Application Resources

Learning Objectives

After studying this unit student should be able to learn:

- Define Application Resources
- List different types of resources
- Understand resource directory hierarchy within android project
- How resources are stored
- Access user resources and system resources programmatically.

Introduction

Resources are the additional files and static content that your code uses, such as bitmaps, layout definitions, user interface strings, animation instructions, and more.

You should always externalize app resources such as images and strings from your code, so that you can maintain them independently. You should also provide alternative resources for specific device configurations, by grouping them in specially-named resource directories. At runtime, Android uses the appropriate resource based on the current configuration. For example, you might want to provide a different UI layout depending on the screen size or different strings depending on the language setting.

Once you externalize your app resources, you can access them using resource IDs that are generated in your project's R class. This document shows you how to group your resources in your Android project and provide alternative resources for specific device configurations, and then access them from your app code or other XML files.
The well-written application accesses its resources programmatically instead of hard coding them into the source code. This is done for a variety of reasons. Storing application resources in a single place is a more organized approach to development and makes the code more readable and maintainable. Externalizing resources such

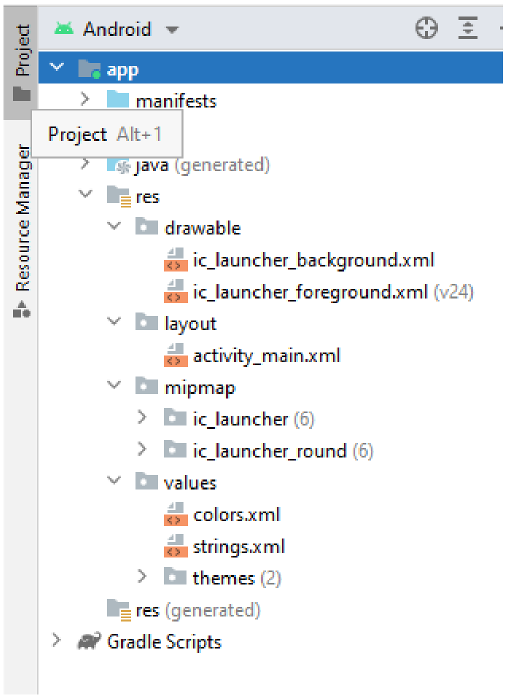
as strings makes it easier to localize applications for different languages and geographic regions.

What are resources?

All Android applications are composed of two things: functionality (code instructions) and data (resources).The functionality is the code that determines how your application behaves. This includes any algorithms that make the application run. Resources include text strings, images and icons, audio files, videos, and other data used by the application. Android resource files are stored separately from the java class files in the Android project. Most common resource types are stored in XML.You can also store raw data files

Resource Directory Hierarchy

Resources are organized in a strict directory hierarchy within the Android project. All resources must be stored under the /res project directory in specially named subdirectories that must be lowercase. Different resource types are stored in different directories. The resource sub-directories generated when you create an Android project are shown in below.



```
app
  manifests
  java (generated)
  res
    drawable
      ic_launcher_background.xml
      ic_launcher_foreground.xml (v24)
    layout
      activity_main.xml
    mipmap
      ic_launcher (6)
      ic_launcher_round (6)
    values
      colors.xml
      strings.xml
      themes (2)
  res (generated)
Gradle Scripts
```

Figure: Resource Hierarchy

Each resource type corresponds to a specific resource subdirectory name. For example, all graphics are stored under the /res/drawable directory structure. Resources can be further organized in a variety of ways using even more specially named directory qualifiers.

For example, the /res/drawable-hdpi directory stores graphics for high-density screens, the /res/drawable-ldpi directory stores graphics for low-density screens, and the /res/drawable-mdpi directory stores graphics for medium-density screens. If you had a graphic resource that was shared by all screens, you would simply store that resource in the /res/drawable directory.

Resource Value Types

Android applications rely on many different types of resources—such as text strings, graphics, and color schemes—for user interface design.

These resources are stored in the /res directory of your Android project in a strict (but reasonably flexible) set of directories and files. All resources filenames must be lowercase and simple (letters, numbers, and underscores only).
The resource types supported by the Android SDK and how they are stored within the project are shown in table below

Resource Type	Directory	Filename	XML Tag
Strings	/res/values/	strings.xml	<string>
String Pluralization	/res/values/	strings.xml	<plurals>, <item>
Arrays of Strings	/res/values/	strings.xml	<string-array>, <item>
Booleans	/res/values/	bools.xml	<bool>
Colors	/res/values/	Colors.xml	<color>
Color State Lists	/res/color/	Examples include buttonstates.xml indicators.xml	<selector>, <item>
Dimensions	/res/values/	Dimens.xml	<dimen>

Resource Type	Directory	Filename	XML Tag
Integers	/res/values/	integers.xml	<integer>
Arrays of Integers	/res/values/	integers.xml	<integer-array>,<item>
Mixed-Type Arrays	/res/values/	Arrays.xml	<array>, <item>
Simple Drawables	/res/values/	drawables.xml	<drawable>
Graphics	/res/drawable/	Examples include icon.png logo.jpg	Supported graphics files or drawable definition XML files such as shapes.
Tweened Animations	/res/anim/	Examples include fadesequence.xml spinsequence.xml	<set>, <alpha>, <scale>, <translate>, <rotate>
Frame-by-Frame Animations	/res/drawable/	Examples include sequence1.xml sequence2.xml	<animation-list>, <item>
Menus	/res/menu/	Examples include mainmenu.xml helpmenu.xml	<menu>
XML Files	/res/xml/	Examples include data.xml data2.xml	Defined by the developer
Raw Files	/res/raw/	Examples include jingle.mp3 somevideo.mp4 helptext.txt	Defined by the developer
Layouts	/res/layout/	Examples include main.xml help.xml	Varies. Must be a layout control.
Styles and Themes	/res/values/	styles.xml themes.xml	<style>

Table-1

Storing Different Resource Value Types

Storing Simple Resource Types Such as Strings

Simple resource value types, such as strings, colors, dimensions, and other primitives, are stored under the /res/values project directory in XML files. Each resource file under the /res/values directory should begin with the following XML header:

<?xml version=”1.0” encoding=”utf-8”?>

Next comes the root node <resources> followed by the specific resource element types such as <string> or <color>. Each resource is defined using a different element name. Although the XML file names are arbitrary, the best practice is to store your resources in separate files to reflect their types, such as strings.xml, colors.xml, and so on. However, there’s nothing stopping the developers from creating multiple resource files for a given type, such as two separate xml files called bright_colors.xml and muted_colors.xml, if they so choose.

Storing Graphics, Animations, Menus, and Files

In addition to simple resource types stored in the /res/values directory, you can also store numerous other types of resources, such as animation sequences, graphics, arbitrary XML files, and raw files. These types of resources are not stored in the /res/values directory, but instead stored in specially named directories according to their type. For example, you can include animation sequence definitions in the /res/anim directory. Make sure you name resource files appropriately because the resource name is derived from the filename of the specific resource. For example, a file called flag.png in the /res/drawable directory is given the name R.drawable.flag.

Accessing Resource Programmatically

When android application is compiled, a R class gets generated, which contains resource IDs for all the resources available in your res/ directory. You can use R class to access that resource using sub-directory and resource name or directly resource ID.

During your application development you will need to access defined resources either in your code, or in your layout XML files. Following section explains how to access your resources in both the scenarios.

Example-1: To access res/drawable/myimage.png and set an ImageView you will use following code −

```
ImageView imageView = (ImageView) findViewById(R.id.myimageview);
imageView.setImageResource(R.drawable.myimage);
```

Here first line of the code make use of R.id.myimageview to get ImageView defined with id myimageview in a Layout file. Second line of code makes use of R.drawable.myimage to get an image with name myimage available in drawable sub-directory under /res.

Example-2: Consider next example where res/values/strings.xml has following definition:

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
  <string  name="hello">Hello, World!</string>
</resources>
```

Now you can set the text on a TextView object with ID msg using a resource ID as follows:

```
TextView msgTextView = (TextView) findViewById(R.id.msg);
msgTextView.setText(R.string.hello);
```

Example-3: Consider a layout res/layout/activity_main.xml with the following definition

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

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
   android:layout_width="fill_parent"
   android:layout_height="fill_parent"
   android:orientation="vertical" >

   <TextView android:id="@+id/text"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:text="Hello, I am a TextView" />

   <Button android:id="@+id/button"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:text="Hello, I am a Button" />
</LinearLayout>
```

This application code will load this layout for an Activity, in the onCreate() method as follows:

```
public void onCreate(Bundle savedInstanceState) {
   super.onCreate(savedInstanceState);
   setContentView(R.layout.activity_main);
}
```

Accessing Resources in XML Layout

Consider the following resource XML res/values/strings.xml file that includes a color resource and a string resource:

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

Now you can use these resources in the following layout file to set the text color and text string as follows:

```
<?xml version="1.0" encoding="utf-8"?>
<EditText xmlns:android="http://schemas.android.com/apk/res/android"
  android:layout_width="fill_parent"
  android:layout_height="fill_parent"
  android:textColor="@color/opaque_red"
  android:text="@string/hello" />
```

Referencing System Resources

You can access system resources in addition to your own resources. The android package contains all kinds of resources, which you can browse by looking in the android.R subclasses. Here you find system resources for

- Animation sequences for fading in and out
- Arrays of email/phone types (home, work, and such)
- Standard system colors
- Dimensions for application thumbnails and icons
- Many commonly used drawable and layout types
- Error strings and standard button text
- System styles and themes

You can reference system resources the same way you use your own; set the package name to android. For example, to set the background to the system color for darker gray, you set the appropriate background color attribute to @android:color/darker_gray.

You can access system resources much like you access your application's resources. Instead of using your application resources, use the Android package's resources under the android.R class.

Let us sum up

Android applications rely on various types of resources, including strings, string arrays, colors, dimensions, drawable objects, graphics, animation sequences, layouts, styles, and themes. Resources can also be raw files. Many of these resources are defined with XML and organized into specially named project directories. Both default and alternative resources can be defined using this resource hierarchy.

Resources are compiled and accessed using the R.java class file, which is automatically generated when the application resources are compiled. Developers access application and system resources programmatically using this special class.

Further Reading

- https://developer.android.com/reference/android/content/res/Resources
- https://developer.android.com/guide/topics/resources/providing-resources

Activity

- Create String Resource for Title and Welcome message for Main Activity and use it at design time and programmatically to set at runtime.

Acknowledgement: "The content in this module is modifications based on work created and shared by the Android Open-Source Project and used according to terms described in the Creative Commons 2.5 Attribution License."