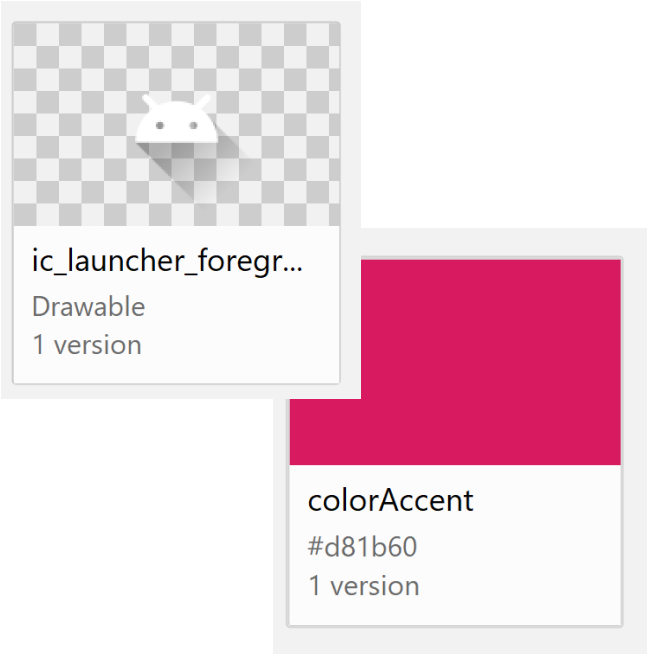
Resources

Resources

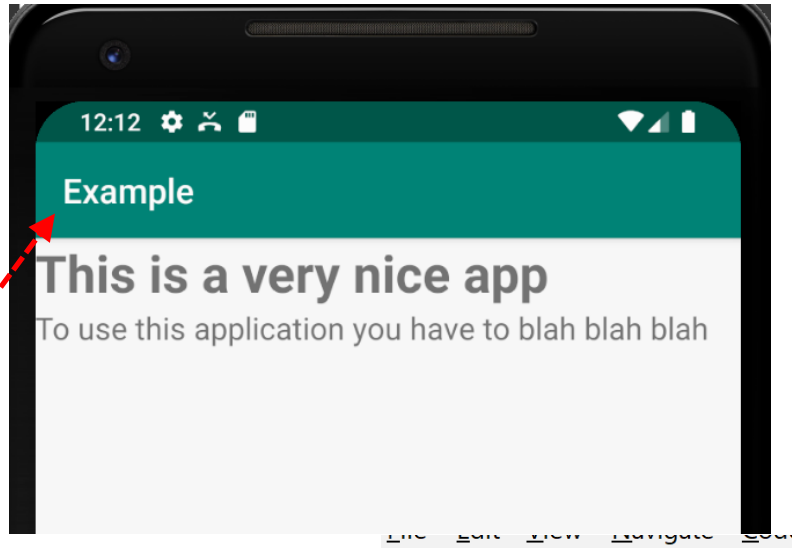
- Resources: strings, dimensions, layout files, menus, images, audio files, etc that your app uses
- Basically app elements defined in other files
  - Easier to update and maintain code

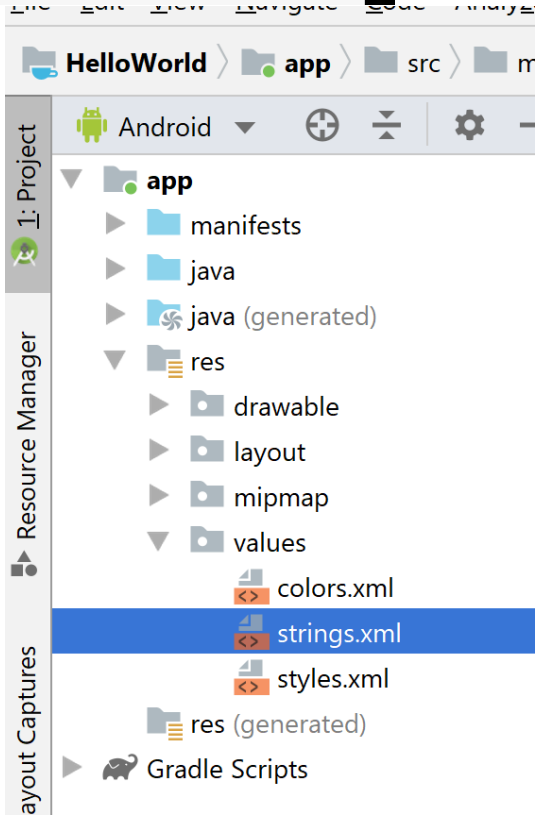



```
<resources>
    <string name="app_name">Example</string>
    <string name="my_screen_title">This is a very nice app</string>
    <string name="my_instructions">To use this application you have
                        to blah blah blah</string>
</resources>
```

Strings

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

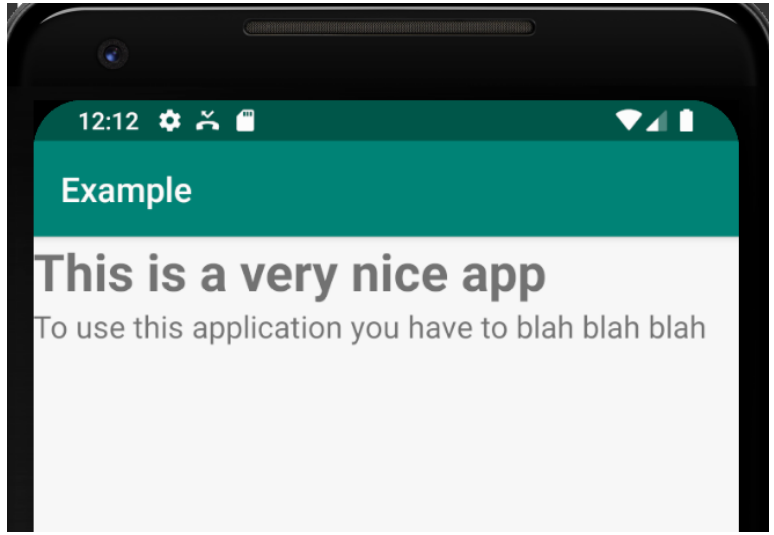





```
<resources>
    <string name="app_name">Example</string>
    <string name="my_screen_title">This is a very nice app</string>
    <string name="my_instructions">To use this application you have to
blah blah blah</string>
</resources>
```

Strings

- Then can be used in any other XML file



Layout file

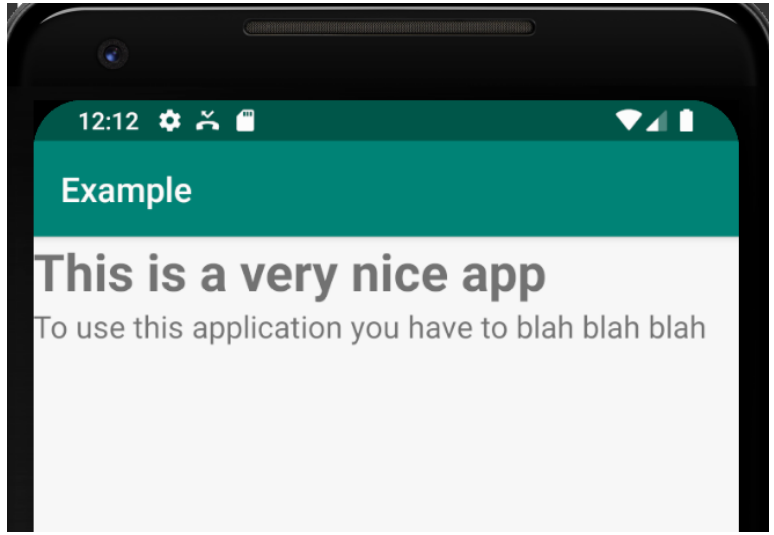
```
. . .
<TextView
    android:id="@+id/textView1"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:text="@string/my_screen_title"
    android:textSize="30sp"
    android:textStyle="bold" />

<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

```
<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 blah</string>
</resources>
```

Strings

- Then can be used in any other XML file



AndroidManifest file

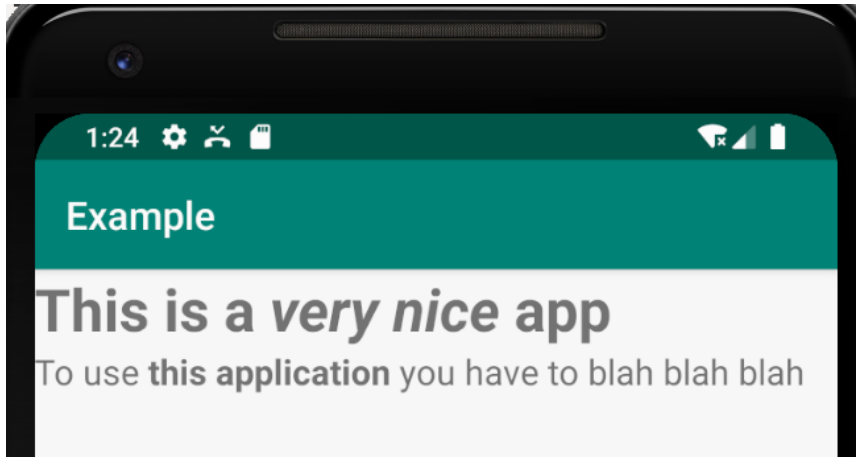
```
. . .
<application
    android:allowBackup="true"
    android:icon="@mipmap/ic_launcher"
    android:label="@string/app_name"
    android:roundIcon="@mipmap/ic_launcher_round"
    android:supportsRtl="true"
    android:theme="@style/AppTheme">
 . . .
```

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 blah</string>
</resources>
```

Strings

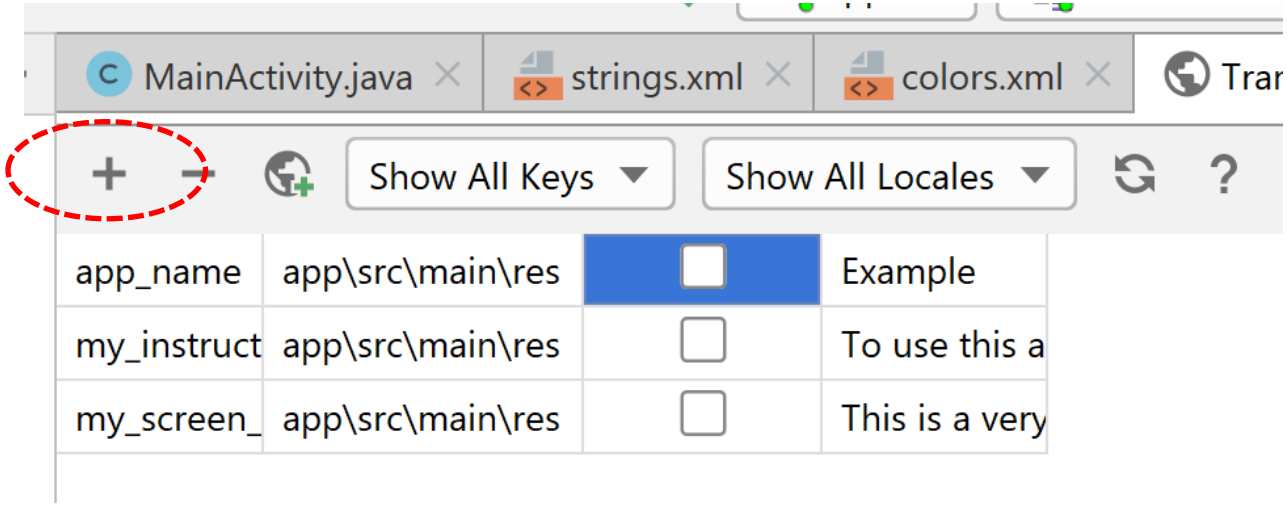
- Some basic HTML formatting can be included
  - ** for bold
  - ** for italics

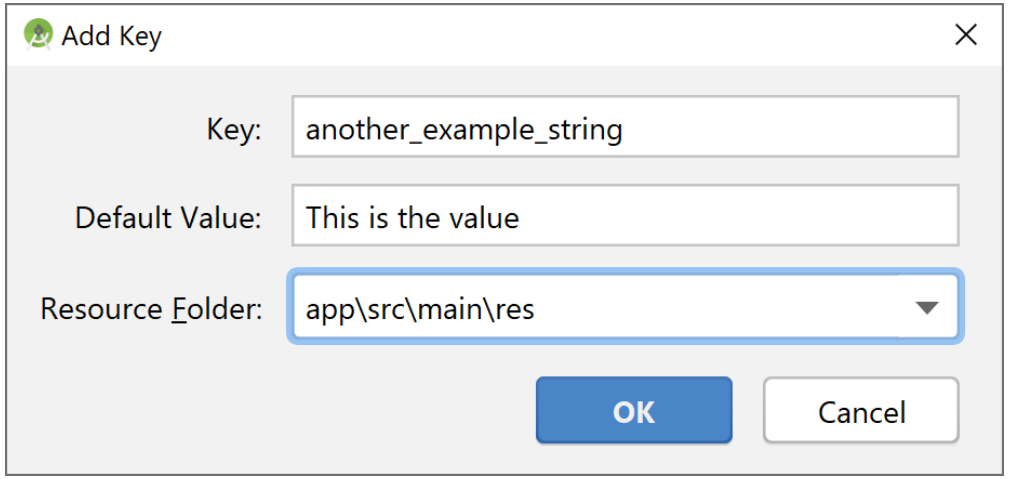



```
<resources>
    <string name="app_name">Example</string>
    <string name="my_screen_title">This is a <i>very nice</i> app</string>
    <string name="my_instructions">To use <b>this application</b> you have to blah
blah blah</string>
</resources>
```

Strings

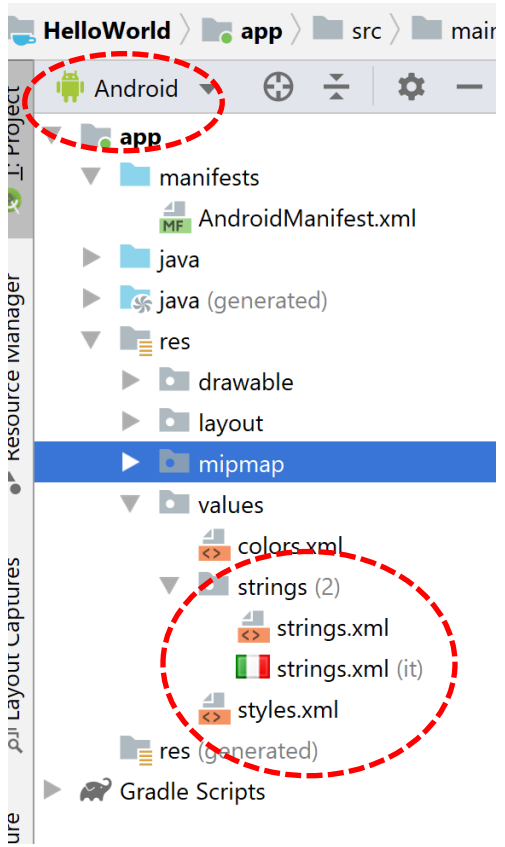
- New strings can be added
  - editing the *strings.xml* file
  - using the tools/wizards provided by AndroidStudio

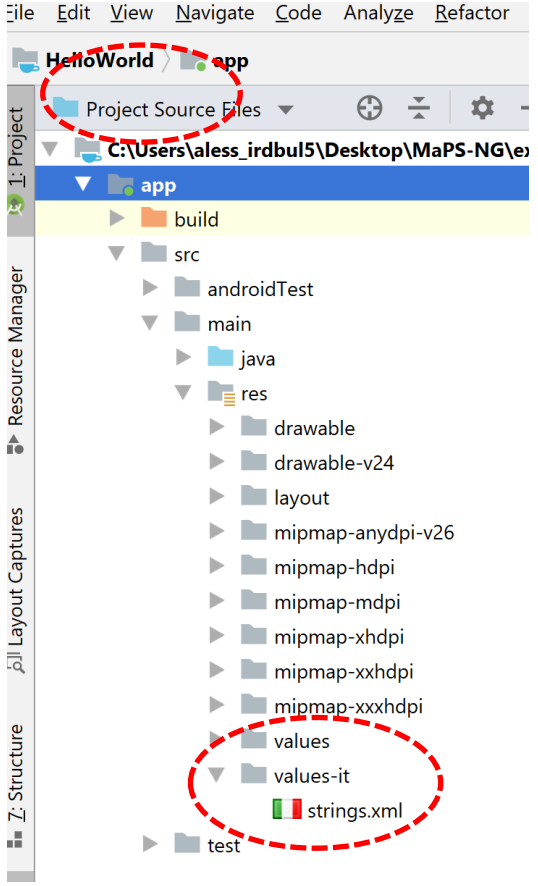


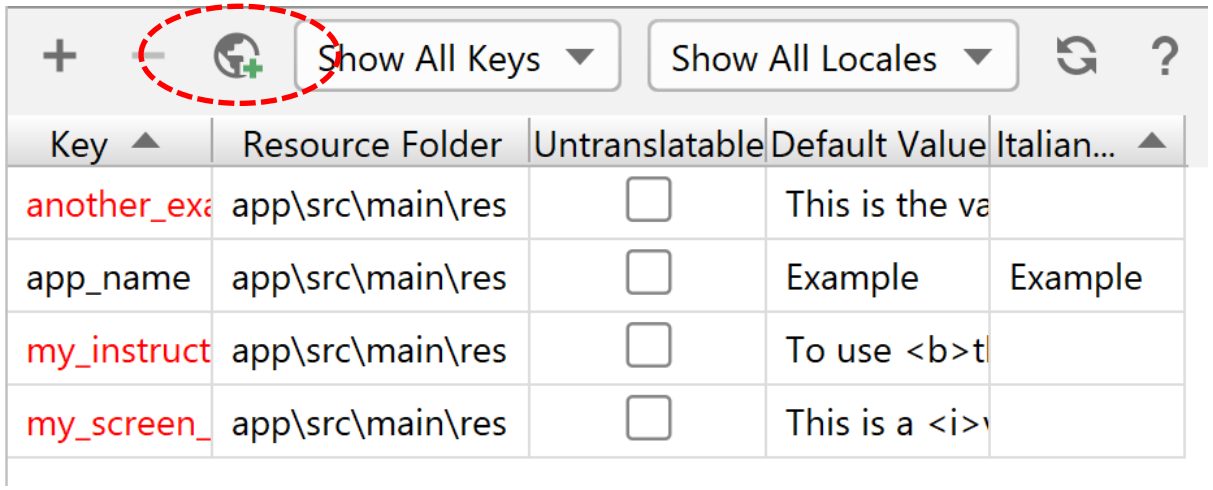


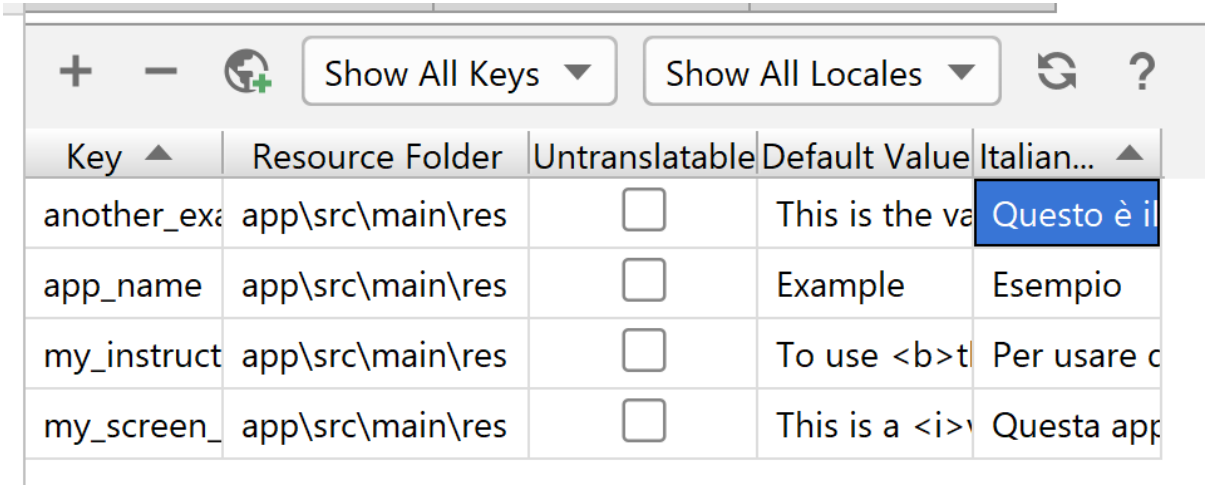
Translations

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



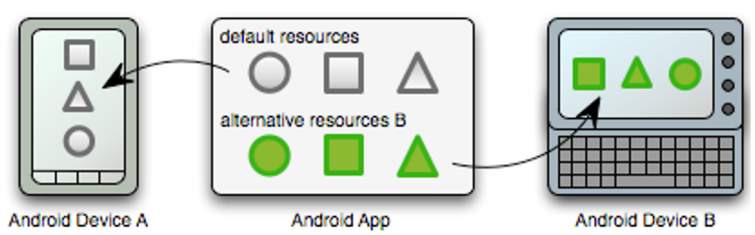






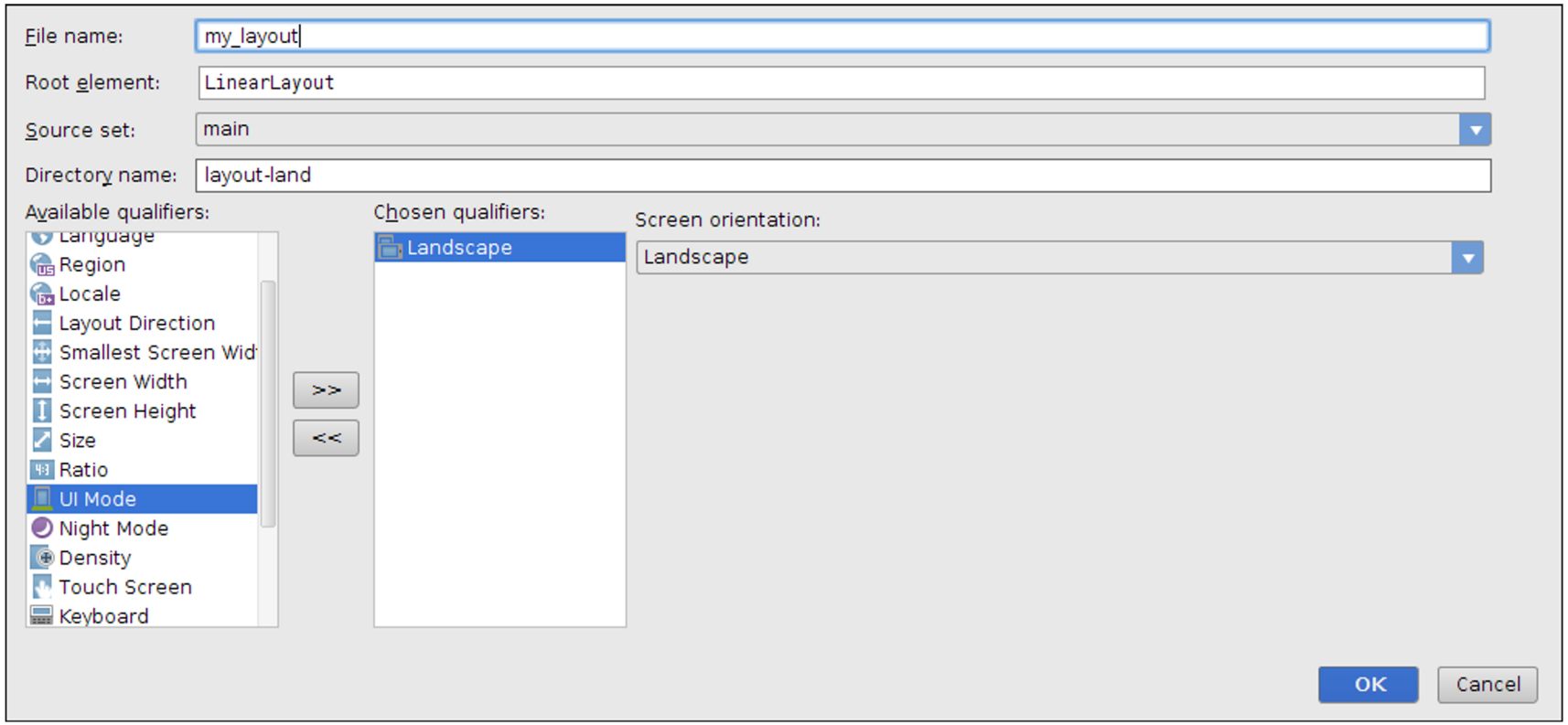
Alternative resources

- 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

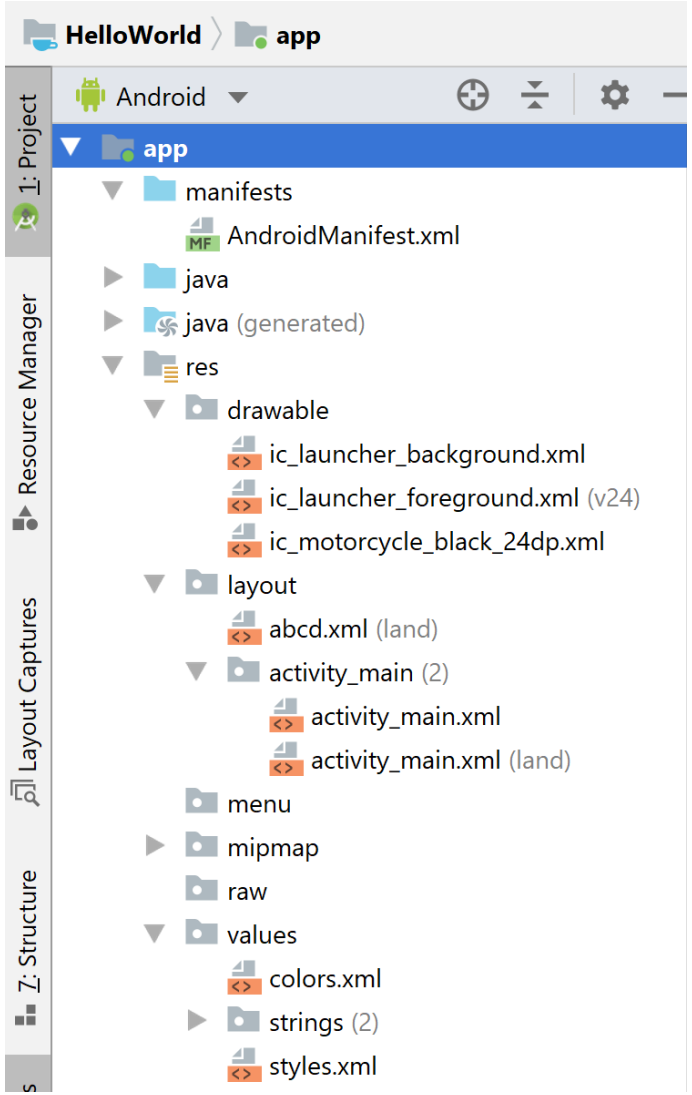
- Examples: night/notnight, landscape/portrait, pixel density, LTR/RTL, ...



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*, …

```xml
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <color name="colorPrimary">#008577</color>
    <color name="colorPrimaryDark">#00574B</color>
    <color name="colorAccent">#D81B60</color>
</resources>
```

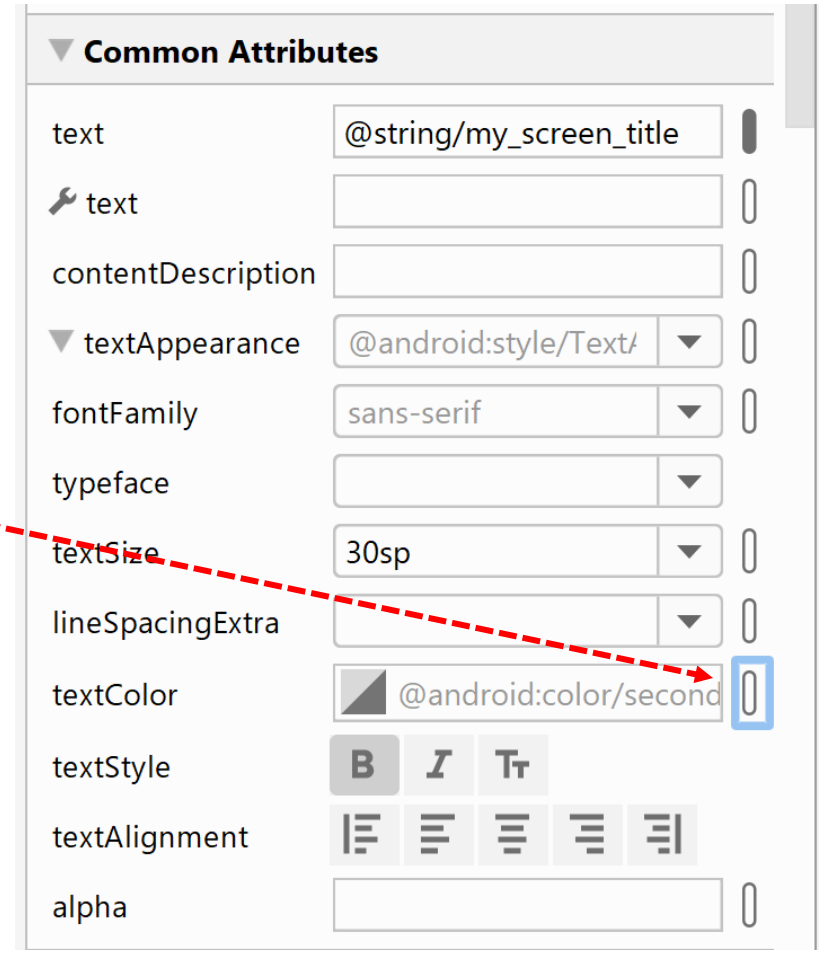
File res/values/colors.xml

```xml
<resources>
    <dimen name="my_border_size">100dp</dimen>
    <dimen name="my_large_font_size">16sp</dimen>
</resources>
```

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"
    android:typeface="monospace"
    android:text="@string/hello" />
```

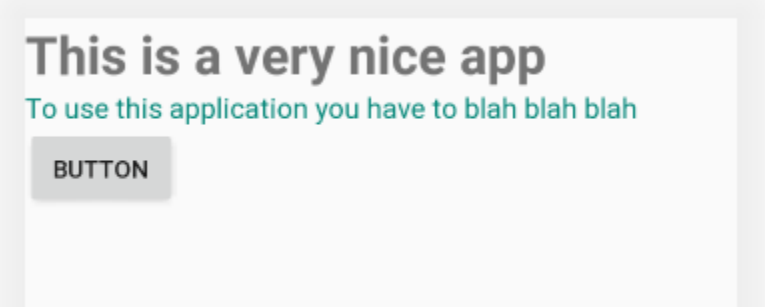
```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <style name="MyFont"
           parent="@android:style/TextAppearance.Medium">
        <item name="android:layout_width">match_parent</item>
        <item name="android:layout_height">wrap_content</item>
        <item name="android:textColor">#00FF00</item>
        <item name="android:typeface">monospace</item>
    </style>
</resources>
```

```
<TextView
    style="@style/MyFont"
    android:text="@string/hello" />
```

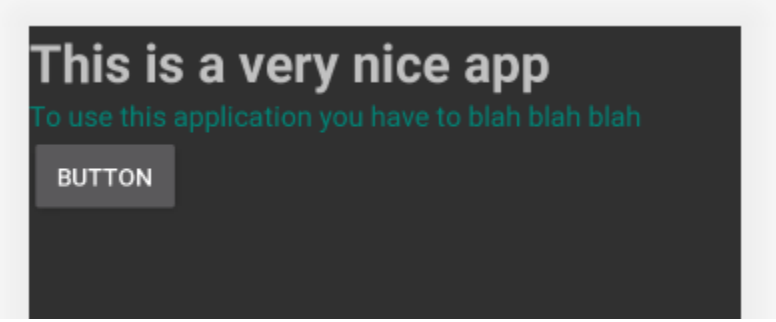
Themes

- 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">
```



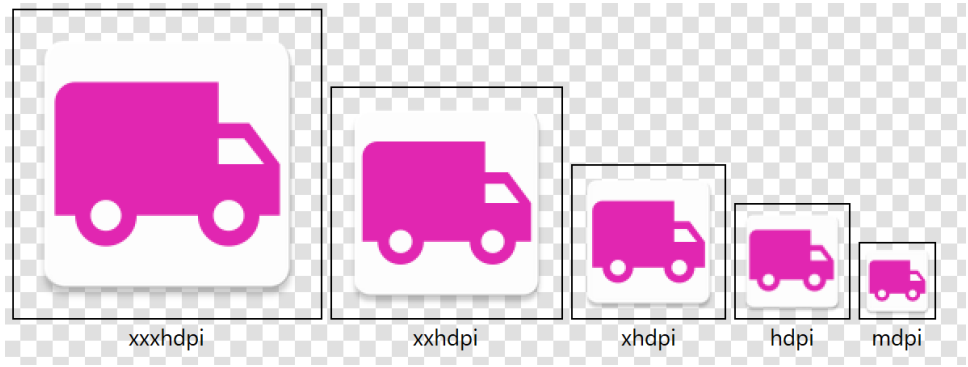
Design.Light



AppCompat.NoActionBar

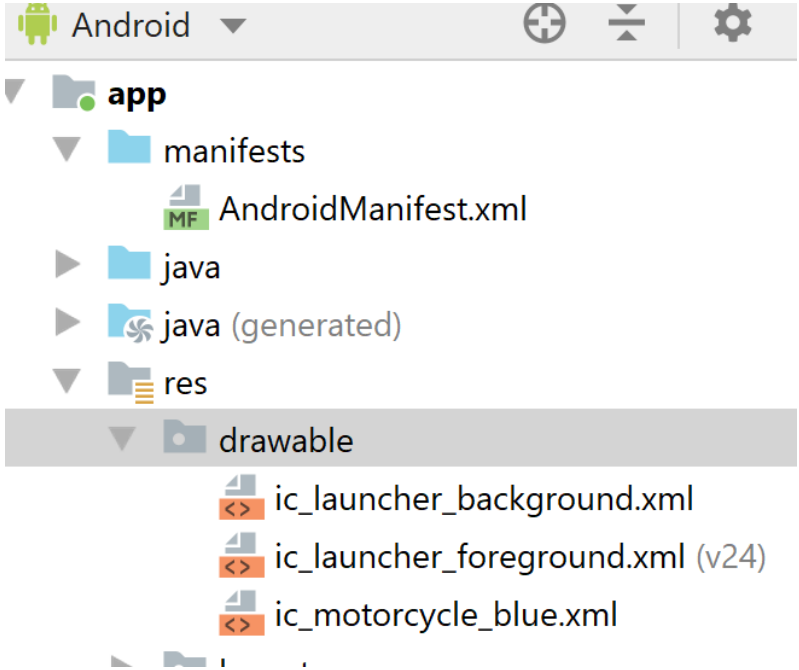
Drawables

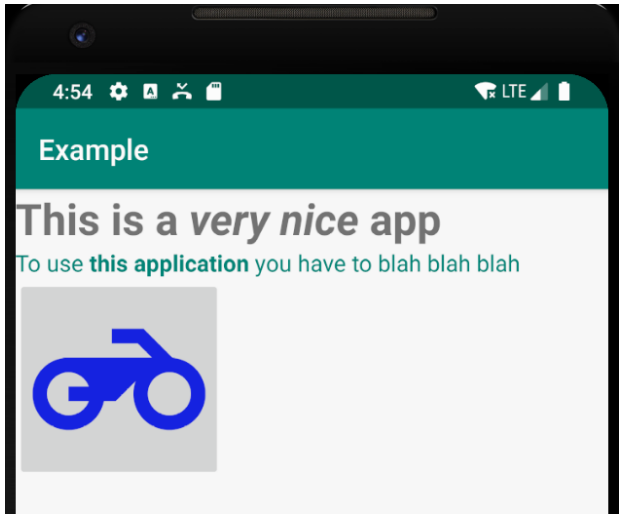
- Android supports images in PNG, JPEG, GIF, and XML formats
- Raster images: put different resolutions of same image into different folders
  - **res/drawable-mdpi**: medium dpi images (~ 160 dpi)
  - **res/drawable-hdpi**: high dpi images (~ 240 dpi)
  - **res/drawable-xhdpi**: extra high dpi images (~ 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

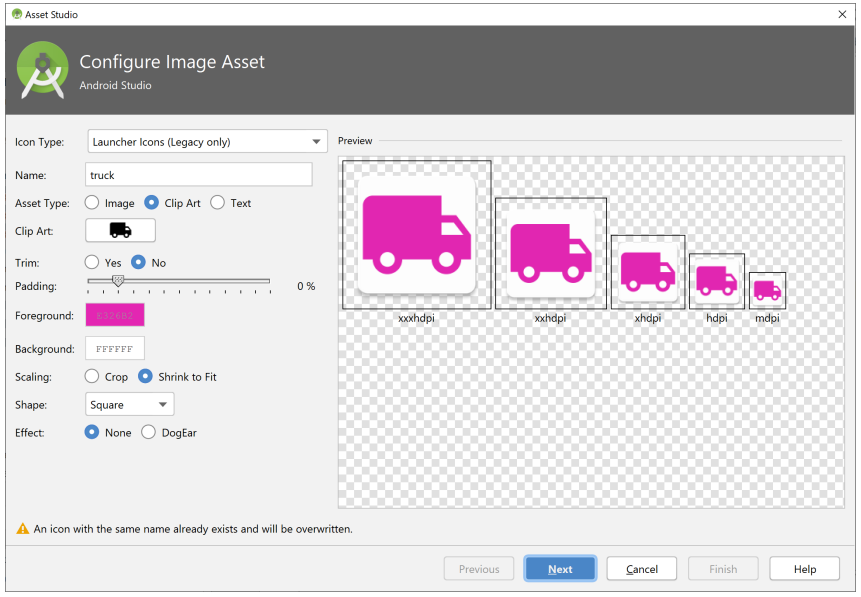





```
<ImageButton
    android:id="@+id/imageButton"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    app:srcCompat="@drawable/ic_motorcycle_blue" />
```

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

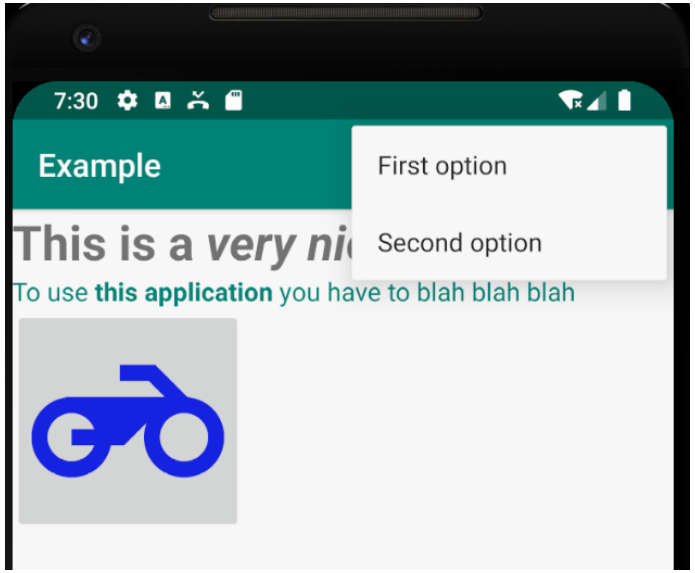
- **Animations**: Android supports two types of animations
  - Tweened: rotate, move, stretch, and fade a *View*;
  - Frame-by-frame: a sequence of *Drawable* images.
- *res/anim* folder, animation's file name is used as its resource identifier

```
<?xml version="1.0" encoding="utf-8"?>
<scale xmlns:android="http://schemas.android.com/apk/res/android"
 android:fromXScale="1.0"
 android:toXScale="0.0"
 android:fromYScale="1.0"
 android:toYScale="0.0"
 android:pivotX="50%"
 android:pivotY="50%"
 android:startOffset="500"
 android:duration="500" />
```

```
<animation-list
xmlns:android="http://schemas.android.com/apk/res/android"
android:oneshot="false">
<item android:drawable="@drawable/rocket1" android:duration="500" />
<item android:drawable="@drawable/rocket2" android:duration="500" />
<item android:drawable="@drawable/rocket3" android:duration="500" />
</animation-list>
```

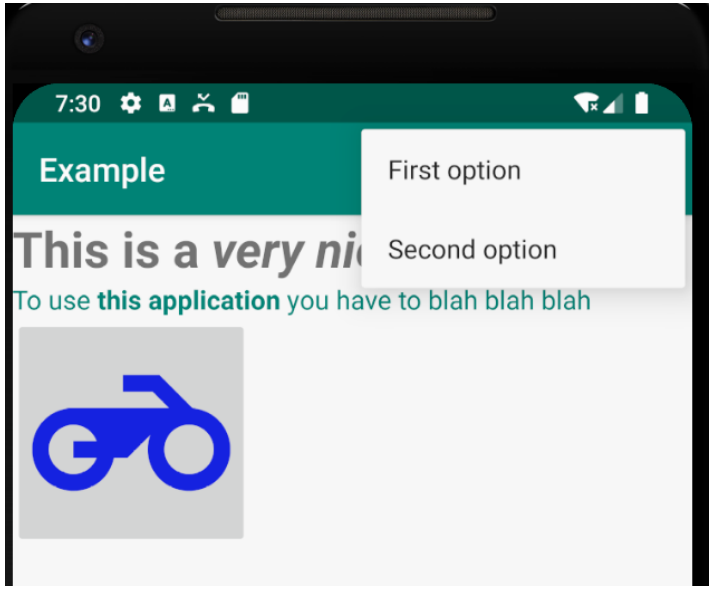
Menus

- 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



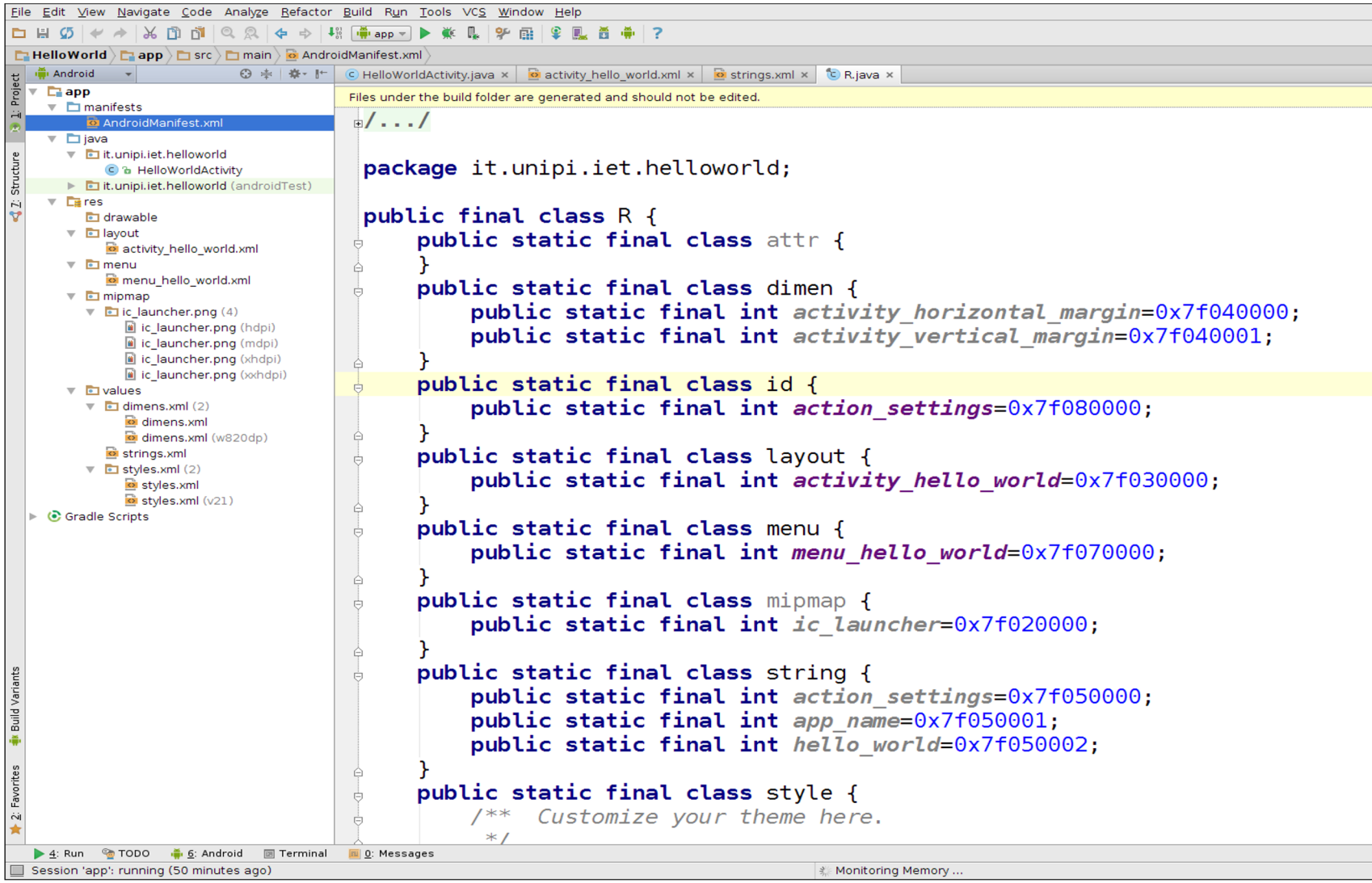

```
public class MainActivity extends AppCompatActivity {

    @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)

Using resources in Java code

- 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



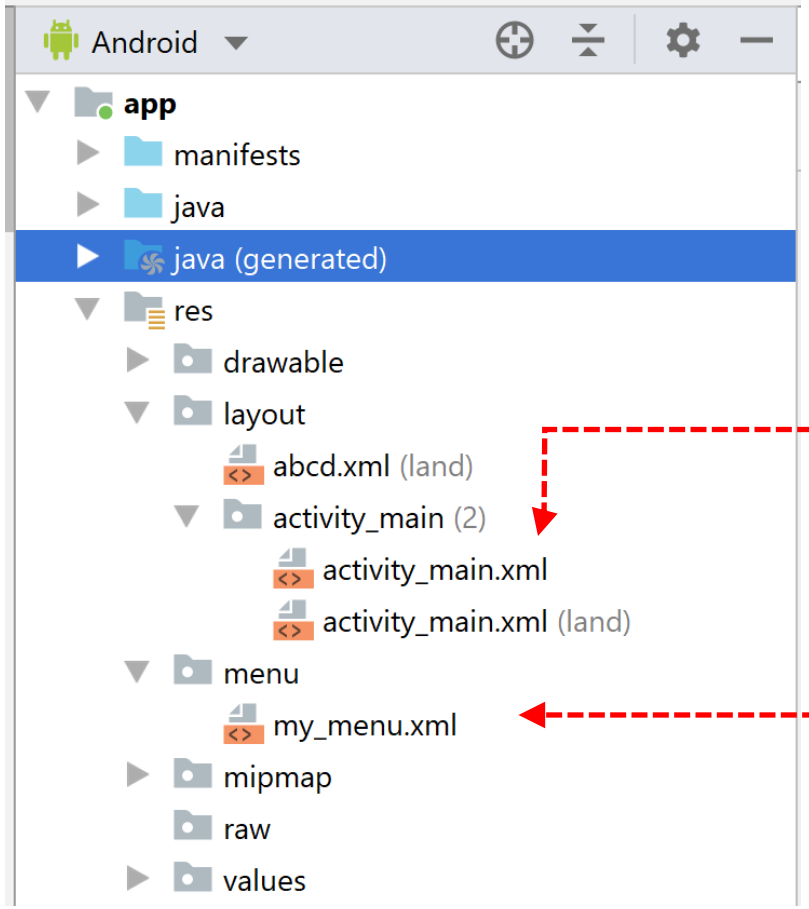
Using resources in Java code

```
<EditText
    android:id="@+id/editText1"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:ems="10"
    android:text="Insert some text"
    android:textSize="24sp" />
```

```
EditText et = (EditText) findViewById(R.id.editText1);
String text = et.getText().toString();
```

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

Using resources in Java code




```
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;
    }
    . . .
}
```

File names

- Layouts, drawables, menues (all resources provided as a file): the resource is identified as

  R.*type_of_resource.file_name*

Using resources in Java code

- 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

- https://developer.android.com
- CS 528 Mobile and Ubiquitous Computing, WPI