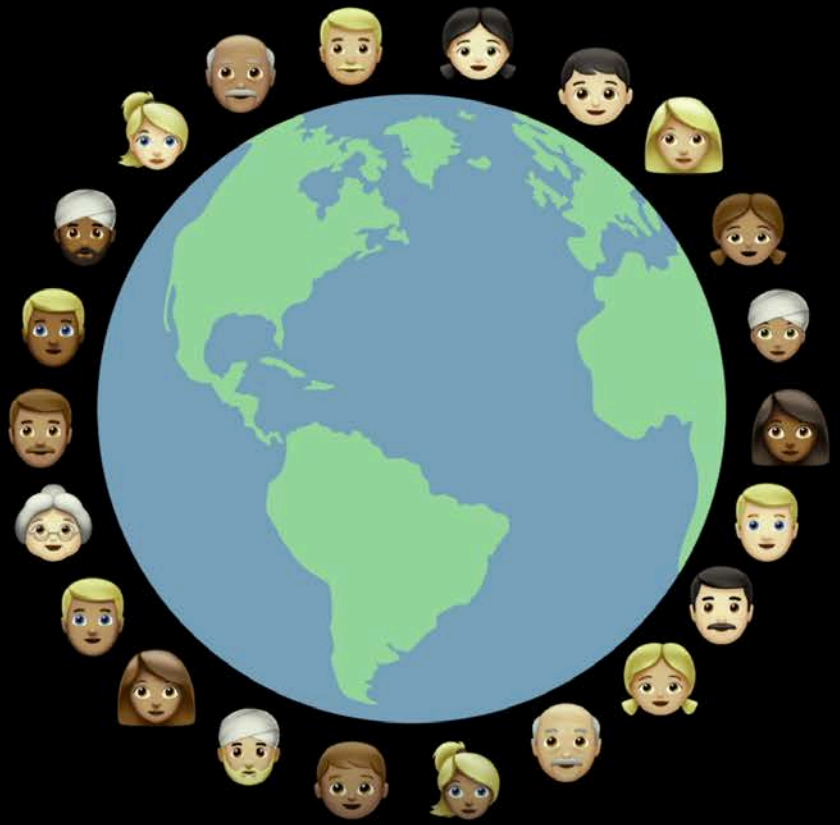
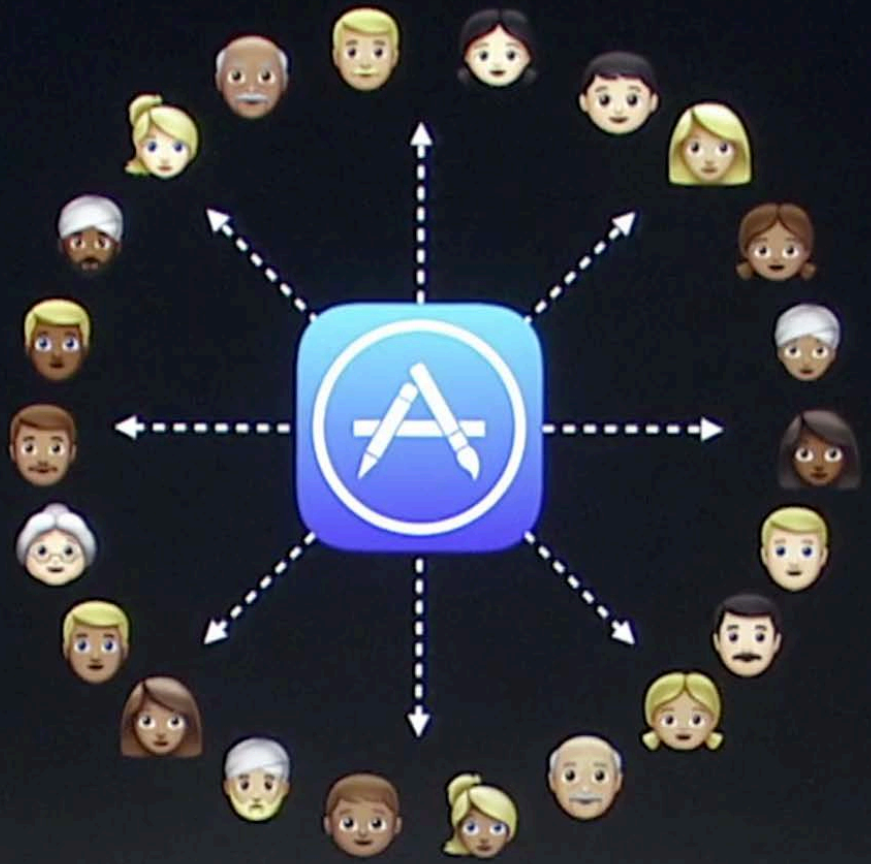
Internationalization
Localization



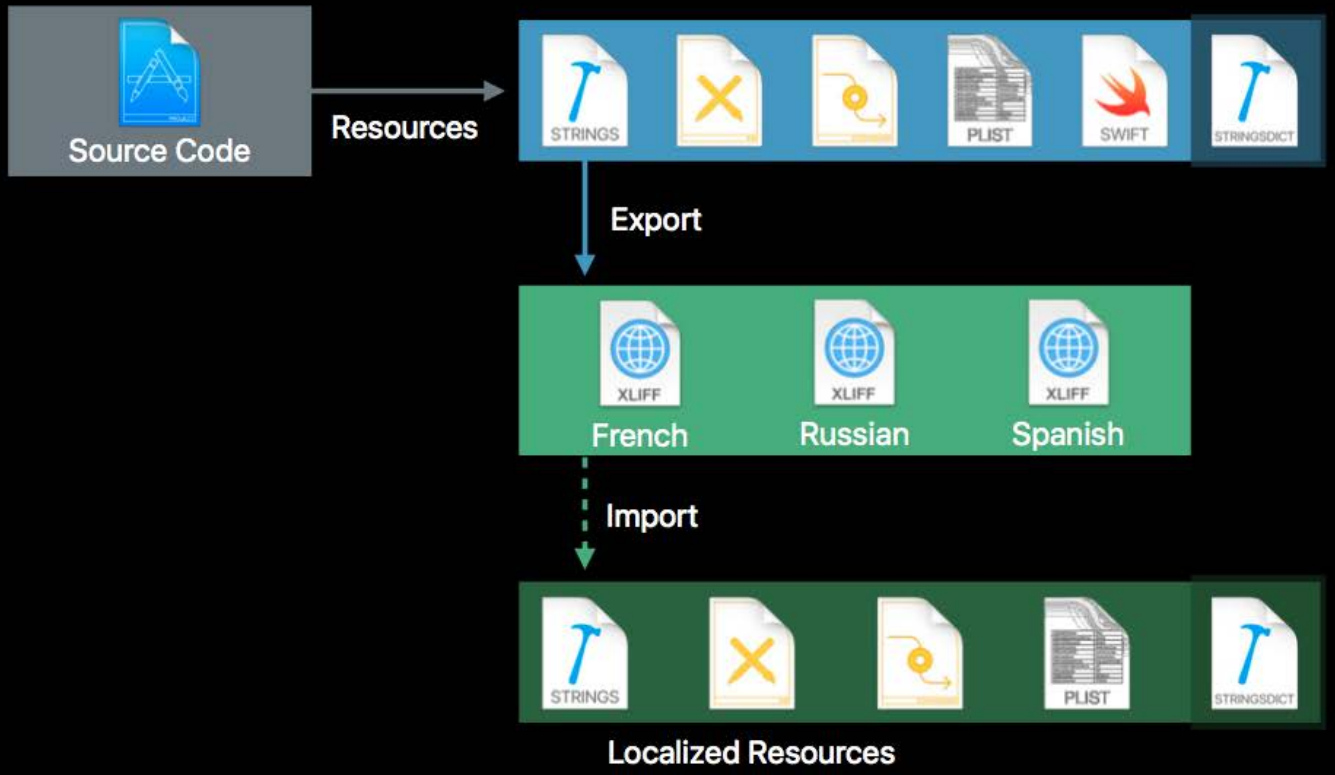


Internationalization

Providing local experiences for global users

- **Strings management**
- **Formatting**
- **User Interface**

Localization Process



Internationalization

Internationalization (i18n) – the dynamic conversion of native cultural information (currency, number and date formats, language, etc.)

Localization (L10n) - is the process of providing appropriate data in the app based on the user's Language and Region Format setting. (German)

Localization APIs
Easy to use in iOS
No precompillation is necessary

- Internationalizing the World Trotter App
- Localizing in Spanish

Internationalizing the app

NumberFormatter (Celsius label in ConversionViewController)

has

Locale property
displays different regions symbols, dates, decimals, metric system, etc.

An instance of locale represents one region's settings for all the these variables.
Once you have that instance you can ask questions"
- Does this region use metric system?
- What currency symbol for this region?

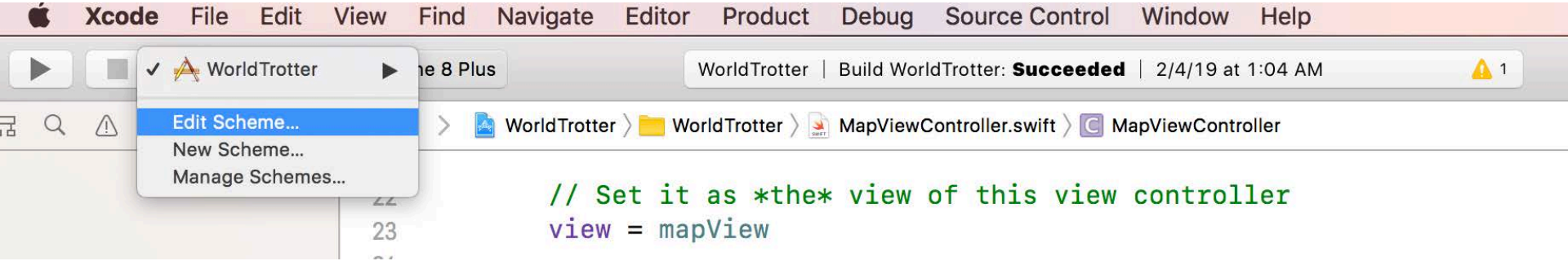
let curentLocale = Locale.current
let isMetric = curentLocle.usesMetricSystem
let currentSymbol = currentLoale.currencySymbol

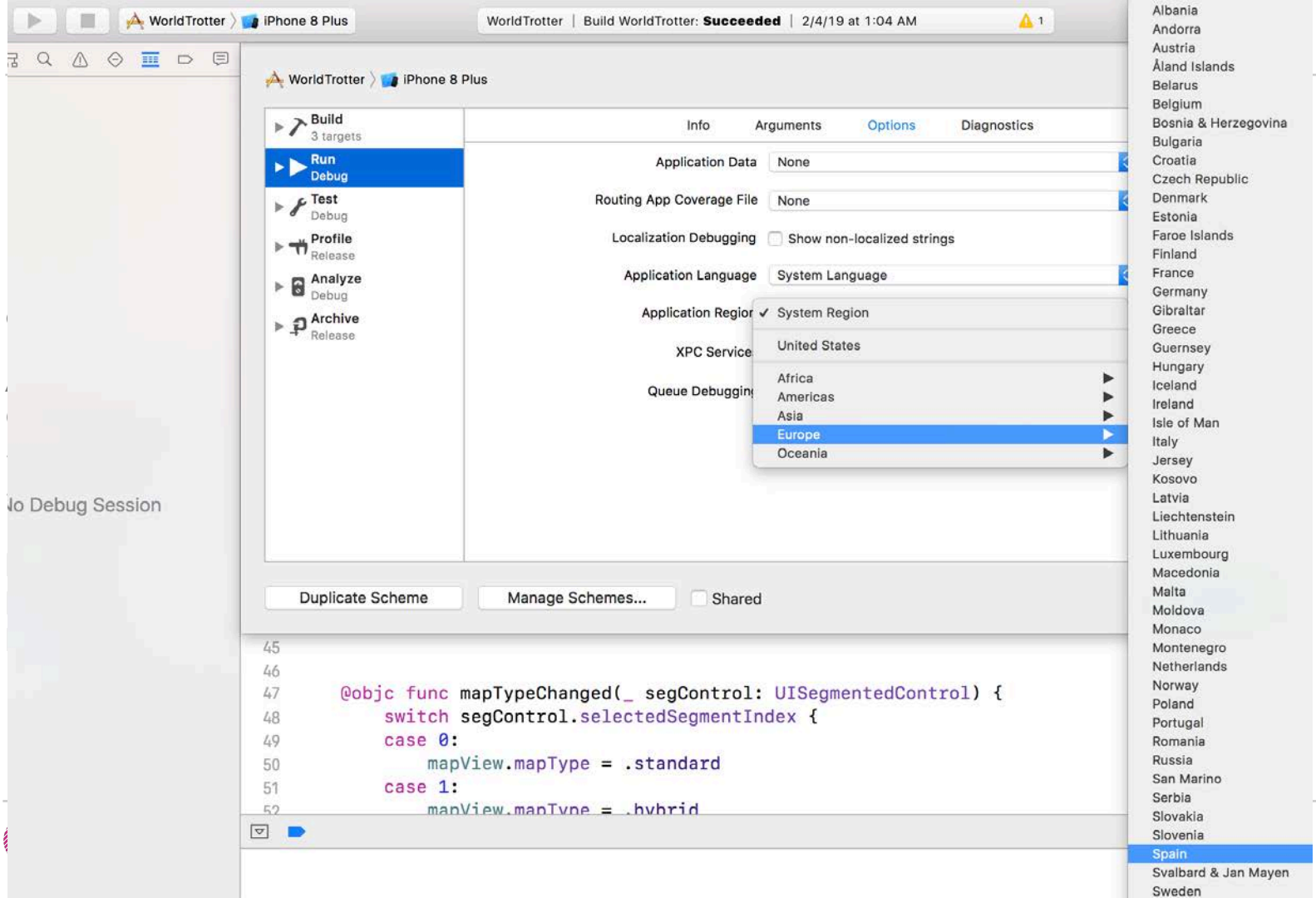
Internationalizing the app

Edit Scheme > options > App Regions > Europe > Spain

Build and run.
Notice the difference in Celsius / Fahrenheit decimals

In Spain decimal separator is comma instead of period
Type in decimal separators and the app will allow it, it only checks for a period instead of using locale-specific decimal separator.

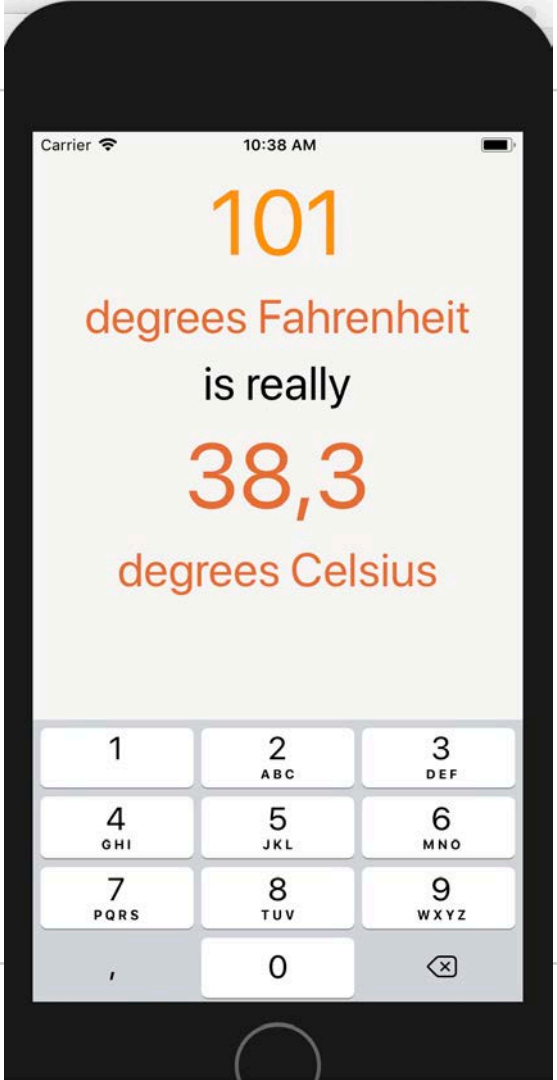



WorldTrotter 〉 iPhone 8 Plus
WorldTrotter | Build WorldTrotter: Succeeded | 2/4/19 at 1:04 AM
1
WorldTrotter 〉 iPhone 8 Plus
Build
3 targets
Run
Debug
Test
Debug
Profile
Release
Analyze
Debug
Archive
Release
Info
Arguments
Options
Diagnostics
Application Data
None
Routing App Coverage File
None
Localization Debugging
Show non-localized strings
Application Language
System Language
Application Regior
System Region
United States
XPC Service
Queue Debuggin
Africa
Americas
Asia
Europe
Oceania
Duplicate Scheme
Manage Schemes...
Shared
No Debug Session
45
46
47 @objc func mapTypeChanged(_ segControl: UISegmentedControl) {
48 switch segControl.selectedSegmentIndex {
49 case 0:
50 mapView.mapType = .standard
51 case 1:
52 mapView.mapType = .hybrid
Albania
Andorra
Austria
Åland Islands
Belarus
Belgium
Bosnia & Herzegovina
Bulgaria
Croatia
Czech Republic
Denmark
Estonia
Faroe Islands
Finland
France
Germany
Gibraltar
Greece
Guernsey
Hungary
Iceland
Ireland
Isle of Man
Italy
Jersey
Kosovo
Latvia
Liechtenstein
Lithuania
Luxembourg
Macedonia
Malta
Moldova
Monaco
Montenegro
Netherlands
Norway
Poland
Portugal
Romania
Russia
San Marino
Serbia
Slovakia
Slovenia
Spain
Svalbard & Jan Mayen
Sweden

Internationalizing the app

Build and run.
Notice the difference in Celsius / Fahrenheit decimals

In Spain decimal separator is comma instead of period
Type in decimal separators and the app will allow it,
it only checks for a period instead of using
locale-specific decimal separator.



Internationalizing the app

Open ConversionController.swift update textfield function:

```
func textField(_ textField: UITextField,
shouldChangeCharactersIn range: NSRange, replacementString string: String) -> Bool {

let existingTextHasDecimalSeparator = textField.text?.range(of: ".")  let
replacementTextHasDecimalSeparator = string.range(of: ".")

        let currentLocale = Locale.current
        let decimalSeparator = currentLocale.decimalSeparator ?? "."
        let existingTextHasDecimalSeparator
            = textField.text?.range(of: decimalSeparator)
        let replacementTextHasDecimalSeparator = string.range(of: decimalSeparator)
    if existingTextHasDecimalSeparator != nil,
Repla
cementTextHasDecimalSeparator != nil {
return false
} else {
return true
}
```

Internationalizing the app

```
15
16      func textField(_ textField: UITextField, shouldChangeCharactersIn range: NSRange, replacementString string: String)
            -> Bool {
17
18          /*print("Current text: \(textField.text)")
19          print("Replacement text: \(string)")
20          return true*/
21       // let existingTextHasDecimalSeparator = textField.text?.range(of: ".")
22       // let replacementTextHasDecimalSeparator = string.range(of: ".")
23
24          let currentLocale = Locale.current
25          let decimalSeparator = currentLocale.decimalSeparator ?? "."
26          let existingTextHasDecimalSeparator
27              = textField.text?.range(of: decimalSeparator)
28          let replacementTextHasDecimalSeparator = string.range(of: decimalSeparator)
29
30
31
32          if existingTextHasDecimalSeparator != nil && replacementTextHasDecimalSeparator != nil {
33              return false
34          } else {
35
36              return true
37          }
38      }
39
```

~~return true~~

}

Internationalizing the app

The app now should allow you to type in multiple decimal separators independent of user's region. But if you type in a number which does not contain a period, the conversion is not happening and ??? Is displayed.

The initializer does not know to to handle something other than a period as decimal separator.

You can fix it using **NumberFormatter** class.

```
~~if let text = textField.text, let value = Double(text) {~~
~~fahrenheitValue = Measurement(value: value, unit: .fahrenheit)~~
 if let text = textField.text, let number = numberFormatter.number(from: text) {
fahrenheitValue = Measurement(value: number.doubleValue, unit: .fahrenheit)
} else {
fahrenheitValue = nil
}
}
```

Internationalizing the app

Build and run

```
    @IBAction func fahrenheitFieldEditingChanged(_ textField: UITextField) {
        //celsiusLabel.text = textField.text
        /*if let text = textField.text, !text.isEmpty {
            celsiusLabel.text = text
        } else {
            celsiusLabel.text = "???"
        }*/
        //if let text = textField.text, let value = Double(text) {
       //     fahrenheitValue = Measurement(value: value, unit: .fahrenheit)

        if let text = textField.text, let number = numberFormatter.number(from: text) {
                fahrenheitValue = Measurement(value: number.doubleValue, unit: .fahrenheit)


        } else {
            fahrenheitValue = nil
        }
}
```

Internationalizing the app

Uses **NumberFormatter** method **number(from:)**
To convert strong into a number.

Because the number formatter is aware of the locale, it is able to convert the string into a number.

If the string contains a valid number, the method returns an instance of NSNumber.

NSNumber is a class that can represent a variety of number types, including Int, Float, Double, and more.

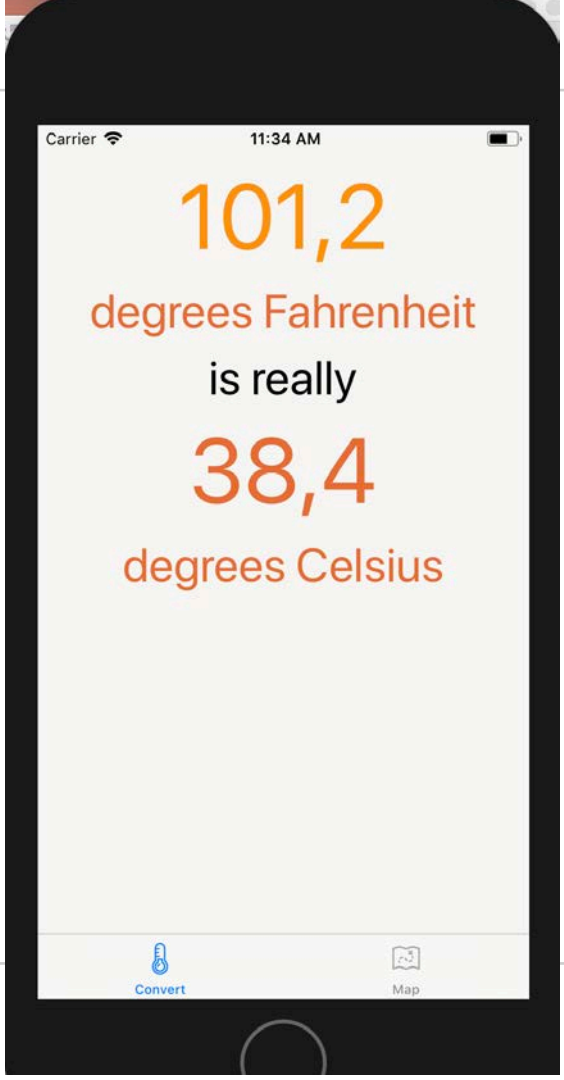
You can ask an instance of NSNumber for its value represented as one of those values.

You are doing that here to get the doubleValue of the number.

Internationalizing the app

Uses **NumberFormatter** method **number(from:)**
To convert strong into a number.

Now that you are converting the string
in a locale-independent way, the text field's value
is properly converted to its Celsius value.



Base Internationalization

Localization usually involves either generating multiple copies of resources (like images, sounds, and interface files) for different regions and languages or creating and accessing strings tables (which you will see later in the chapter) to translate text into different languages.

When you build a target in Xcode, an application bundle is created. All of the resources that you added to the target in Xcode are copied into this bundle along with the executable itself. This bundle is represented at runtime by an instance of Bundle known as the main bundle. Many classes work with the Bundle to load resources.

Localizing a resource puts another copy of the resource in the application bundle. These resources are organized into language-specific directories, known as lproj directories. Each one of these directories is the name of the localization suffixed with lproj. For example, the American English localization is en_US, where en is the English language code and US is the United States of America region code, so the directory for American English resources is en_US.lproj. (The region can be omitted if you do not need to make regional distinctions in your resource files.) These language and region codes are standard on all platforms, not just iOS.

Base Internationalization

When a bundle is asked for the path of a resource file, it first looks at the root level of the bundle for a file of that name. If it does not find one, it looks at the locale and language settings of the device, finds the appropriate lproj directory, and looks for the file there.

Thus, just by localizing resource files, your application will automatically load the correct file.

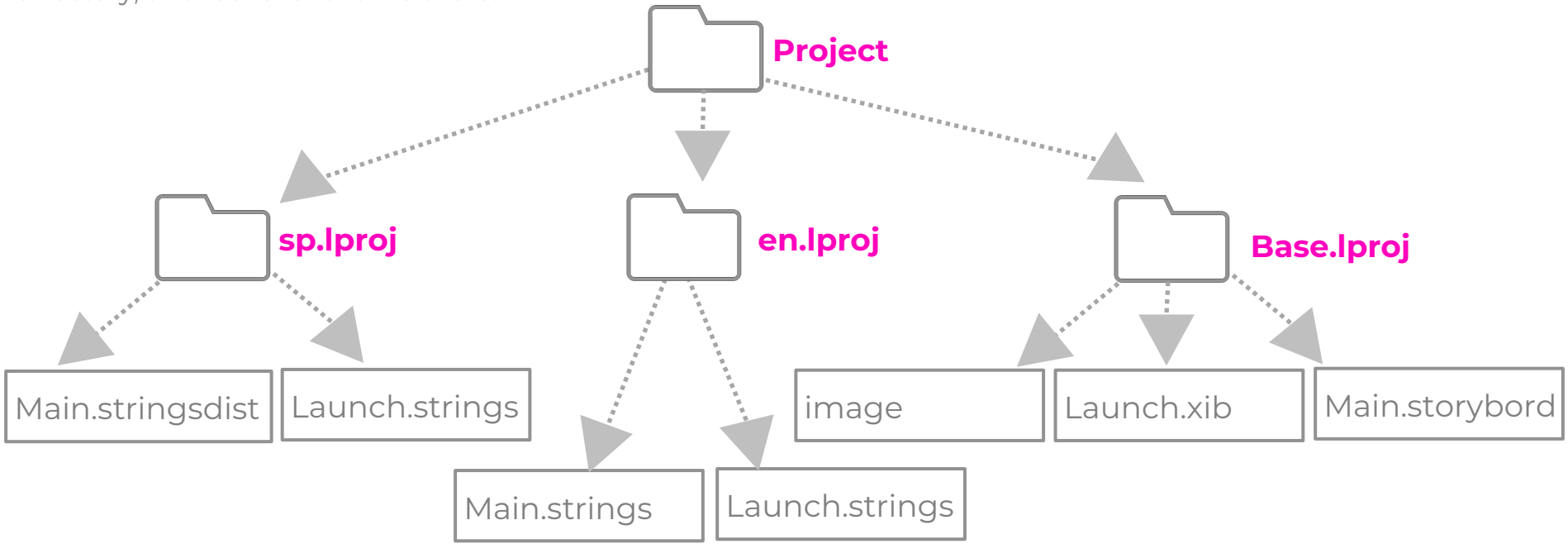
One option for localizing resource files is to create separate storyboard files and manually edit each string in each file. However, this approach does not scale well if you are planning multiple localizations. What happens when you add a new label or button to your localized storyboard? You have to add this view to the storyboard for every language. Not fun.

To simplify the process of localizing interface files, Xcode has **base internationalization.**

Base internationalization creates the Base.lproj directory, which contains the main interface files. Localizing individual interface files can then be done by creating just the Localizable.strings files. It is still possible to create the full interface files, in case localization cannot be done by changing strings alone.

Localized resources structure

Base internationalization creates the Base.lproj directory, which contains the main interface files. When a bundle is asked for the path of a resource file, it first looks at the root level of the bundle for a file of that name. If it does not find one, it looks at the locale and language settings of the device, finds the appropriate lproj directory, and looks for the file there.



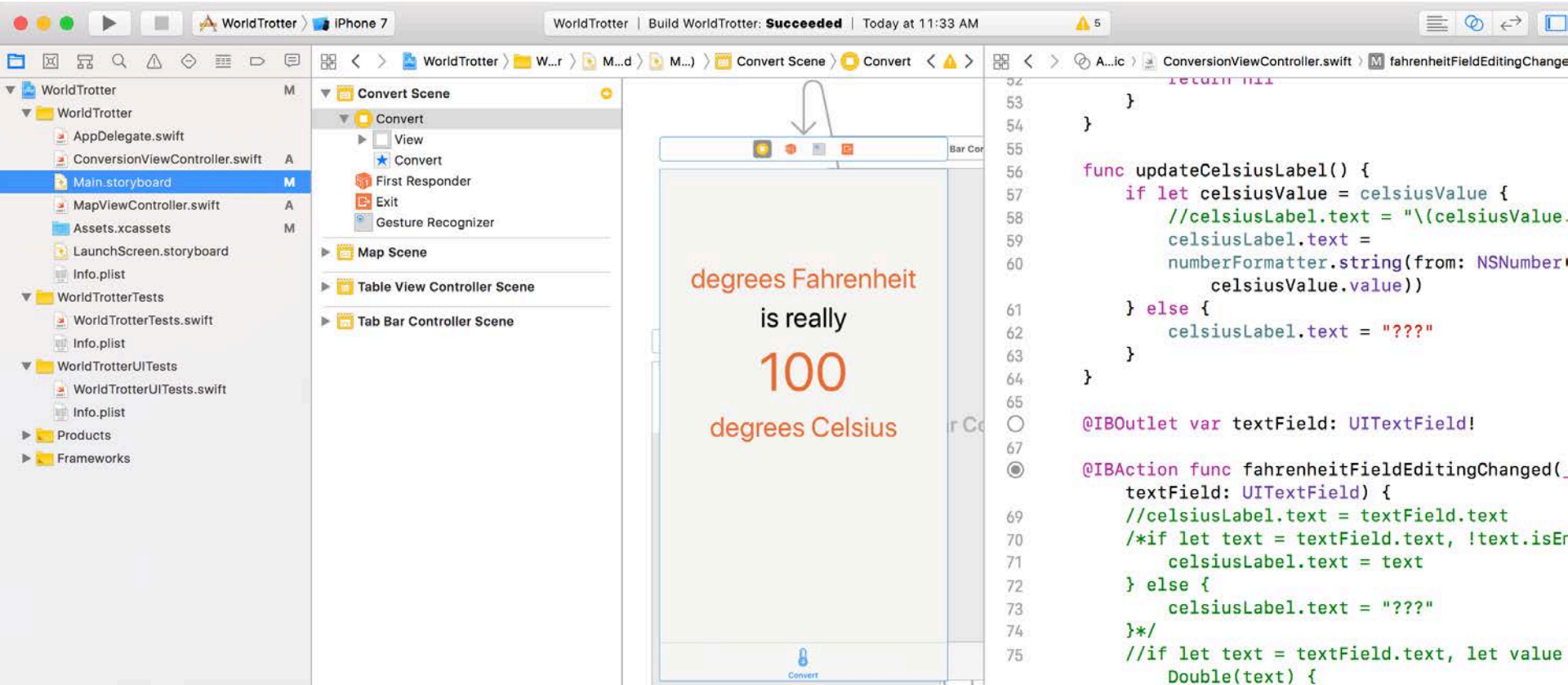
Base Internationalization

Apple's Localization video

https://developer.apple.com/videos/play/wwdc2017/401

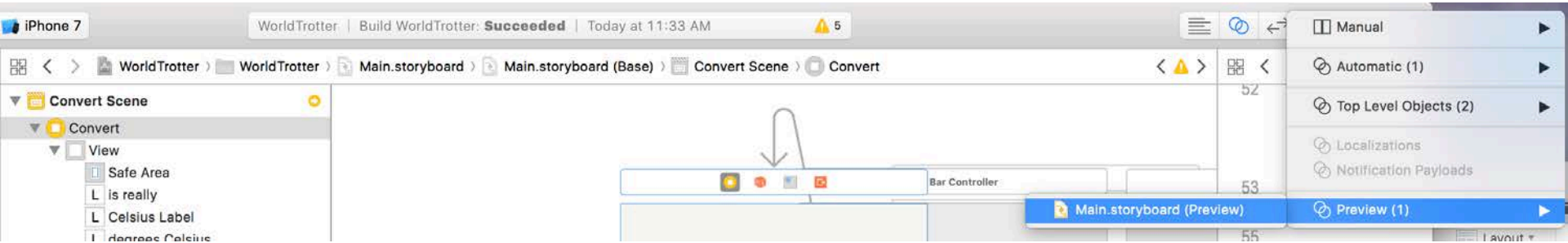
Base Internationalization

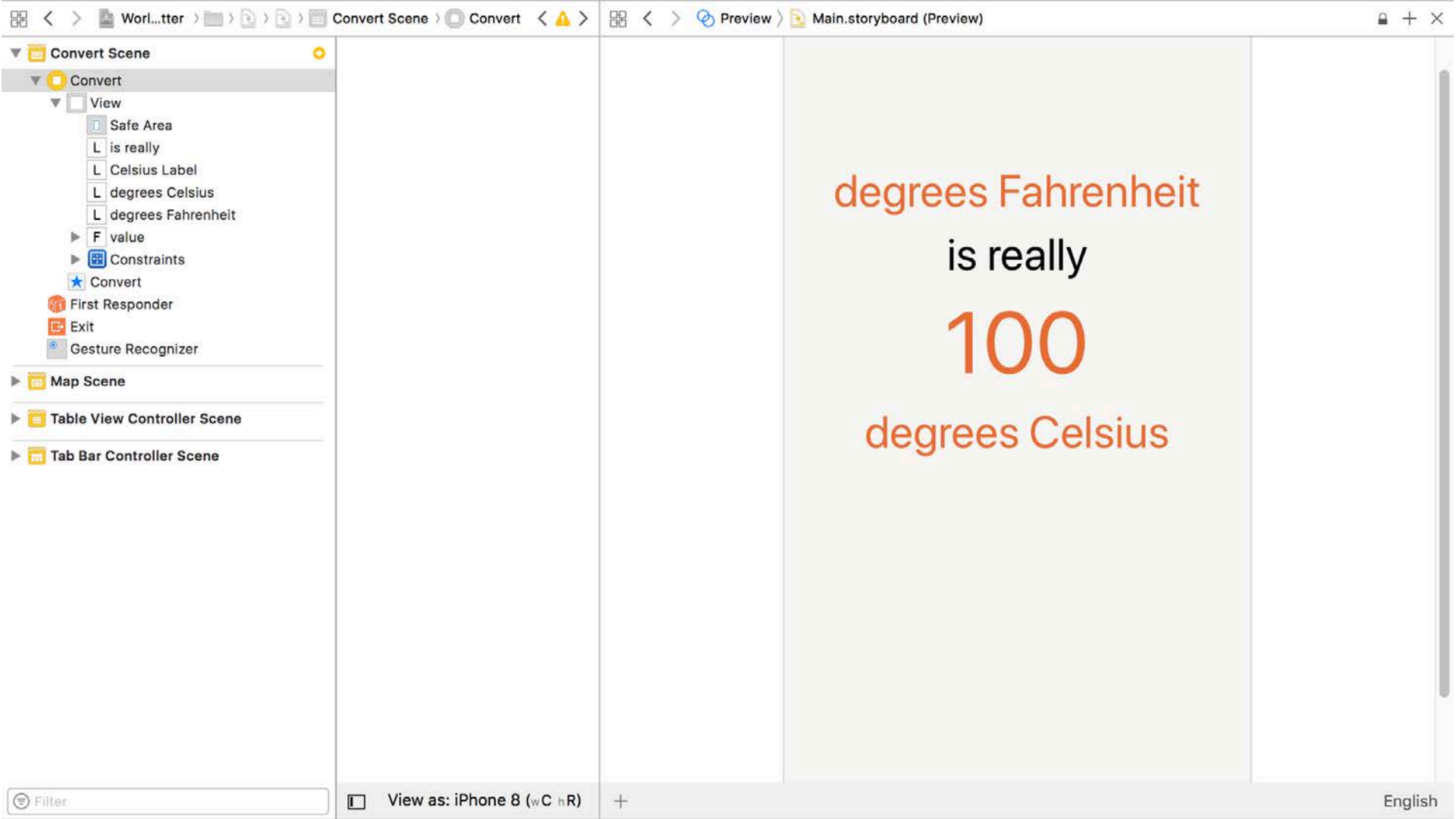
Select Main.storyboard
Option+Command+return to open Preview



Base Internationalization

Select Main.storyboard
Option+Command+return to open Preview




Worl...tter
Convert Scene
Convert
Preview
Main.storyboard (Preview)
Convert Scene
Convert
View
Safe Area
is really
Celsius Label
degrees Celsius
degrees Fahrenheit
value
Constraints
Convert
First Responder
Exit
Gesture Recognizer
Map Scene
Table View Controller Scene
Tab Bar Controller Scene
Filter
View as: iPhone 8 (wC hR)
degrees Fahrenheit
is really
100
degrees Celsius
English

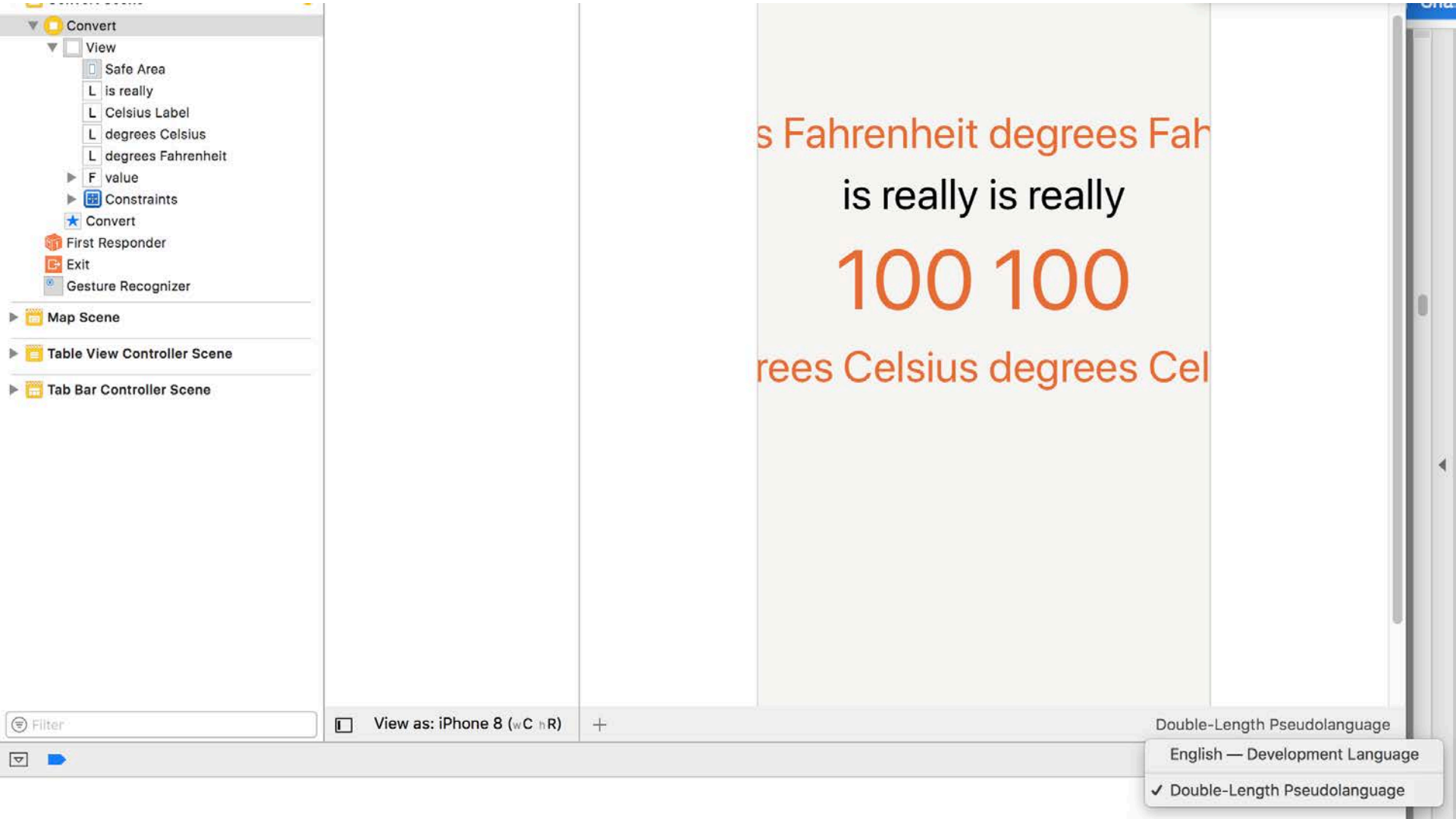
Base Internationalization

The + button on the left side allows you to add additional screen sizes to the preview canvas. The button on the right side allows you to select a language to preview this interface in.

Xcode supplies the built-in pseudolanguage to help you internationalize apps before receiving translations for all of strings and assets.

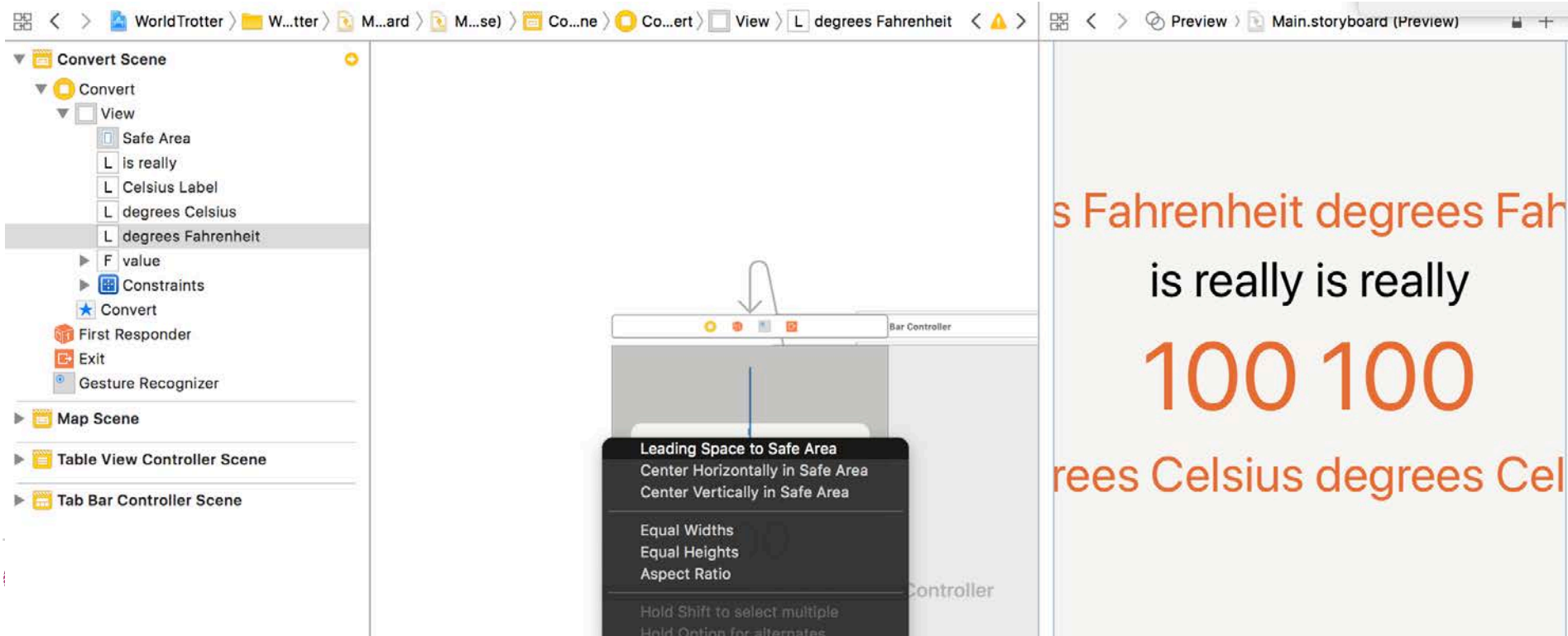
Pseudolanguage mimics languages that are more verbose by repeating whatever text string is in the text element. So, for example, “is nice” becomes “is nice is nice.”

Select Pseudolanguage


Convert
View
Safe Area
is really
Celsius Label
degrees Celsius
degrees Fahrenheit
value
Constraints
Convert
First Responder
Exit
Gesture Recognizer
Map Scene
Table View Controller Scene
Tab Bar Controller Scene
Filter
s Fahrenheit degrees Fah
is really is really
100 100
rees Celsius degrees Cel
View as: iPhone 8 (wC hR)
Double-Length Pseudolanguage
English — Development Language
Double-Length Pseudolanguage

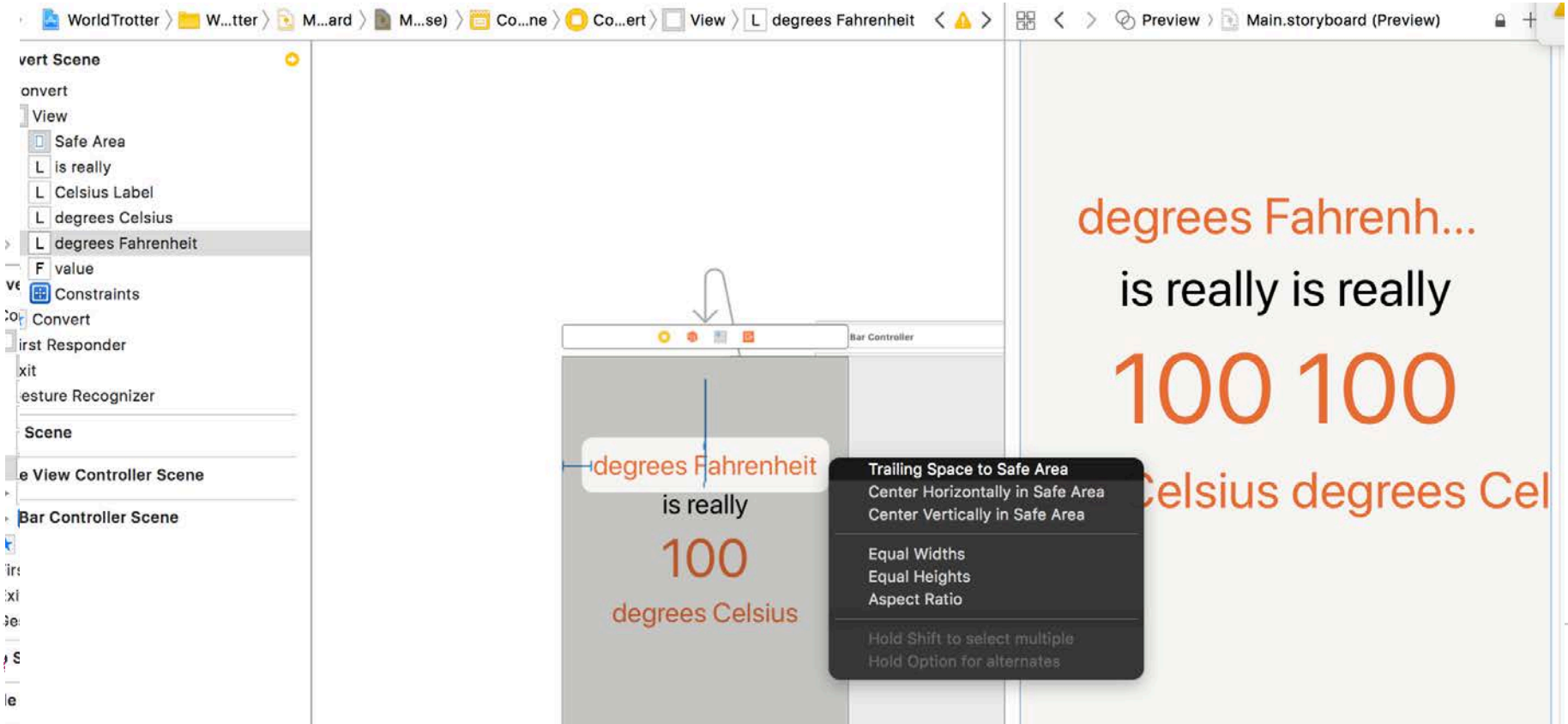
Base Internationalization

select the degrees Fahrenheit label. Control-drag from the label to the left side of the superview. When you do, a context-sensitive pop-up will appear giving you the constraints that make sense for this direction. Select Leading Space to Safe Area (in the book ”to Container Margin”)



Base Internationalization

Control-drag from the degrees Fahrenheit label to the right side of the superview and select Trailing Space to Safe Area (in the book “to Container Margin”.)



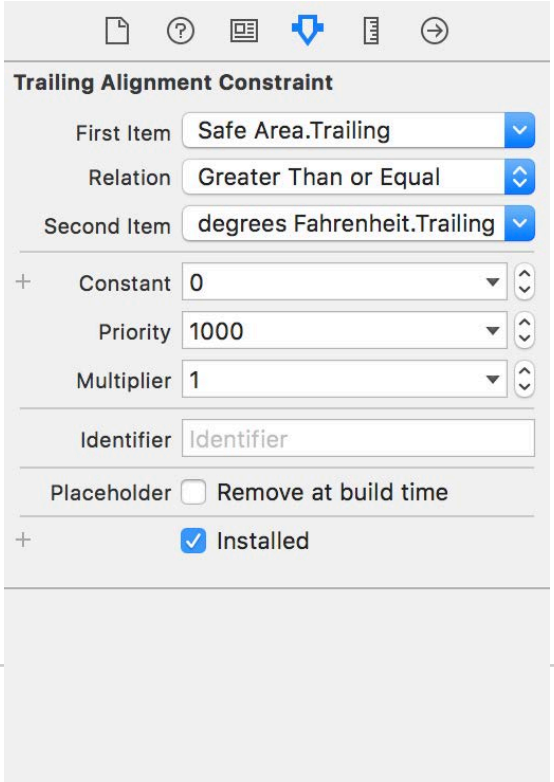
Base Internationalization

Select the leading constraint by clicking on the I-bar to the left of the label.
Open its attributes inspector and change the Relation to Greater Than or Equal and the Constant to 0.

Do the same for the trailing constraint.

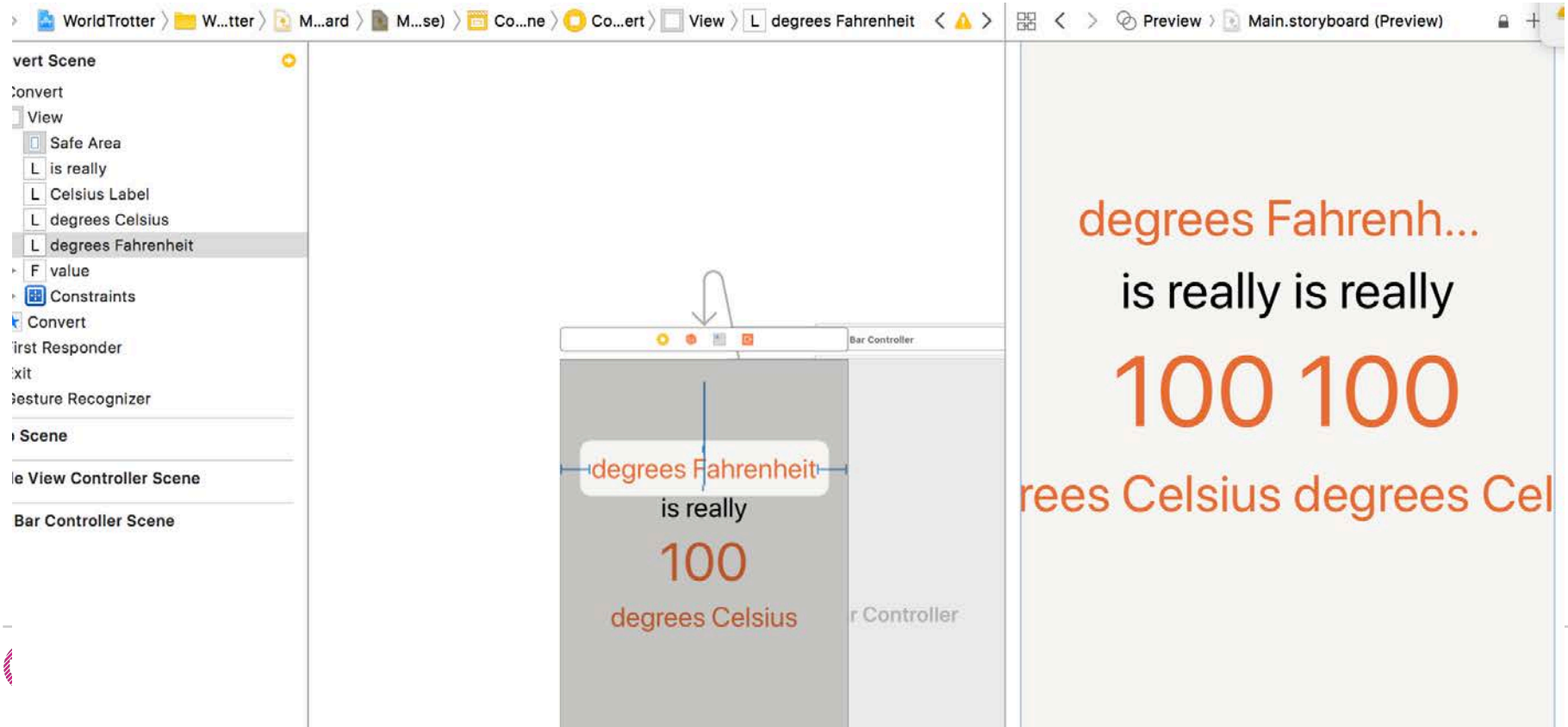
Select the label and open its attributes inspector.
Change the Lines count to 0.

Now take a look at the preview assistant;
the label is no longer being truncated
and instead the text flows to a second line.



Base Internationalization

Control-drag from the degrees Fahrenheit label to the right side of the superview and select Trailing Space to Safe Area (in the book “to Container Margin”.)



Base Internationalization

Repeat steps for each label:

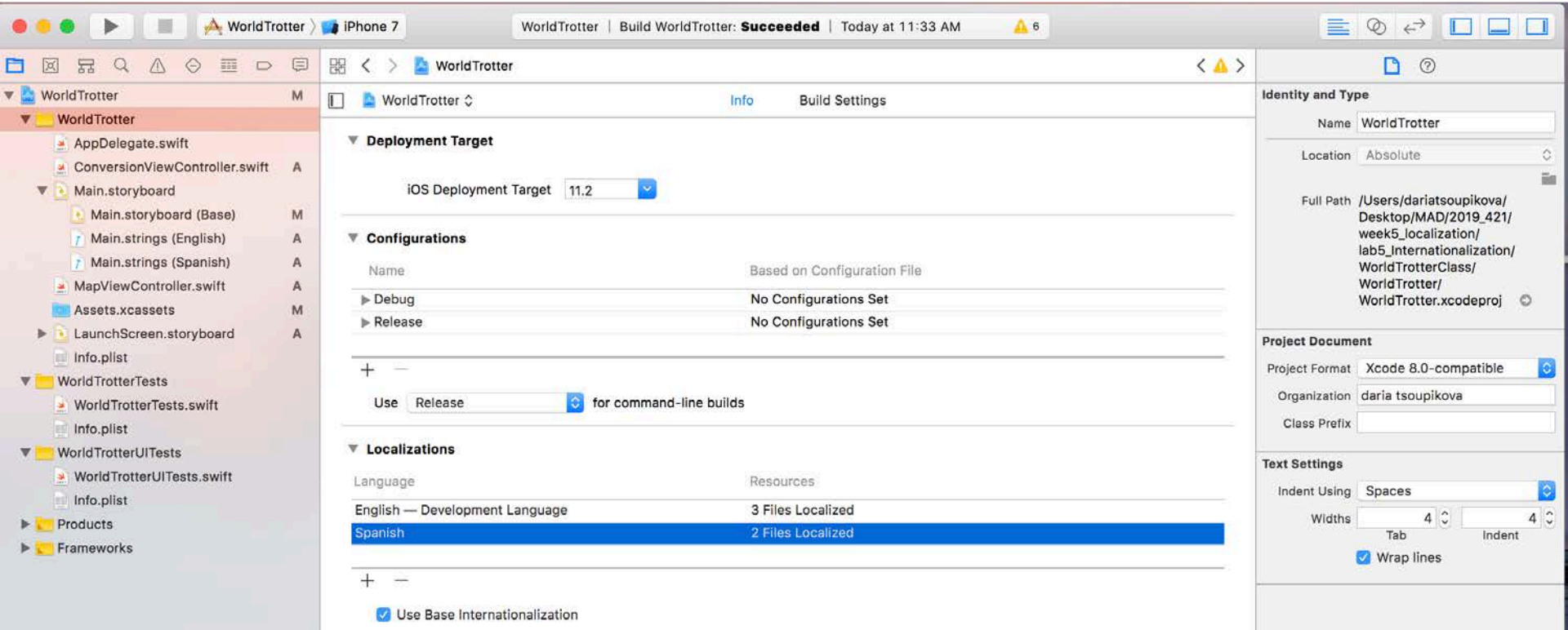
- Add a leading and trailing constraint to each label.
- Set the constraints' relation to Greater Than or Equal and the constant to 0.
- Change the label's line count to 0.

Close the Preview window after done.

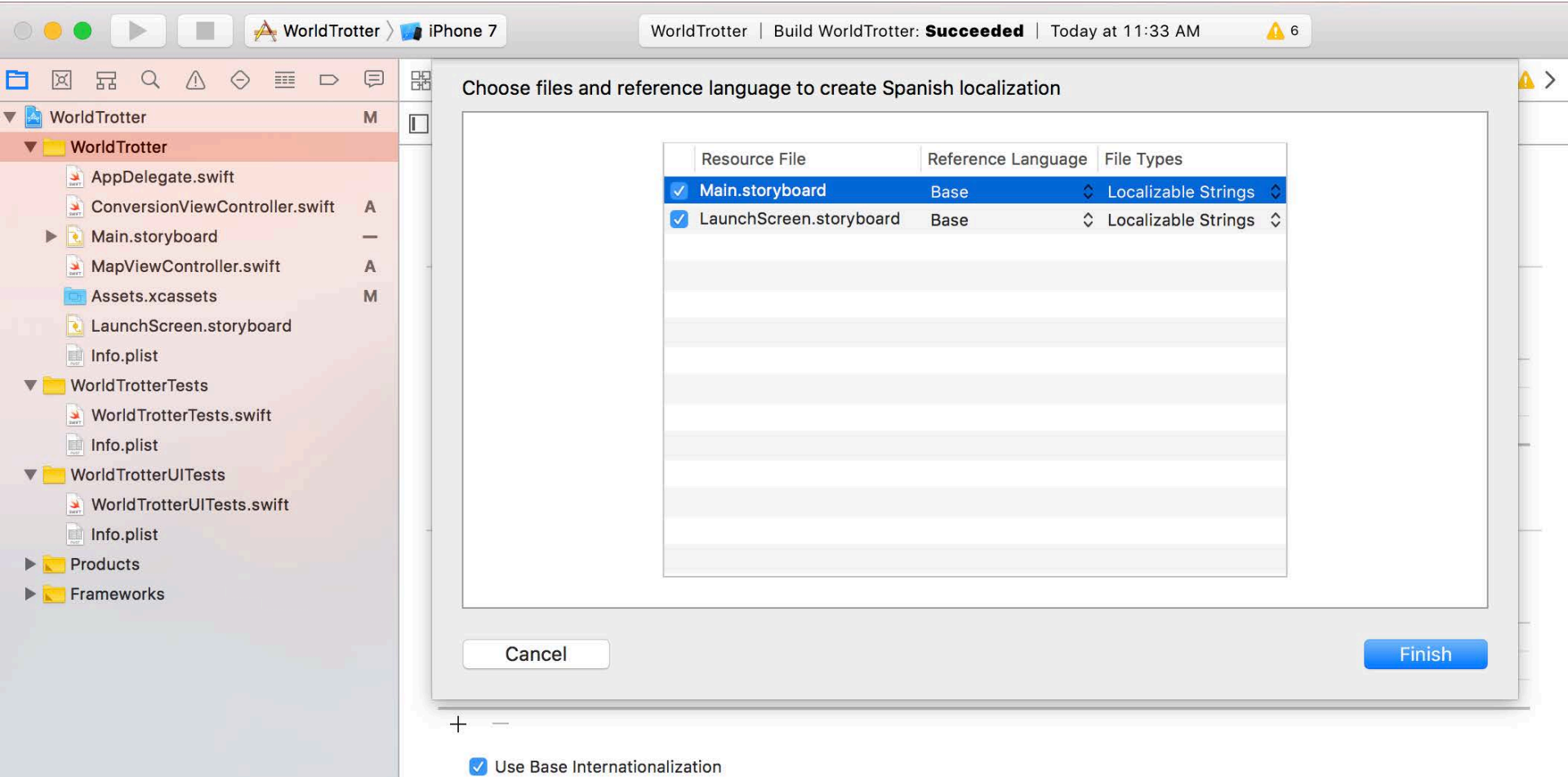
The app's its interface is able to adapt to various languages and regions.

Localizing the app

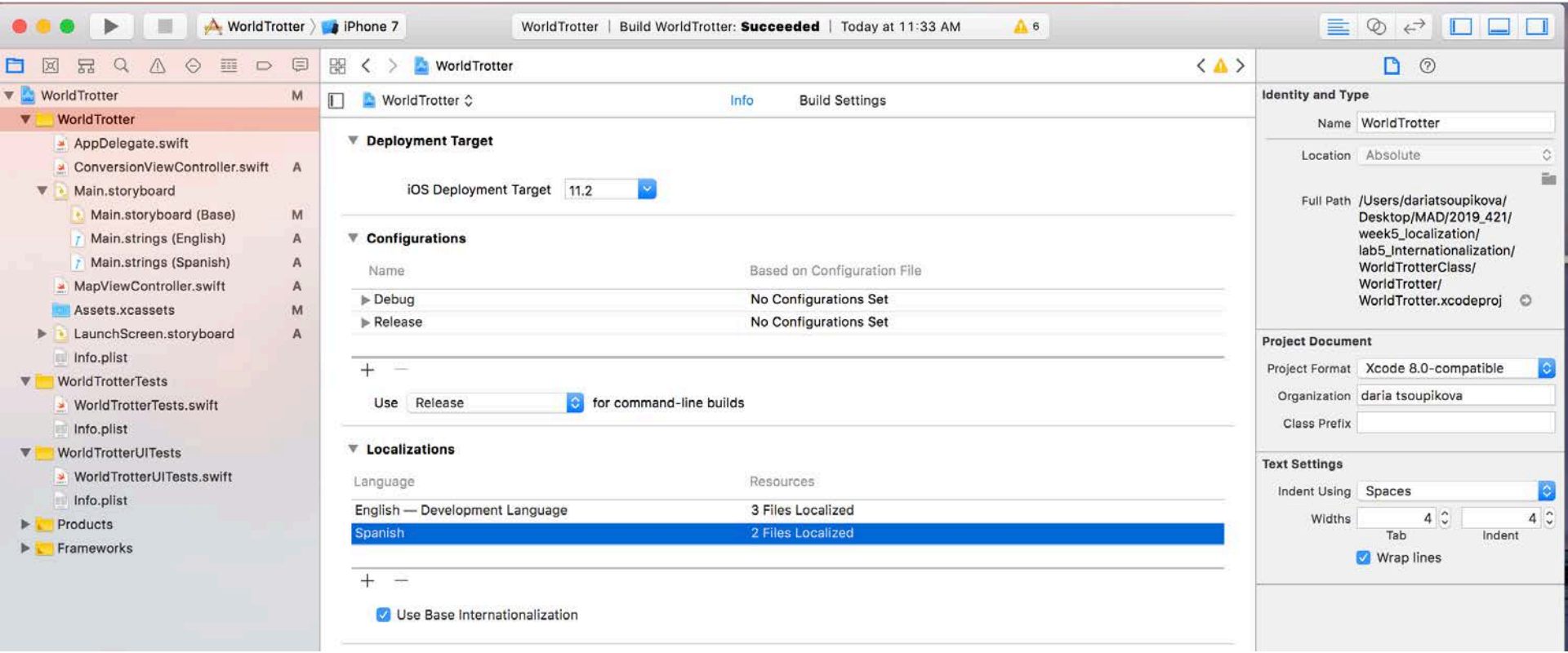
Click the + button under Localizations and select Spanish (es). In the dialog, you can uncheck the LaunchScreen.storyboard file; keep the Main.storyboard file checked. Make sure that the reference language is Base and the file type is Localizable Strings. Click Finish.



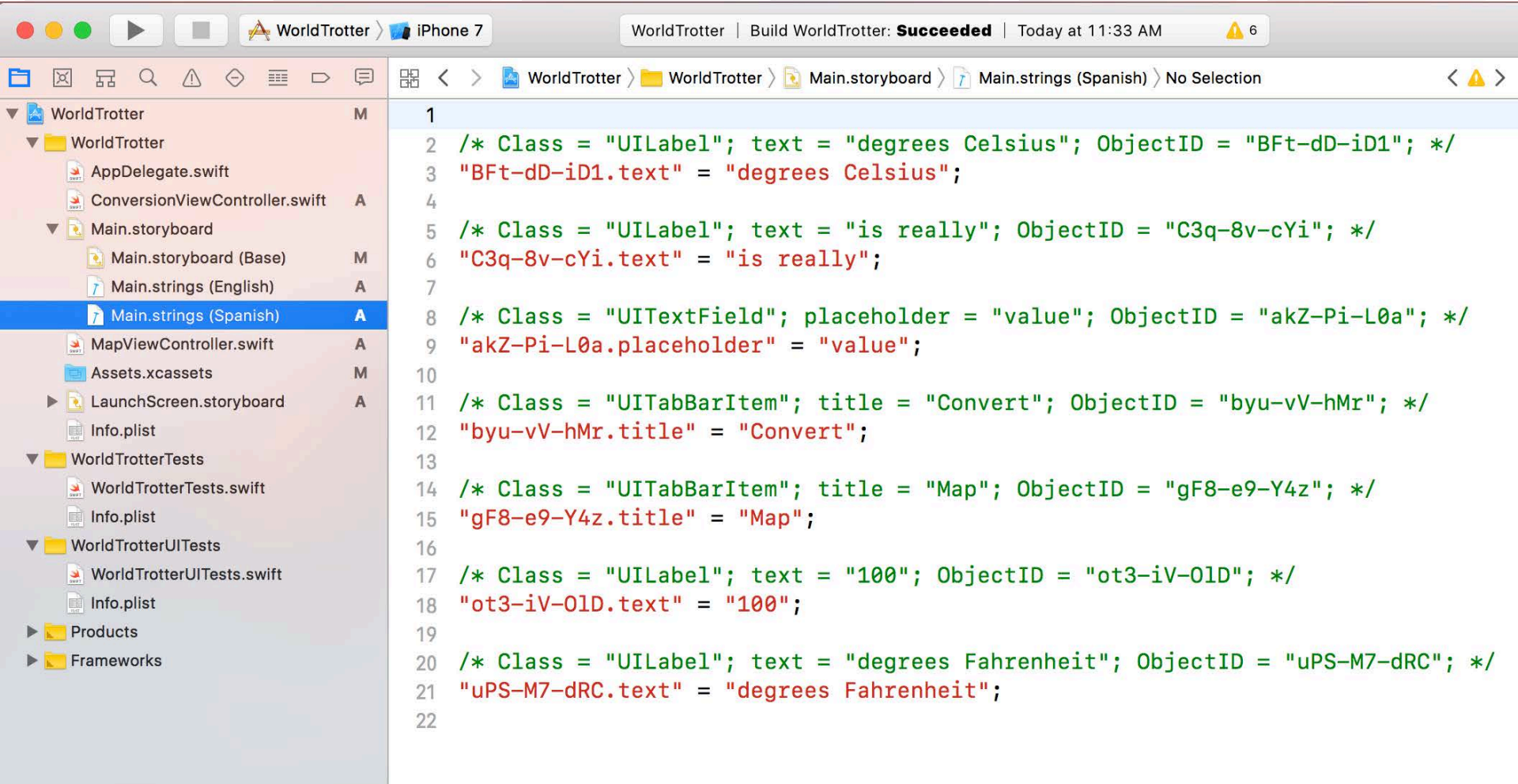
Localizing the app



Localizing the app



Localizing the app



Localizing the app

You have to translate localized files yourself; Xcode is not that smart.

/* Class = "UITabBarItem"; title = "Map"; ObjectID = "6xh-o5-yRt"; */"6xh-o5-yRt.title" = "Map" **"Mapa";**
/* Class = "UILabel"; text = "degrees Celsius"; ObjectID = "7la-u7-mx6"; */"7la-u7-mx6.text" = "degrees Celsius" "**grados Celsius";**
/* Class = "UILabel"; text = "degrees Fahrenheit"; ObjectID = "Dic-rs-P0S"; */"Dic-rs-P0S.text" = "degrees Fahrenheit" "**grados Fahrenheit";**
/* Class = "UILabel"; text = "100"; ObjectID = "Eso-Wf-EyH"; */"Eso-Wf-EyH.text" = "100";
/* Class = "UITextField"; placeholder = "value"; ObjectID = "On4-jV-YlY"; */"On4-jV-YlY.placeholder" = "value" **"valor";**
/* Class = "UILabel"; text = "is really"; ObjectID = "wtF-xR-gbZ"; */"wtF-xR-gbZ.text" = "is really" **"es realmente";**
/* Class = "UITabBarItem"; title = "Convert"; ObjectID = "zLY-50-CeX"; */"zLY-50-CeX.title" = "Convert" **"Convertir";**

Localizing the app

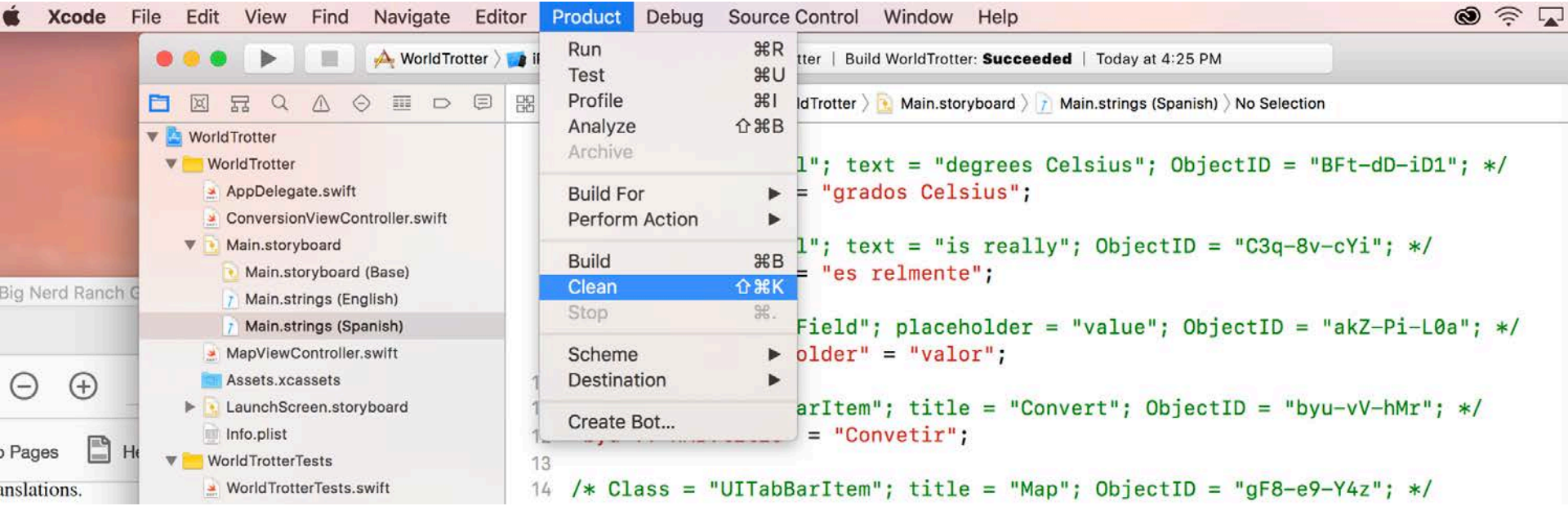
```
1
2  /* Class = "UILabel"; text = "degrees Celsius"; ObjectID = "BFt-dD-iD1"; */
3  "BFt-dD-iD1.text" = "grados Celsius";
4
5  /* Class = "UILabel"; text = "is really"; ObjectID = "C3q-8v-cYi"; */
6  "C3q-8v-cYi.text" = "es relmente";
7
8  /* Class = "UITextField"; placeholder = "value"; ObjectID = "akZ-Pi-L0a"; */
9  "akZ-Pi-L0a.placeholder" = "valor";
10
11 /* Class = "UITabBarItem"; title = "Convert"; ObjectID = "byu-vV-hMr"; */
12 "byu-vV-hMr.title" = "Convetir";
13
14 /* Class = "UITabBarItem"; title = "Map"; ObjectID = "gF8-e9-Y4z"; */
15 "gF8-e9-Y4z.title" = "Mapa";
16
17 /* Class = "UILabel"; text = "100"; ObjectID = "ot3-iV-OlD"; */
18 "ot3-iV-OlD.text" = "100";
19
20 /* Class = "UILabel"; text = "degrees Fahrenheit"; ObjectID = "uPS-M7-dRC"; */
21 "uPS-M7-dRC.text" = "grados Fahrenheit";
22
```

Localizing the app

1. Exit and restart Xcode to rebuild the app with new localization resources.

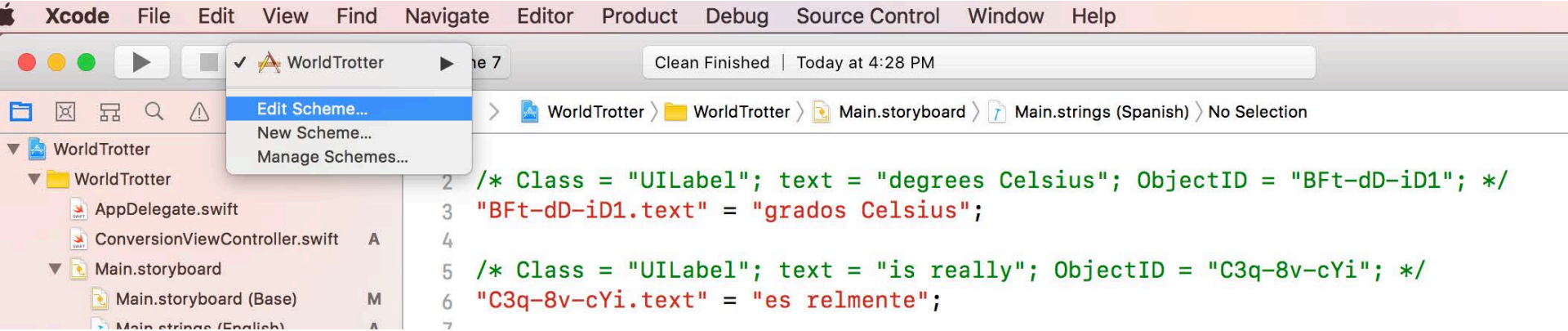
2. Product > Clean

3. Press and hold the Option key while opening the Product menu and choose Clean Build Folder to entirely recompile, rebundle, and reinstall the app.



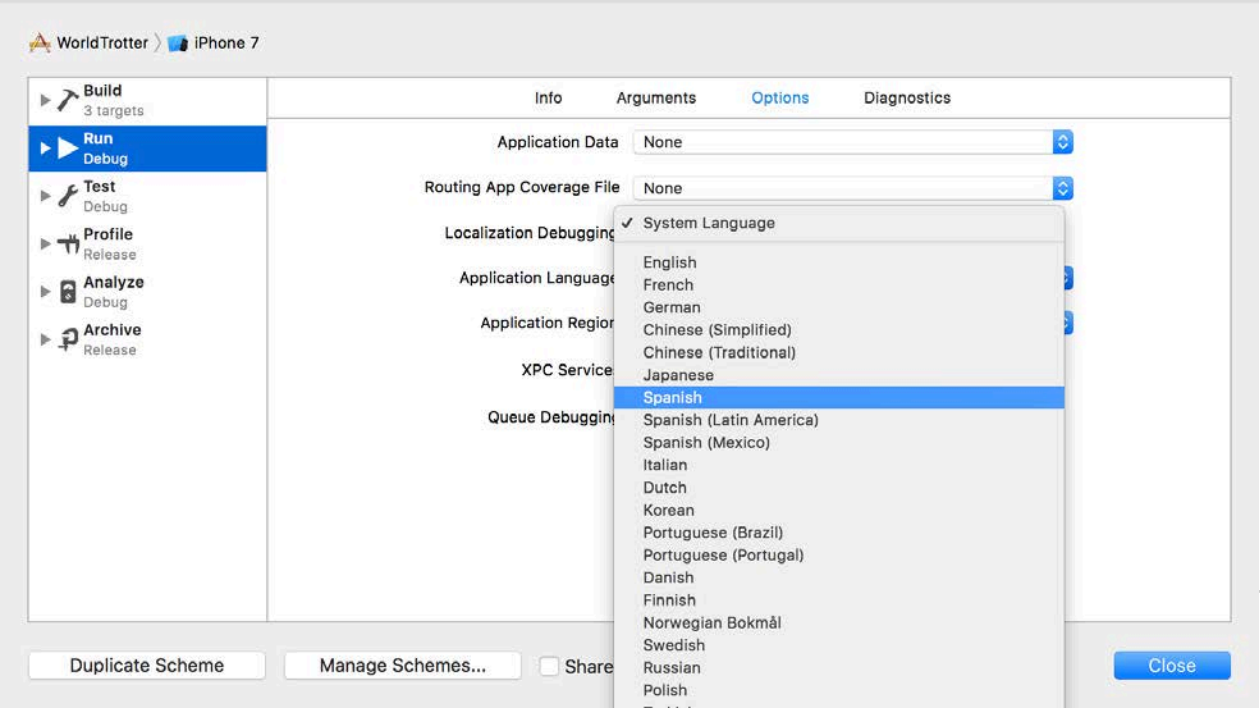
Localizing the app

Open the active scheme pop-up and select Edit Scheme.
Make sure Run is selected on the lefthand side and open the Options tab.
Open the Application Language pop-up and select Spanish.
Spain is selected from the Application Region pop-up.



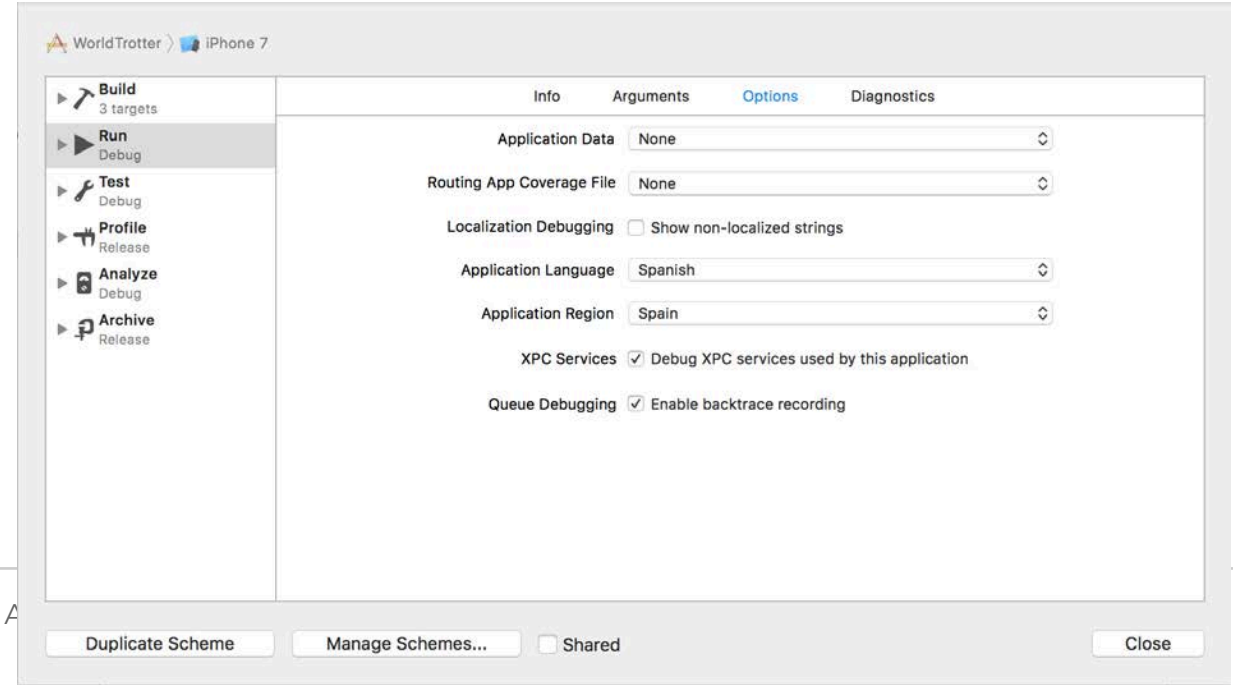
Localizing the app

Open the active scheme pop-up and select Edit Scheme.
Make sure Run is selected on the lefthand side and open the Options tab.
Open the Application Language pop-up and select Spanish.
Spain is selected from the Application Region pop-up.



Localizing the app

Open the active scheme pop-up and select Edit Scheme.
Make sure Run is selected on the lefthand side and open the Options tab.
Open the Application Language pop-up and select Spanish.
Spain is selected from the Application Region pop-up.



Localizing the app

Open the active scheme pop-up and select Edit Scheme.
Make sure Run is selected on the lefthand side and open the Options tab.
Open the Application Language pop-up and select Spanish.
Spain is selected from the Application Region pop-up.

Build and run.

The constraints on the labels accommodate different lengths of text, and resize labels to fit.

String table

To dynamically display translated versions of these strings, you must create a strings table.

A strings table is a file containing a list of key-value pairs for all of the strings that your application uses and their associated translations. It is a resource file that you add to your application, but you do not need to do a lot of work to get data from it.
You might use a string in your code like this:

```
let greeting = "Hello!"
```

To internationalize the string in your code, you replace literal strings with the function NSLocalizedString(_:comment:).

```
let greeting = NSLocalizedString("Hello!", comment: "The greeting for the user")
```

This function takes two arguments: a key and a comment that describes the string's use.

- The key is the lookup value in a strings table.
- At runtime, NSLocalizedString(_:comment:) will look through the strings tables bundled with your application for a table that matches the user's language settings.
- Then, in that table, the function gets the translated string that matches the key.

String table

In MapViewController.swift, update the loadView() method:

```
override func loadView() {
// Create a map view    mapView = MKMapView()
// Set it as *the* view of this view controller    view = mapView

~~let segmentedControl~~
~~= UISegmentedControl(items: ["Standard", "Satellite", "Hybrid"])~~
      let standardString = NSLocalizedString("Standard", comment: "Standard map view")
      let satelliteString
         = NSLocalizedString("Satellite", comment: "Satellite map view")
      let hybridString = NSLocalizedString("Hybrid = UISegmentedControl(items:
[standardString, satelliteString, hybridString])", comment: "Hybrid map view")
      let segmentedControl= UISegmentedControl(items: [standardString, satelliteString,
hybridString])
```

String table

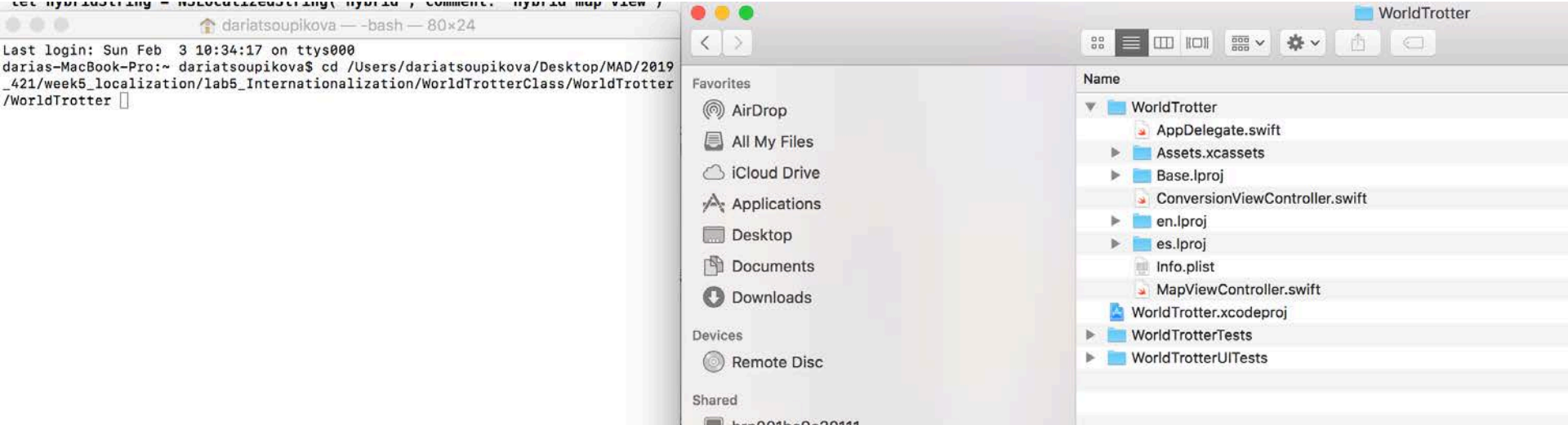
In MapViewController.swift, update the loadView() method:

```
18      override func loadView() {
19          // Create a map view
20          mapView = MKMapView()
21
22          // Set it as *the* view of this view controller
23          view = mapView
24
25       /*   let segmentedControl = UISegmentedControl(items: ["Standard", "Hybrid", "Satellite"])*/
26
27          let standardString = NSLocalizedString("Standard", comment: "Standard map view")
28          let satelliteString
29              = NSLocalizedString("Satellite", comment: "Satellite map view")
30          let hybridString = NSLocalizedString("Hybrid = UISegmentedControl(items: [standardString, satelliteString,
                hybridString])", comment: "Hybrid map view")
31          let segmentedControl= UISegmentedControl(items: [standardString, satelliteString, hybridString])
32
33
34          segmentedControl.backgroundColor = UIColor.white.withAlphaComponent(0.5)
35          segmentedControl.selectedSegmentIndex = 0
36
```

String table

1. Open Terminal, type the following:
cd
followed by a space. (Do not press Return yet.)

2. open Finder and locate MapViewController.swift and the folder that contains it. Drag the icon of that folder onto the Terminal window. Terminal will fill out the path for you. It will look something like this:
cd /Users/cbkeur/iOSDevelopment/WorldTrotter/WorldTrotter/
Press Return.



String table

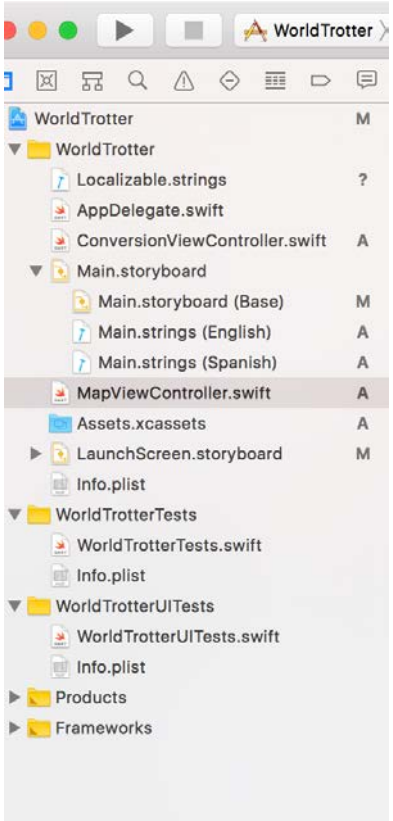
3. Use the terminal command **ls** to print out the contents of the working directory and confirm that MapViewController.swift is in that list.

4. To generate the strings table,
enter the following into Terminal and press Return:

genstrings MapViewController.swift

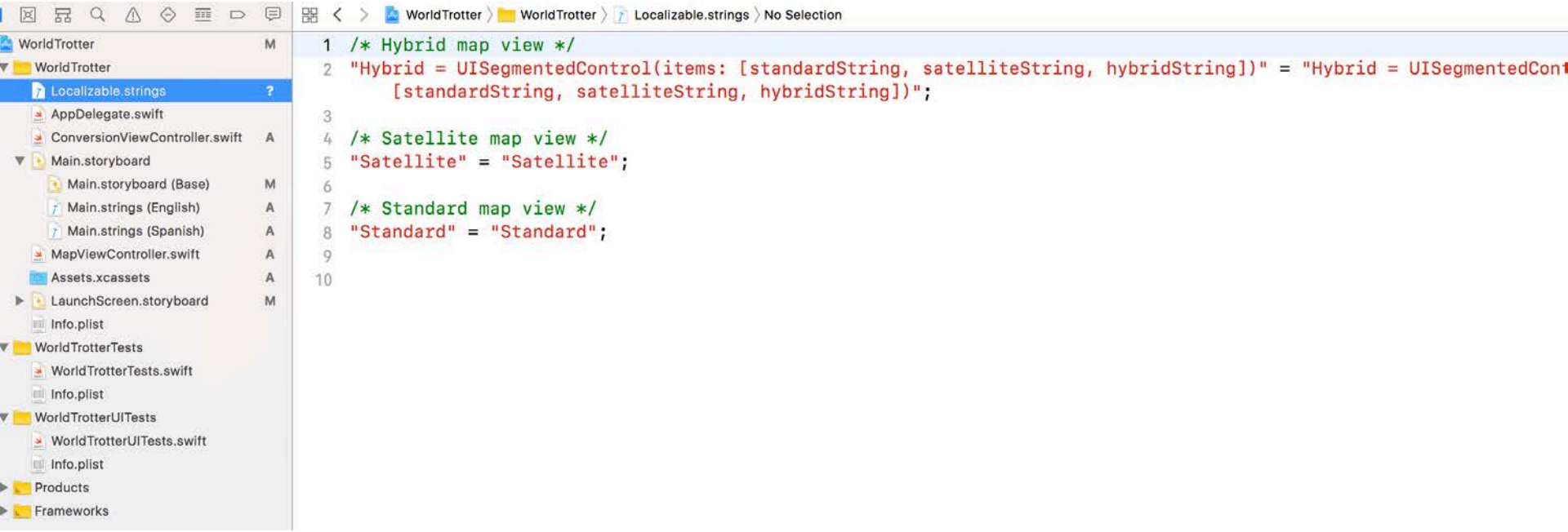
The resulting file, Localizable.strings, contains the strings from MapViewController.

5. Drag this new file from Finder into the project navigator (or use the File → Add Files to "WorldTrotter"... menu item). When the application is compiled, this resource will be copied into the main bundle.



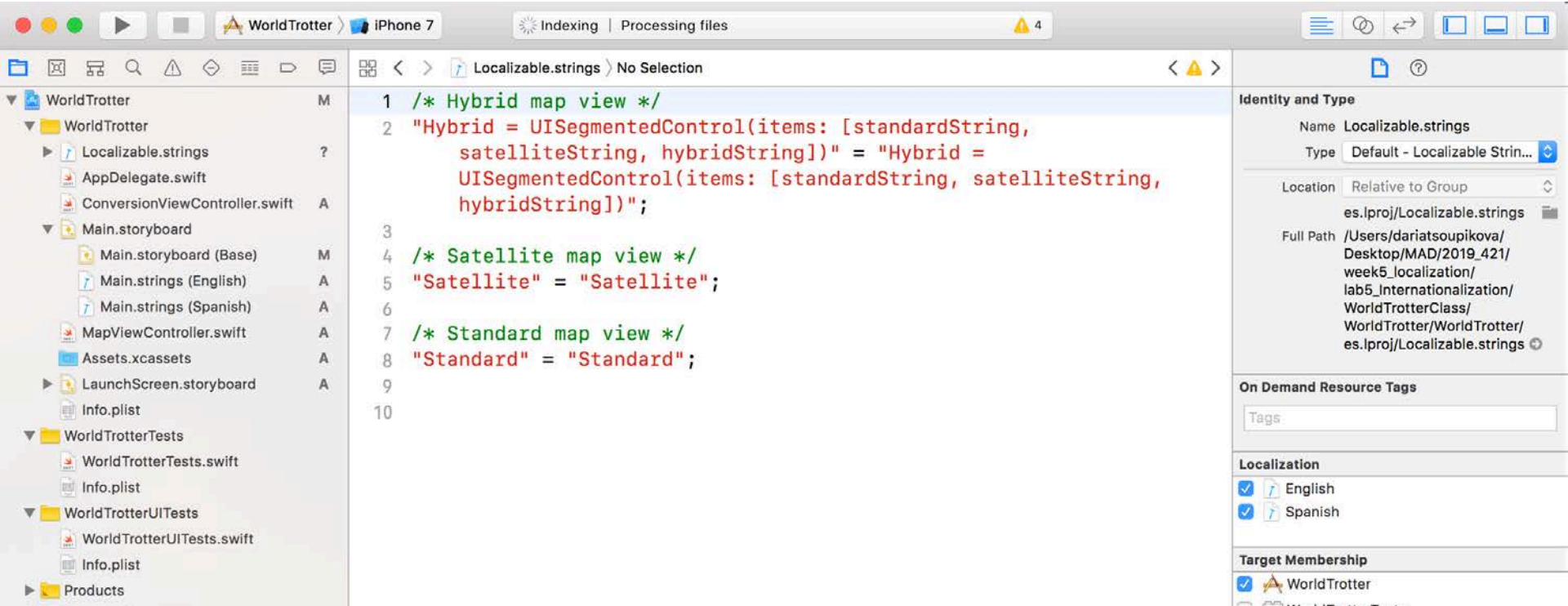
String table

Localizable.strings file



String table

to localize it in Xcode:
Open its file inspector and click the Localize. button in the utility area.
Make sure Base is selected from the pop-up and click Localize. Add the Spanish and English localization by checking the box next to each language.



String table

In the project navigator, click on the disclosure triangle that now appears next to Localizable.strings. Open the Spanish version.

The string on the lefthand side is the key that is passed to the NSLocalizedString(_:comment:) function, and the string on the righthand side is what is returned.

Change the text on the righthand side to the Spanish translations shown below.

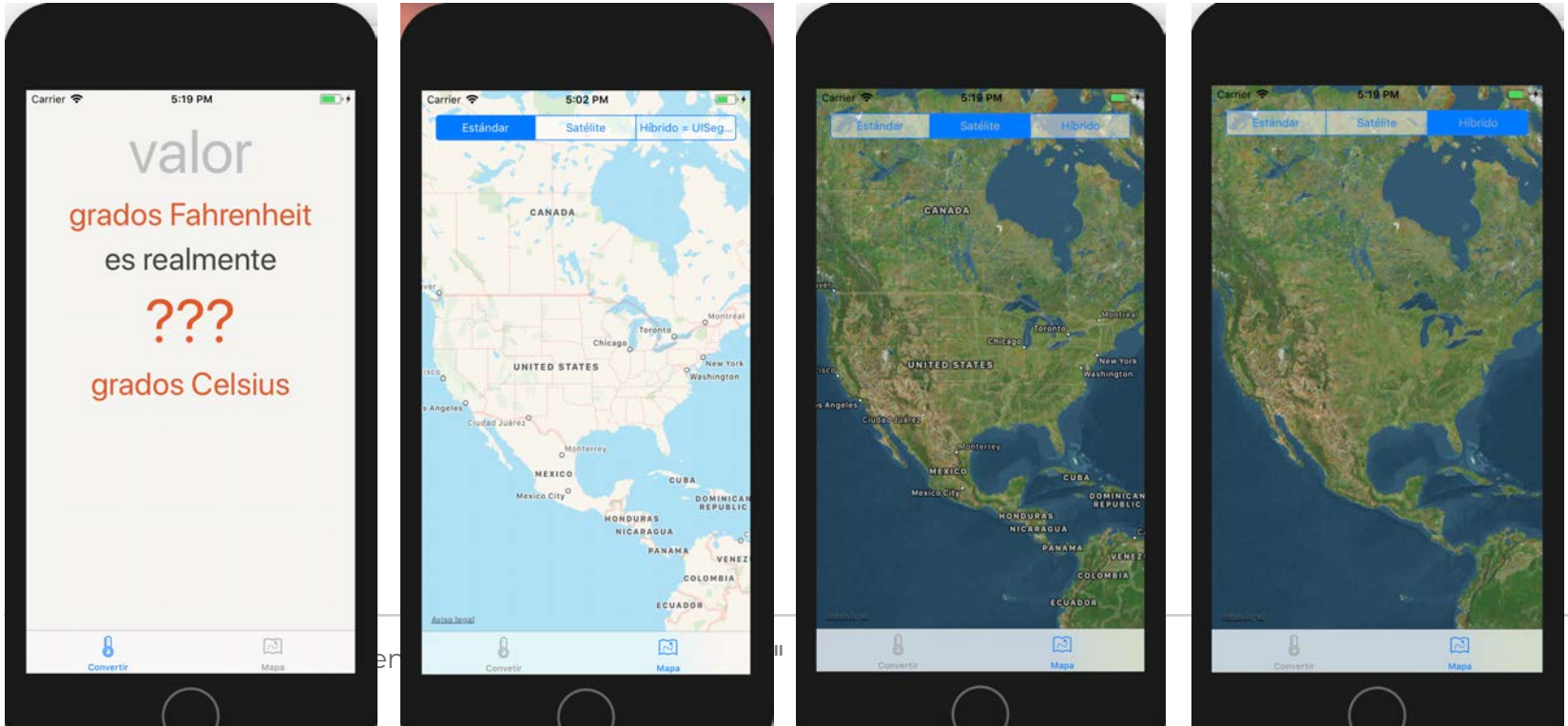
/* Hybrid map view */"Hybrid" = "Hybrid" "Híbrido";

/* Satellite map view */"Satellite" = "Satellite" "Satélite";

/* Standard map view */"Standard" = "Standard" "Estándar";

String table

check your scheme language setting, build and run. Now all titles will appear in Spanish.



String table

Internationalization and localization are important for greatest public outreach.

Most of the apps are internationalized and localized for global market.

The app now
- converts between Celsius and Fahrenheit
- displays a map in a few different modules
- scales on all screen sizes
- is localized into another language

Assignment 3
- Internationalize and localize (In German) the Main screen of the Estimator
- Include metric/ imperial conversion
- (ignore the conversion of other measures for now)