# Resources

#### Resources

- Resources: strings, dimensions, layout files, menus, images, audio files, etc that your app uses
- Basically app elements defined in other files
  - Easier to update and maintain code

```
<resources>
   <string name="app name">Example</string>
   <string name="my screen title">This is a very nice app</string>
   <string name="my_instructions">To use this application you have
                           to blah blah </string>
</resources>
```









people outline





















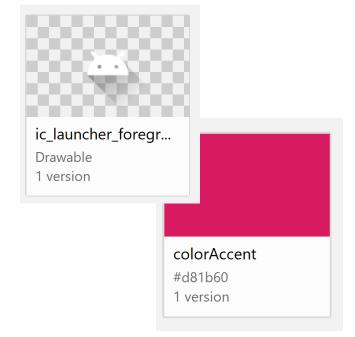




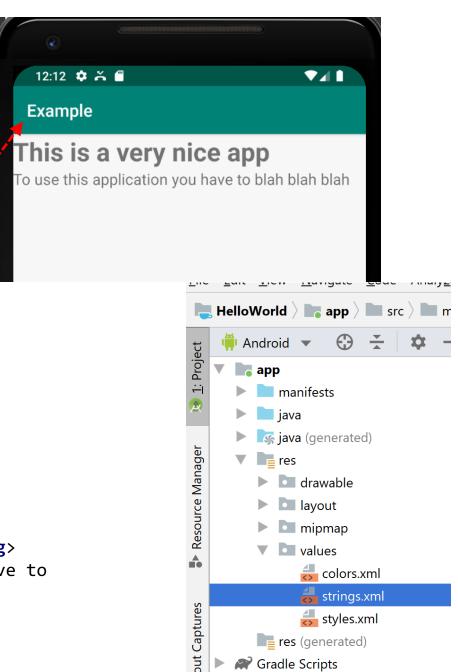








- String resources
  - Decouple UI content from app logic
  - Supporting different languages (EN, IT, FR, ...) is straightforward
- Strings are declared in strings.xml

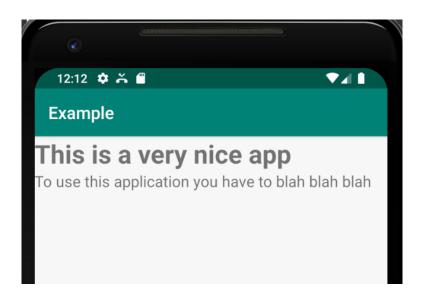


Then can be used in any other XML file

#### **Layout file**

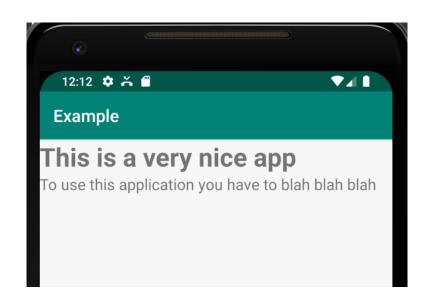
```
<TextView
    android:id="@+id/textView1"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:text="@string/my_screen_title"
    android:textSize="30sp"
    android:textStyle="bold" />

<TextView
    android:id="@+id/textView2"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:text="@string/my_instructions"
    android:textSize="18sp" />
. . .
```



#### strings.xml

Then can be used in any other XML file



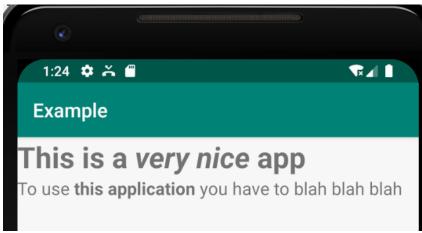
#### AndroidManifest file

#### <application</pre> android:allowBackup="true" android:icon="@mipmap/ic\_launcher" android:label="@string/app name" android:roundIcon="@mipmap/ic\_launcher\_round" android:supportsRtl="true" android:theme="@style/AppTheme">

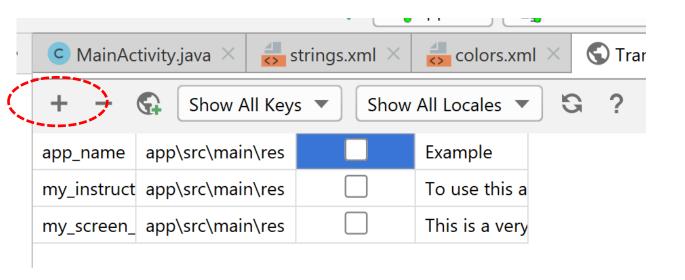
#### strings.xml

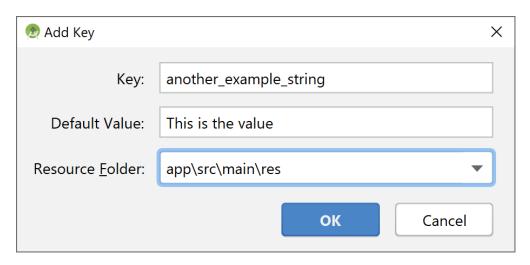
```
<resources>
   <string name="app name">Example</string>
    <string name="my screen title">
      This is a very nice app</string>
    <string name="my instructions">
      To use this application you have
      to blah blah </string>
</resources>
```

- Some basic HTML formatting can be included
  - <*b>*</*b>* for bold
  - <i></i> for italics



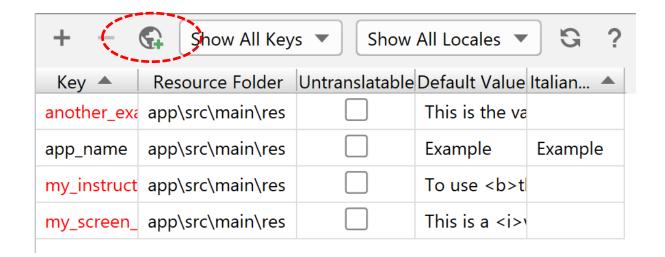
- New strings can be added
  - editing the *strings.xml* file
  - using the tools/wizards provided by AndroidStudio

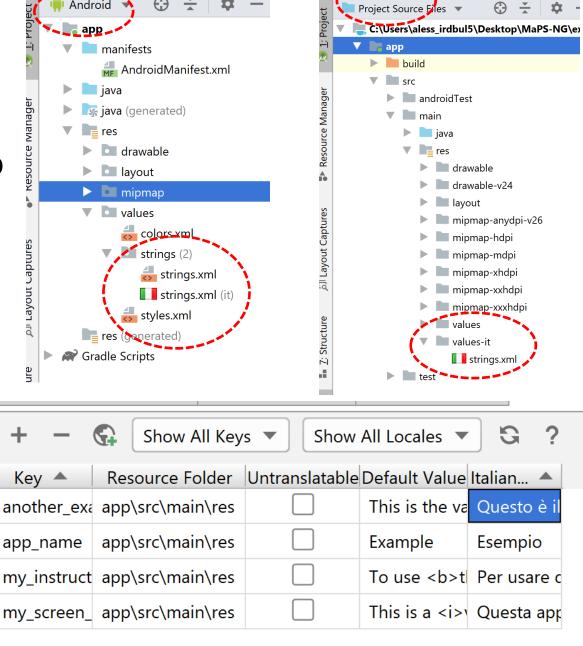




#### **Translations**

- To translate your app, you just have to provide versions of the *strings.xml* file
- AndroidStudio supports translation
- Extensions for different languages





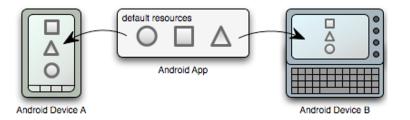
HelloWorld > app > src > mair

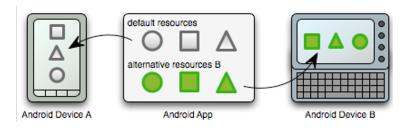
Edit View Navigate Code Analyze Refactor

HelioWorld app

#### **Alternative resources**

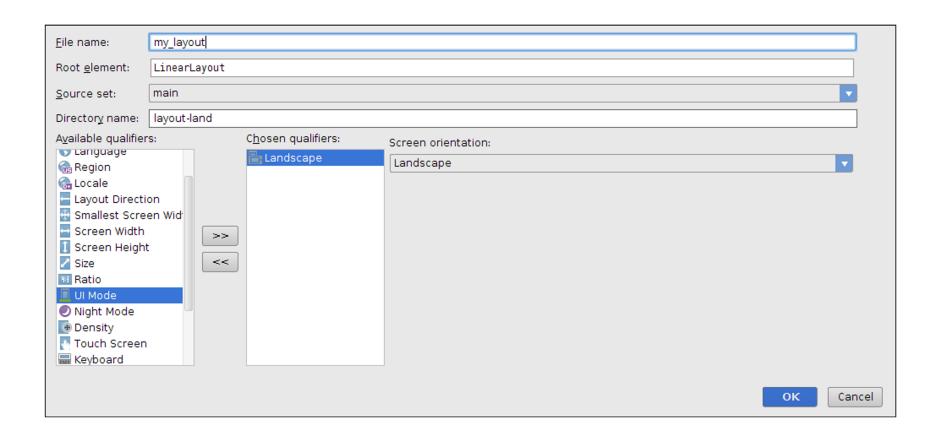
- Alternative resources support specific device configurations
  - different languages
  - screen sizes
  - day/night mode
- Default resources
  - used regardless of the device configuration
  - or when there are no alternative resources
- For example:
  - res/layout/
  - res/layout-land/





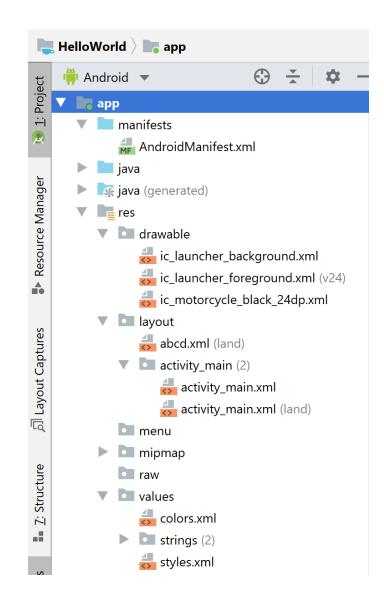
#### **Alternative resources**

Examples: night/notnight, landscape/portrait, pixel density, LTR/RTL, ...



# Resource types

- Main resource types:
  - simple values (strings, colors, styles, dimensions)
  - drawables
  - mipmaps
  - layouts
  - animations
  - menus
  - raw resources
- Stored in one of the subfolders of res/



#### **Colors and dimensions**

#### • Colors:

- expressed as RGB + transparency
- click on coloured square to change it

#### • Dimensions:

• different measurement units: dp, mm, in, px, sp, ...

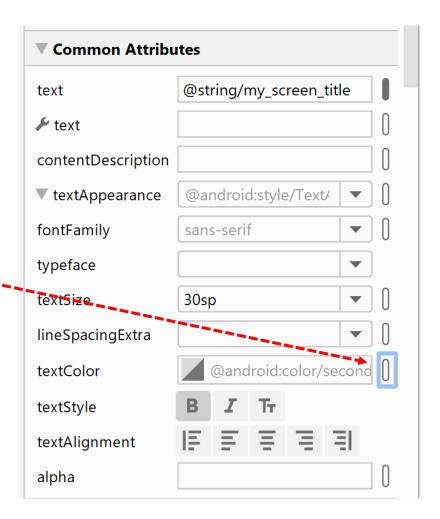
File res/values/colors.xml

File res/values/dimensions.xml

#### **Colors and dimensions**

- Use them in widget properties
  - Write XML
  - Select resource in design

# <TextView android:id="@+id/textView2" android:layout\_width="match\_parent" android:layout\_height="wrap\_content" android:text="@string/my\_instructions" android:textSize="@dimen/my\_large\_font\_size" android:textColor="@color/colorPrimary"/>



# **Styles**

- Collection of properties that specify the look of widgets
  - e.g. height, padding, font color, font size, background color,...
- Defined in an XML resource

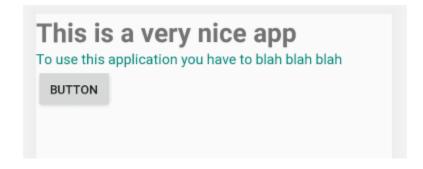
```
<TextView
    android:layout width="match parent"
    android:layout height="wrap content"
   android:textColor="#00FF00"
                                                    <?xml version="1.0" encoding="utf-8"?>
   android:typeface="monospace"
                                                    <resources>
    android:text="@string/hello" />
                                                        <style name="MyFont"</pre>
                                                               parent="@android:style/TextAppearance.Medium">
                                                            <item name="android:layout width">match parent</item>
                                                            <item name="android:Layout height">wrap content</item>
                                                            <item name="android:textColor">#00FF00</item>
                                                            <item name="android:typeface">monospace</item>
                                                        </style>
    <TextView
                                                    </resources>
          style="@style/MyFont"
          android:text="@string/hello" />
```

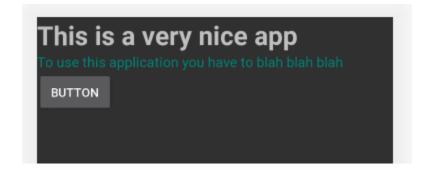
#### **Themes**

- A style can be applied to a single widget
- A theme is a style applied to a whole screen (activity) or application
- You can define new themes and/or select existing ones (in design)

```
<application android:theme="@style/MyTheme">
```

<activity android:theme="@android:style/Theme.Translucent">



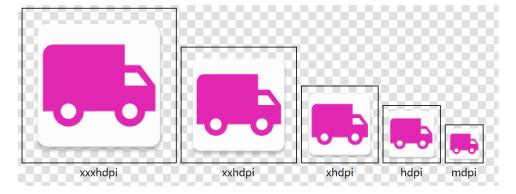


Design.Light

AppCompat.NoActionBar

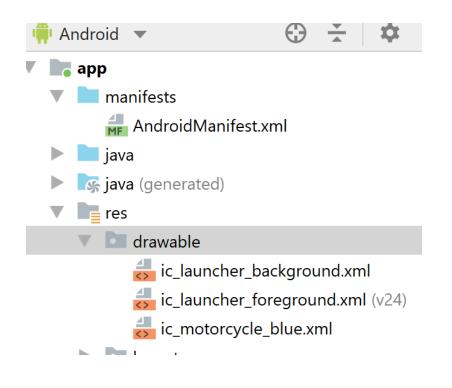
#### **Drawables**

- Android supports images in PNG, JPEG, GIF, and XML formats
- Raster images: put different resolutions of same image into different folders
  - res/drawable-mdpi: medium dpi images (~ 160 dpi)
  - res/drawable-hdpi: high dpi images (~ 240 dpi)
  - res/drawable-xhdpi: extra high dpi images (~ 320 dpi)
  - res/drawable-xxhdpi: extra extra high dpi images (~ 480 dpi)
  - res/drawable-xxxhdpi: high dpi images (~ 640 dpi)
- Vector images (XML) can be scaled without problems



#### **Drawables**

• The resource identifier for a *Drawable* resource is the file name without the extension





```
This is a very nice app
To use this application you have to blah blah

Colored

This is a very nice app
To use this application you have to blah blah blah
```

4:54 🌣 🚨 👸 🖽

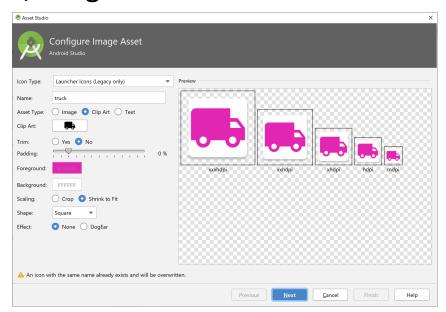
Example

T LTE

```
<ImageButton
    android:id="@+id/imageButton"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    app:srcCompat="@drawable/ic_motorcycle_blue" />
```

#### **Drawables**

- At run-time, Android chooses which resolution/directory (e.g. -mdpi) depending on phone resolution
- Image Asset Studio: generates icons in various densities from original image
  - https://developer.android.com/studio/write/image-asset-studio.html



#### **Animations**

- Animations: Android supports two types of animations
  - Tweened: rotate, move, stretch, and fade a *View*;
  - Frame-by-frame: a sequence of *Drawable* images.
- res/anim folder, animation's file name is used as its resource identifier

```
<?xml version="1.0" encoding="utf-8"?>
<scale xmlns:android="http://schemas.android.com/apk/res/android"</pre>
 android:fromXScale="1.0"
 android:toXScale="0.0"
                                                               <animation-list
 android:fromYScale="1.0"
                                                               xmlns:android="http://schemas.android.com/apk/res/android"
 android:toYScale="0.0"
                                                               android:oneshot="false">
 android:pivotX="50%"
                                                               <item android:drawable="@drawable/rocket1" android:duration="500" />
 android:pivotY="50%"
                                                               <item android:drawable="@drawable/rocket2" android:duration="500" />
                                                               <item android:drawable="@drawable/rocket3" android:duration="500" />
 android:startOffset="500"
                                                               </animation-list>
 android:duration="500" />
```

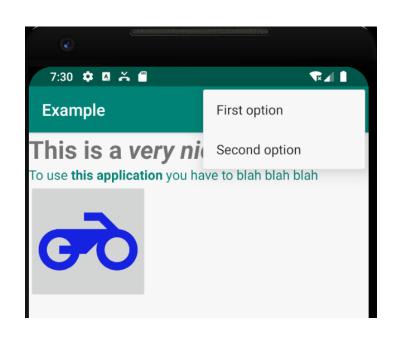
#### Menus

- XML files that define the content and structure of menues
- Can be built using AndroidStudio design
- Must be «inflated» in Activity
- onClick() specifies the method to be invoked

```
<menu xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:android="http://schemas.android.com/apk/res/android">
    <item
        android:id="@+id/first_item"
        android:onClick="mymethod1"
        android:title="First option" />
        <item
        android:id="@+id/second_item"
        android:onClick="mymethod2"
        android:title="Second option" />
    </menu>
```



#### Menus



```
public class MainActivity extends AppCompatActivity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        getMenuInflater().inflate(R.menu.my_menu, menu);
        return true;
    public void mymethod1(MenuItem v) {
        Log.i("Example", "First option");
    public void mymethod2(MenuItem v) {
        Log.i("Example", "Second option");
```

# **Using resources**

- Resources can be
  - used from Java code (e.g. to retrieve input from an EditText)
  - be referenced from within other XML resources (e.g., an image in a layout)

- The R class is the connection between Java code and resources (in XML or provided as files)
- The R class is automatically generated by AndroidStudio

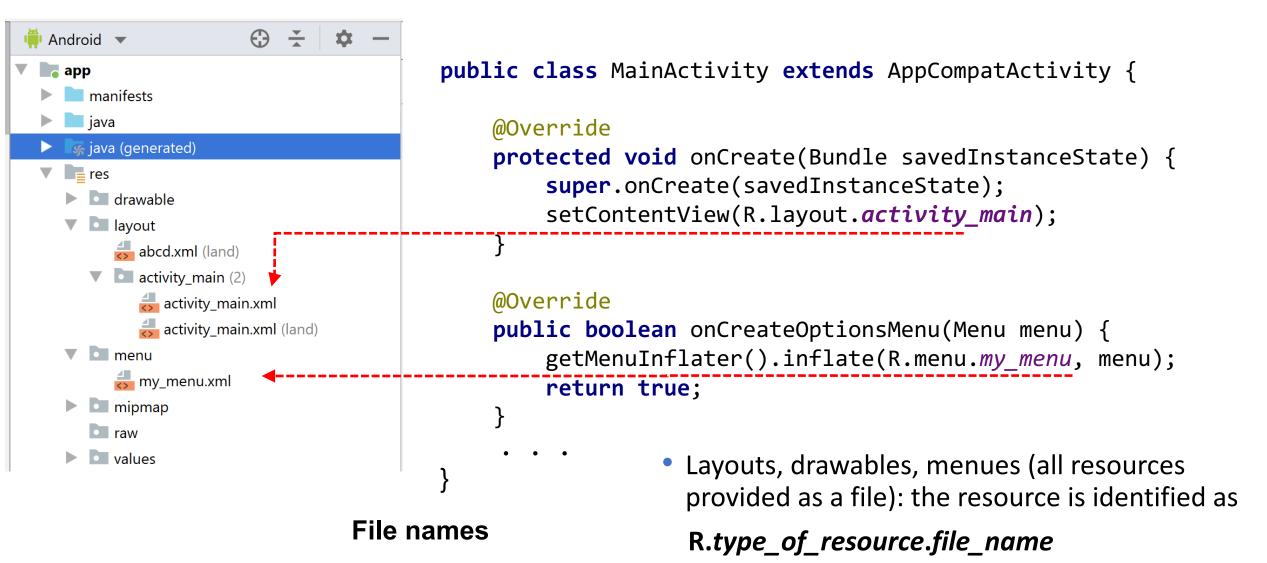
```
File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help
       😂 🛊 🔅 🕒 © HelloWorldActivity.java × 🔯 activity hello_world.xml × 🔯 strings.xml × 🐮 R.java ×
                               Files under the build folder are generated and should not be edited

▼ it.unipi.iet.helloworld

                                 package it.unipi.iet.helloworld;
        © % HelloWorldActivity
     it.unipi.iet.helloworld (androidTest)
                                 public final class R {

▼ layout
                                     public static final class attr {
        activity hello world.xml
     menu_hello_world.xml
                                     public static final class dimen {
     mipmap
                                          public static final int activity horizontal margin=0x7f040000;
      ▼ ic launcher.png (4)
          ic launcher.png (hdpi)
                                          public static final int activity vertical margin=0x7f040001;
          ic_launcher.png (mdpi)
          ic launcher.png (xhdpi)
          ic launcher.png (xxhdpi)
                                     public static final class id {
     ▼ li values
                                          public static final int action settings=0x7f080000;
          dimens.xml
          dimens.xml (w820dp)
        strings.xml
                                     public static final class layout {
       ▼ i styles.xml (2)
                                          public static final int activity hello world=0x7f030000;
          styles.xml
          styles.xml (v21)
   Gradle Scripts
                                     public static final class menu {
                                          public static final int menu hello world=0x7f070000;
                                     public static final class mipmap {
                                          public static final int ic launcher=0x7f020000;
                                     public static final class string {
                                          public static final int action settings=0x7f050000;
                                          public static final int app name=0x7f050001;
                                          public static final int hello world=0x7f050002;
                                     public static final class style {
                                          /** Customize your theme here.
  ▶ <u>4</u>: Run 🔏 TODO 📫 <u>6</u>: Android
Session 'app': running (50 minutes ago)
```

 For widgets: a reference can be obtained by using the findViewById() method and providing the resource identifier



For simple values (strings, colors, etc): the resource is identified by means
of the name attribute

R.type\_of\_simple\_value.name\_of\_resource

```
<string name="my_string">Press this
button to continue</string>
```

```
String s = getResources().getString(R.string.my_string);
```

## Using resources in other XML resources

 You can use resources as attribute values in other XML resources attribute="@resourcetype/resource\_identifier"

```
<string name="my_string">Press this
button to continue</string>
```

<EditText
android:id="@+id/myEditText"
android:layout\_width="fill\_parent"
android:layout\_height="wrap\_content"
android:text="@string/my\_string"
android:textColor="@color/opaque\_blue"
/>

# Using system's resources

- Resources can be divided in
  - Your project's resources
  - Android OS' resources
- To use Android resources use android.R (in Java code) or @android: rather than the application-specific R class
- Examples:

```
CharSequence httpError = getString(android.R.string.httpErrorBadUrl);

<EditText
android:id="@+id/myEditText"
android:layout_width="fill_parent"
android:layout_height="wrap_content"
android:text="@android:string/httpErrorBadUrl"
android:textColor="@android:color/darker_gray"
```

/>

### References

- <a href="https://developer.android.com">https://developer.android.com</a>
- CS 528 Mobile and Ubiquitous Computing, WPI