Resources: introduction

Application **Resources** Definition

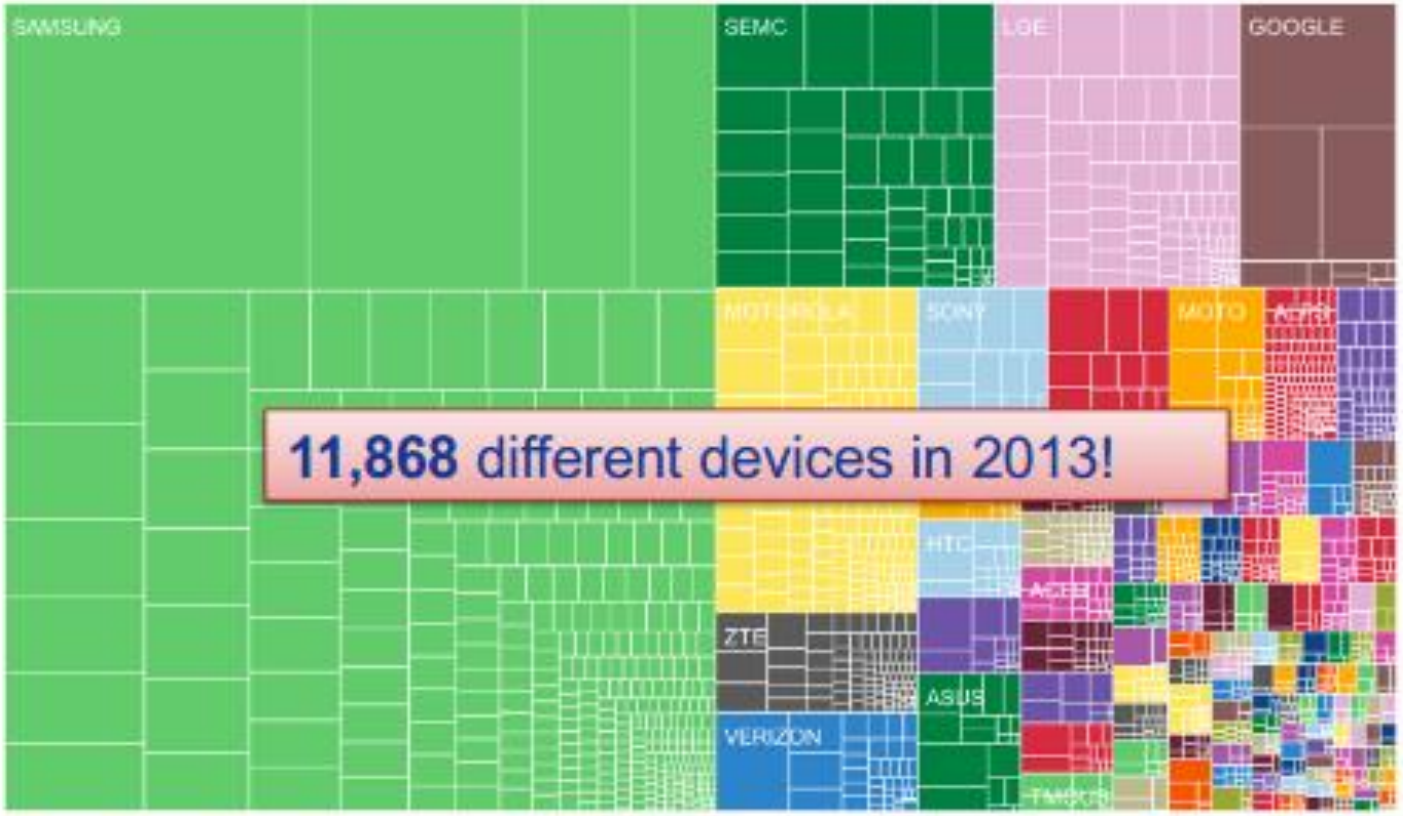
- An Application is composed of: **code** and **resources**.

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

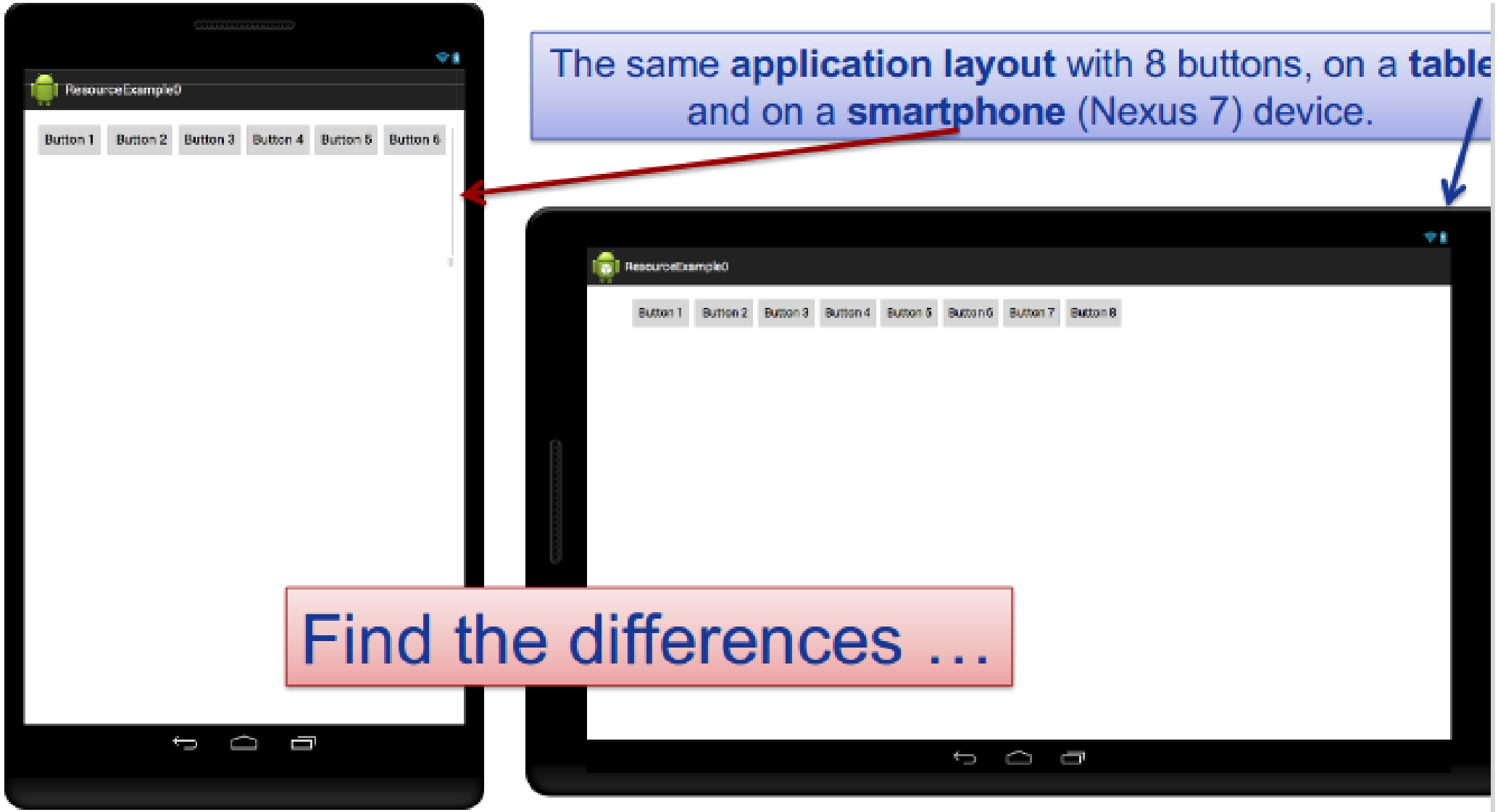
- Utilization of Resources...why?
  - **Separate** data presentation (layout) from data management
  - **Provide** alternative resources to support specific device configurations (e.g. different language packs)
  - **Re-compile** only when strictly needed!

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



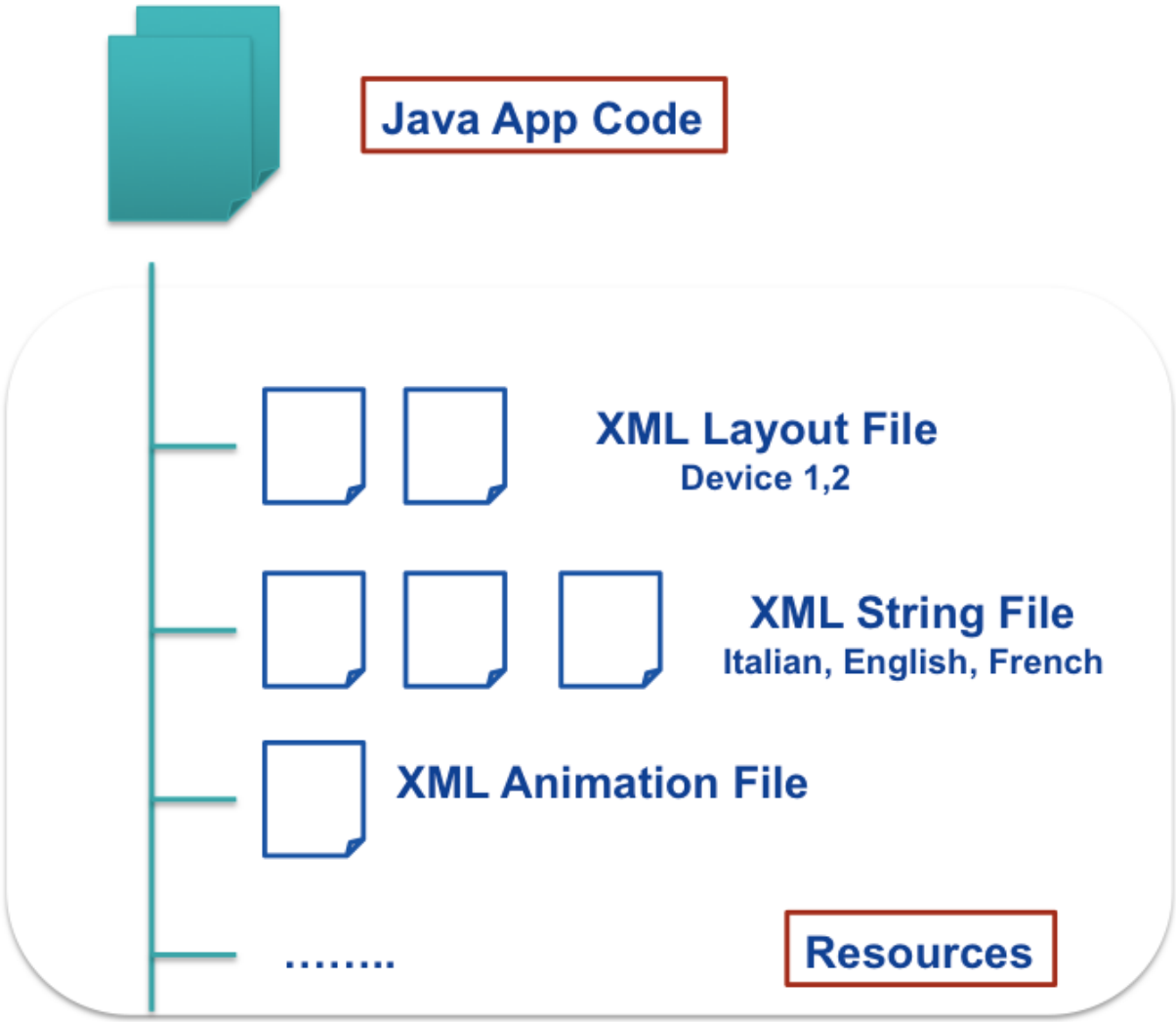
The same **application layout** with 8 buttons, on a **table** and on a **smartphone** (Nexus 7) device.

Find the differences …

Resources

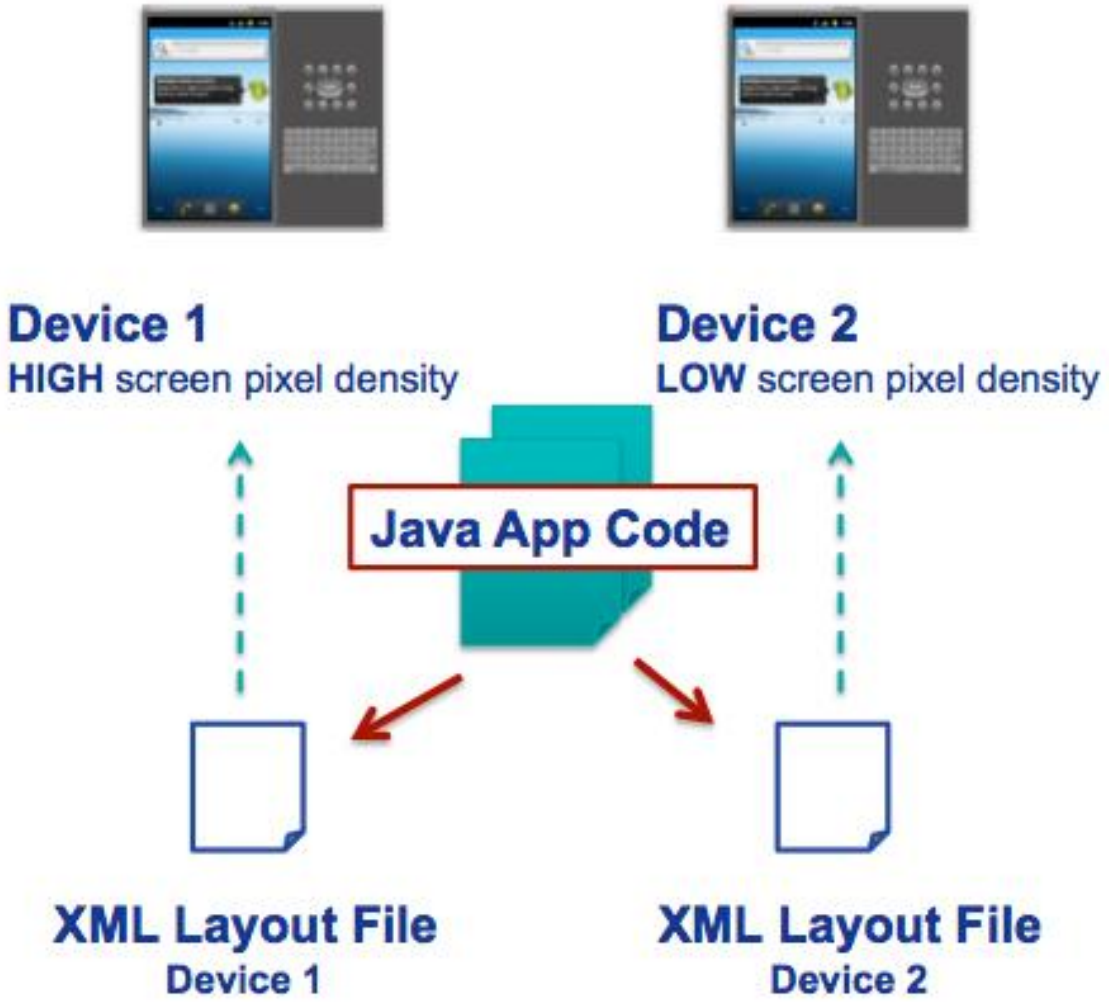
- An Android application might run on heterogenous devices with different characteristics (screen size, language support, keyboard type, input devices,….).
- **Traditional solution**:
  - Foresee all the alternatives in Java code (if-else cases)
  - Recompile when need to change layout, add a new language package or extend support.
- **Android solution**:
  - Separate code from application resources (declarative XML-based approach to define resources)
  - Automatic resource selection mechanism.

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

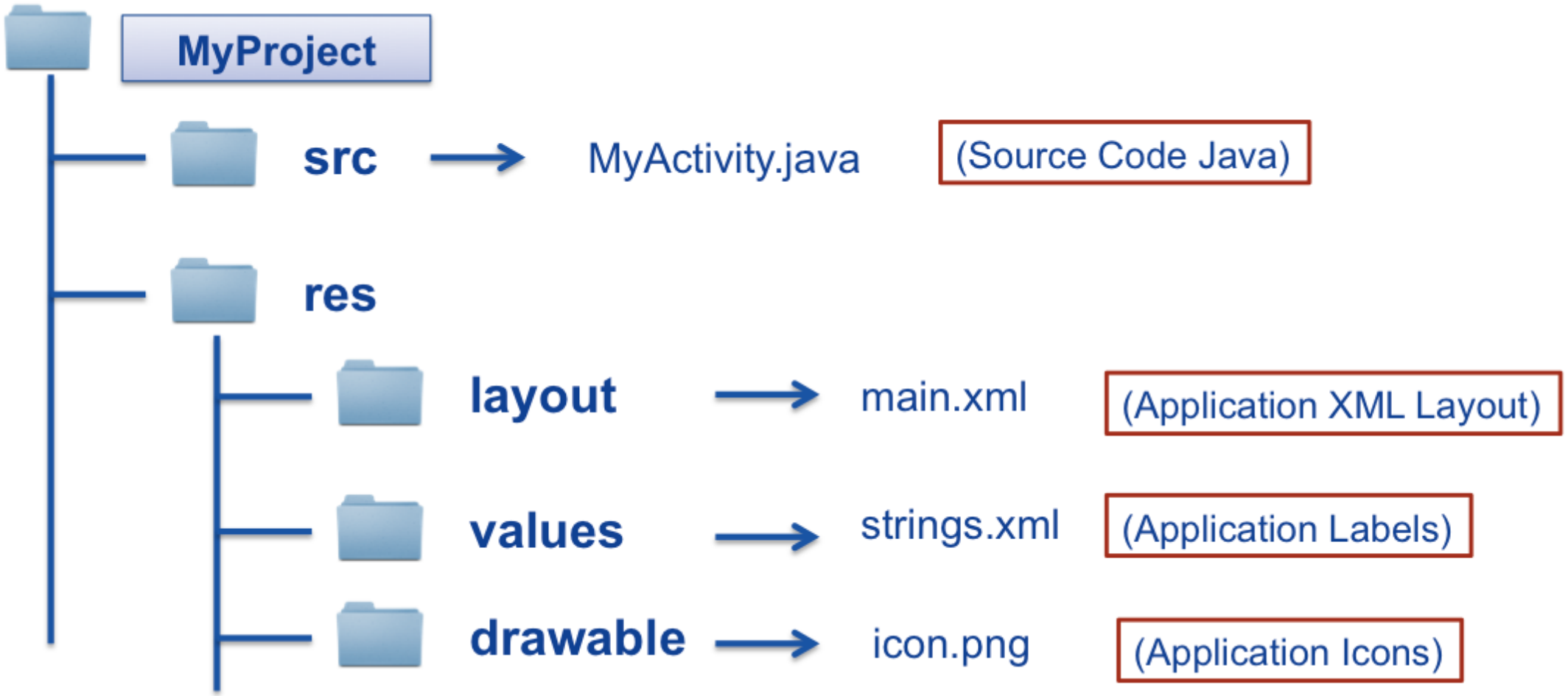


- Build the ***application layout*** through XML files (like HTML)
- Define ***two*** different XML ***layouts*** for two different devices
- At ***runtime***, Android detects the current device configuration and loads the appropriate resources for the application
- No need to recompile
- Just add a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

Application resources definition

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

res folder


MyProject
src
MyActivity.java
(Source Code Java)
res
layout
main.xml
(Application XML Layout)
values
strings.xml
(Application Labels)
drawable
icon.png
(Application Icons)

Java/res folders

KotlinApplication – MainActivity.kt [KotlinApplication.app]

File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help

MainActivity | 4 WVGA (Nexus S) API 24

KotlinApplication 〉 app 〉 src 〉 main 〉 java 〉 it 〉 uninsubria 〉 pdm 〉 kotlinapplication 〉 MainActivity

Android

- manifests
- java
  - it.uninsubria.pdm.kotlinapplication
    - MainActivity
  - it.uninsubria.pdm.kotlinapplication (androidTest)
  - it.uninsubria.pdm.kotlinapplication (test)
- java (generated)
- res
  - drawable
  - layout
    - activity_main.xml
  - mipmap

activity_main.xml | MainActivity.kt | AndroidManifest.xml

```
2
3   import android.os.Bundle
4   import android.util.Log
5   import androidx.appcompat.app.AppCompatActivity
6
7
8   class MainActivity : AppCompatActivity() {
9       val TAG = "MainActivity"
10      override fun onCreate(savedInstanceState: Bundle?) {
11          super.onCreate(savedInstanceState)
12          setContentView(R.layout.activity_main)
13          Log.v(TAG, msg: "onCreate")
14      }
15
```

TODO | Terminal | Database Inspector | 4: Run | Profiler | Build | 6: Logcat | Event Log | Layout Inspector

Success: Operation succeeded (24 minutes ago) 7:1 LF UTF-8 4 spaces

Available Resource Types

- different types of resources that can be externalized from code
- **String Resources**
  - Define **strings**, **string arrays**, ...
  - Saved in res/values/ and accessed from the **R**.*string*, **R**.*array*, …
- **Layout Resource**
  - Define the layout for your application UI.
  - Saved in res/layout/ and accessed from the **R**.*layout* class.
- **Color State List Resource**
  - Define a color resources that changes based on the View state
  - Saved in res/color/ and accessed from the **R**.*color* class.
- Etc..
  for a complete list: https://developer.android.com/

Resource Types

Resource Type	Resource contained
res/animator	*XML files that define property animations.*
res/anim	*XML files that define tween animations.*
res/colors	*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.*

- Different types of application resource that you can provide in your resources directory (res/).

XML - structure

An XML **element** is everything

Attributes are designed to contain data related to a specific element.

element

child element

start tag

end tag

attribute

value

text

```
<ORDER>
    <SOLD-TO>
        <PERSON ID="123">
            <LASTNAME>Malan</LASTNAME>
            <FIRSTNAME>David</FIRSTNAME>
        </PERSON>
    </SOLD-TO>
    <SOLD-ON>20050621</SOLD-ON>
    <ITEM>
        <PRICE>29.95</PRICE>
        <BOOK>
            <TITLE>Harry Potter and the
            Order of the Phoenix</TITLE>
            <AUTHOR>J.K. Rowling</AUTHOR>
        </BOOK>
    </ITEM>
</ORDER>
```

Attribute **values** must always be quoted.

Tags mark the start and end of an element.

Resource definition

- Each resource has a **name=”identifier”**.

string.xml contains all the text that the application uses.
E.g: the name of buttons, labels. default text, …

attribute

name=identifier

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

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

</resources>
```

element=resource type

value

Resources in res/values/

- res/values/ contains XML files that contain simple values, such as strings, integers, and colors.
- Files in the values/ directory describe multiple resources.
- For a file in this directory, each child of the <resources> element defines a single resource.

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string name="tvBottomText">Bottom text</string>
    <color name="llBottomColor">#339966</color>
</resources>
```

- For example,
  - a <string> element creates an ***R**.string* resource
  - a <color> element creates an ***R**.color* resource.

Resources in res/values/

- Because each resource in res/values/ is defined with its own XML element, you can **name the file whatever you want** and place **different resource types** in one file.
- For example,

pippo.xml

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string name="tvBottomText">Bottom text</string>
    <color name="llBottomColor">#339966</color>
</resources>
```

- For clarity, place unique resource types in different files.
  - arrays.xml, colors.xml, dimens.xml, strings.xml, styles.xml

Mapping between android resources and resources ID

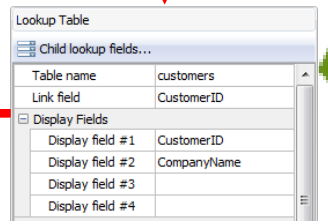
layout1.xml

```
...
<Button android:id="@+id/button1" >
...
```

aapt tool - **A**ndroid **A**sset **P**ackaging **T**ool: Is used to generate unique IDs for each of our resources and store them in a lookup table.

R.java

```
...
public static final int button1=0x7f05000b;
...
```



resources.arsc

When we call:
findViewById(***R***.*id.button1*);
we are essentially still do the search based on the ID, searching a number like 0x7f05000b.

Resources: access from code

- Android generates a class `R.txt` which provide access to resources via an `int` ***constant***.
  - `R.txt` is an automatically generated file (Do not modify it)

Var name match resource identifier

int id updateTextButton 0x7f08016b

The value of these variables is an integer that represents each resource's location in the ***resource table***.

Using resources in code

- Some constructors/methods accepts a resource identifier as arguments.

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

```
//Inflate a layout resource.
setContentView(R.layout.main)

// Display a transient dialog box that displays the
// error message string resource.
Toast.makeText(this, R.string.app_error, Toast.LENGTH_LONG).show()
```

Accessing resource objects

- You can get an ***instance*** of the resource itself using helper methods to extract them from the resource table.
- The ***resource table*** is represented within applications as an instance of the `Resources`:
  - obtained by the getResources() method
  - it provides getters for all available resource types.

```
val myResources = getResources()  // get the resource object
// lookup on the resource table
val styledText = myResources.getText(R.string.stop_message)
```

Resources: string, integer and arrays

Values definition

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
<string name="app_title">Example</string>
<string name="label" >Hello world!</string>

<integer name="age" >35</integer>

<string-array name="nameArray">
   <item>Mauro</item>
   <item>Paolo</item>
</string-array>

<integer-array name="intArray">
   <item>1</item>
   <item>2</item>
</integer-array>
</resources>
```

Resources: string, integer and arrays

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
<string name="app_title">Example</string>
<string name="label" >Hello world!</string>

<integer name="age" >35</integer>

</string-array>

<integer-array name="intArray">
   <item>1</item>
   <item>2</item>
</integer-array>
</resources>
```

Values definition

```
// Access the string value
val title = getResources().getString(R.string.app_title)
```

Resource in A01_HelloWorld

Add new Strings to Resources

Auto Generate @String reference in Android Studio

- Auto create a @string reference in strings.xml.
- Alt+Enter, Extract String Resource while the caret is inside the hardcoded string:

```
9   @Override
10  protected void onCreate(Bundle savedInstanceState) {
11      super.onCreate(savedInstanceState);
12      setContentView(R.layout.activity_main);
13      setTitle("hello");
14  }
15
16
```

Extract string resource
Inject Language

- The same works in XML files:

```
122 <TextView
123     android:id="@+id/s_hour"
124     android:layout_width="wrap_content"
125     android:layout_height="wrap_content"
126     android:text="00"
127
```

Extract string resource

Add “Hello World!” to strings.xml from Java code

strings.xml

```
<resources>
    <string name="app_name">A01_HelloWorld</string>
    <string name="action_settings">Settings</string>
    <string name="hello_world_str">Hello World!</string>
    <string name="to_be_implemented">Replace with your own action</string>
</resources>
```

```
FloatingActionButton fab = (FloatingActionButton) findViewById(R.id.fab);
fab.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        Snackbar.make(view, R.string.to_be_implemented, Snackbar.LENGTH_LONG)
                .setAction("Action", null).show();
    }
});
```

MainActivity

```
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        Toolbar toolbar = (Toolbar) findViewById(R.id.toolbar);
        setSupportActionBar(toolbar);

        FloatingActionButton fab = (FloatingActionButton) findViewById(R.id.fab);
        fab.setOnClickListener((view) -> {
            Snackbar.make(view, "Replace with your own action", Snackbar.LENGTH_LONG)
                    , null).show();
```

- Extract string resource
- Insert App Indexing API Code
- Make method call chain into call sequence
- Inject language or reference

```
    menu) {
        // Inflate the menu; this adds items to the action bar if it is present.
        getMenuInflater().inflate(R.menu.menu_main, menu);
        return true;
    }
```

- values
  - colors.xml
  - dimens.xml (2)
  - strings.xml (2)
    - strings.xml
    - strings.xml (it)
  - styles.xml (2)

Extract Resource

Resource name: to_be_implemented

Resource value: place with your own action

Source set: main

File name: strings.xml

Create the resource in directories:

- [x] values
- [ ] values-it
- [ ] values-v21
- [ ] values-w820dp

OK Cancel

Add “Hello World!” to strings.xml from XML layout

strings.xml

```
<resources>
    <string name="app_name">A01_HelloWorld</string>
    <string name="action_settings">Settings</string>
    <string name="hello_world_str">Hello World!</string>
</resources>
```

content_main.xml

```
    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/hello_world_str" />
</RelativeLayout>
```

- values
  - colors.xml
  - dimens.xml (2)
  - strings.xml (2)
    - strings.xml
    - strings.xml (it)
  - styles.xml (2)

- app
  - manifests
  - java
    - it.insubria.pdm.a01_helloworld
      - MainActivity
    - it.insubria.pdm.a01_helloworld (androidTest)
    - it.insubria.pdm.a01_helloworld (test)
  - res
    - layout
      - activity_main.xml
      - content_main.xml
    - menu
    - mipmap
    - values
      - colors.xml
      - dimens.xml (2)
      - strings.xml
      - styles.xml (2)
- Gradle Scripts

RelativeLayout | TextView

```
1  <?xml version="1.0" encoding="utf-8"?>
2  <RelativeLayout xmlns:android="http://schemas.android.com/apk/res/a
3      xmlns:app="http://schemas.android.com/apk/res-auto"
4      xmlns:tools="http://schemas.android.com/tools"
5      android:id="@+id/content_main"
6      android:layout_width="match_parent"
7      android:layout_height="match_parent"
8      android:paddingBottom="16dp"
9      android:paddingLeft="16dp"
10     android:paddingRight="16dp"
11     android:paddingTop="16dp"
12     app:layout_behavior="android.support.design.widget.AppBarLayout
13     tools:context="it.insubria.pdm.a01_helloworld.MainActivity"
14     tools:showIn="@layout/activity_main">
15
16     <TextView
17         android:layout_width="wrap_content"
18         android:layout_height="wrap_content"
19         android:text="Hello World!" />
20
21
```

- Extract string resource
- Override Resource in Other Configuration...
- Inject language or reference

Extract Resource

Resource name: hello_world_str

Resource value: Hello World!

Source set: main

File name: strings.xml

Create the resource in directories:

- [x] values
- [x] values-it
- [ ] values-v21
- [ ] values-w820dp

OK | Cancel