Supporting Different Languages

THIS CLASS TEACHES YOU TO

1. Create Locale Directories and String Files
2. Use the String Resources

YOU SHOULD ALSO READ

- Localization Checklist
- Localization with Resources

It's always a good practice to extract UI strings from your app code and keep them in an external file. Android makes this easy with a resources directory in each Android project.

If you created your project using the Android SDK Tools (read Creating an Android Project (/training/basics/firstapp/creating-project.html)), the tools create a `res/` directory in the top level of the project. Within this `res/` directory are subdirectories for various resource types. There are also a few default files such as `res/values/strings.xml`, which holds your string values.

Create Locale Directories and String Files

To add support for more languages, create additional `values` directories inside `res/` that include a hyphen and the ISO country code at the end of the directory name. For example, `values-es/` is the directory containing simple resourcess for the Locales with the language code "es". Android loads the appropriate resources according to the locale settings of the device at run time.

Once you've decided on the languages you will support, create the resource subdirectories and string resource files. For example:

```
MyProject/
    res/
       values/
           strings.xml
       values-es/
           strings.xml
       values-fr/
           strings.xml
```

Add the string values for each locale into the appropriate file.

At runtime, the Android system uses the appropriate set of string resources based on the locale currently set for the user's device.

For example, the following are some different string resource files for different languages.

English (default locale), `/values/strings.xml`:

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string name="title">My Application</string>
    <string name="hello_world">Hello World!</string>
</resources>
```

Spanish, `/values-es/strings.xml`:

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string name="title">Mi Aplicación</string>
    <string name="hello_world">Hola Mundo!</string>
</resources>
```

French, `/values-fr/strings.xml`:

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string name="title">Mon Application</string>
    <string name="hello_world">Bonjour le monde !</string>
</resources>
```

Note: You can use the locale qualifier (or any configuration qualifer) on any resource type, such as if you want to provide localized versions of your bitmap drawable. For more information, see Localization (/guide/topics/resources/localization.html).

Use the String Resources

You can reference your string resources in your source code and other XML files using the resource name defined by the `<string>` element's `name` attribute.

In your source code, you can refer to a string resource with the syntax `R.string.<string_name>`. There are a variety of methods that accept a string resource this way.

For example:

```
// Get a string resource from your app's Resources
String hello = getResources().getString(R.string.hello_world);

// Or supply a string resource to a method that requires a string
TextView textView = new TextView(this);
textView.setText(R.string.hello_world);
```

In other XML files, you can refer to a string resource with the syntax `@string/<string_name>` whenever the XML attribute accepts a string value.

For example:

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