



MODULE 5

RESOURCE

TABLE OF CONTENT

- What is a Resource?
- Types of Resources
- String Resource
- Internationalization
- Webography

What is a Resource?

- ▶ Resources include text data , bitmaps, audio, videos, ...
- ▶ Better to define resources in external files
 - Easier to maintain
- ▶ XML files
 - In **res** directory
- ▶ Accessed in Java code through IDs assigned to them

What is a Resource?

- ▶ Apps applicable to diverse hardware systems
 - By creating several resources files supporting different hardwares
- ▶ Resources mainly divided into drawables, layouts and values

Types of Resources

▶ **Drawable** Resources

- Define various graphics with bitmaps or XML
 - Icons and graphics resources according to the screen resolutions
- Saved in **res/drawable/**
 - drawable-xhdpi folder : for 320dpi
 - drawable-hdpi folder : for 240dpi
 - drawable-mdpi folder : for 160dpi
 - drawable-ldpi folder : for 120dpi
- Accessed from the **R.drawable** class

Types of Resources

- ▶ **Layout** Resource
 - Define the layout for application UI
 - Saved in **res/layout/**
 - Accessed from the **R.layout** class
- ▶ **Menu** Resource
 - Define the contents of application menus
 - Saved in **res/menu/**
 - Accessed from the **R.menu** class

Types of Resources

► MipMap Resources

- For placing the app icons only
 - Any other drawable assets are placed in drawable folders
- Saved in `res/mipmap/`
- Accessed from the `R.mipmap` class

Types of Resources

- ▶ **Values** Resource: **String** Resources
 - Define strings, string arrays, and plurals (and include string formatting and styling)
 - Saved in **res/values/**
 - Accessed from the **R.string**, **R.array** and **R.plurals** classes
- ▶ **Values** Resource: **Style** Resources
 - Define the look and format for UI elements
 - Saved in **res/values/**
 - Accessed from the **R.style** class

Types of Resources

▶ Animation Resources

- Tween animations
 - Saved in **res/anim/**
 - Accessed from the **R.anim** class
- Frame animations
 - Saved in **res/drawable/**
 - Accessed from the **R.drawable** class

Types of Resources

- ▶ **Color** State List Resource
 - Define a color resources that changes based on the View state
 - Saved in **res/color/**
 - Accessed from the **R.color** class

Types of Resources

► More Resource Types

- Define values such
 - Bool : XML resource that carries a boolean value
 - Color : XML resource that carries a hexadecimal color value
 - Dimension : XML resource that carries a dimension value (with a unit of measure)
 - ID : XML resource that provides a unique identifier for application resources and components
 - Integer : XML resource that carries an integer value
 - Integer Array : XML resource that provides an array of integers
 - Typed Array : XML resource that provides a TypedArray
- Saved in **res/values/**

Types of Resources

- ▶ More Resource Types (continued)
 - But each accessed from unique R sub-classes
 - Bool : Accessed from the **R.bool** class
 - Color : Accessed from the **R.color** class
 - Dimension : Accessed from the **R.dimen** class
 - ID : Accessed from the **R.id** class
 - Integer : Accessed from the **R.integer** class
 - Integer Array : Accessed from the **R.array** class
 - Typed Array : Accessed from the **R.array** class

String Resource

- ▶ Do not hardcode String values!
 - Better to define String resources in external file
 - Easier to maintain
 - E.g, for internationalization

Using String Resource

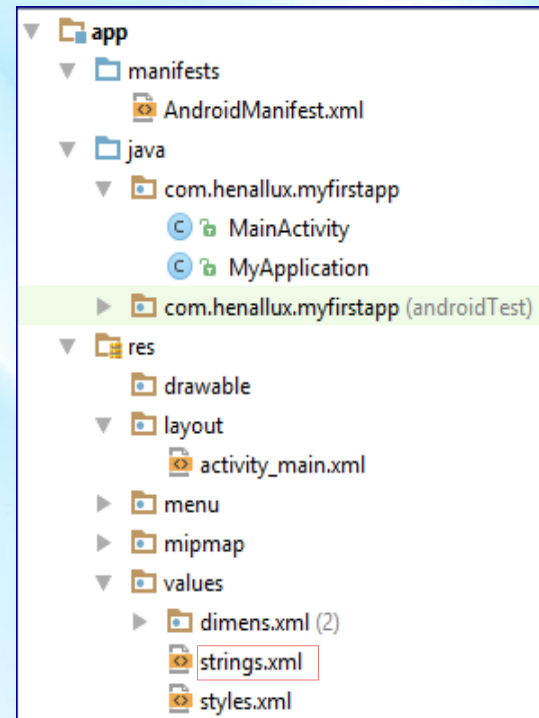
- ▶ Add new resource entry in strings.xml file
 - Through **<string>** tag
 - Attribute **name** : identifies resource

▶ E.g,

```
<resources>  
  <string name="app_name">MyFirstApp</string>  
  <string name="action_settings">Settings</string>  
  <string name="hello_world">Hello world</string>  
</resources>
```

Resource identifier

String value



String Resource

► Use String resource

◦ In XML files

- Through `@string/resourceIdentifier`

- E.g,

```
<TextView
    android:id="@+id/helloText"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="@string/hello_world" />
```

◦ In Java

- Through **R.string**

- E.g,

```
String message = getString(R.string.hello_world);
```

```
Toast.makeText(MainActivity.this, R.string.hello_world, Toast.LENGTH_LONG).show();
```


Internationalization

- ▶ Put default language text in `res/values/strings.xml`
- ▶ To support additional languages:
 - Create additional values directories inside **res/**
 - Include a hyphen and the ISO language code at the end of the directory name
 - E.g, ***res/values-fr***
 - Copy strings.xml files in those directories with values in right language
 - E.g, ***res/values-fr/strings.xml***
- ▶ Android loads the appropriate resources according to the locale settings of the device at runtime

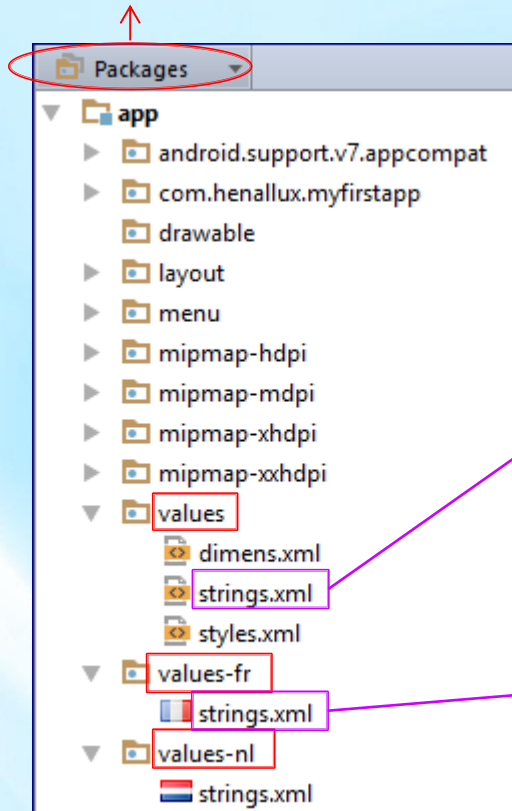
Internationalization

- ▶ The Iso language code is defined by
 - A two-letter ISO 639-1 language code
 - Optionally followed by a two letter ISO 3166-1-alpha-2 region code (preceded by lowercase "r")
 - E.g, **en**, **fr**, **en-rUS**, **fr-rFR**, **fr-rCA**

Internationalization

► E.g.,

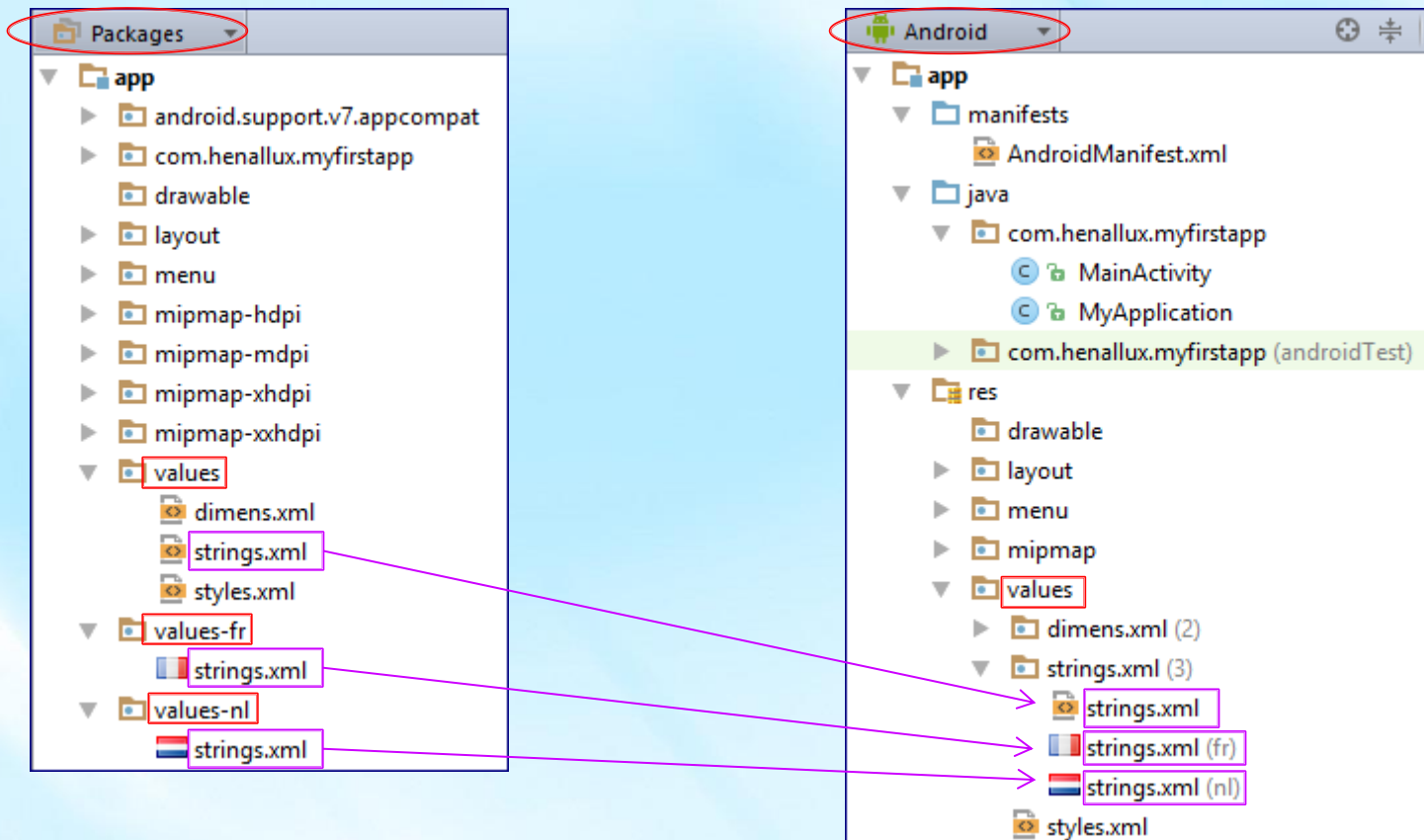
In Android studio: change to Package view



```
<?xml version="1.0" encoding="utf-8"?>
  <resources>
    <string name="app_name">My first application</string>
    <string name="hello_world">Hello world!</string>
  </resources>
```

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
  <string name="app_name">Ma première application</string>
  <string name="hello_world">Bonjour à tous!</string>
</resources>
```

Internationalization



Webography

- ▶ <http://developer.android.com/guide/topics/resources/index.html>
- ▶ <http://developer.android.com/guide/topics/resources/available-resources.html>
- ▶ <http://developer.android.com/guide/topics/resources/localization.html>
- ▶ <http://developer.android.com/training/basics/supporting-devices/languages.html>