



# Programming with Android:
# Application Resources

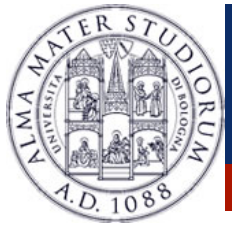
**Luca Bedogni** **Marco Di Felice**

**Dipartimento di Scienze dell'Informazione**

**Università di Bologna**

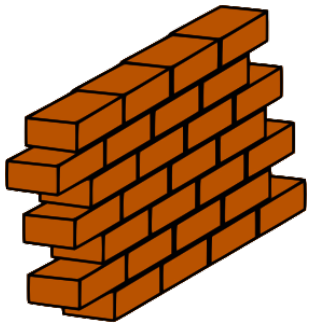
# Outline

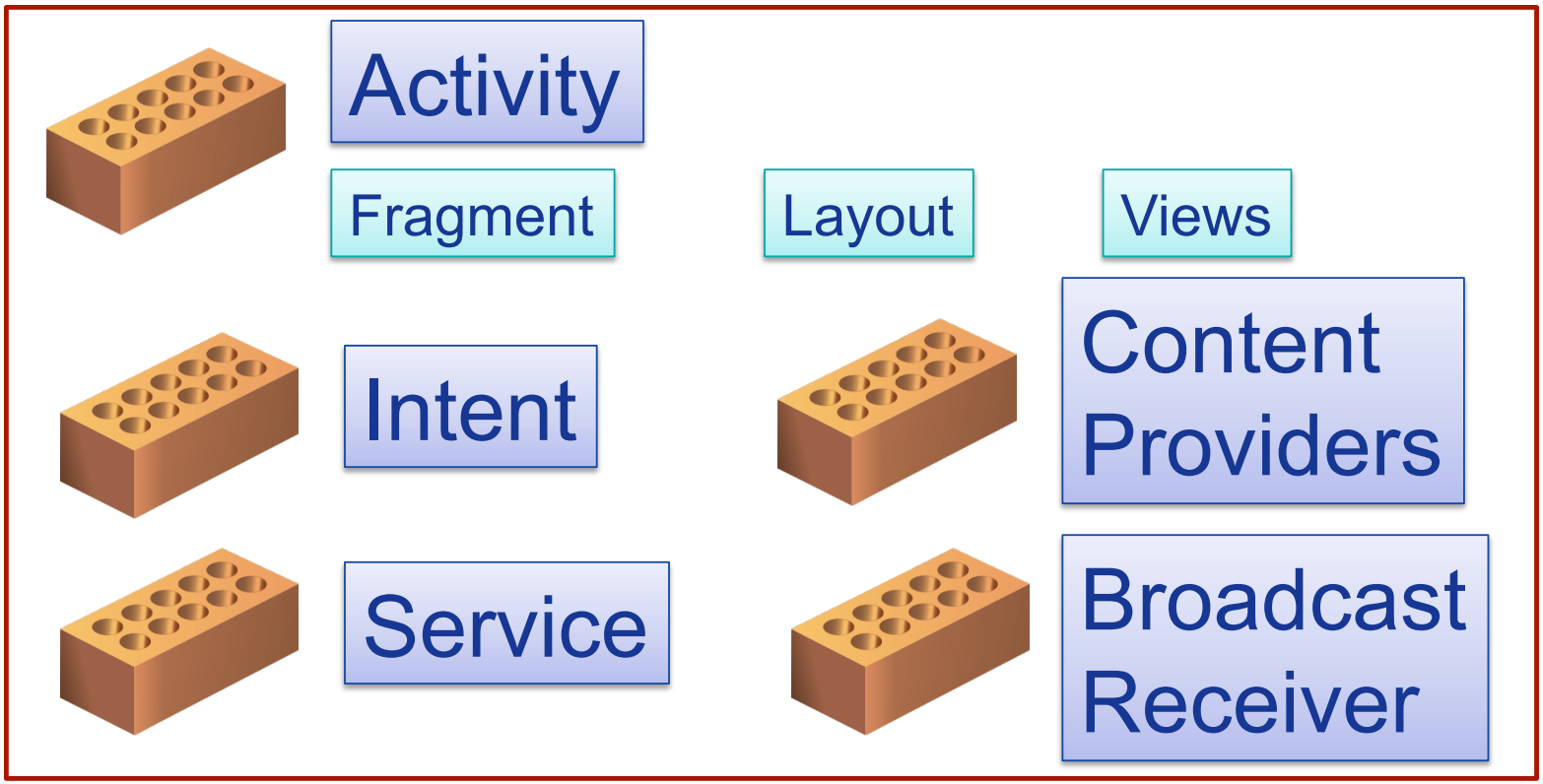
- What is a **resource**?
- **Declaration** of a resource
- Resource **type**: *integer*, *string*, *array, color, style*
- Resource **type**: *layout*
- Resource **type**: *drawable*, *raw*, *xml*
- Accessing resources from **XML** and **Java** code
- Providing **Configuration-specific** resources



# Android Applications Design

- Developing an Android Application means using in a proper way the **Android basic components** …



# Application Resources Definition

An Android Application is composed of: **code** and **resources**.


**Java code**

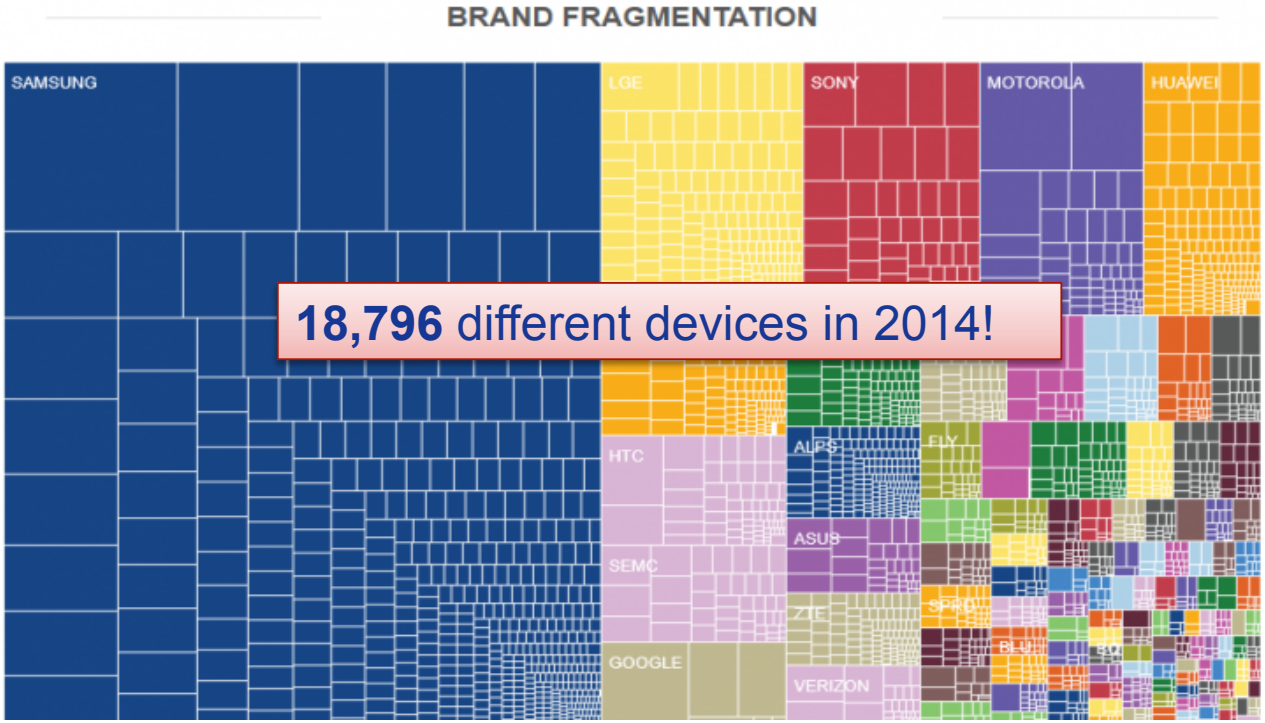
**Static data:**
- Text strings
- Colors
- Layout
- Sound
- Images
- …

Utilization of Resources… why?

- **Separate** data presentation (layout) from data management
- **Provide** alternative resources to support specific device configurations (e.g. different language packs)

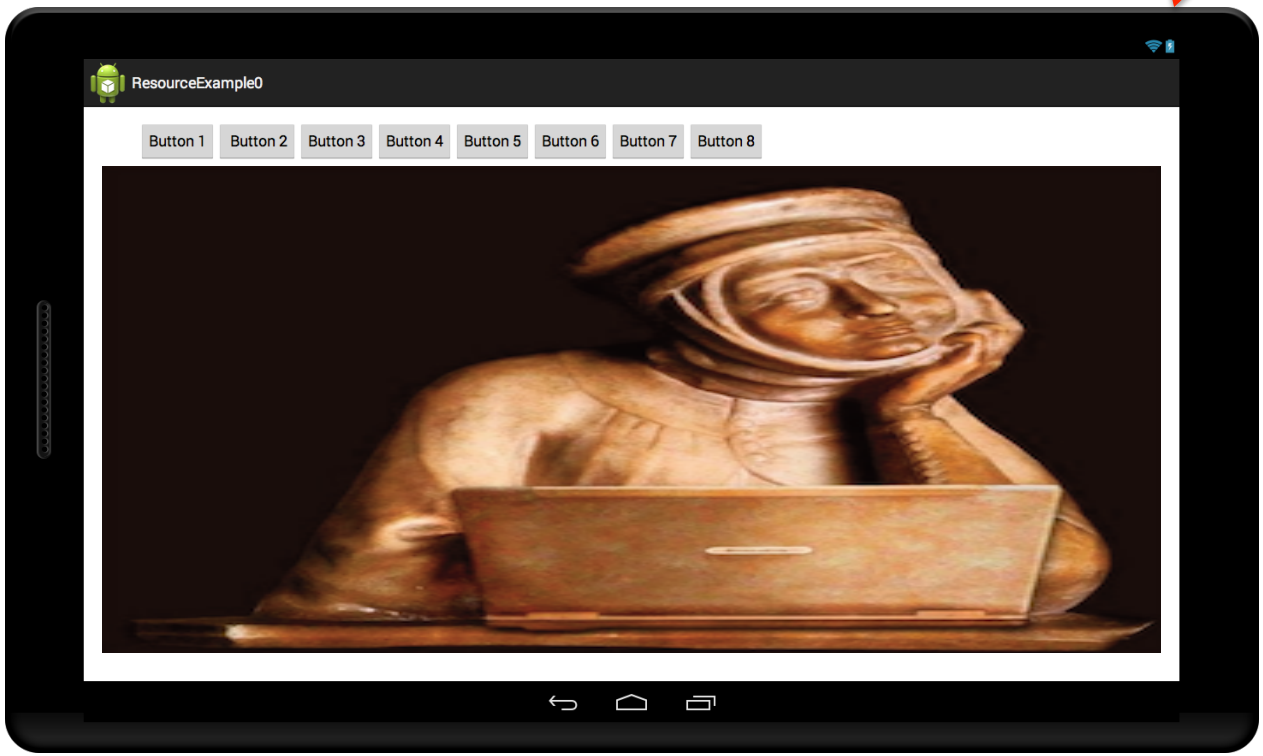
# Application Resources Definition

**PROBLEM.** An Android application might run on heterogenous devices with different characteristics (e.g. screen size, language support, keyboard type, input devices, etc).


# Application Resources Definition


The same **application layout** with 8 buttons, on a **tablet** and on a **smartphone** device.



# Application Resources Definition

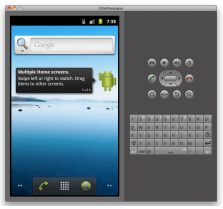
Handle the **devices' hetereogenity** writing Java code?

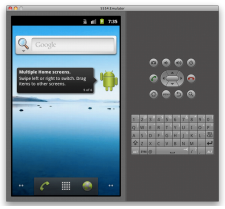
1. Provide different versions of the Mobile Application → **UNFEASIBLE**
2. Support all configurations within a single App→ **OK, but code is full of if-then-else**

**ANDROID WAY**. Separate code from UI. Use declative XML-based approch to define resources (images, files, layout, text, etc)

# Application Resources Definition

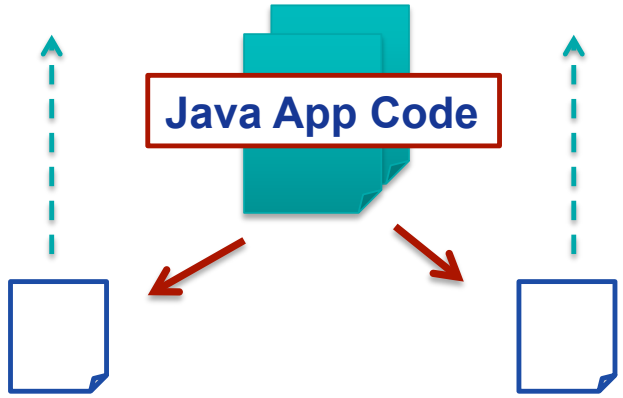
EXAMPLE





**Device 1**
**HIGH** screen pixel density

**Device 2**
**LOW** screen pixel density



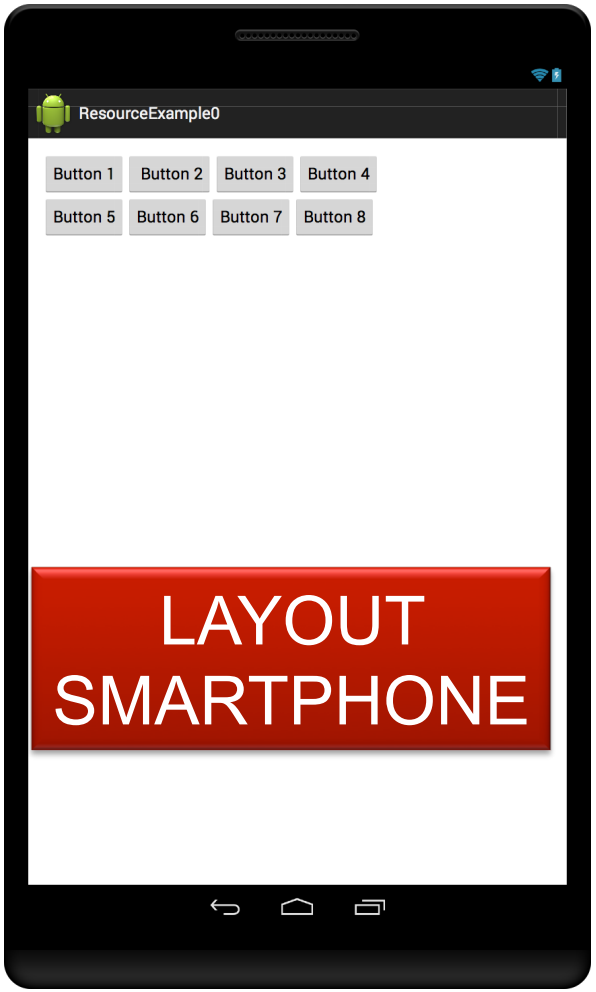

**XML Layout File**
**Device 1**

**XML Layout File**
**Device 2**

- Build the **application layout** through XML files (like HTML)
- Define **two** different XML **layouts** for two different devices
- At **runtime**, Android detects the current device configuration and loads the appropriate resources for the application
- **No need to recompile**!
- Just add a new XML file if you need to support a new device

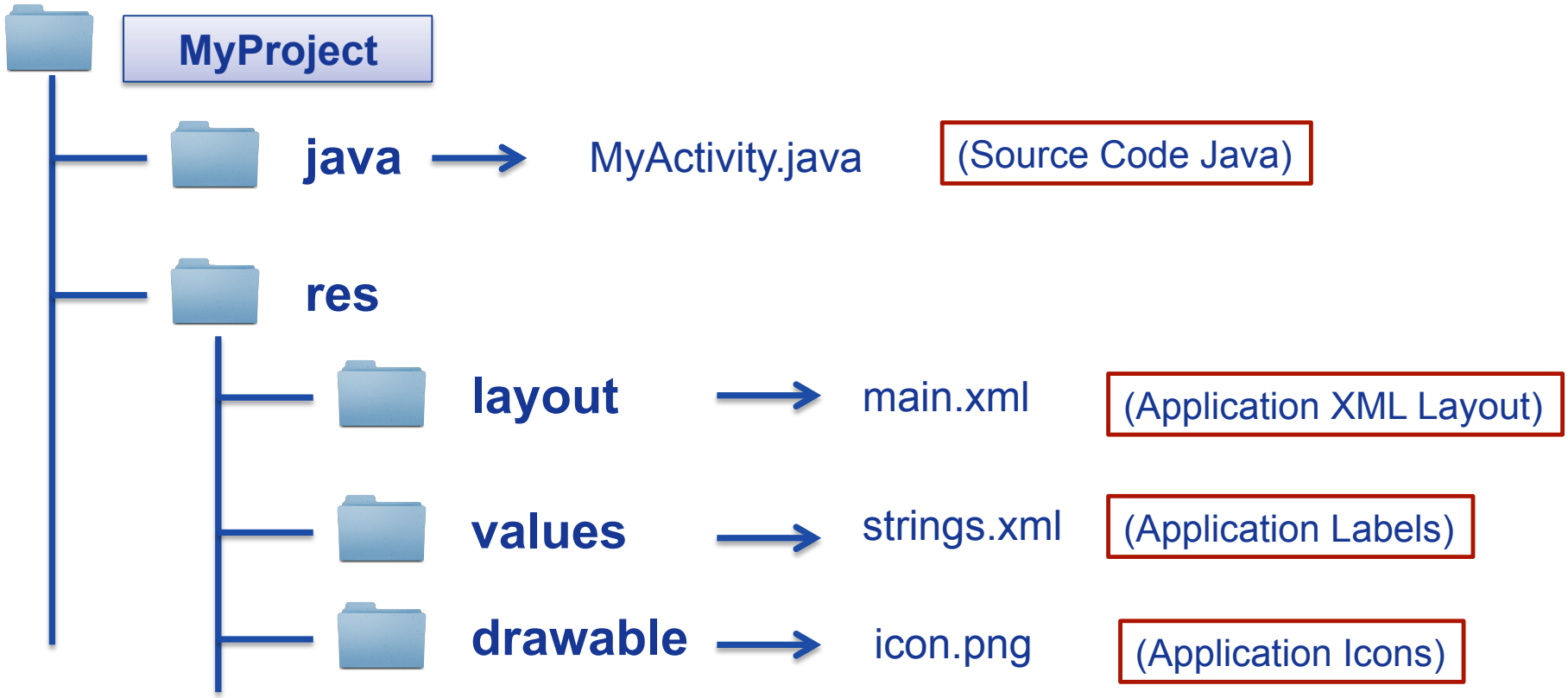
# Application Resources Definition

The same **application layout** with 8 buttons, on a **tablet** and on a **smartphone** (Nexus 7) device.




# Application Resources Definition

- Resources are defined in the **res**/ folder of the project.



# Application Resources Definition

| Resource Type | Resource contained |
|---|---|
| **res/animator** | *XML files that define property animations.* |
| **res/anim** | *XML files that define tween animations.* |
| **res/drawable** | *Bitmap files (.png, .jpg, .gif) or XML files that are compiled into other resources.* |
| **res/layout** | *XML files that define a user interface layout.* |
| **res/menu** | *XML files that define application menus.* |
| **res/mipmap** | *Drawable files for different launcher icon densities* |
| **res/raw** | *Arbitrary files to save in their raw form.* |
| **res/values** | *XML files that contain simple values, such as strings, integers, array.* |
| **res/xml** | *Arbitrary XML files.* |

# Application Resources Definition

(COMMON) **CHARACTERISTICS** of RESOURCES

- **Resources** can be: XML files, XML tags, or raw files (e.g. a picture or an audio file)
- Each resource has a unique **name or identifier** (used to identify/retrieve it from Java/XML code).
- There might be **multiple alternatives** of the same resource for different devices configuration (e.g. two pictures, with same name, for different screen's size).

# Application Resources Definition

| Resource Type | Resource contained |
|---|---|
| **res/animator** | *XML files that define property animations.* |
| **res/anim** | *XML files that define tween animations.* |
| **res/drawable** | *Bitmap files (.png, .jpg, .gif) or XML files that are compiled into other resources.* |
| **res/layout** | *XML files that define a user interface layout.* |
| **res/menu** | *XML files that define application menus.* |
| **res/mipmap** | *Drawable files for different launcher icon densities* |
| **res/raw** | *Arbitrary files to save in their raw form.* |
| **res/values** | *XML files that contain simple values, such as strings, integers, array.* |
| **res/xml** | *Arbitrary XML files.* |

# Resources Types: string and array

| Resource Type | File | Java constant | XML tag | Description |
|---|---|---|---|---|
| **string** | Any file in the res/values/ | R.string.<key> | <string> | **String** value associated to a key. |
| **integer** | Any file in the res/values/ | R.integer.<key> | <integer> | **Integer** value associated to a key. |
| **array** | Any file in the res/values/ | R.array.<key> | <string-array><br><item><br><item><br></string-array> | **Array of strings**. Each element is a described by an <item> |
| **array** | Any file in the res/values/ | R.array.<key> | <integer-array><br><item><br><item><br></integer-array> | **Array of integers**. Each element is a described by an <item> |

# Resources Types: string and array

MYVALUES.XML

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

    <string name="app_title"> My Application
    </string>

    <string name="label" > Hello world!
    </string>

    <integer name="myvalue"> 53 </integer>
</resources>
```

# Resources Types: string and array

MYVALUES.XML

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string-array name="nameArray">
        <item> John </item>
        <item> Michael </item>
    </string-array>
    <integer-array name="valArray">
        <item> 1 </item>
        <item> 2 </item>
    </integer-array>
</resources>
```

# Resources Types: color, dimension, style

| Resource Type | File | Java constant | XML tag | Description |
|---|---|---|---|---|
| **color** | Any file in the res/values/ | R.color.<key> | <color> | Definition of **colors** used in the GUI |
| **dimension** | Any file in the res/values/ | R.dimen.<key> | <dimen> | Dimension **units** of the GUI components |
| **style**<br>**theme** | Any file in the res/values/ | R.style.<key> | <style> | **Themes** and **styles** used by applications or by components |

# Resources Types: color, dimension, style

VALUES.XML

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <color name="red"> #FF0000
    </color>
    <color name="red_trasparent"> #66DDCCDD
    </color>
</resources>
```

- Color values can be defined based on one of these syntax rules: **#RGB**, **#ARGB**, **#RRGGBB**, **#AARRGGBB** (R=*red*, G=*green*, B=*blue*, A=*transparency*).

# Resources Types: color, dimension, style

| Code | Description |
|---|---|
| px | Pixel units |
| in | Inch units |
| mm | Millimeter units |
| pt | Points of 1/72 inch |
| dp | Abstract unit, independent from pixel density of a display |
| sp | Abstract unit, independent from pixel density of a display (font) |

These units are relative to a 160 dpi (dots per inch) screen, on which 1dp is roughly equal to 1px. When running on a higher density screen, the number of pixels used to draw 1dp is scaled up by a factor appropriate for the screen's dpi.

# Resources Types: color, dimension, style

VALUES.XML

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
     <dimen name="widthButton"> 50dp
     </dimen>
     <dimen name="sizeText"> 14sp
     </dimen>
     <dimen name="heightButton"> 14px
     </dimen>
</resources>
```

# Resources Types: color, dimension, style

MYVALUES.XML

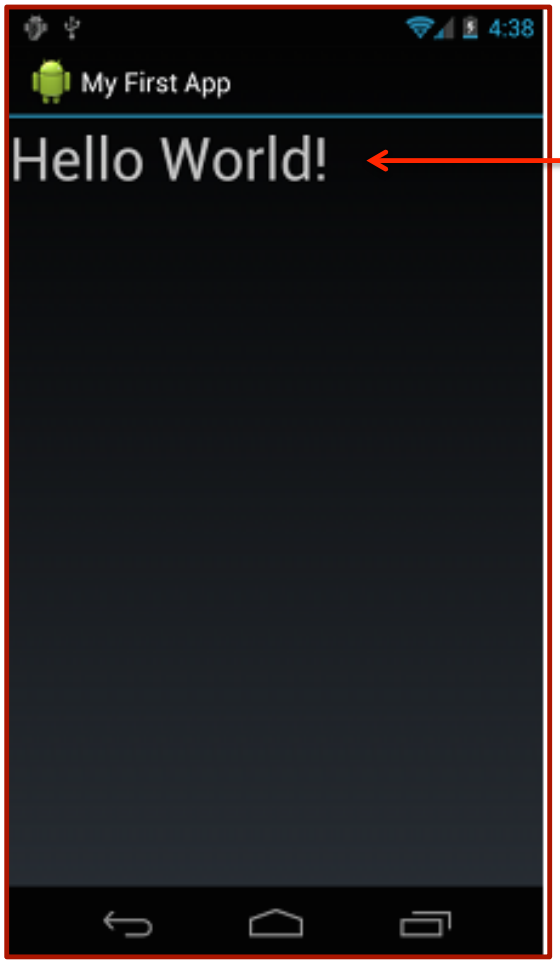
```
<?xml version="1.0" encoding="utf-8"?>
<resources>
      <style name="CustomText" parent="@style/Text">
            <item name="android:textSize">20sp
            </item>
            <item name="android:textColor">#008
            </item>
      </style>
</resources>
```

A **Style** is a set of **attributes** that can be applied to a specific component of the GUI (View) or to the whole screen or application (in this case, it is also referred as "*theme*").

# Application Resources Definition

| Resource Type | Resource contained |
|---|---|
| **res/animator** | *XML files that define property animations.* |
| **res/anim** | *XML files that define tween animations.* |
| **res/drawable** | *Bitmap files (.png, .jpg, .gif) or XML files that are compiled into other resources.* |
| **res/layout** | *XML files that define a user interface layout.* |
| **res/menu** | *XML files that define application menus.* |
| **res/mipmap** | *Drawable files for different launcher icon densities* |
| **res/raw** | *Arbitrary files to save in their raw form.* |
| **res/values** | *XML files that contain simple values, such as strings, integers, array.* |
| **res/xml** | *Arbitrary XML files.* |

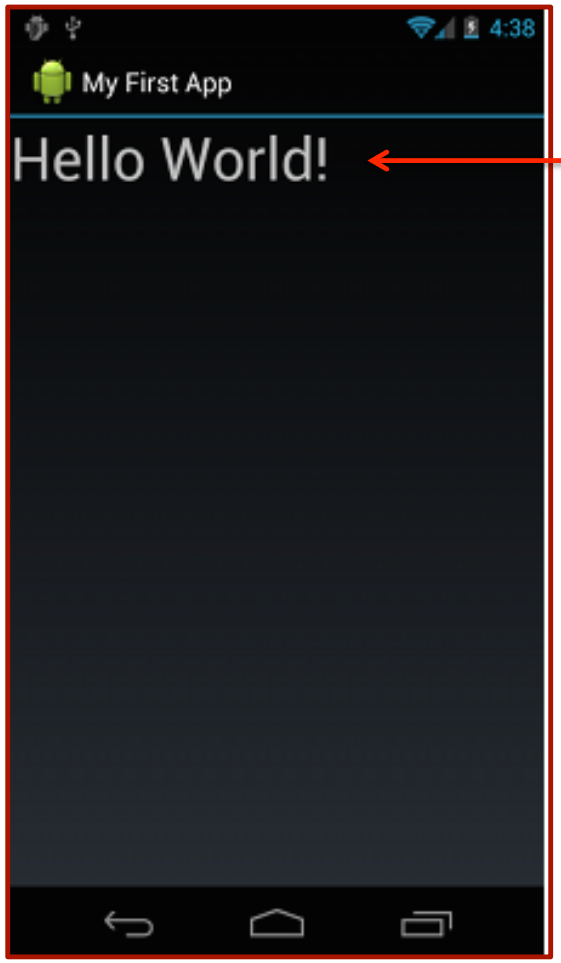
# Resources Types: layout




A **Layout** resource is an XML file describing the **User Interface** of the Component (e.g. Activity)

```
<TextView
 android:layout_width="wrap_content"
 android:layout_height="wrap_content"
 android:text=”Hello World!
 android:id="@+id/mytext" />
```

# Resources Types: layout




A Layout File is composed of **Views**

Each **XML Tag** is a View

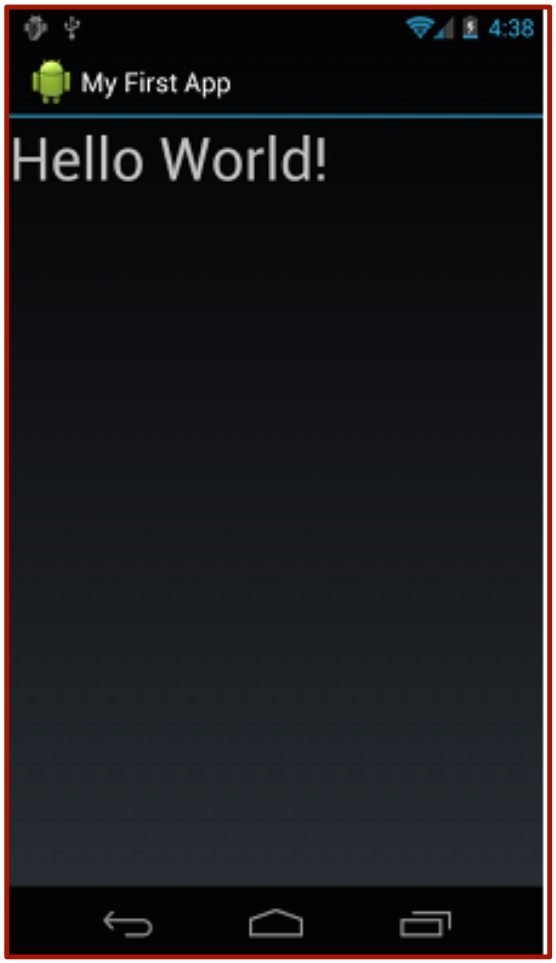
View **type**

View **properties**

```
<TextView
  android:layout_width="wrap_content"
  android:layout_height="wrap_content"
  android:text=”Hello World!
  android:id="@+id/mytext" />
```

View **ID**

# Resources Types: layout

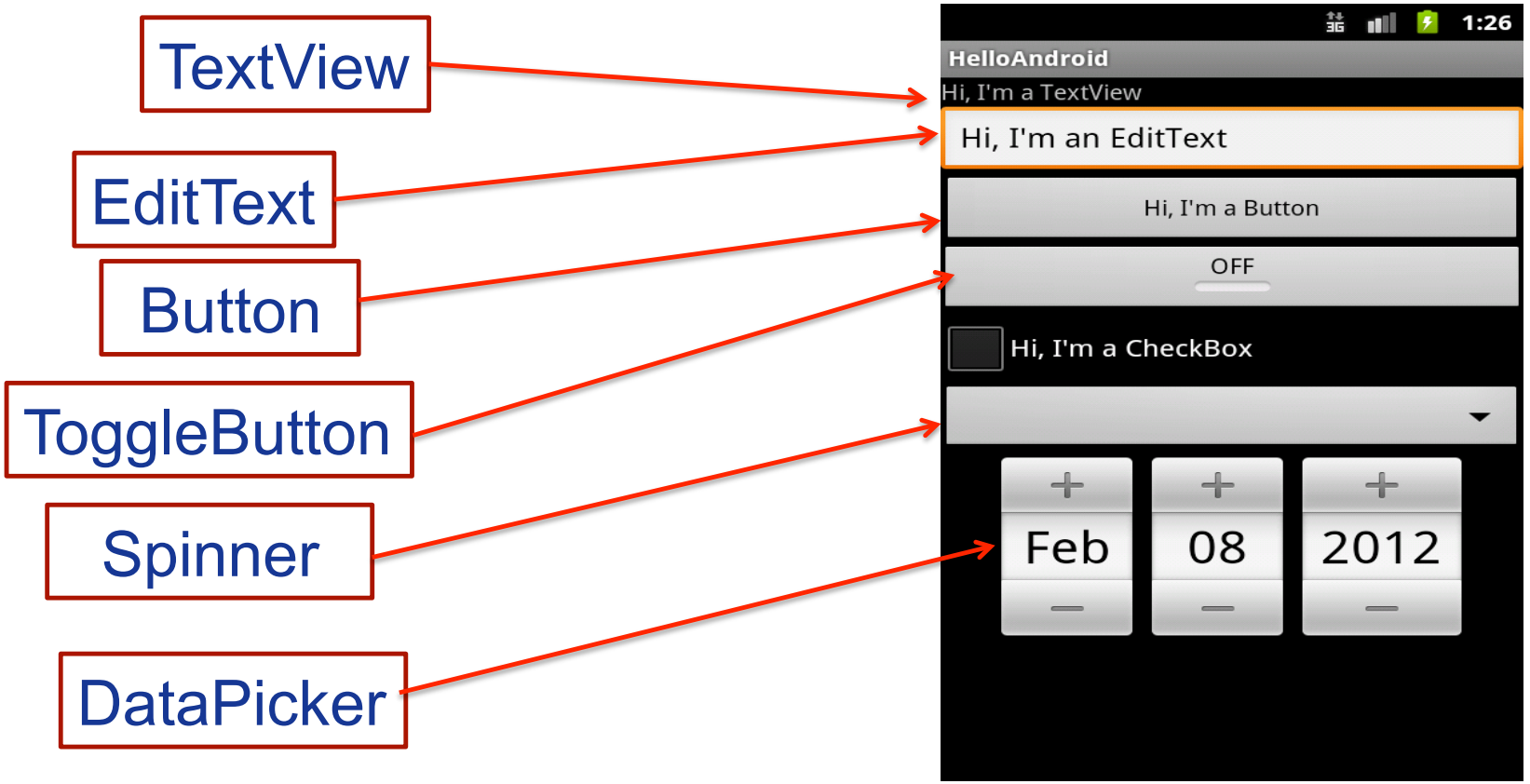



A Layout File is composed of **ViewGroup**

```
<LinearLayout
 android:orientation="vertical"
 android:layout_width="fill_parent"
 android:layout_height="fill_parent”>

<TextView
  android:layout_width="wrap_content"
  android:layout_height="wrap_content"
  android:text=”Hello World!
  android:id="@+id/mytext" />
<LinearLayout/>
```

# Resources Types: layout



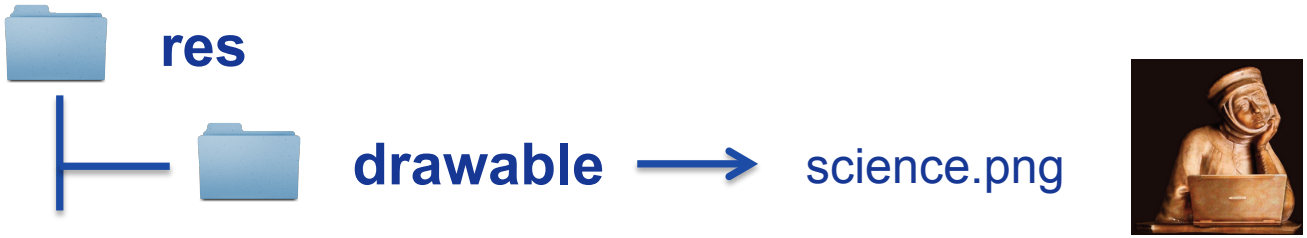
# Application Resources Definition

| Resource Type | Resource contained |
|---|---|
| **res/animator** | *XML files that define property animations.* |
| **res/anim** | *XML files that define tween animations.* |
| **res/drawable** | *Bitmap files (.png, .jpg, .gif) or XML files that are compiled into other resources.* |
| **res/layout** | *XML files that define a user interface layout.* |
| **res/menu** | *XML files that define application menus.* |
| **res/mipmap** | *Drawable files for different launcher icon densities* |
| **res/raw** | *Arbitrary files to save in their raw form.* |
| **res/values** | *XML files that contain simple values, such as strings, integers, array.* |
| **res/xml** | *Arbitrary XML files.* |

# Resources Types: drawable

A **Drawable** resource is a general concept for a graphic that can be drawn to the screen.

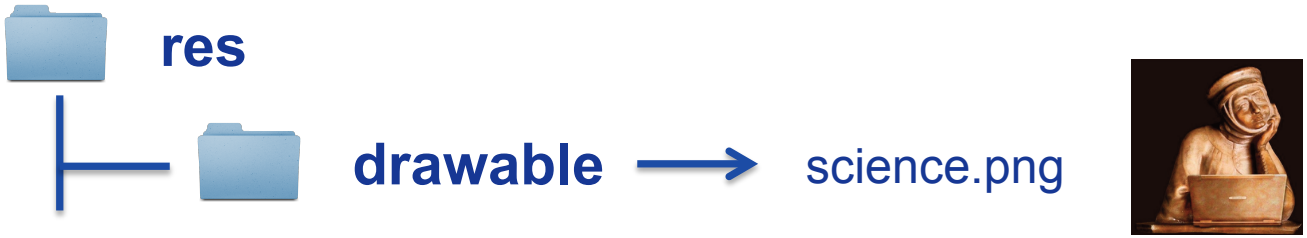
- **BitMapFile**: .jpeg, .png, .gif file
- **XML File**: define a shape, a graphic effect, a transition between different states, etc



# Resources Types: drawable

A **Drawable** resource is a general concept for a graphic that can be drawn to the screen.

- **BitMapFile**: .jpeg, .png, .gif file
- **XML File**: define a shape, a graphic effect, a transition between different states, etc



# Resources Types: drawable

A **Drawable** resource is a general concept for a **graphic** that can be drawn to the screen.

| Drawable type | Description |
|---|---|
| **BitMap File** | A bitMap Graphic file (.png, .gif. .jpeg) |
| **Nine-Patch File** | A PNG file with stretchable regions to allow resizing |
| **Layer List** | A Drawable managing an array of other drawable |
| **State List** | A Drawable that references different graphics based on the states |
| **Level List** | An XML managing alternate Drawables. Each assigned a value |
| **Transition** | A Drawable that can cross-fade between two Drawable |
| **Inset** | A Drawable that insets another Drawable by a specific distance |
| **Clip** | A Drawable that clips another Drawable based on its current level |
| **Scale** | A Drawable that changes the size of another Drawable |
| **Shape** | An XML file that defines a geometric shape, colors and gradients |

# Resources Types: drawable

A **Drawable** resource is a general concept for a graphic that can be drawn to the screen.

MYSHAPE.XML

```
<?xml version="1.0" encoding="utf-8"?>
<shape xmlns:android="http://schemas.android.com/apk/res/android"
    android:shape="rectangle">
    <gradient
        android:startColor="#FFFF0000"
        android:endColor="#80FF00FF"
        android:angle="45"/>
    <padding android:left="7dp"
        android:top="7dp"
        android:right="7dp"
        android:bottom="7dp" />
    <corners android:radius="8dp" />
</shape>
```

# Resources Types: drawable

A **Drawable** resource is a general concept for a graphic that can be drawn to the screen.

MYSELECTOR.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" />
    <item android:state_focused="true"
          android:drawable="@drawable/button_focused" />
    <item android:state_hovered="true"
          android:drawable="@drawable/button_focused" />
    <item android:drawable="@drawable/button_normal" />
</selector>
```

# Access to Application Resources

- Each Resource is associated with an **Identifier** (ID), that is composed of two parts:
  - The resource **type**: Each resource is grouped into a "type," (e.g. string, color, menu, drawable, layout, etc)
  - The resource **name**, which is either: the filename, excluding the extension; or the value in the XML **<android:name>** attribute.
  - Identifiers must be unique!!
- *Two ways to access resources*:
  - From the **XML** Code
  - From the **Java** Code

# Access to Application Resources

- Each Resource is associated with an **Identifier** (ID), that is composed of two parts:
  - The resource **type**: Each resource is grouped into a "type," (e.g. string, color, menu, drawable, layout, etc)
  - The resource **name**, which is either: the filename, excluding the extension; or the value in the XML **<android:name>** attribute.
  - Identifiers must be unique!!
- *Two ways to access resources*:
  - **From the XML Code**
  - From the **Java** Code

# Access to Application Resources: XML

**@[<*package_name*>:]<*resource_type*>/<*resource_name*>**

- <**package_name**> is the name of the package in which the resource is located (not required when referencing resources from the same package)
- <r**esource_type**> is the the name of the resource type
- <**resource_name**> is either the resource filename without the extension or the android:name attribute value in the XML element.

# Access to Application Resources: XML

VALUE RESOURCE /res/value folder

STRINGS.XML

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string name=”labelButton”> Submit </string>
    <string name=”labelText”> Hello world! </string>
</resources>
```

LAYOUT RESOURCE /res/layout folder

MAIN_ACTIVITY.XML

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <Textview android:id=“@+id/label1”
android:text=“@string/labelText” />
     <Button android:id=“@+id/button1”
android:text=“@string/labelButton”/>
  </resources>
```

# Access to Application Resources: XML

LAYOUT RESOURCE /res/layout folder

MAIN_ACTIVITY.XML

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <ImageView
      android:id="@+id/label1"
      android:src="@drawable/science" />
  </resources>
```

DRAWABLE RESOURCE /res/drawable folder


# Access to Application Resources

- Each Resource is associated with an **Identifier** (ID), that is composed of two parts:
  - The resource **type**: Each resource is grouped into a "type," (e.g. string, color, menu, drawable, layout, etc)
  - The resource **name**, which is either: the filename, excluding the extension; or the value in the XML **<android:name>** attribute.
  - Identifiers must be unique!!
- *Two ways to access resources*:
  - From the **XML** Code
  - **From the Java Code**

# Access to Application Resources: Java

Each **XML Resource** has a correspective **Java Object**


```
<TextView
 android:layout_width="wrap_content"
 android:layout_height="wrap_content"
 android:text=”Hello World!
 android:id="@+id/mytext" />
```

```
class TextView extends View {
...
CharSequence getText();
void getText(CharSequence);
}
```

# Access to Application Resources: Java

Each **XML Resource** has a correspective **Java Object**


**PROBLEM.** How to refer to a specific **XML** resource with a given **name**?

**ANSWER. Through the R class**, that works as a **glue** between the world of java and the world of resources

# Access to Application Resources: Java

STRINGS.XML

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

R.JAVA GENERATED **AUTOMATICALLY**, NO NEED TO MODIFY IT!

R.JAVA

```
public final class R {
    public static final class string {
        public static final int hello=0x7f040001;
         public static final int lbutton=0x7f040005;
    }
}
```

**R** contains **resource IDs** for all the resources in the **res**/ directory.

# Access to Application Resources: Java

**[<*package_name*>.]R.<*resource_type*>.<*resource_name*>**

- <**package_name**> is the name of the package in which the resource is located (not required when referencing resources from the same package)
- <r**esource_type**> is the **R** subclass for the resource type
- <**resource_name**> is either the resource filename without the extension or the android:name attribute value in the XML element.

# Access to Application Resources: Java

**STEP0**: ***Declare*** *resources in res/*

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

   <string name="hello"> Hello </string>
   <string name="label1"> Label  </string>
</resources>
```

**XML**-Based, Declarative Approach

**STEP1**: ***Compile*** *the project*

**STEP2**: **Access** *resources through R class*

```
public final class R {

    public static final class string {
        public static final int hello=0x7f040001;
         public static final int label1=0x7f040005;
    }

}
```

**Java** Code, Programmatic Approach

# Access to Application Resources: Java

**`getResources()`**

Get accesses to all resources of the application

**`getResources().getTTT(Name)`**

Get accesses to all resources of the application of type TTT and with name equal to Name

- `String getResources().getString(name)`
- `int getResouces().getInt(name)`
- `int getResources().getColor(name)`
- `Drawable getResources().getDrawable(name)`
- …

# Access to Application Resources: Java

EXAMPLE of JAVA code accessing RESOURCES

```
// Get a string resource named hello
String hello=getResources().getString(R.string.hello);

// Get a color resource named opaque_red
int color=getResources().getColor(R.color.opaque_red);

// Access an integer-array resource
int val[]=getResources().getIntArray(R.array.valArray);

// Access a Drawable resource
Drawable d=getResources().getDrawable(R.drawable.img);
```

# Access to Application Resources: Java

**`View findViewById(ID)`**

Get accesses to a **View** object with identifier ID

EXAMPLE of JAVA code accessing **LAYOUT** RESOURCES

```
// Access to a TextView with ID equal to label1
TextView msgTextView = (TextView)
     findViewById(R.id.label1);

msgTextView.setText(getResources().getString(R.string
.stringText));
```

# Access to Application Resources: Java

**View findViewById(ID)**

Get accesses to a **View** object with identifier ID

EXAMPLE of JAVA code accessing **LAYOUT** RESOURCES

```
// Access to a Button with ID equal to button1
Button myButton= (Button)
     findViewById(R.id.button);

myButton.setTextColor(getResources().getColor(R.color
.deepblue));
```

# Resources Alternatives

Android applications might provide **alternative resources** to support specific device configurations (e.g. different languages).

- At runtime, Android **detects** the current device configuration and **loads** the appropriate resources for the application.

- To specify configuration-specific alternatives:

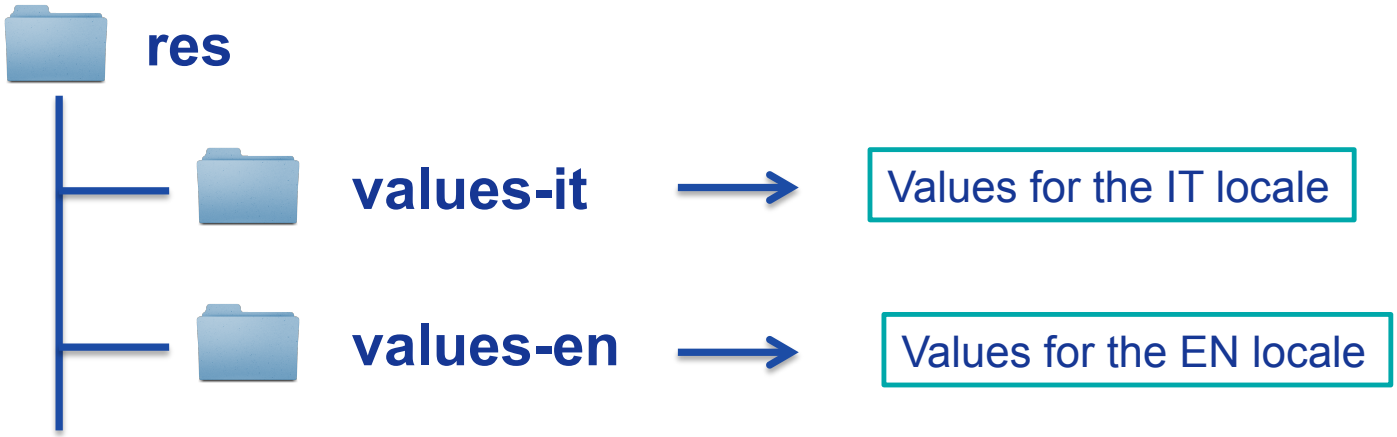
  1. Create a new directory in **res**/ named in the form ***<resources_name>-<config_qualifier>***

  2. Save the respective alternative resources in this new directory

# Resources Alternatives

Name of the folder: **<resources_name>-<config_qualifier>**.

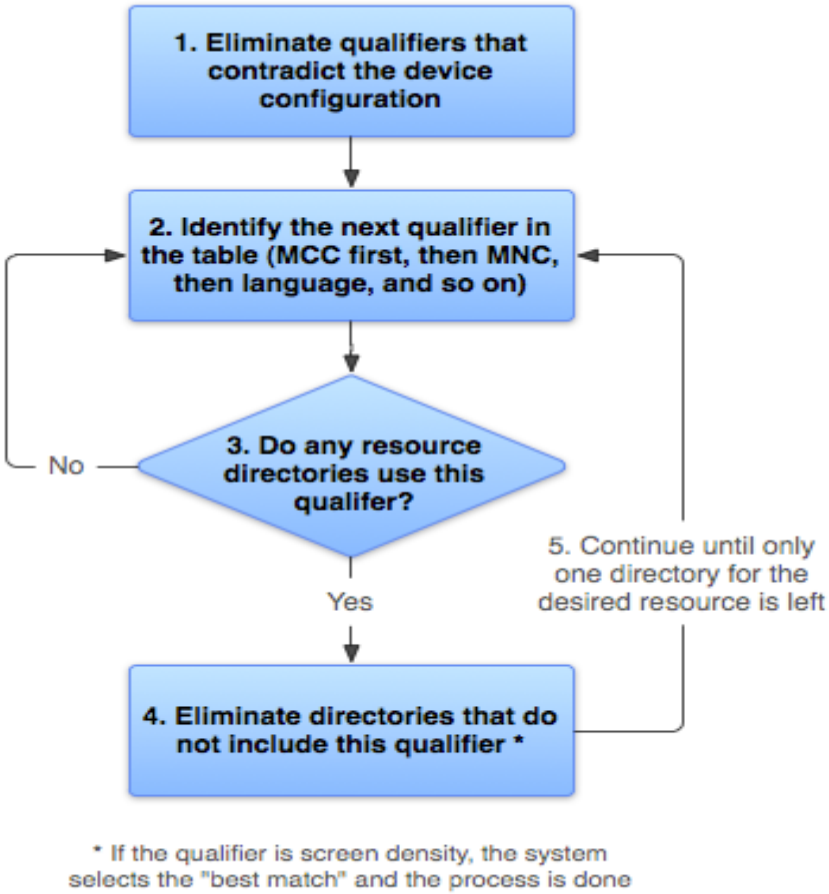
- <*resources_name*> is the directory name of the corresponding default resources (see previous slides).
- <*qualifier*> is a name that specifies an individual configuration for which these resources are to be used (see next slide).



# Resources Alternatives: Qualifiers

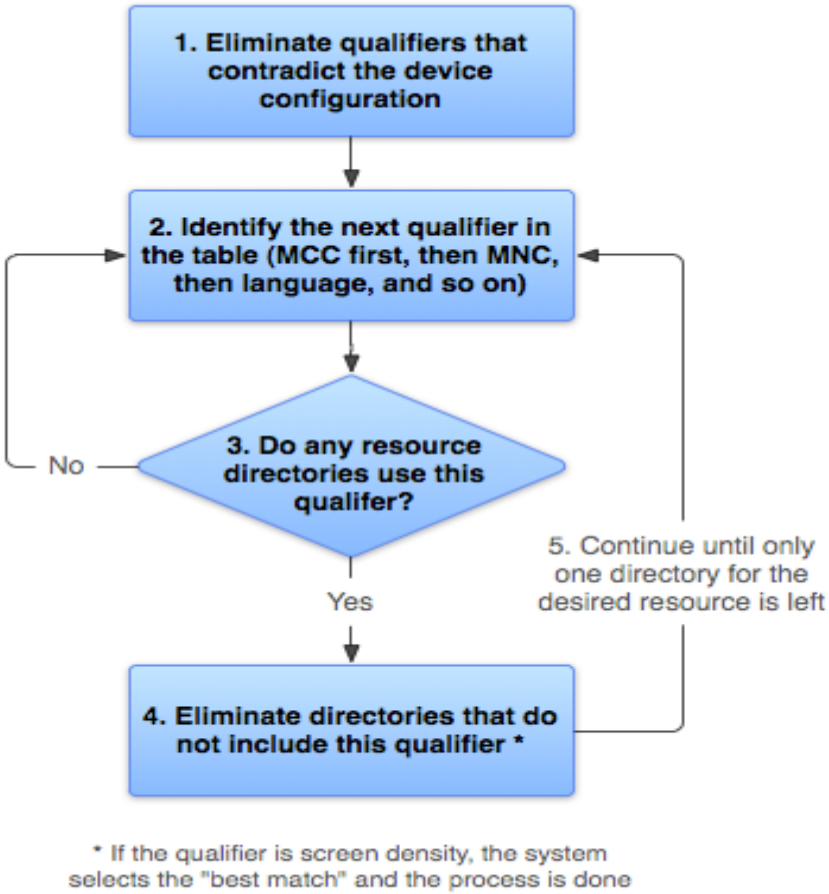
| Configuration | Values Example | Description |
|---|---|---|
| **MCC and MNC** | mcc310, mcc208, etc | mobile country code (MCC) |
| **Language and region** | en, fr, en-rUS, etc | ISO 639-1 language code |
| **smallestWidth** | sw320dp, etc | shortest dimension of screen |
| **Available width** | w720dp, w320dp, etc | minimum available screen width |
| **Available height** | h720dp, etc | minimum available screen height |
| **Screen size** | small, normal, large | screen size expressed in dp |
| **Screen aspect** | long, notlong | aspect ratio of the screen |
| **Screen orientation** | port, land | screen orientation (can change!) |
| **Screen pixel density (dpi)** | ldpi, mdpi, hdpi | screen pixel density |
| **Keyboard availability** | keysexposed, etc | type of keyword |
| **Primary text input method** | nokeys, qwerty | availability of qwerty keyboard |
| **Navigation key availability** | navexposed, etc | navigation keys of the application |
| **Platform Version (API level)** | v3, v4, v7, etc | API supported by the device |

# Resources Alternatives Matching




- When the application requests a resource for which there are multiple alternatives, **Android selects which alternative resource to use** at runtime, depending on the current device configuration, **through the algorithm shown in the Figure**.

# Resources Alternatives Matching

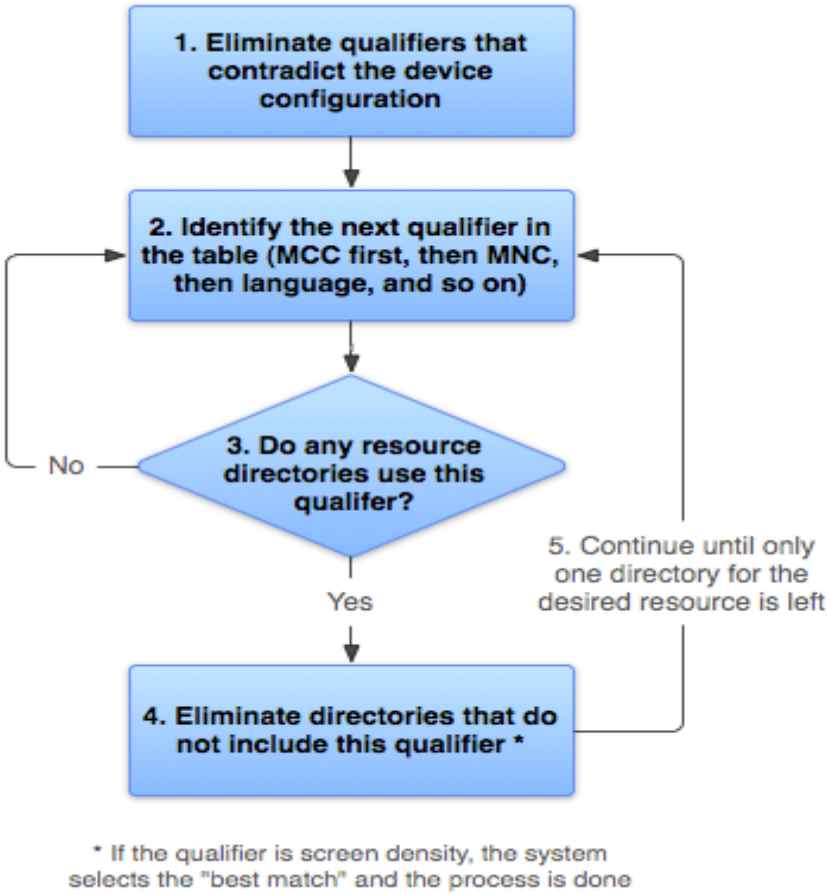



DEVICE CONFIGURATION

Locale = it
Screen orientation = port
Screen pixel density = hdpi
Touchscreen type = notouch
Primary text input method = 12key

drawable/
drawable-it/
drawable-fr-rCA/
drawable-it-port/
drawable-it-notouch-12key/
drawable-port-ldpi/
drawable-port-notouch-12key/

# Resources Alternatives Matching

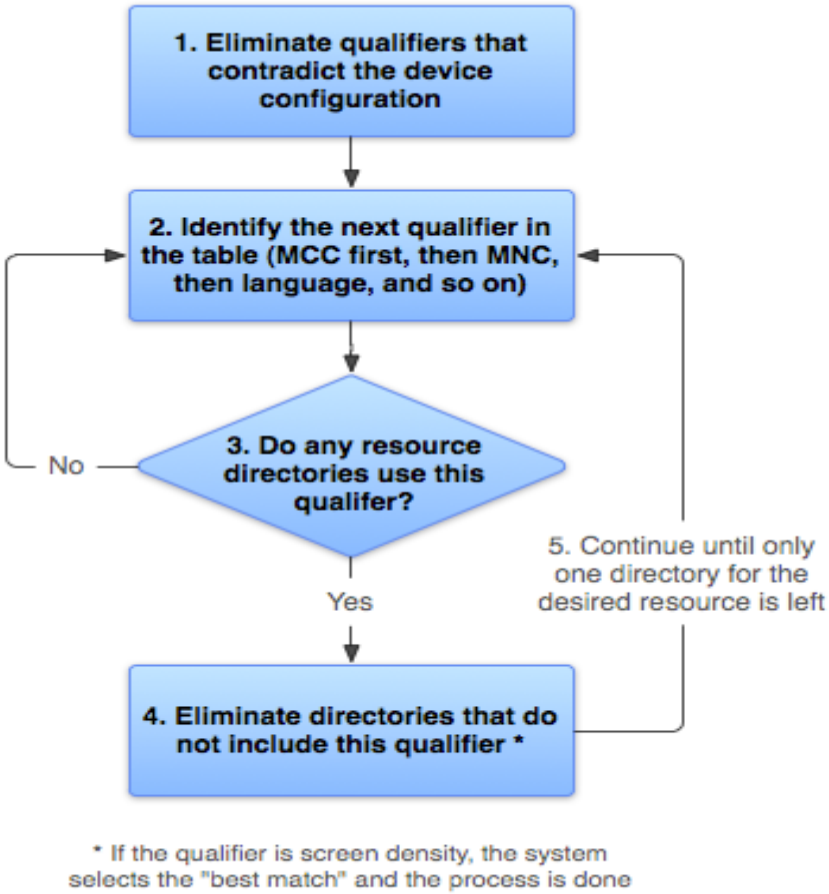



DEVICE CONFIGURATION

Locale = it
Screen orientation = port
Screen pixel density = hdpi
Touchscreen type = notouch
Primary text input method = 12key

~~drawable/~~
drawable-it/
~~drawable-fr-rCA/~~
drawable-it-port/
drawable-it-notouch-12key/
~~drawable-port-ldpi/~~
~~drawable-port-notouch-12key/~~

# Resources Alternatives Matching




**DEVICE** CONFIGURATION

Locale = it
Screen orientation = port
Screen pixel density = hdpi
Touchscreen type = notouch
Primary text input method = 12key

~~drawable/~~
drawable-it/
~~drawable-fr-rCA/~~
**drawable-it-port/**
~~drawable-it-notouch-12key/~~
~~drawable-port-ldpi/~~
~~drawable-port-notouch-12key/~~

# Resources Alternatives

**BEST PRACTICES**

- Provide **default** resources for your application.
- Provide **alternative resources** based on the target market of your application.
- Avoid **unnecessary or unused** resources alternatives.
- Use **alias** to reduce the duplicated resources.