Android Programming

MODULE TWO

www.teachics.org

teachics.

The Computer Science Learning Platform.

Table of Contents

- Understanding android resources - String resources, Layout resources, Resource reference syntax.
- Defining own resource IDs - Enumerating key android resources,string arrays, plurals, Colour resources, dimension resources, image resources.

Android Resources

- A resource in Android is a file or a value that is bound to an executable application.
- You can change them or provide alternatives without recompiling the application.
- Examples - strings, colors, bitmaps, and layouts.
- You should place each type of resource in a specific subdirectory of your project's ***res/*** directory.
- Resource directories supported inside project res/ directory are,
  - *animator/*
  - *color/*
  - *drawable/*
  - *mipmap/*
  - *layout/*
  - *menu/*
  - *raw/*
  - *values/* - arrays.xml, colors.xml, dimens.xml, strings.xml, styles.xml
  - *xml/*
  - *font/*

String Resources

- A string resource provides text strings for your application with optional text styling and formatting.
- Three types
  - **String** - XML resource that provides a single string.
  - **String Array** - XML resource that provides an array of strings.
  - **Quantity Strings** (Plurals) - XML resource that carries different strings for pluralization.

String

- A single string that can be referenced from the application or from other resource files.
- File location - ***res/values/strings.xml***
- Syntax

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string name="string_name">text_string</string>
</resources>
```

- Example

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

- Applying String to a view - `android:text="@string/hello"`
- Retrieve string in code - `android:text="@string/hello"`

String Arrays

- An array of strings that can be referenced from the application.
- File location - ***res/values/strings.xml***
- Syntax

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string-array name="string_array_name">
        <item>text_string</item>
    </string-array>
</resources>
```

Example

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string-array name="planets_array">
        <item>Mercury</item>
        <item>Venus</item>
        <item>Earth</item>
        <item>Mars</item>
    </string-array>
</resources>
```

- Retrieve string arrays in code

```
Resources res = getResources();
String[] planets = res.getStringArray(R.array.planets_array);
```

Plurals (Quantity Strings)

- The resource plurals is a set of strings.
- These strings are various ways of expressing a numerical quantity, such as how many pages are there in a book.
  - There is 1 egg.
  - There are 2 eggs.
  - There are 0 eggs.
- Full set supported by android is zero, one, two, few, many and other.
- Syntax

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <plurals name="plural_name">
       <item
           quantity=["zero" | "one" | "two" | "few" | "many" | "other"]>text_string</item>
    </plurals>
</resources>
```

Plurals (Quantity Strings)

- File location - ***res/values/strings.xml***

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <plurals name="numberOfSongsAvailable">
          <item quantity="one">%d song found.</item>
          <item quantity="other">%d songs found.</item>
   </plurals>
</resources>
```

- Usage

```
int count = getNumberOfSongsAvailable();
Resources res = getResources();
String songsFound = res.getQuantityString(R.plurals.numberOfSongsAvailable, count, count);
```

String Formatting

When a string contains characters that have special usage in XML or Android, you must escape the characters.

- @ \@
- ? \?
- < <
- & &
- Single quote (')
  - '
  - \'
  - Enclose the entire string in double quotes ("This'll work", for example)
- Double quote (")
  - "
  - \"

String Formatting

You can add styling to your strings with HTML markup.

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string name="welcome">Welcome to <b>Android</b>!</string>
</resources>
```

Layout Resource

- A layout resource defines the architecture for the UI in an Activity or a component of a UI.
- File location - ***res/layout/filename.xml***
- Syntax

```
<?xml version="1.0" encoding="utf-8"?>
<ViewGroup
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@[+][package:]id/resource_name"
    android:layout_height=["dimension" | "match_parent" | "wrap_content"]
    android:layout_width=["dimension" | "match_parent" | "wrap_content"]
    [ViewGroup-specific attributes] >
    <View
        android:id="@[+][package:]id/resource_name"
        android:layout_height=["dimension" | "match_parent" | "wrap_content"]
        android:layout_width=["dimension" | "match_parent" | "wrap_content"]
        [View-specific attributes] >
    </View>
<ViewGroup >
```

Layout Resource

- **<ViewGroup>** - A container for other View elements. Different kinds of ViewGroup objects include LinearLayout, RelativeLayout, and FrameLayout.
- **<View>** - An individual UI component, generally referred to as a "widget". Different kinds of View objects include TextView, Button, and CheckBox.

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
              android:layout_width="match_parent"
              android:layout_height="match_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
```

Color Resource

- A color value defined in XML.
- The color is specified with an RGB value and alpha channel.
- File location - ***res/values/colors.xml***
- The value always begins with a # character and then followed by the Alpha-Red-Green-Blue information in one of the following formats:
  - #RGB
  - #ARGB
  - #RRGGBB
  - #AARRGGBB
- Syntax

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <color name="color_name">hex_color</color>
</resources>
```

Color Resource

- File location - ***res/values/colors.xml***

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
   <color name="opaque_red">#f00</color>
   <color name="translucent_red">#80ff0000</color>
</resources>
```

- Usage

```
Resources res = getResources();
int color = res.getColor(R.color.opaque_red);
```

- This layout XML applies the color to an attribute:

```
<TextView
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:textColor="@color/translucent_red"
    android:text="Hello"/>
```

Dimension Resource

- A dimension value defined in XML.
- A dimension is specified with a number followed by a unit of measure (10px, 2in, 5sp).
- The following units of measure are supported by Android:
  - **in** - inches.
  - **mm** - Millimeters.
  - **px** - pixels (Corresponds to actual pixel on the screen)
  - **pt** - Points (1/72 of an inch based on the physical size of the screen)
  - **dp** - Density-independent Pixels - pixels based on a 160dpi (pixel density per inch) screen (dimensions adjust to screen density)
  - **sp** - Scale-independent Pixels - This is like the dp unit, but it is also scaled by the user's font size preference.
- File location - ***res/values/dimens.xml***

Dimension Resource

- Syntax

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <dimen name="dimension_name">dimension</dimen>
</resources>
```

- Example

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <dimen name="textview_height">25dp</dimen>
    <dimen name="textview_width">150dp</dimen>
    <dimen name="font_size">16sp</dimen>
</resources>
```

- This application code retrieves a dimension

```
Resources res = getResources();
float fontSize = res.getDimension(R.dimen.font_size);
```

- This layout XML applies dimensions to attributes

```
android:textSize="@dimen/font_size"
```

Image Resource

- A drawable resource is a general concept for a graphic that can be drawn to the screen.
- Android supports bitmap files in three formats: .png (preferred), .jpg (acceptable), .gif (discouraged).
- Android creates a Drawable resource for any of these files when you save them in the ***res/drawable/*** directory.
- An image is saved at ***res/drawable/myimage.png***.
  This layout XML applies the image to a View:

```
<ImageView
    android:layout_height="wrap_content"
    android:layout_width="wrap_content"
    android:src="@drawable/myimage" />
```

- The code that retrieves the image as a Drawable:

```
Resources res = getResources();
Drawable drawable = ResourcesCompat.getDrawable(res, R.drawable.myimage, null);
```

Resource Reference Syntax

- Regardless of the type of resource, all Android resources are identified (or referenced) by their IDs in Java source code.
- The syntax that is used to allocate an ID to a resource in the XML file is called resource reference syntax.
- All resource IDs are defined in your project's R class, which the aapt tool automatically generates.
- This resource reference has the following formal structure:

  `@[<package_name>:]<resource_type>/<resource_name>`
- <package_name> is the name of the package in which the resource is located (not required when referencing resources from the same package).
- <resource_type> is the R subclass for the resource type.(drawable, id, layout, array, etc.)
- <resource_name> is either the resource filename without the extension or the android:name attribute value in the XML element (for simple values).

Defining Own Resource IDs

- The general pattern for allocating an ID is either to create a new one or to use the one created by the Android package.
- However, it is possible to create IDs beforehand and use them later in your own packages
- The line `<TextView android:id="@+id/text"/>` indicates that an ID named text is used if it already exists.
- If the ID doesn't exist, a new one is created.
- Predefining an ID

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <item type="id" name="id_name"/>
</resources>
```

- Reusing a Predefined ID

```
<Button android:id="@id/button_ok"/>
```

Enumerating Key Android Resources

Resource Type	Location	Description
Colors	/res/values/any-file	Represents color identifiers pointing to color codes. These resource IDs are exposed in R.java as R.color.*.
Strings	/res/values/any-file	Represents string resources. String resources allow Java-formatted strings and raw HTML in addition to simple strings. These resource IDs are exposed in R.java as R.string.*.
String arrays	/res/values/any-file	Represents a resource that is an array of strings. These resource IDs are exposed in R.java as R.array.*.
Plurals	/res/values/any-file	Represents a suitable collection of strings based on the value of a quantity. The quantity is a number. In various languages, the way you write a sentence depends on whether you refer to no objects, one object, few objects, or many objects. The resource IDs are exposed in R.java as R.plural.*.
Dimensions	/res/values/any-file	Represents dimensions or sizes of various elements or views in Android. Supports pixels, inches, millimeters, density independent pixels, and scale independent pixels. These resource IDs are exposed in R.java as R.dimen.* .
Images	/res/drawable/multiplefiles	Represents image resources. Supported images include .jpg, .gif, .png, and so on. Each image is in a separate file and gets its own ID based on the file name. These resource ids are exposed in R.java as R.drawable.*.
Color drawables	/res/values/any-file also /res/drawable/multiplefiles	Represents rectangles of colors to be used as view backgrounds or general drawables like bitmaps. This can be used in lieu of specifying a single-colored bitmap as a background. In Java, this is equivalent to creating a colored rectangle and setting it as a background for a view.
Arbitrary XML files	/res/xml/*.xml	Android allows arbitrary XML files as resources. These files are compiled by the AAPT compiler. These resource IDs are exposed in R.java as R.xml.*.
Arbitrary raw resources	/res/raw/*.*	Android allows arbitrary noncompiled binary or text files under this directory. Each file gets a unique resource ID. These resource IDs are exposed in R.java as R.raw.* .
Arbitrary raw assets	/assets/*.*/*.*	Android allows arbitrary files in arbitrary subdirectories starting at the /assets subdirectory. These are not really resources, just raw files.

Thank You.

teachics.

The Computer Science Learning Platform.

References

- Komatineni, S. and MacLean, D., 2012. **Pro Android 4**. New York: Apress L.P.
- Android Developers. **App Resources Overview**. [online]
  Available at: <https://developer.android.com/guide/topics/resources/providing-resources>