Programming with Android: Application Resources

Federico Montori

Dipartimento di Informatica, Scienza e Ingegneria

Università di Bologna

Outline

- What is a **resource**?
- **Declaration** of a resource
- Resource **type**: *integer*, *string*, *array*
- Resource **type**: *color*, *dimension*, *style*
- Resource **type**: *drawable*, *raw*, *xml*
- Defining **Configuration-specific** resources
- Providing the **Best resources** for a device

Application Resources Definition

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

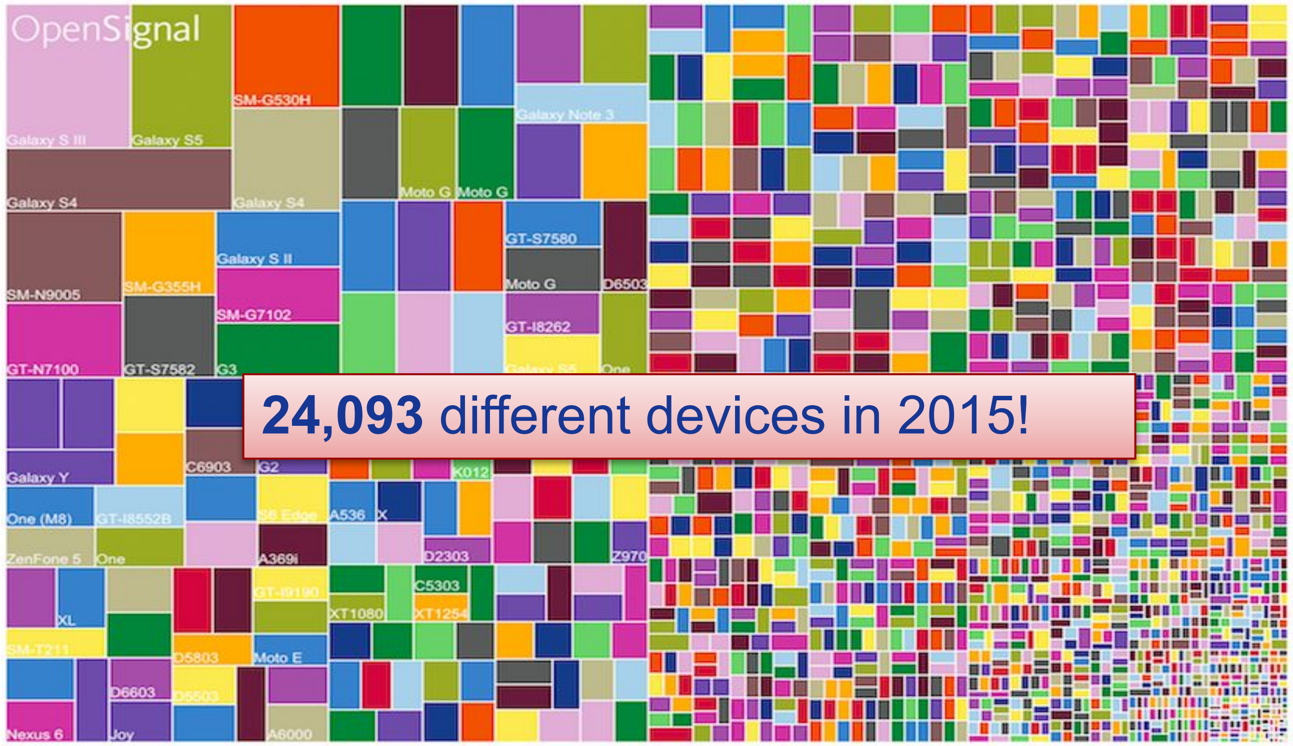
DEF. **Resources** **<u>are everything that is not code</u>** (including: XML layout files, language packs, images, audio/video files, etc)

Utilization of Resources… why?

- **Separate** data presentation (layout) from data management
- **Provide** alternative resources to support specific device configurations (e.g. different language packs)
- **Re-compile** <u>only when strictly needed</u>!

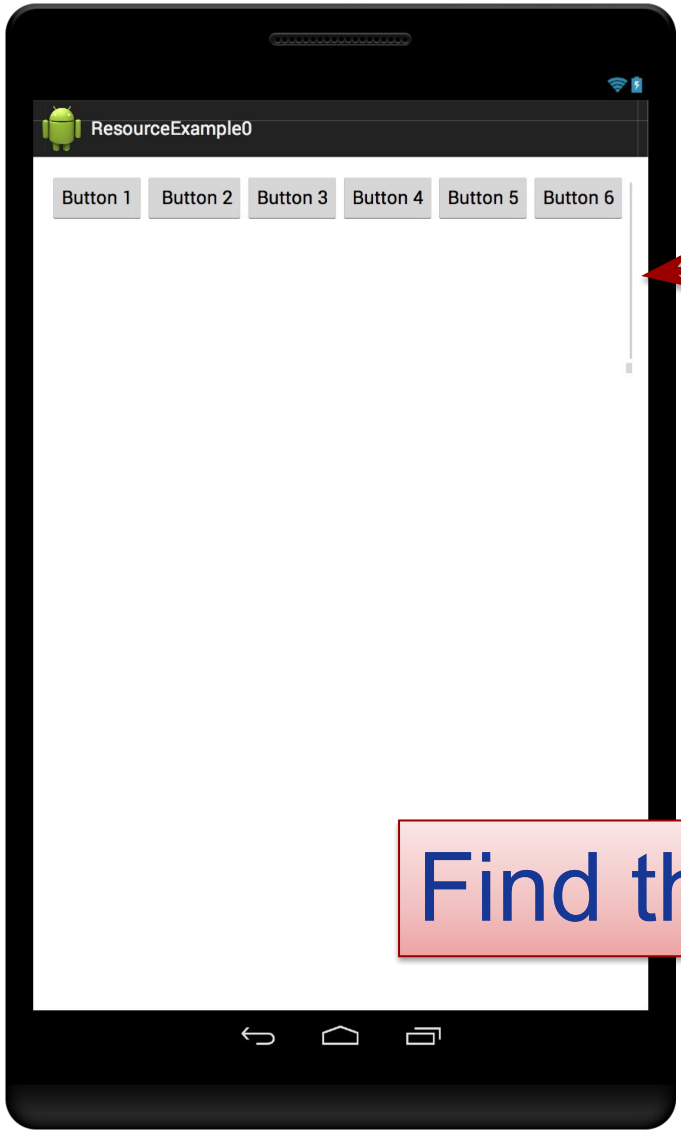
Application **Resources** Definition

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

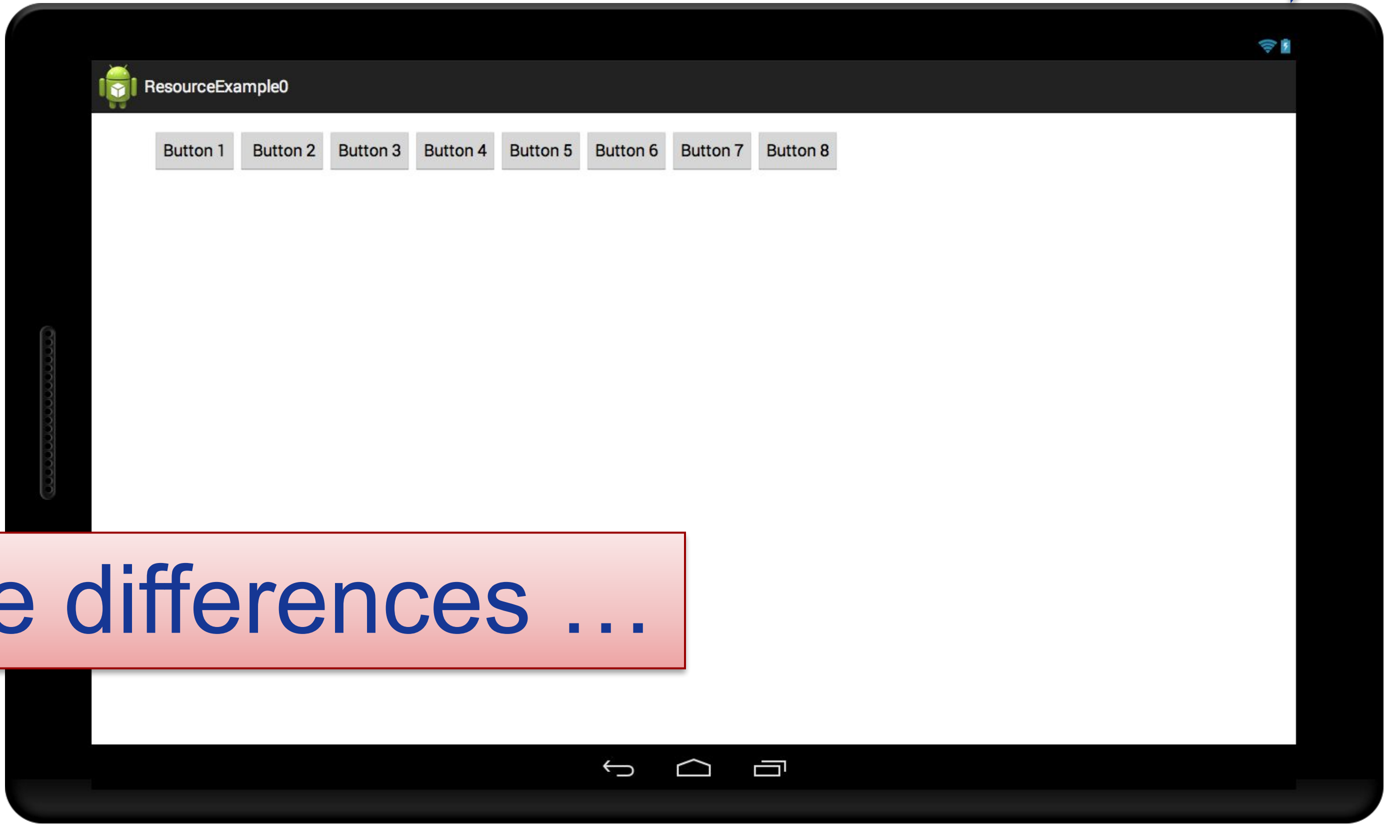


24,093 different devices in 2015!

Application Resources Definition



The same **application layout** with 8 buttons, on a **tablet**



Find the differences …

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).

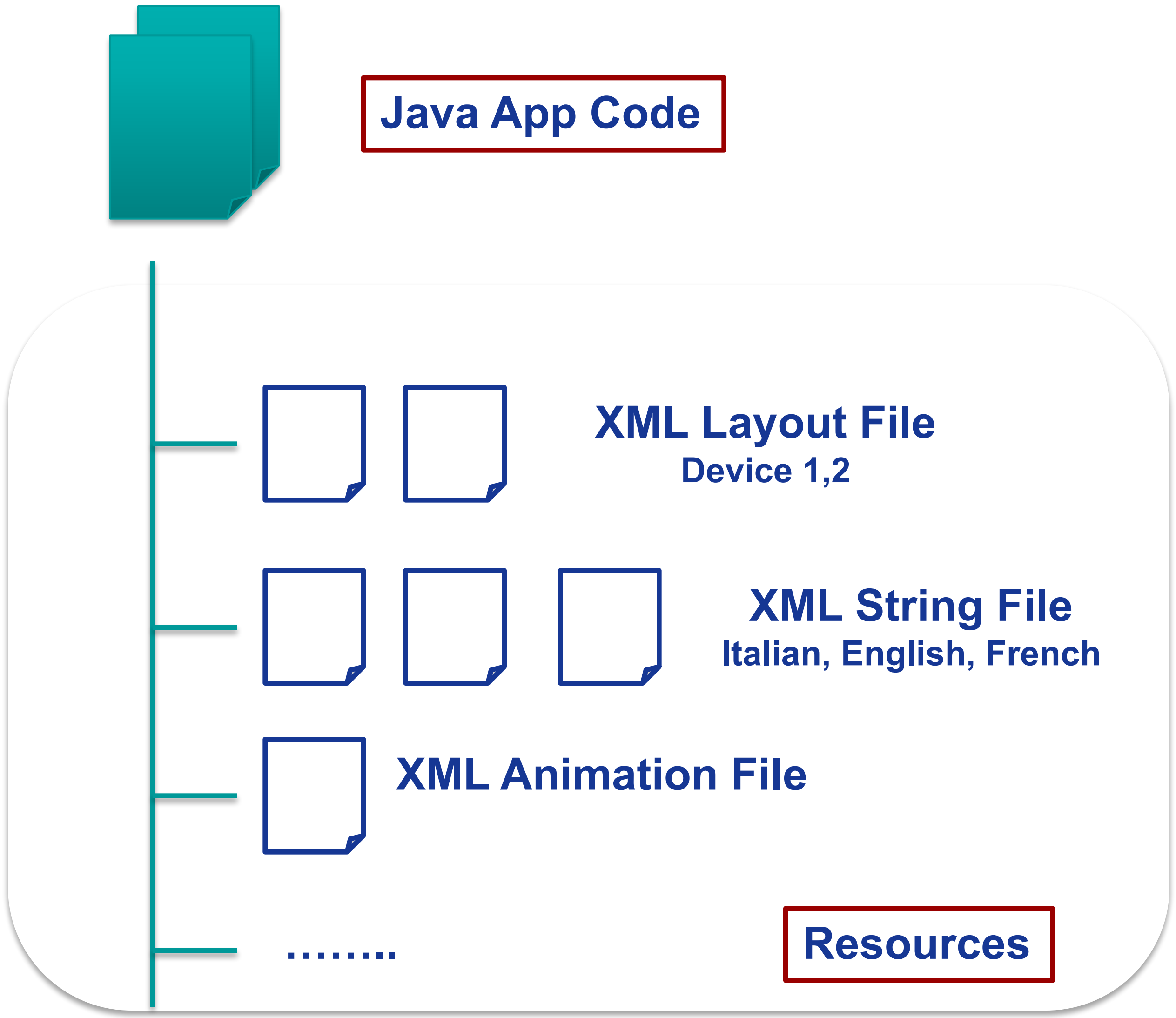
TRADITIONAL SOLUTION. Foresee all the alternatives in Java code

- The code is full of **if-else** cases
- Recompile when need to change layout or add a new language package.

ANDROID SOLUTION. Separate code from application resources

- Use declarative XML-based approach to define resources (images, files, layout, text, etc)

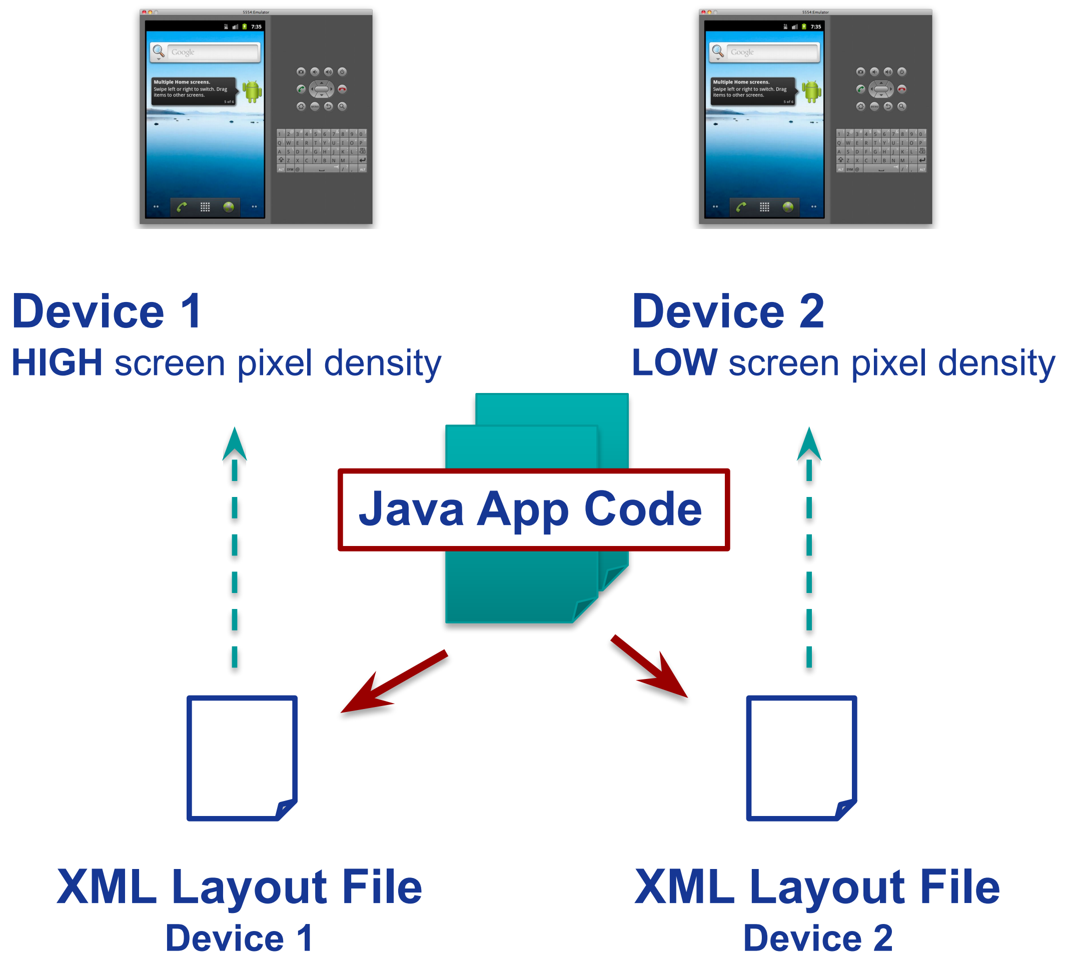
Application Resources Definition



- Use XML files to define (**declarative approach**):
  - *Application Layout*
  - *Text used in the applications*
  - *Application Menu*
  - *Animations*
  - …
- Foresee different **resources alternatives** for different device configurations (e.g. screen resolution, language, input devices. etc)

Application **Resources** Definition

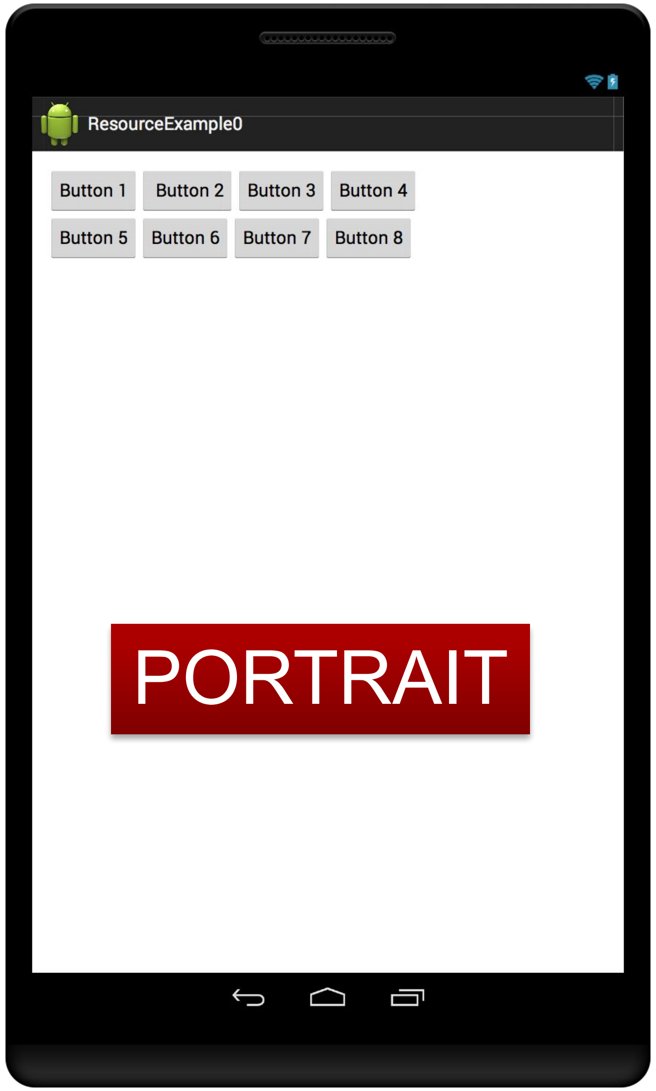
EXAMPLE

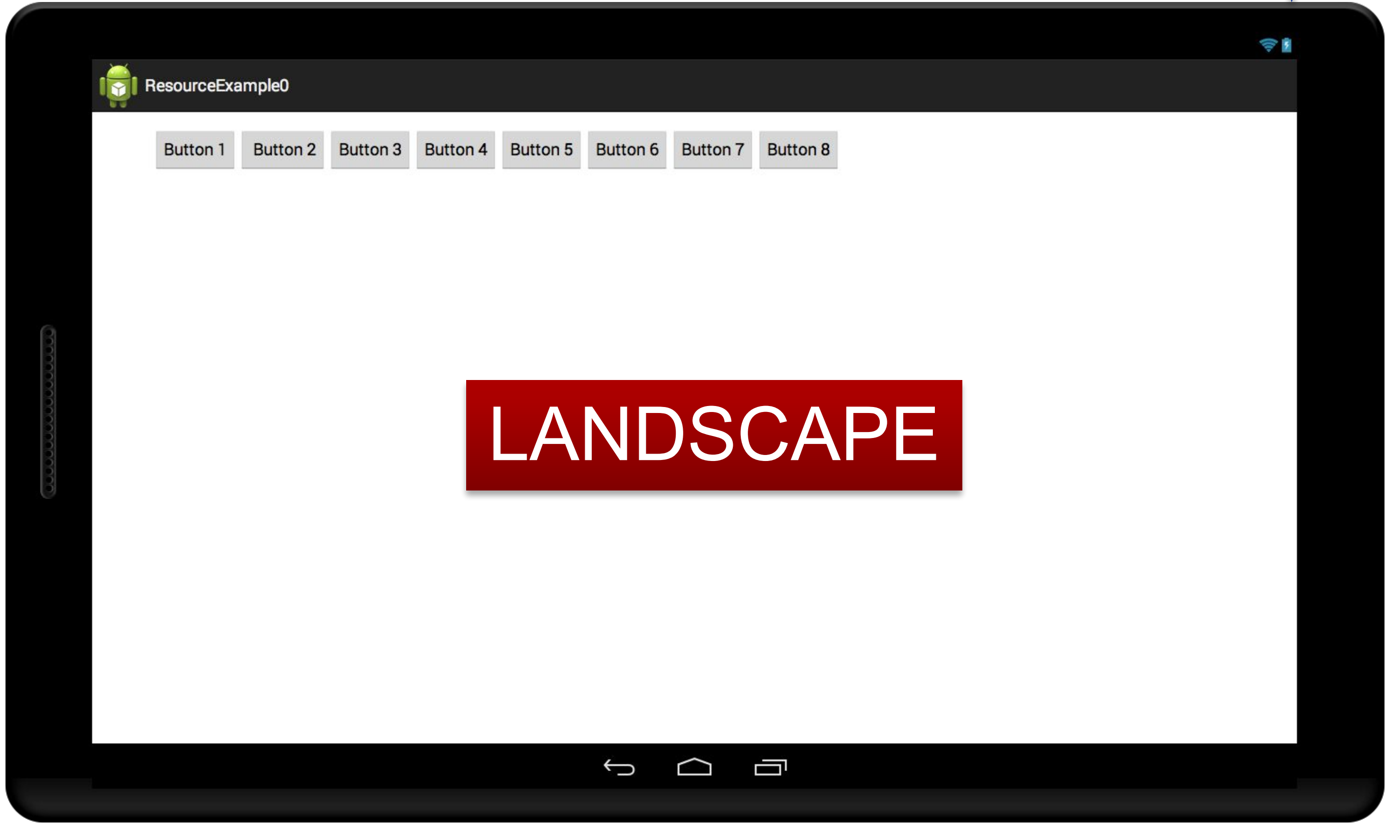


- Build the **application layout** through XML files (like HTML)
- Define **two** different XML **layouts** for two different devices
- At **runtime**, Android detects the current device configuration and loads the appropriate resources for the application
- **No need to recompile**!
- Just add a new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

Application Resources Definition

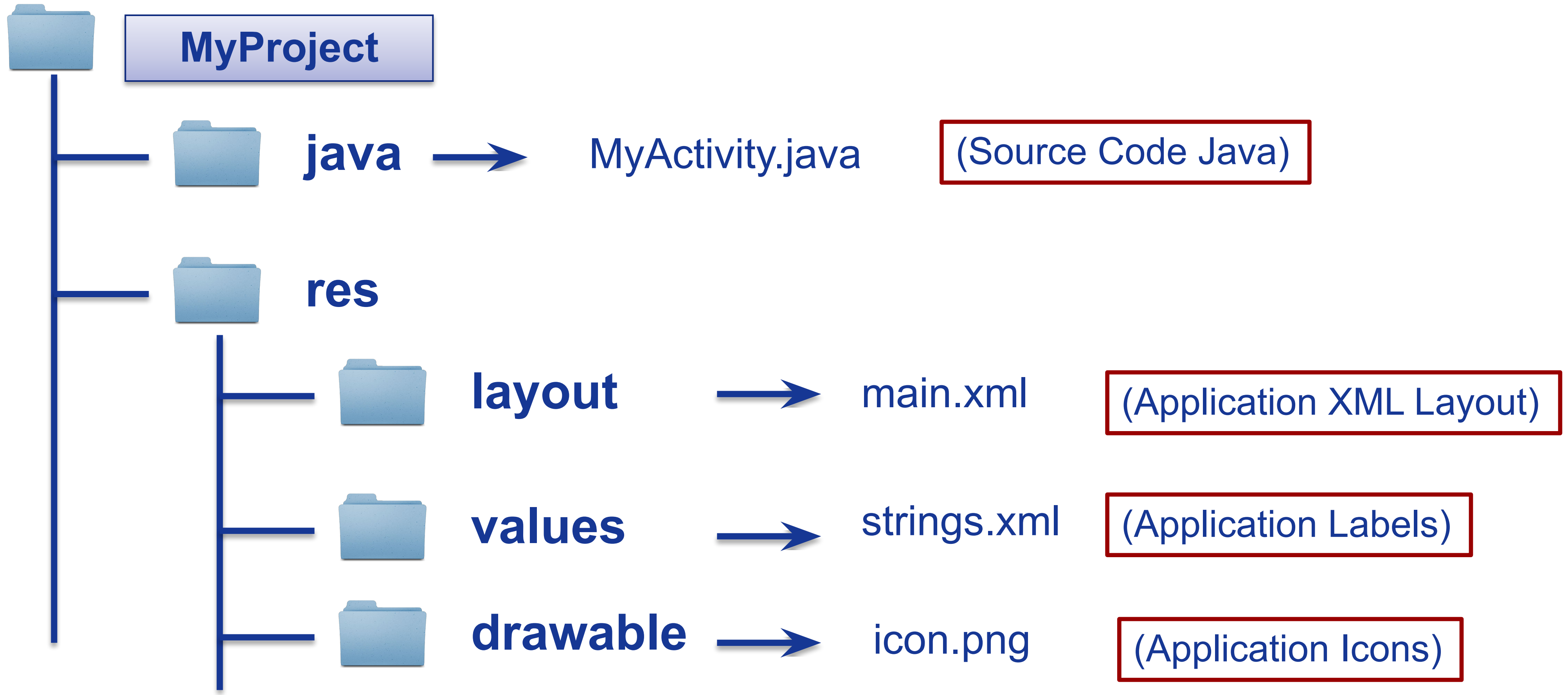
The same **application layout** with 8 buttons, on a **tablet** and on a **smartphone** (Nexus 7) device.





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/color**	*XML files that define a state list of colors.*
+ **res/drawable**	*Bitmap files (.png, .9.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/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

- Resources are defined in a **declarative** way through **XML**.
- Each resource has a name/identifier (see details later).

Example: **string.xml** contains all the text that the application uses. For example, the name of buttons, labels. default text, etc

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

   <string name="hello"> Hello world! </string>
   <string name="labelButton"> Insert your username </string>

</resources>
```

Resource type
(string)

Application Resources Definition

- Resource can be accessed in the **Java** code through the **R class**, that works as a **glue** between the world of java and the world of resources.
- **Automatically generated** file, no need to modify it.
- **Recreated** in case of changes in the **res**/ directory.

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

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

Application Resources Definition

- Resources can be accessed from Java code by using the **R** class and methods of the **Activity** class (details later).
- We just need to know the **resource Identifier** (ID) … how to know it? (see next slides)

```
…
final String hello=getResources().getString(R.string.hello);
final String label=getResources().getString(R.string.labelButton);
Log.i(STRING_TAG,“ String1 ” + hello);
Log.i(STRING_TAG,“ String2 ” + label);
…
…
```

Application Resources Definition

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**

- 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 **Java** Code
  - From the **XML** files

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)
- <**resource_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

STRING.XML

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <color name="opaque_red">#f00</color>
    <string name="labelButton"> Submit </string>
    <string name="labelText"> Hello world! </string>
</resources>
```

MAIN.XML

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

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)
- <**resource_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

```
// Get a string resource from the string.xml file
final String hello=getResources().getString(R.string.hello);

// Get a color resource from the string.xml file
final int color=getResources().getColor(R.color.opaque_red);

// Load a custom layout for the current screen
setContentView(R.layout.main_screen);

// Set the text on a TextView object using a resource ID
TextView msgTextView = findViewById(R.id.label1);
msgTextView.setText(R.string.labelText);
```

Access to Application Resources: Java

```
// Set the text on a TextView object using a resource ID
TextView msgTextView = findViewById(R.id.label1);
msgTextView.setText(R.string.labelText);
```

You will notice that each of your Views and ViewGroups has a number of attributes:

```
android:id="@+id/label_1"
```

@ means: “parse and expand the rest of the string as an id resource.

+ means: “this is going to be added as a **new** id in R.java

21

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> <item> <item> </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> <item> <item> </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"> Example Application </string>
    <string name="label" > Hello world!   </string>
    <integer name="val" > 53 </integer>
    <string-array name="nameArray">
        <item> John Bonham </item>
        <item> Frank Zappa </item>
    </string-array>
    <integer-array name="valArray">
        <item> 1 </item>
        <item> 2 </item>
    </integer-array>

</resources>
```

Resources Types: string and array

MYFILE.JAVA

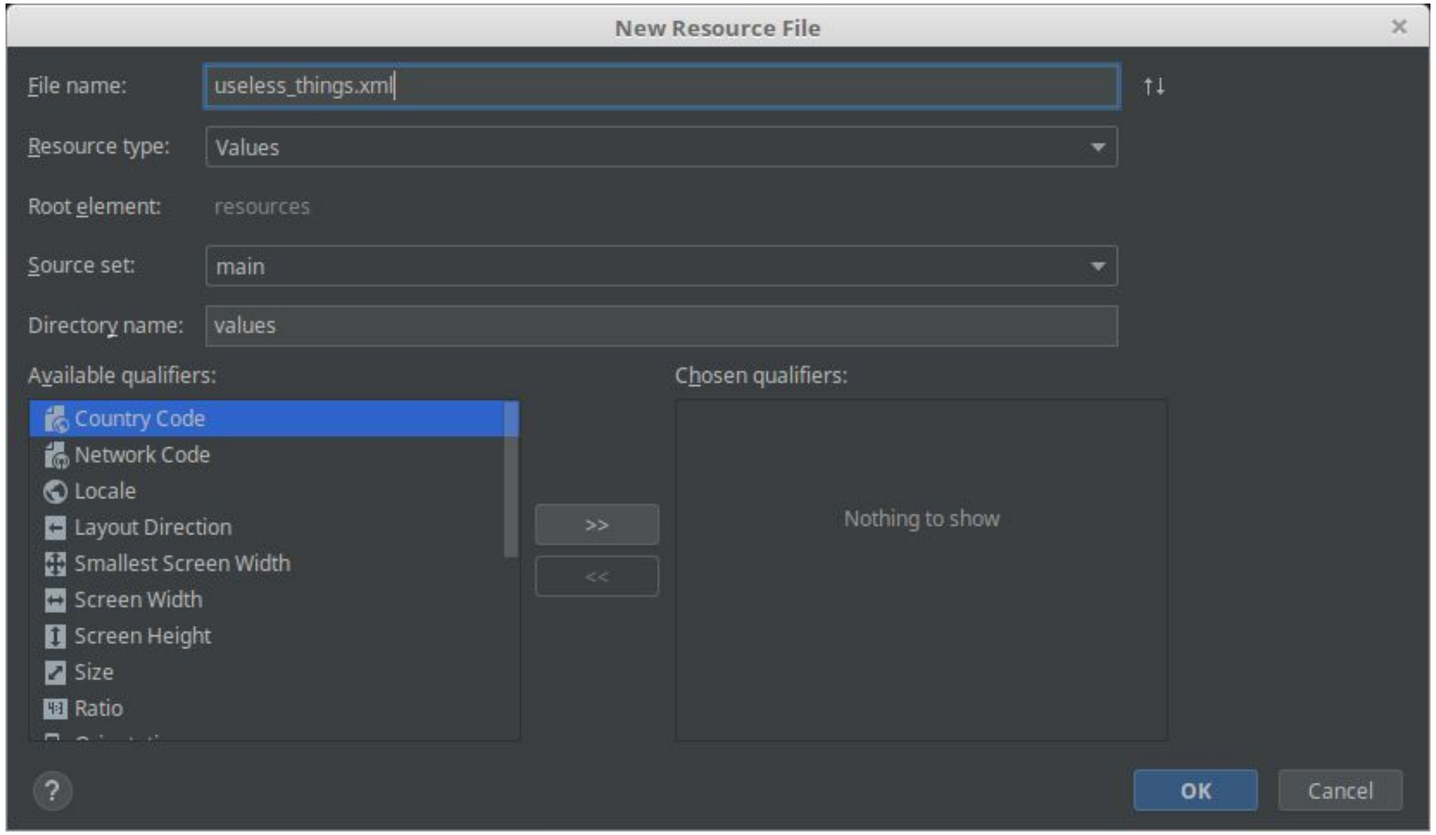
```
// Access the string value
final String hello=getResources().getString(R.string.app_title);

// Access the string-array values
final String[] nameS=getResources().getStringArray
(R.array.nameArray);

// Access the integer-array values
final int[] val=getResources().getIntArray(R.array.valArray);
```

Resources Types: string and array

- Resources can be defined in the **res/string.xml** or in any **other file** defined by the users (File →New →Values resource file)

Focus on **values** must be active.

For older versions hit File → New → Android XML File



New Resource File

File name: useless_things.xml

Resource type: Values

Root element: resources

Source set: main

Directory name: values

Other Resources Types

- Some other resources types (we will meet later … or never)

Resource Type	File	Java constant	XML tag	Description
layout	Any file in the res/layout/	R.layout.<key>	<layout>	Defines a layout of the screen
animation	Any file in the res/animator/	R.animator.<key>	<animator>	Defines a property animation (not the only method!)
menu	Any file in the res/menu/	R.menu.<key>	<menu>	User-defined menus with multiple options

Resources Types: color, dimension, style

- All “subtypes” of **values**

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/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

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

    <color name="red"> #FF0000 </color>
    <color name="red_trasparent" > #66DDCCDD</color>

</resources>
```

COLORS.XML

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

```
int redTransparent=getResources.getColor(R.color.red_transparent)
```

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. Likewise, when on a lower density screen, the number of pixels used for 1dp is scaled down

Resources Types: color, dimension, style

MYVALUES.XML

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

      <dimen name="textview_height">25dp</dimen>
      <dimen name="textview_width">150dp</dimen>
      <dimen name="font_size">16sp</dimen>
</resources>
```

- **Applying dimensions** to attributes in the XML layout:

MAIN.XML

```
<TextView
   android:layout_height="@dimen/textview_height"
   android:layout_width="@dimen/textview_width"
   android:textSize="@dimen/font_size"/>
```

Resources Types: color, dimension, style

- 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*”).

- A style is an XML resource that is referenced using the value provided in the **name** attribute.

- Styles can be organized in a **hierarchical** structure. A style can inherit properties from another style, through the **parent** attribute.

- Use **<style></style>** tags to define a style in the **res**/ folder. Use **<item>** to define the attributes of the style.

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>
```

- **Applying a style** to a View in the XML layout:

MAIN.XML

```
<EditText style="@style/CustomText"
   android:layout_width="match_parent"
   android:layout_height="wrap_content"
   android:text="Hello, World!" />
```

Resources Types: drawable

Resource Type	File	Java constant	XML tag	Description
drawable	Any file in the res/drawable/	R.drawable.<key>	<drawable>	Images and everything that can be drawn

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

- Images
- XML resources with attributes such as **android:drawable** and **android:icon** (e.g. a Button can have a drawable resource as background)

Complete list of drawable resource type can be found here:
http://developer.android.com/guide/topics/resources/drawable-resource.html

Resources Types: drawable

- A **BitMap** file is a **.png**, **.jpg** or a **.gif** file.
- Android creates a **BitMap** resource for any of these files saved in the **res/drawable** directory.

This layout XML applies the file myimage.png saved in res/drawable to a **View**.

```
<ImageView
   android:layout_width="match_parent"
   android:layout_height="wrap_content"
   android:src="@drawable/myimage" />
```

Retrieve the image as a Drawable from Java:

Drawable draw=res.**getDrawable**(R.drawable.myimage);

Resources Types: drawable

- An **XMLBitmap** is an XML resource that **points to a bitmap file**.
- Usage: (*i*) **Alias** to the raw bitmap file, (*ii*) Specify additional properties such as **dithering** and **tiling**.

```
<?xml version="1.0" encoding="utf-8"?>
<bitmap xmlns:andoid=http://schemas.android.com/apk/res/android”
andoid:src=“@drawable/tile”
andoid:tileMode=“repeat”>
```

Some properties of an XMLBitmap:
android:src, android:antialias, android:dither, android:filter, android:gravity

Resources Types: drawable

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 drawables
State List	A Drawable that references different graphics based on the states
Level List	An XML managing alternate Drawables. Each assigned to 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

Complete list of drawable resource type can be found here:
http://developer.android.com/guide/topics/resources/drawable-resource.html

Resources Types: xml and raw

Resource Type	File	Java constant	XML tag	Description
xml	Any file in the res/xml/	R.xml.<key>	<xml>	User-specific XML file with name equal to key
raw	Any file in the res/raw/	R.raw.<key>	<raw>	Raw resources, accessible through the R class but not optimized

Used to define resources for which no run-time optimization must be performed (e.g. audio/video files). They can be accessed an a stream of bytes, by using Java **InputStream** objects:

InputStream is= getResources().**openRawResource**(R.raw.videoFile)

Resources Types: xml and raw

- The res/xml folder might contain arbitrary XML files that can be read at runtime through the R.xml.<filename> constant.

- It is possible to parse the XML file through a **XMLResourceParser** object, that implements an XML parser:

XMLResourceParser parser=getResources().**getXML**(R.xml.myfile)

```
<?xml version="1.0" encoding="utf-8"?>
<names>
      <name code=”1234”>Marco Di Felice </item>
       <name code=4324">Luca Bedogni </item>
 </names>
```

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:

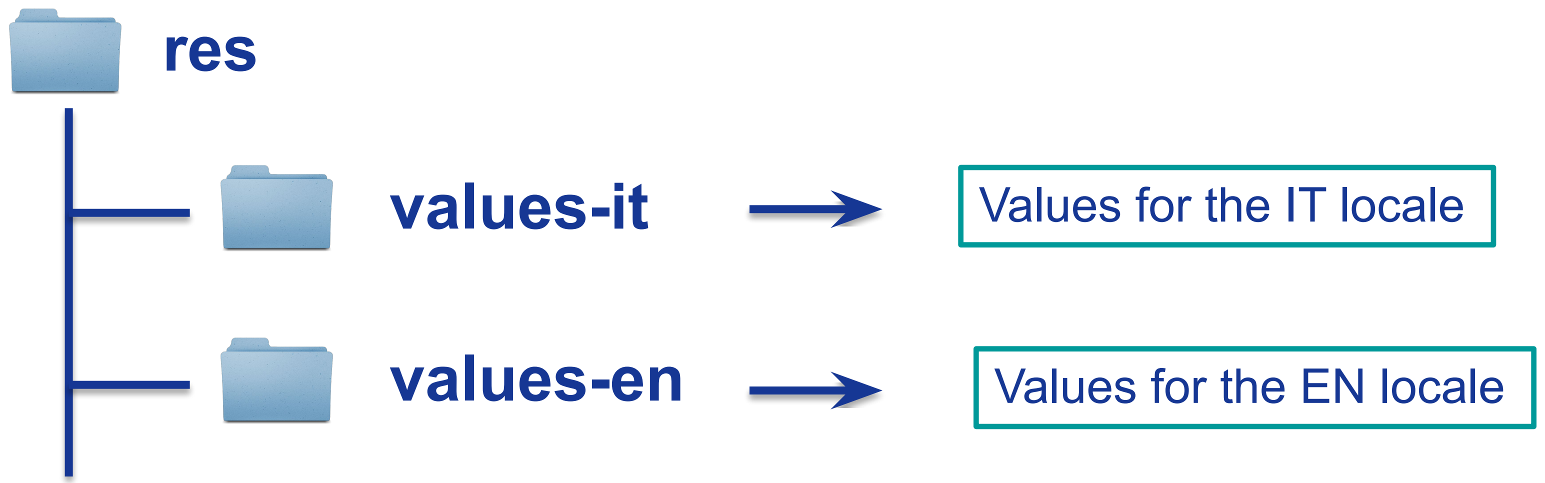
  1. Create a new directory in **res**/ named in the form <***resources_name***>-<***config_qualifier***>

  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).

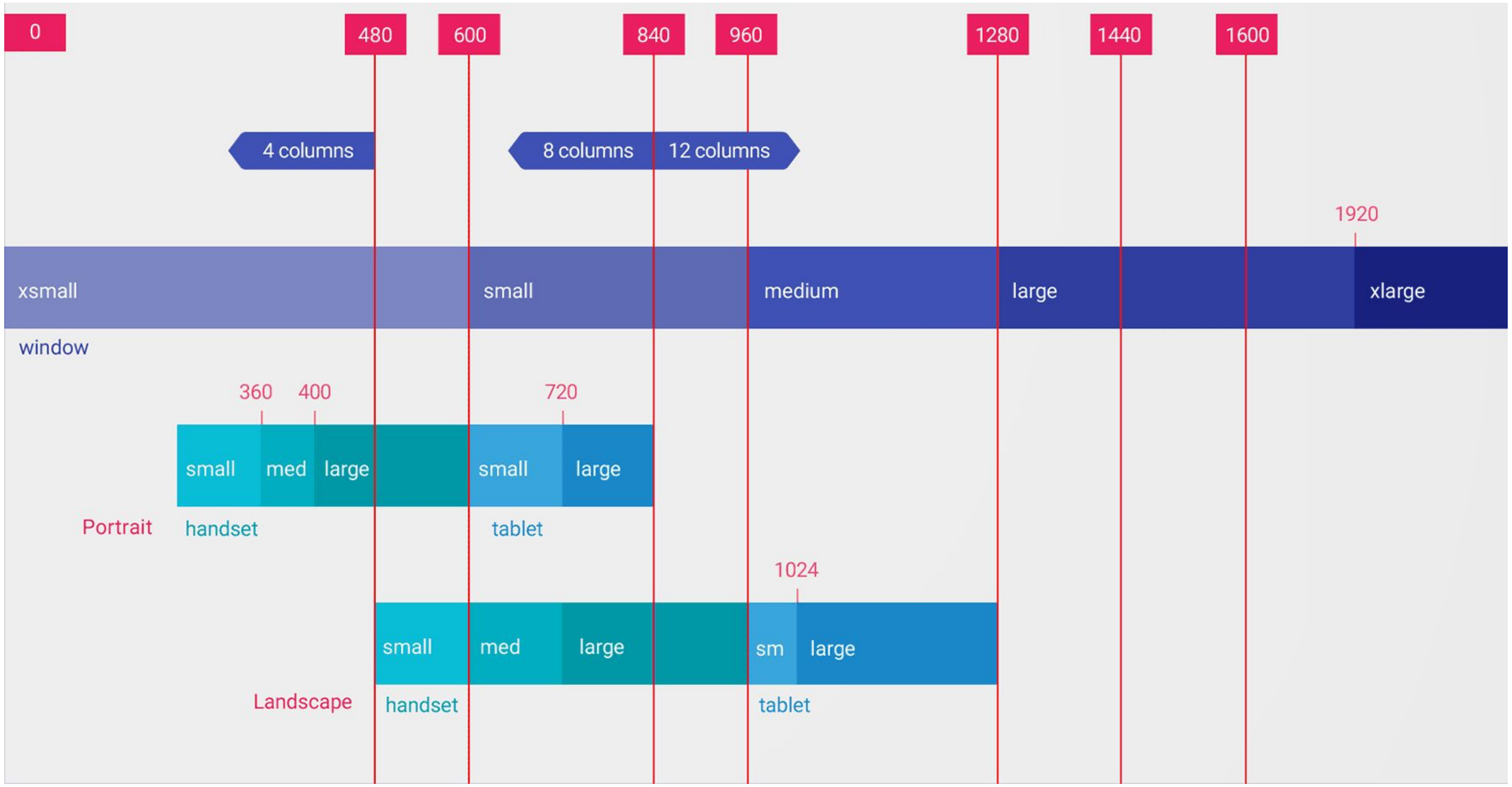


Resources Alternatives: Qualifiers

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 module width
Available height	h720dp, etc	minimum available module height
Screen size	small, normal, large, ...	screen size
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

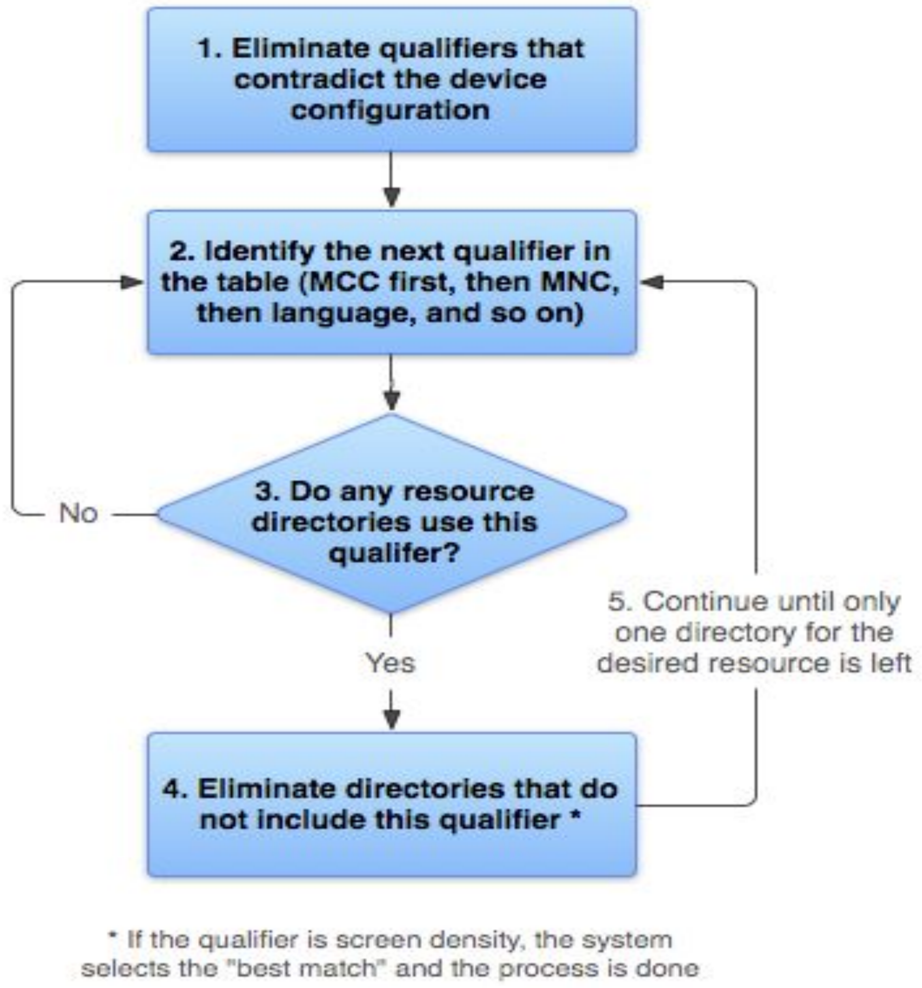
Resources Alternatives: smallestWidth

- All values in dp (density-independent pixels) != dpi



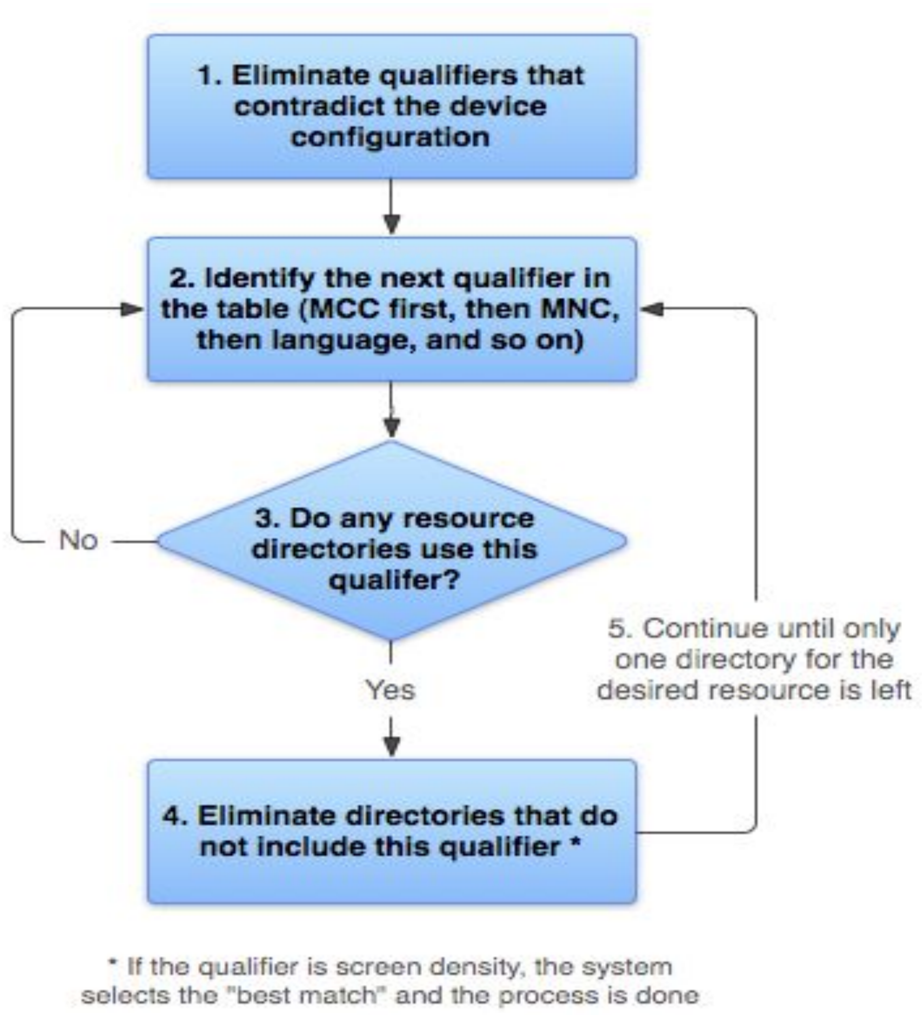
- This is **one way** of dealing with different screen sizes.
- NOT the only one (see layouts)

Resources Alternatives Matching


1. Eliminate qualifiers that contradict the device configuration
2. Identify the next qualifier in the table (MCC first, then MNC, then language, and so on)
3. Do any resource directories use this qualifer?
No
Yes
4. Eliminate directories that do not include this qualifier *
5. Continue until only one directory for the desired resource is left
* If the qualifier is screen density, the system selects the "best match" and the process is done

- 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



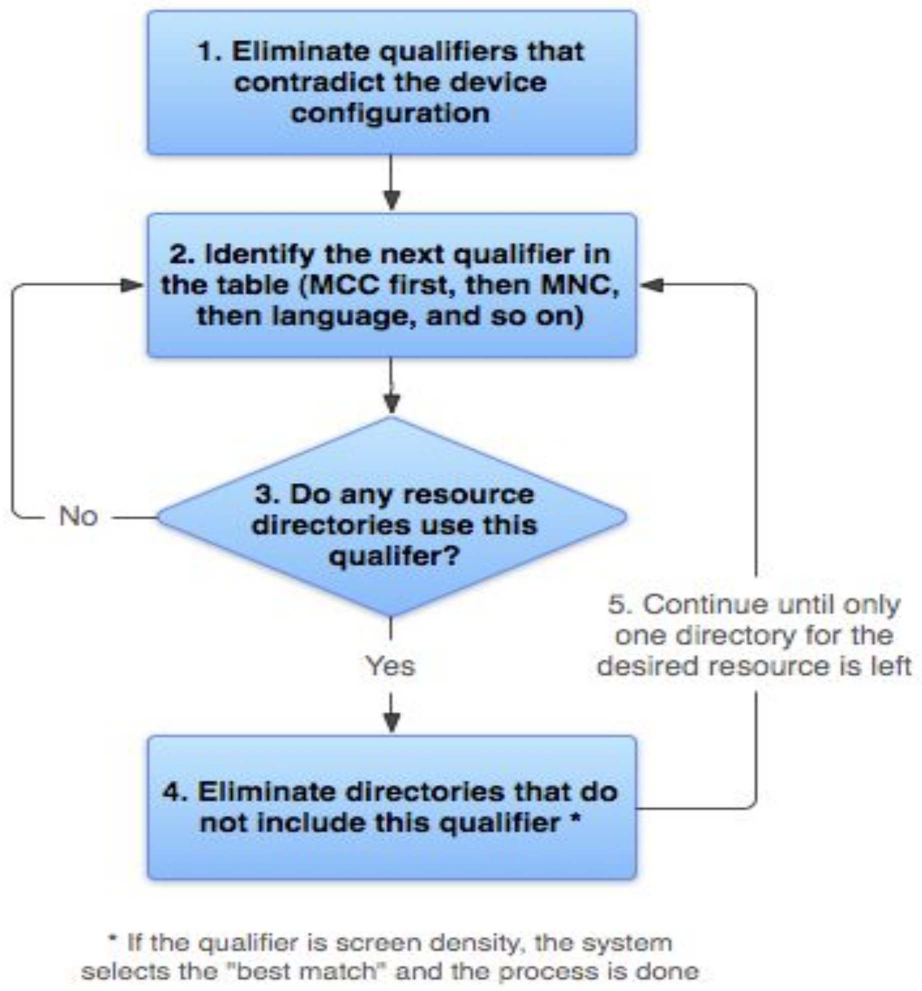
1. Eliminate qualifiers that contradict the device configuration
2. Identify the next qualifier in the table (MCC first, then MNC, then language, and so on)
3. Do any resource directories use this qualifer?
No
Yes
4. Eliminate directories that do not include this qualifier *
5. Continue until only one directory for the desired resource is left
* If the qualifier is screen density, the system selects the "best match" and the process is done

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-land-notouch-12key/

Resources Alternatives Matching



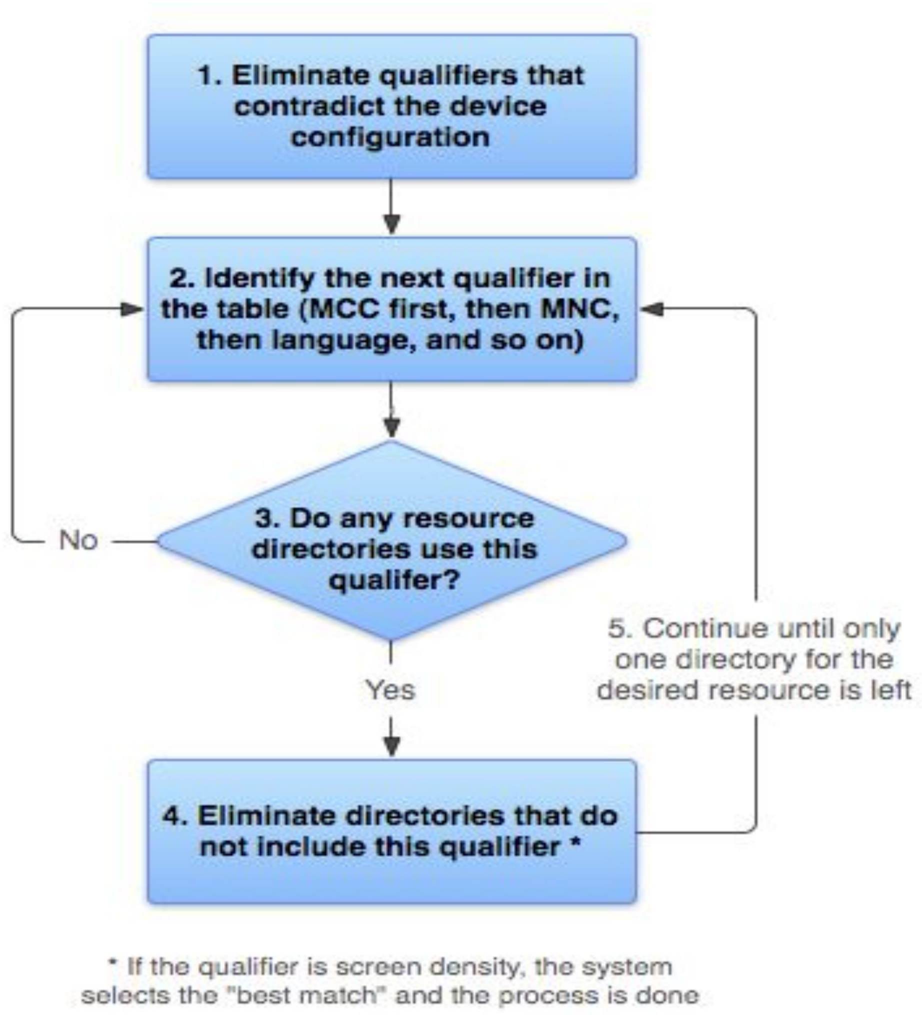
1. Eliminate qualifiers that contradict the device configuration
2. Identify the next qualifier in the table (MCC first, then MNC, then language, and so on)
3. Do any resource directories use this qualifer?
No
Yes
4. Eliminate directories that do not include this qualifier *
5. Continue until only one directory for the desired resource is left
* If the qualifier is screen density, the system selects the "best match" and the process is done

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-land-notouch-12key/~~

Resources Alternatives Matching



1. Eliminate qualifiers that contradict the device configuration
2. Identify the next qualifier in the table (MCC first, then MNC, then language, and so on)
3. Do any resource directories use this qualifer?
No
Yes
4. Eliminate directories that do not include this qualifier *
5. Continue until only one directory for the desired resource is left
* If the qualifier is screen density, the system selects the "best match" and the process is done

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-land-notouch-12key/~~

Resources Alternatives

BEST PRACTICE

- 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.

Resources Alternatives: example

- How to change the splash screen depending on the language and location

Location / Language	US	France	Canada	Italy	Germany	Rest of the world
English	Hello		Hello			Hello
French		Bonjour	Bonjour			Bonjour
Italian				Ciao		Ciao
German					Hallo	Hallo
Rest of the languages						Hello

Resources Alternatives: example

- How to change the splash screen depending on the language and location

res/

values	Hello
values-it	Ciao
values-fr	Bonjour
values-de	Hallo
drawable	
drawable-en-rUS	🇺🇸
drawable-en-rCA	🇨🇦
drawable-it-rIT	🇮🇹
drawable-de-rDE	🇩🇪
drawable-fr-rFR	🇫🇷
drawable-fr-rCA	🇨🇦