Localization

In this unit, you will create a "Hello World" app that can be localized to other languages, as well as pseudolocalized to LTR (left-to-right) and RTL (right-to-left) pseudolocales.

Notes

You might be surprised at how easy it is to create a localized application on the Android platform. Android Studio provides all the tools you need, including an emulator if you don't have an Android device. This unit assumes you have Android Studio installed.

References

- Getting Started | Android Developers

Build generic "Hello World" app

The first step to create a localizable app for our purposes is to follow Google's instructions to create a simple "Hello World" app with Android Studio.

Notes

For the purposes of this unit, launching your app in an emulator is more useful than installing it on a phone or tablet. Android Studio comes with an emulator module built in.

References

- Building Your First App | Android Developers

Make localizable "hello" string

You are going to replace the generic "Hello, world!" in the beginner app with a custom string. The text field on your app's layout should contain a reference to a string in the default `strings.xml` file. You can name the new string something like `hello`, and it can contain anything you can distinguish from the old string, such as "Hello, hello!" – just so you can distinguish it from the old string when you run your app again.

References

- Providing Resources | Android Developers
- Accessing Resources | Android Developers

Make your app pseudolocalizable

To pseudolocalize your app, you must set the `pseudoLocalesEnabled` flag in your `build.gradle` file. Fortunately, this is a simple procedure.

References

- Pseudolocalization: Visiting Android's Bizarro World

Add two pseudolocales

Add the pseudolocales `en_XA` and `ar_XB` to the emulator with the **Custom Locales** application.

Notes

The `en_XA` pseudolocale is a LTR locale, and the `ar_XB` pseudolocale is RTL.

Test pseudolocalization

If you've done everything so far correctly, all you have to do now to see your two custom pseudolocales in action is select them (one at a time) and launch your app.

Localize app with another language

Completing the localization process is only slightly more complex. You'll need to make a copy of the `values` folder with the name of the new locale appended, and then copy the default `strings.xml` file into it, translating each string it contains into your target language.

Figure 1: Creating a custom pseudolocale

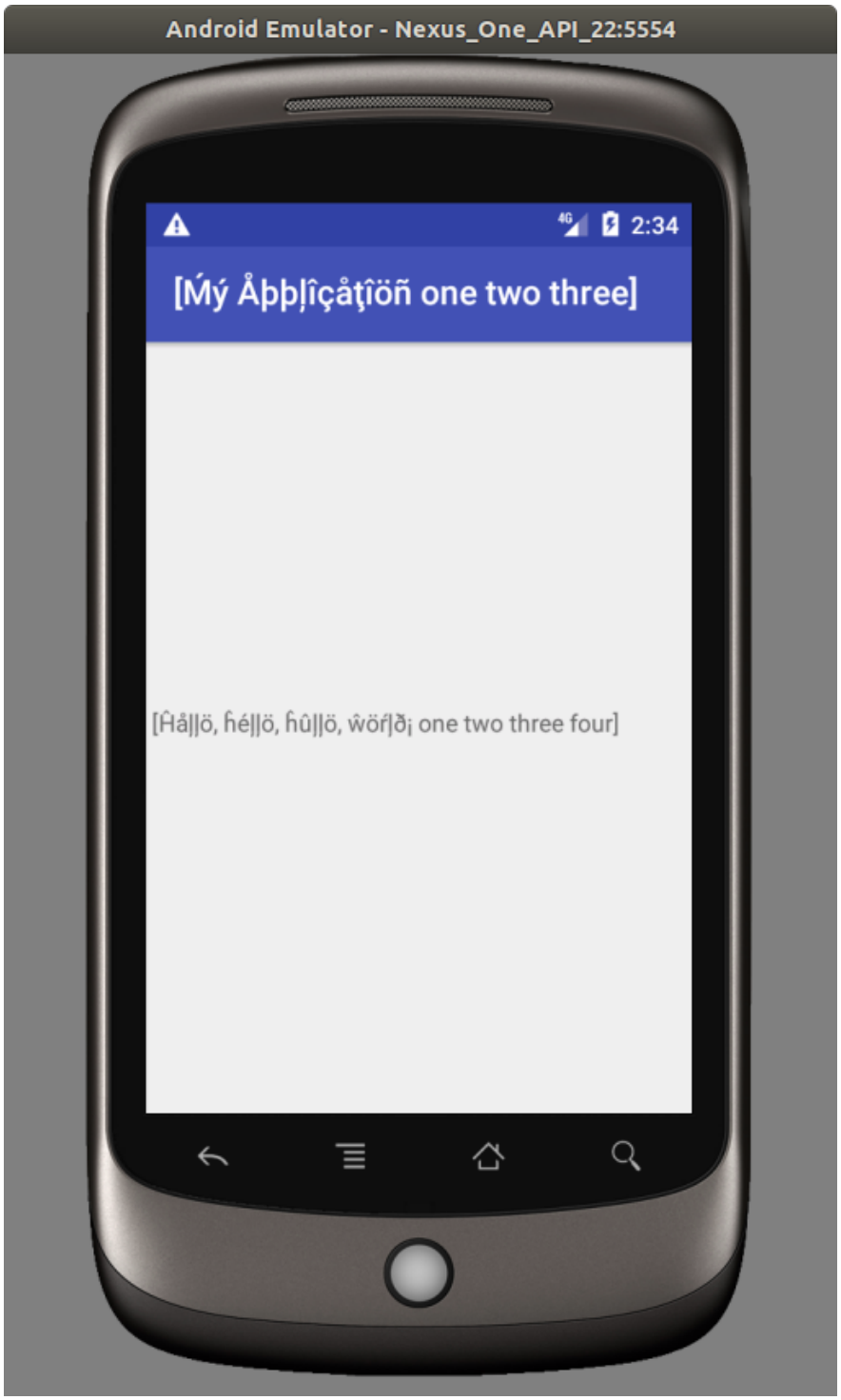

Figure 2: The en_XA pseudolocale

Figure 3: The ar_XB pseudolocale

Notes

If you do not translate a string, the corresponding default string will be used. **Important:** If there is no corresponding default string, your app will crash when it tries to display the missing string.

References

- Supporting Different Languages and Cultures | Android Developers

Test localization

Much as with the pseudolocales above, you can test your app's new locale simply by selecting it in the **Custom Locales** app and restarting your application.

Notes

This has been a short demonstration of how software is localized on the Android. Localizing a complete application would be more complex, of course, but the steps above provide the basics to do so.

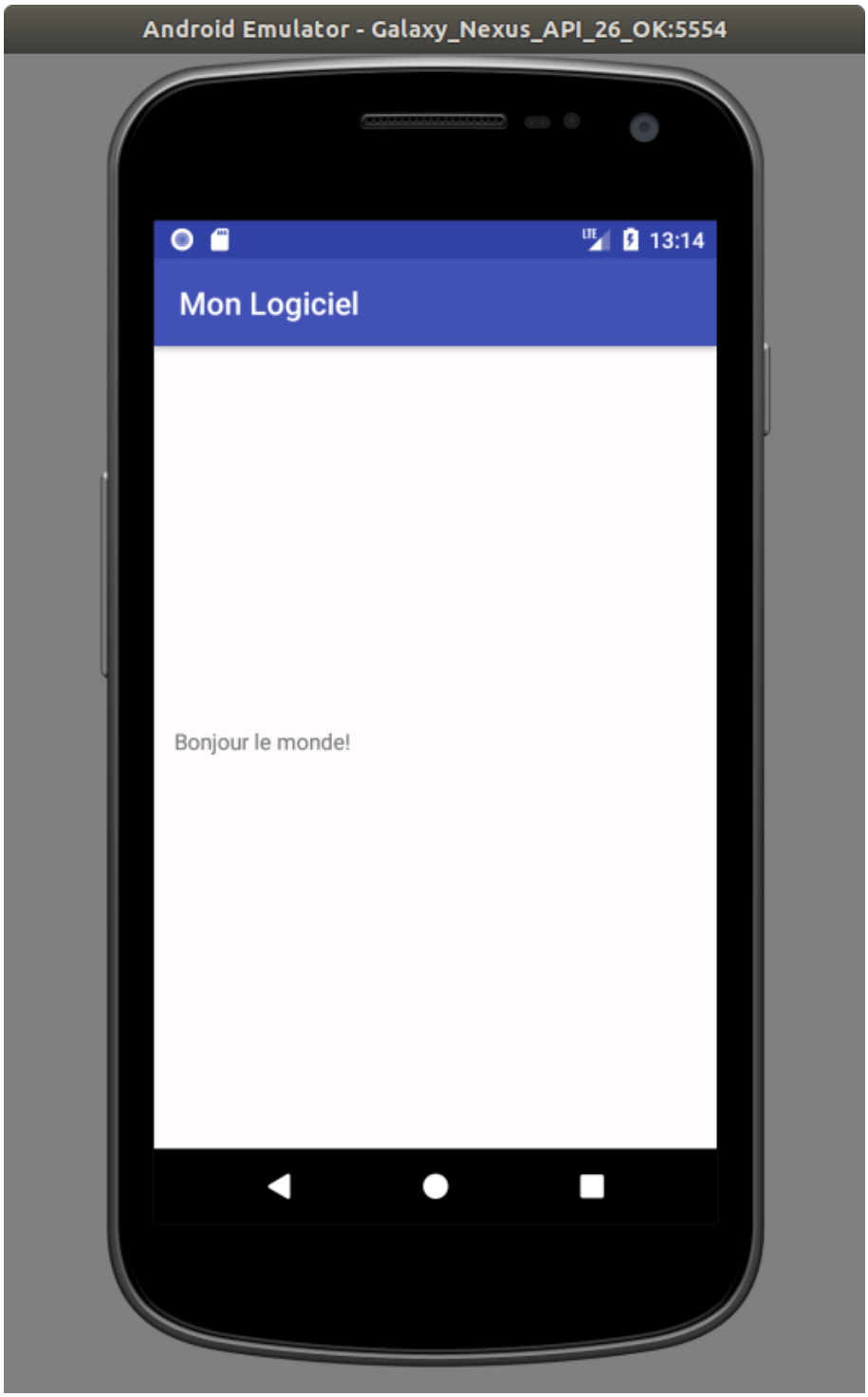

Figure 4: The fr_FR locale