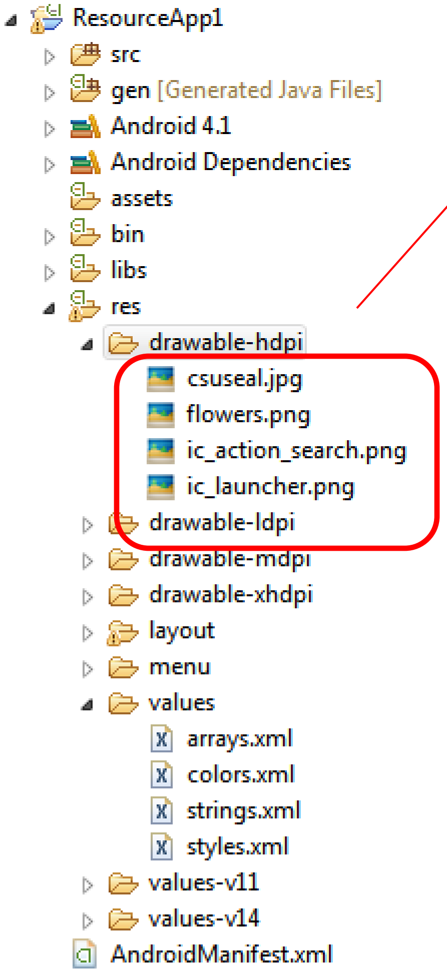


```
16-ResourceDemo1
  src
    cis493.resources
      ResourceDemo1.java
  gen [Generated Java Files]
    cis493.resources
      BuildConfig.java
      R.java
  Google APIs [Android 4.1]
  assets
  bin
  res
    drawable
    layout
    raw
    values
      arrays.xml
      colors.xml
      strings.xml
    xml
  AndroidManifest.xml
  project.properties
```

Android Resources

Java Statements for Using Resources

Retrieving a **drawable image** from: **res/drawable/**



```
ResourceApp1
    src
    gen [Generated Java Files]
    Android 4.1
    Android Dependencies
    assets
    bin
    libs
    res
        drawable-hdpi
            csuseal.jpg
            flowers.png
            ic_action_search.png
            ic_launcher.png
        drawable-ldpi
        drawable-mdpi
        drawable-xhdpi
        layout
        menu
        values
            arrays.xml
            colors.xml
            strings.xml
            styles.xml
        values-v11
        values-v14
    AndroidManifest.xml
```

```
// same as xml layout attribute
// android:src="@drawable/android_green_3d"

imageView1.setImageResource(
                        R.drawable.flowers);
```

Android Resources

Example1. Using Embedded Resources (drawable, string, array).

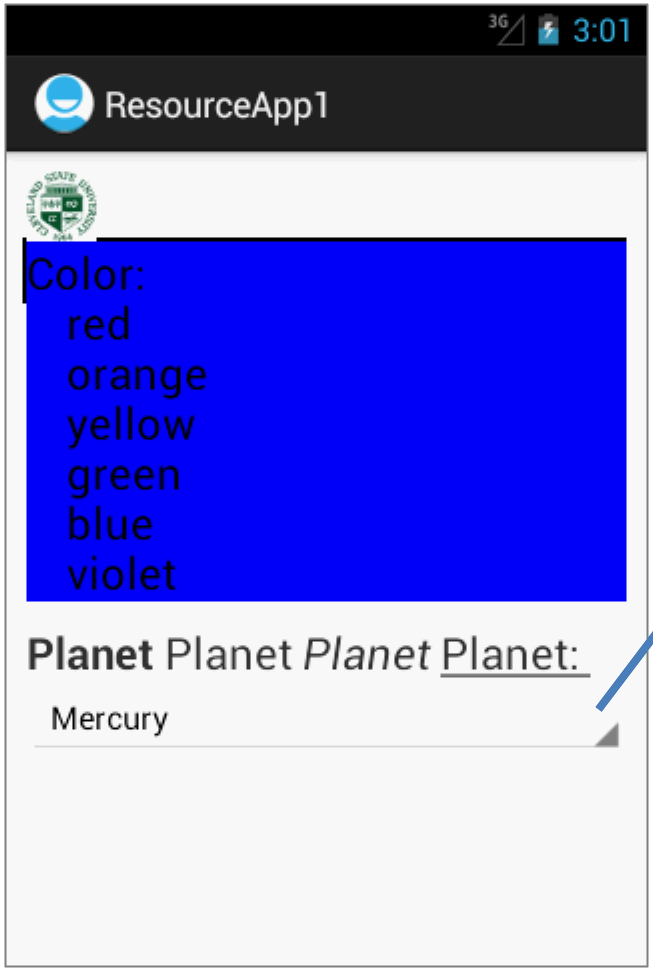
```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:padding="10dip" >

    <ImageView
        android:id="@+id/ImageView01"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content" />

    <EditText
        android:id="@+id/txtColorBox"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:background="@color/solid_blue"
        android:textSize="22dp" />

    <TextView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginTop="10dip"
        android:text="@string/spinner_1_planet"
        android:textSize="22dp" />

    <Spinner
        android:id="@+id/spinner2"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:drawSelectorOnTop="true"
        android:prompt="@string/spinner_1_planet_prompt" />
</LinearLayout>
```

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <drawable name="red">#7f00</drawable>
    <drawable name="blue">#770000ff</drawable>
    <drawable name="green">#7700ff00</drawable>

    <color name="solid_red">#f00</color>
    <color name="solid_blue">#0000ff</color>
    <color name="solid_green">#f0f0</color>
    <color name="solid_yellow">#ffffff00</color>
</resources>
```

/res/values/colors.xml

Android Resources

Example1. Using Embedded Resources (drawable, string, array).



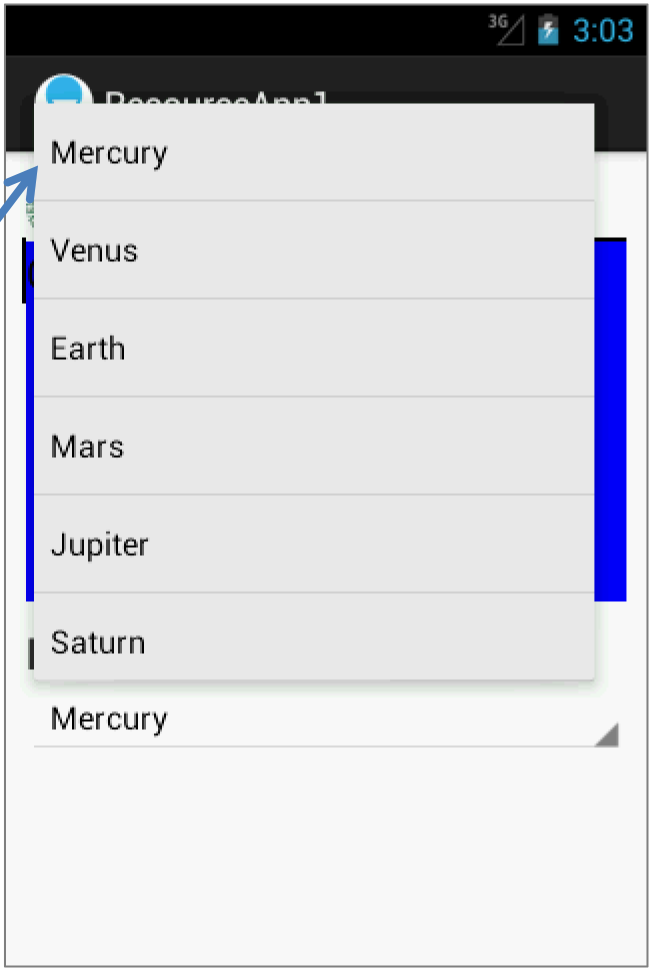
3:01

ResourceApp1

Color:
red
orange
yellow
green
blue
violet

Planet Planet *Planet* <u>Planet:</u>

Mercury



3:03

ResourceApp1

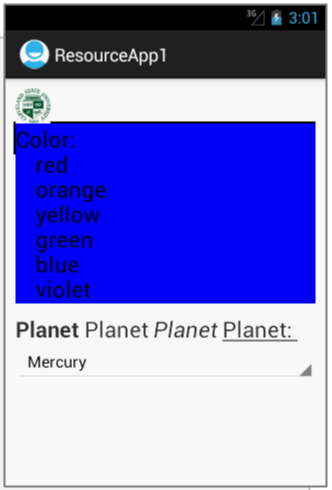
Mercury
Venus
Earth
Mars
Jupiter
Saturn

Mercury

Android Resources

Example1. Using Embedded Resources (drawable, string, array).

```java
// using Resources (adapted from Android - ApiDemos)
public class ResourceApp1 extends Activity {

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main_layout);

      EditText txtColorBox = (EditText) findViewById(R.id.txtColorBox);

      ImageView imageView1 = (ImageView) findViewById(R.id.ImageView01);

      // similar to xml layout android:src="@drawable/android_green_3d"
      imageView1.setImageResource(R.drawable.csuseal);

      String myColors[] = this.getResources().getStringArray(R.array.colors);

      String msg = this.getString(R.string.spinner_1_color);
      for (int i = 0; i < myColors.length; i++) {
          msg += "\n\t" + myColors[i];
      }
      txtColorBox.setText(msg);
```

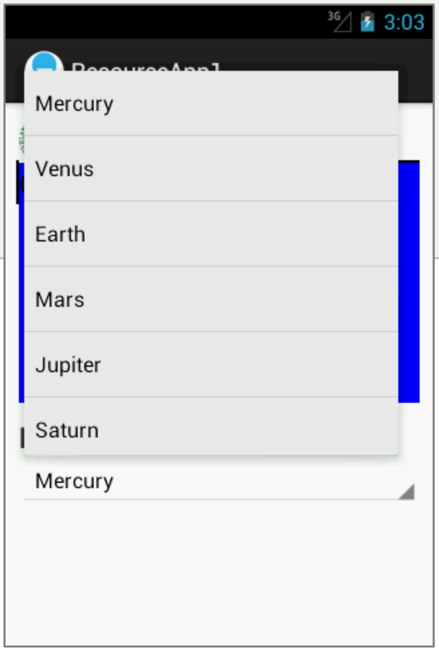


```
3:01
ResourceApp1
Color:
   red
   orange
   yellow
   green
   blue
   violet
Planet Planet Planet Planet:
Mercury
```

Android Resources

Example1. Using Embedded Resources (drawable, string, array).

```
      Spinner s2 = (Spinner) findViewById(R.id.spinner2);

      ArrayAdapter<CharSequence> adapter = ArrayAdapter.createFromResource(
                                              this, R.array.planets,
                                              android.R.layout.simple_spinner_item);

      adapter.setDropDownViewResource(android.R.layout.simple_spinner_dropdown_item);

      s2.setAdapter(adapter);

    }//onCreate

}//class
```



Resources

Questions ?