Accessing Resources

Once you provide a resource in your application (discussed in Providing Resources), you can apply it by referencing its resource ID. All resource IDs are defined in your project's `R` class, which the `aapt` tool automatically generates.

When your application is compiled, `aapt` generates the `R` class, which contains resource IDs for all the resources in your `res/` directory. For each type of resource, there is an `R` subclass (for example, `R.drawable` for all drawable resources) and for each resource of that type, there is a static integer (for example, `R.drawable.icon`). This integer is the resource ID that you can use to retrieve your resource.

Although the `R` class is where resource IDs are specified, you should never need to look there to discover a resource ID. A resource ID is always composed of:

- The *resource type*: Each resource is grouped into a "type," such as `string`, `drawable`, and `layout`. For more about the different types, see Resource Types.
- The *resource name*, which is either: the filename, excluding the extension; or the value in the XML `android:name` attribute, if the resource is a simple value (such as a string).

There are two ways you can access a resource:

- **In code:** Using an static integer from a sub-class of your `R` class, such as:

  ```
  R.string.hello
  ```

  `string` is the resource type and `hello` is the resource name. There are many Android APIs that can access your resources when you provide a resource ID in this format. See Accessing Resources in Code.
- **In XML:** Using a special XML syntax that also corresponds to the resource ID defined in your `R` class, such as:

  ```
  @string/hello
  ```

  `string` is the resource type and `hello` is the resource name. You can use this syntax in an XML resource any place where a value is expected that you provide in a resource. See Accessing Resources from XML.

Quickview

- Resources can be referenced from code using integers from `R.java`, such as `R.drawable.myimage`
- Resources can be referenced from resources using a special XML syntax, such as `@drawable/myimage`
- You can also access your app resources with methods in `Resources`

Key classes

`Resources`

In this document

See also

Providing Resources
Resource Types

Accessing Resources in Code

You can use a resource in code by passing the resource ID as a method parameter. For example, you can set an `ImageView` to use the `res/drawable/myimage.png` resource using `setImageResource()`:

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

You can also retrieve individual resources using methods in `Resources`, which you can get an instance of with `getResources()`.

Access to Original Files

Syntax

Here's the syntax to reference a resource in code:

While uncommon, you might need access your original files and directories. If you do, then saving your files in `res/` won't work for you, because the only way to read a resource from `res/` is with the resource ID. Instead, you can save your resources in the `assets/` directory.

Files saved in the `assets/` directory are *not* given a resource ID, so you can't reference them through the `R` class or from XML resources. Instead, you can query files in the `assets/` directory like a normal file system and read raw data using `AssetManager`.

However, if all you require is the ability to read raw data (such as a video or audio file), then save the file in the `res/raw/` directory and read a stream of bytes using `openRawResource()`.

```
[<package_name>.]R.<resource_type>.<resource_name>
```

- *`<package_name>`* is the name of the package in which the resource is located (not required when referencing resources from your own package).
- *`<resource_type>`* is the `R` subclass for the resource type.
- *`<resource_name>`* is either the resource filename without the extension or the `android:name` attribute value in the XML element (for simple values).

See Resource Types for more information about each resource type and how to reference them.

Use cases

There are many methods that accept a resource ID parameter and you can retrieve resources using methods in `Resources`. You can get an instance of `Resources` with `Context.getResources()`.

Here are some examples of accessing resources in code:

```
// Load a background for the current screen from a drawable resource
getWindow().setBackgroundDrawableResource(R.drawable.my_background_image) ;

// Set the Activity title by getting a string from the Resources object, because
//  this method requires a CharSequence rather than a resource ID
getWindow().setTitle(getResources().getText(R.string.main_title));

// Load a custom layout for the current screen
setContentView(R.layout.main_screen);

// Set a slide in animation by getting an Animation from the Resources object
mFlipper.setInAnimation(AnimationUtils.loadAnimation(this,
        R.anim.hyperspace_in));

// Set the text on a TextView object using a resource ID
TextView msgTextView = (TextView) findViewById(R.id.msg);
msgTextView.setText(R.string.hello_message);
```

Caution: You should never modify the `R.java` file by hand—it is generated by the `aapt` tool when your project is compiled. Any changes are overridden next time you compile.

Accessing Resources from XML

You can define values for some XML attributes and elements using a reference to an existing resource. You will often do this when creating layout files, to supply strings and images for your widgets.

For example, if you add a `Button` to your layout, you should use a string resource for the button text:

```
<Button
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:text="@string/submit" />
```

Syntax

Here is the syntax to reference a resource in an XML resource:

`@[<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
- `<resource_name>` is either the resource filename without the extension or the `android:name` attribute value in the XML element (for simple values).

See Resource Types for more information about each resource type and how to reference them.

Use cases

In some cases you must use a resource for a value in XML (for example, to apply a drawable image to a widget), but you can also use a resource in XML any place that accepts a simple value. For example, if you have the following resource 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>
```

You can use these resources in the following layout file to set the text color and text string:

```
<?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" />
```

In this case you don't need to specify the package name in the resource reference because the resources are from your own package. To reference a system resource, you would need to include the package name. For example:

```
<?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="@android:color/secondary_text_dark"
    android:text="@string/hello" />
```

> **Note:** You should use string resources at all times, so that your application can be localized for other languages. For information about creating alternative resources (such as localized strings), see Providing Alternative Resources.

You can even use resources in XML to create aliases. For example, you can create a drawable resource that is an alias for another drawable resource:

```
<?xml version="1.0" encoding="utf-8"?>
<bitmap xmlns:android="http://schemas.android.com/apk/res/android"
    android:src="@drawable/other_drawable" />
```

This sounds redundant, but can be very useful when using alternative resource. Read more about Creating alias resources.

Referencing style attributes

A style attribute resource allows you to reference the value of an attribute in the currently-applied theme. Referencing a style attribute allows you to customize the look of UI elements by styling them to match standard variations supplied by the current theme, instead of supplying a hard-coded value. Referencing a style attribute essentially says, "use the style that is defined by this attribute, in the current theme."

To reference a style attribute, the name syntax is almost identical to the normal resource format, but instead of the at-symbol (`@`), use a question-mark (`?`), and the resource type portion is optional. For instance:

```
?[<package_name>:][<resource_type>/]<resource_name>
```

For example, here's how you can reference an attribute to set the text color to match the "primary" text color of the system theme:

```
<EditText id="text"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:textColor="?android:textColorSecondary"
    android:text="@string/hello_world" />
```

Here, the `android:textColor` attribute specifies the name of a style attribute in the current theme. Android now uses the value applied to the `android:textColorSecondary` style attribute as the value for `android:textColor` in this widget. Because the system resource tool knows that an attribute resource is expected in this context, you do not need to explicitly state the type (which would be `?android:attr/textColorSecondary`)—you can exclude the `attr` type.

Accessing Platform Resources

Android contains a number of standard resources, such as styles, themes, and layouts. To access these resource, qualify your resource reference with the `android` package name. For example, Android provides a layout resource you can use for list items in a `ListAdapter`:

```
setListAdapter(new ArrayAdapter<String>(this, android.R.layout.simple_list_item_1,
myarray));
```

In this example, `simple_list_item_1` is a layout resource defined by the platform for items in a `ListView`. You can use this instead of creating your own layout for list items. (For more about using `ListView`, see the List View Tutorial.)

← Back to Application Resources   ↑ Go to top