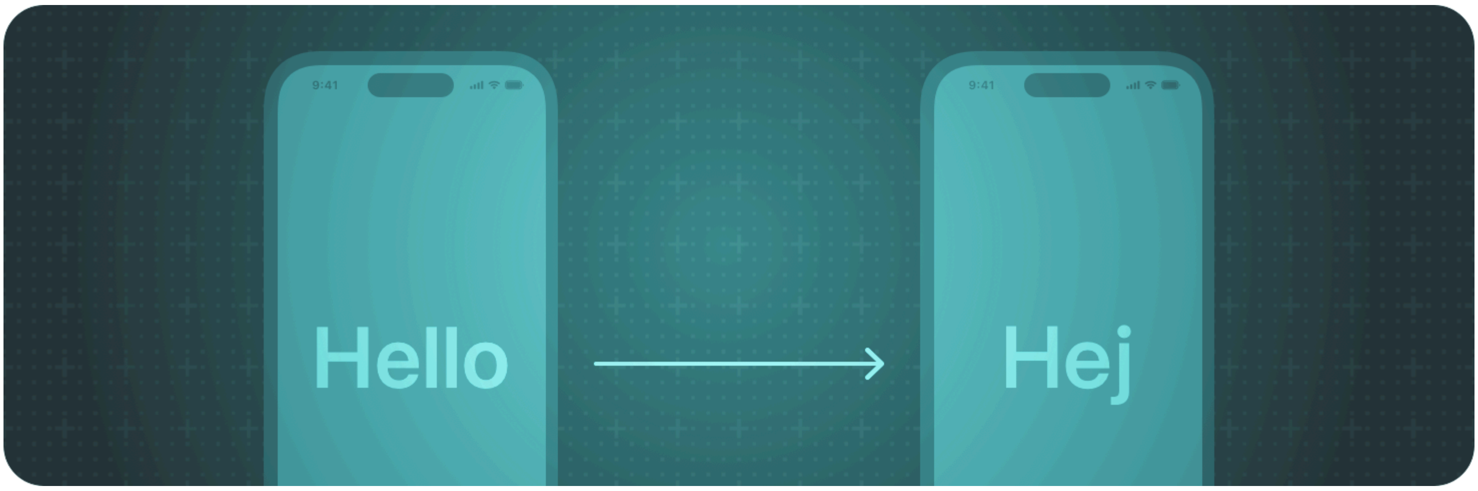
Article

Localizing and varying text with a string catalog

Use a string catalog to translate text, handle plurals, and vary the text your app displays on specific devices.

Overview

Your app delivers the best experience when it runs well in a person's locale and displays content in their native language. Supporting multiple languages is more than translating text. It includes handling plurals for nouns and units, as well as displaying the right form of text on specific devices.



Use a string catalog to localize and translate all your app's text in a visual editor right in Xcode. A string catalog automatically tracks all the localizable strings from your code, and keeps your translations in one place.

Use string catalogs to host translations, vary strings that contain plurals for different regions and locales, and change how text appears on different devices.

Related session from WWDC25

Session 225: Code-along: Explore localization with Xcode

Localize your app's text

Before you can translate text, you need to make it localizable. This involves wrapping the user-facing strings in your app in constructs that make them translatable.

In SwiftUI, all string literals within a view are automatically localizable.

```
// SwiftUI localizable text.
Text("Title")
```

Make general text localizable using the `String(localized:)` initializer. For apps targeting older platforms, use the `NSLocalizedString` macro.

```
// General localizable text.
String(localized: "Add a description for your collection here.")
```

Add comments to give context and assist localizers when translating your text. Alternatively, you can use coding intelligence later to generate comments after you create a string catalog.

```
// Localizable text with comments.
Text("Edit", comment: "The text label on a button to switch to editor mode.")
String(localized: "North America", comment: "The name of a continent.")
```

When your code isn't running in the main bundle, use the `bundle` parameter to tell the system where to find the string at runtime. You can pass the explicit bundle name or pass the `bundle()` macro to refer to the bundle that contains resources for the current target.

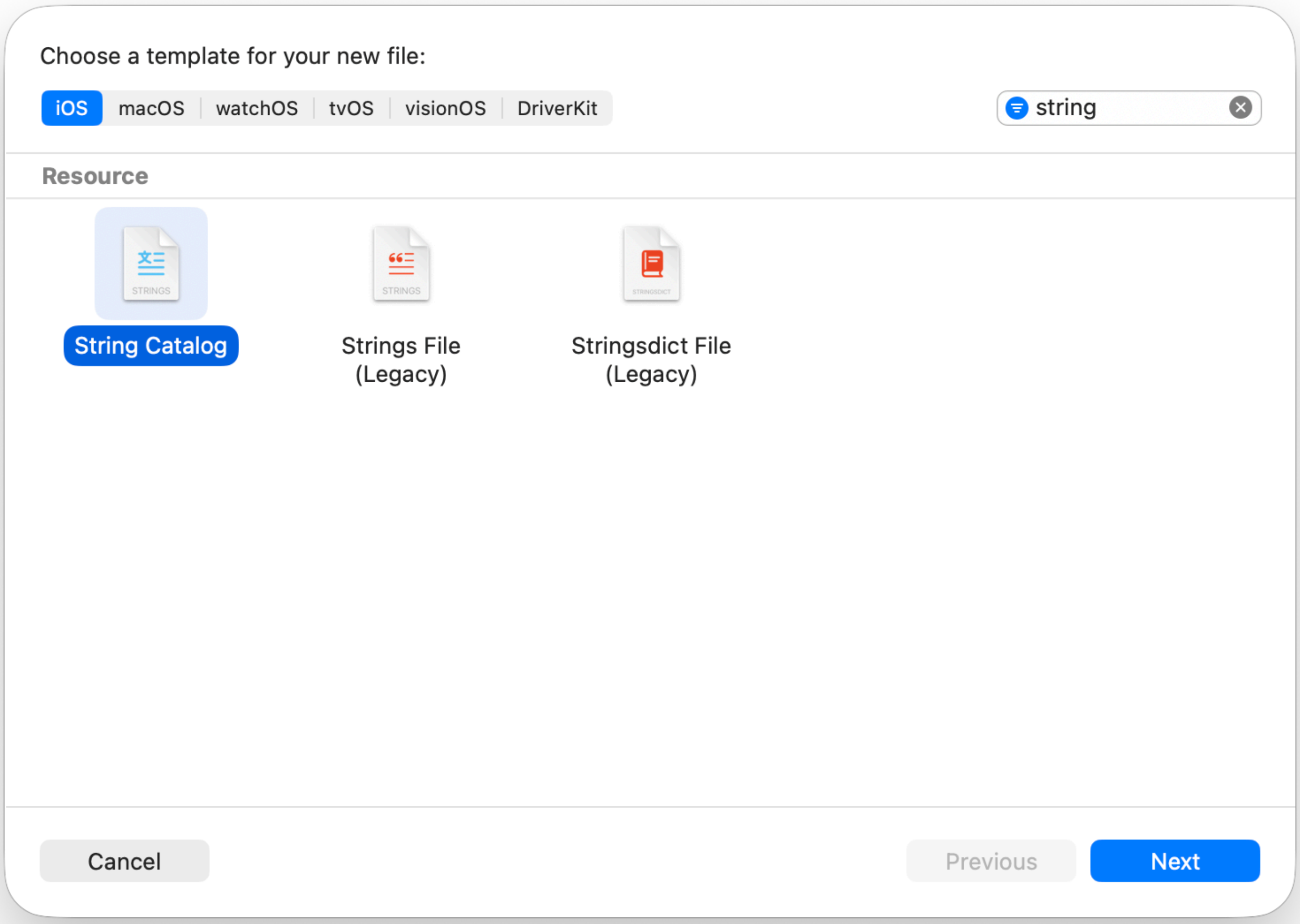
```
// Localizable text in a Swift package or framework.
Text("My Collections" bundle: #bundle, comment: "Section title above user-created co
```

To refer to the main app, pass `Bundle.main`, which is also the default bundle.

Add a string catalog to your project

To add a string catalog to your project, choose File > New > File from Template.

In the sheet that appears, select the platform, enter `string` in the filter field, select String Catalog under Resource, and click Next. In the dialog that appears, accept the default name `Localizable`, choose the Resources folder in your project as the location, and click Create.



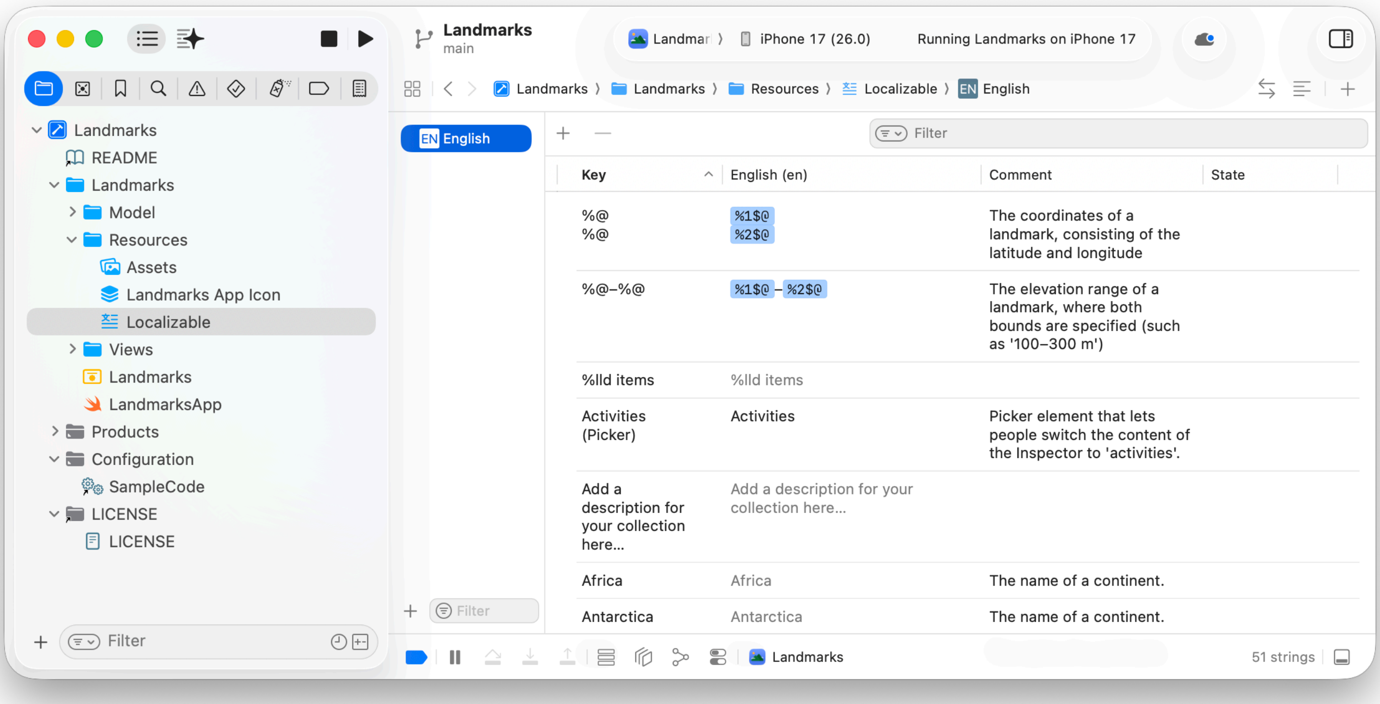
If your string catalog gets too big, you can create multiple string catalog files within a single Xcode project, and give each a unique name. Then choose which string catalog to use for each translation by passing the string catalog name to the `tableName` or `table` parameter to the respective localization API as follows:

```
// A SwiftUI localization example pointing to a specific string catalog.
Text("Explore", tableName: "Navigation")

// A general text localization example pointing to a specific string catalog.
String(localized: "Gorgeous mountain peaks!", table: "LandmarkCollectionData")
```

Add your localizable text to the string catalog

Xcode automatically keeps your string catalog and app in sync each time you build your project. To populate your string catalog with the localizable text from your app, choose Product > Build. Xcode identifies strings in your app that support localization, and then adds them to your string catalog.



Landmarks
main

Landmar 〉 iPhone 17 (26.0) Running Landmarks on iPhone 17

Landmarks 〉 Landmarks 〉 Resources 〉 Localizable 〉 EN English

- Landmarks
  - README
  - Landmarks
    - Model
    - Resources
      - Assets
      - Landmarks App Icon
      - Localizable
    - Views
    - Landmarks
    - LandmarksApp
  - Products
  - Configuration
    - SampleCode
  - LICENSE
    - LICENSE

EN English

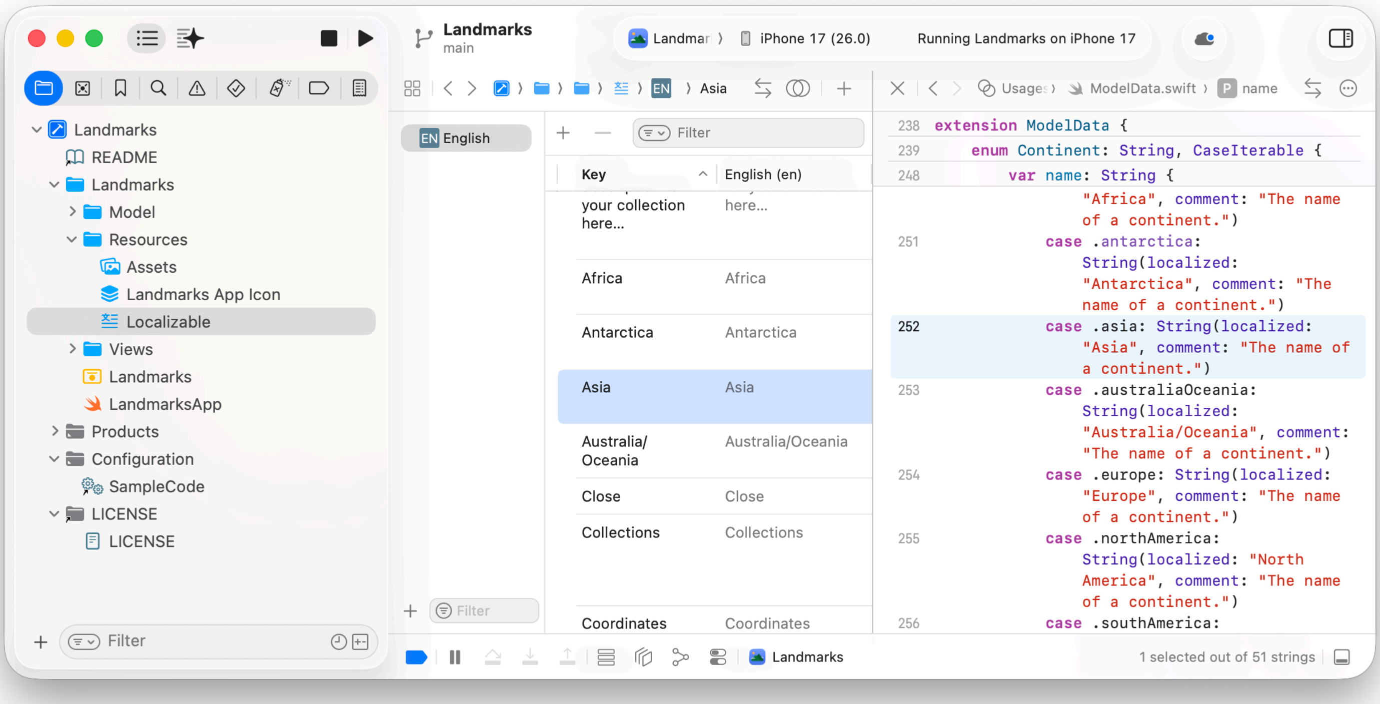
Key	English (en)	Comment	State
%@ %@	%1$@ %2$@	The coordinates of a landmark, consisting of the latitude and longitude	
%@–%@	%1$@ – %2$@	The elevation range of a landmark, where both bounds are specified (such as '100–300 m')	
%lld items	%lld items		
Activities (Picker)	Activities	Picker element that lets people switch the content of the Inspector to 'activities'.	
Add a description for your collection here…	Add a description for your collection here…		
Africa	Africa	The name of a continent.	
Antarctica	Antarctica	The name of a continent.	

Landmarks 51 strings

Review the source of localizable text in the assistant

After you populate your string catalog, you can review the location of localizable text in your app before you add languages and translations.

In the Project navigator, select the string catalog file and choose Editor > Assistant. Then select a key in the catalog to see the source of the localizable text in the assistant.



Key	English (en)
your collection here…	here…
Africa	Africa
Antarctica	Antarctica
Asia	Asia
Australia/ Oceania	Australia/Oceania
Close	Close
Collections	Collections
Coordinates	Coordinates

```
extension ModelData {
    enum Continent: String, CaseIterable {
        var name: String {
                "Africa", comment: "The name
                of a continent.")
            case .antarctica:
                String(localized:
                "Antarctica", comment: "The
                name of a continent.")
            case .asia: String(localized:
                "Asia", comment: "The name of
                a continent.")
            case .australiaOceania:
                String(localized:
                "Australia/Oceania", comment:
                "The name of a continent.")
            case .europe: String(localized:
                "Europe", comment: "The name
                of a continent.")
            case .northAmerica:
                String(localized: "North
                America", comment: "The name
                of a continent.")
            case .southAmerica:
```

If necessary, make edits to the localizable text in the source editor and choose Product > Build to update the string catalog.

Alternatively, to jump to the text in the source editor, click the arrow button that appears next to a key when you hover over it in the string catalog.

Generate comments automatically

You can let Xcode automatically create comments for your localizable strings to provide more context for translators. Comments should describe the interface element, surrounding interface, placeholder content, and variables in the string.

To generate a comment for a specific key:

1. In the string catalog editor, Control-click the key you want to add a comment to.

2. Choose Generate Comment from the context menu.

You can also enable automatic comment generation for all projects. In Xcode > Settings > Editing, turn on “Automatically generate string catalog comments.”

Later, when you export your localization files, an “auto-generated” note appears next to any automatically generated comments to distinguish them from manually written comments.

Add variants for strings that contain plurals

Languages have different grammatical rules for handling plurals of nouns and units. For example, in English, you can return `1 item` when the value of `%lld` is `1`. And you can return `%lld items` for all other cases. Other languages can have fewer or more plural variants, depending on their region and locale.

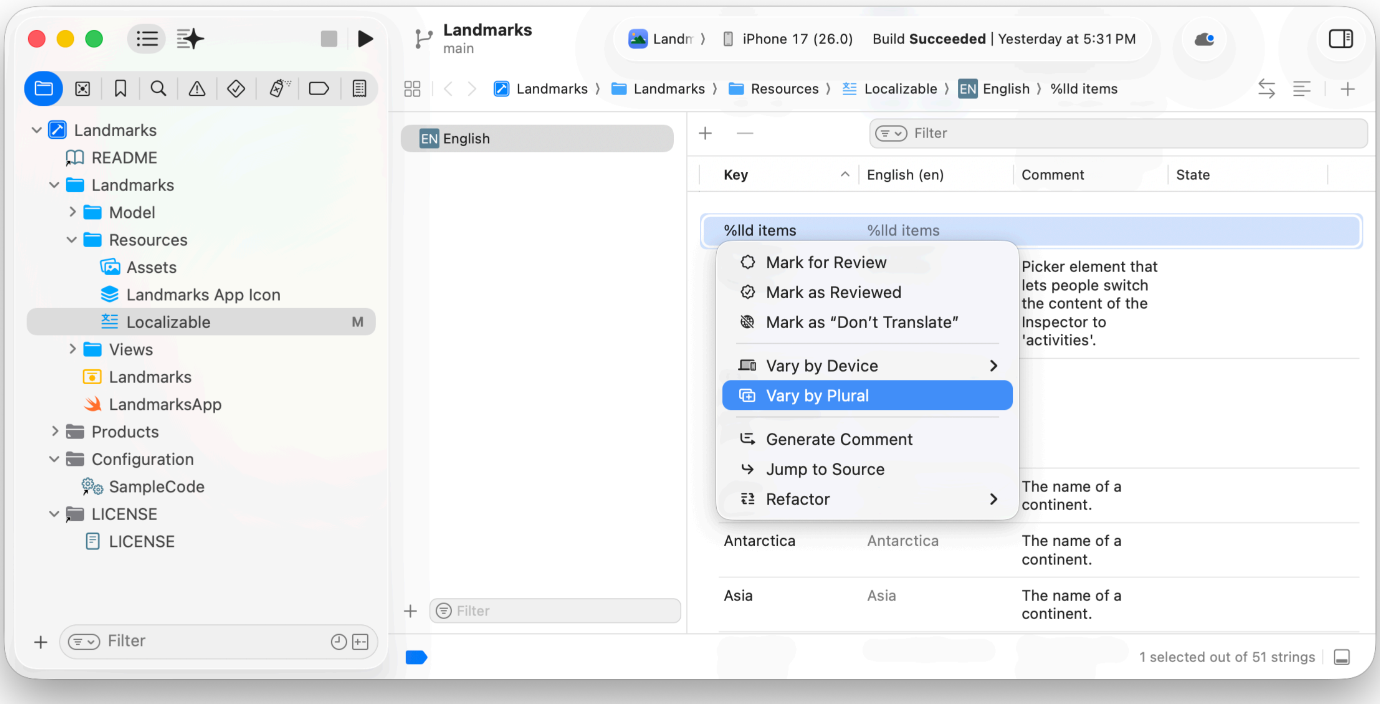
Plural	Text
One	`%lld item`
Other	`%lld items`

First localize the text in your app using the value the string is dependent on, using string interpolation.

```
Text("\(collection.landmarks.count) items")
```

Then, add plural variants to the source localization to assist localizers when translating your text in other languages. When you add other languages, Xcode automatically adds the language-specific variations for those languages to the strings file too. For example, Xcode adds One and Other variants for English, and One, Few, Many, and Other variants for Russian.

In the string catalog editor, Control-click the key that contains variables and select Vary by Plural from the contextual menu.



Key	English (en)	Comment	State
%lld items	%lld items		
		Picker element that lets people switch the content of the Inspector to 'activities'.	
		The name of a continent.	
Antarctica	Antarctica	The name of a continent.	
Asia	Asia	The name of a continent.	

When you add plural variants, the system does the following:

- Adds all the plural forms for that language to the string catalog editor.
- Determines which specifier to use for the interpolated string (`%lld` representing a 64-bit integer in this case).
- Prepopulates the variant fields with the value of that key.

Note

If you add plural variants to the source localization after you add languages, the system propagates the variants to the other languages, if possible. If you add variants to a nonsource localization, that change affects only that language.

Click the language in the sidebar, click the text field in the Language column for each variation of that string key, and then enter the text for the system to use when that plural displays.

When you run the app, the plural variants update based on the value of the interpolated string.

Vary strings by device

When you need to alter the text that displays on a device due to the available space, or because it has a different interaction, use the Vary by Device option in the string catalog editor.

For example, suppose you want to display two different messages depending on whether your app is running on iPhone or a Mac.

Operating system	Message
iOS	Tap to learn more
macOS	Click to learn more

To vary the text string based on device, select the string catalog file, along with the language and key representing the message you want to vary. Control-click the key, select Vary by Device, and then select the device you want to add a specific message for.

Enter the text you want to display for that selected device, while retaining the existing text for all other devices. Add more devices if you need more variations.

When the app runs on the selected device, the system displays the new message for that device.

Add languages to your project

To support multiple languages in your app, add additional languages to your project. For each language, select the string catalog in the Project navigator, click the Add button (+) near the bottom of the string catalog editor, and select a language from the pop-up menu.

You can also add and remove languages using the Info pane in the project editor. Under Localizations, click the Add button (+), or select a language and click the Remove button (-).

Add your translations using the string catalog editor

If you know the translations for the languages that you add, you can enter them directly in the string catalog editor. Otherwise, you can export the languages, send them to a third-party for translation, and then import them.

To enter translations, select the language (German in this example) that you want to add translations for in the sidebar. Then click the text field for each key in that language's column and enter the translation for that key.

Newly added strings that require translation appear with a New icon in the State column. When you add a translation, the New icon changes to a green checkmark. As you add translations, the percent symbol beside the language updates, displaying the translation percentage for that language. When a string catalog language reaches full translation, the percent symbol changes to a green checkmark.

For information about exporting and importing translations in Xcode, see Exporting localizations and Importing localizations.

Test your translations

To test your translations in the simulator:

1. Change the app language of your current scheme by choosing Product > Scheme > Edit Scheme.
2. In the dialog that appears, select the Run action on the left and click the Options tab.
3. Choose the language you want to test from the App Language drop-down menu and click Close.

You can also navigate to Settings on the individual simulated device and change the device's language there. When your app runs, the translations for the selected language appear in the simulator.

For more information on testing localizations, see Previewing localizations and Testing localizations when running your app.

See Also

Essentials

- Supporting multiple languages in your app

  Internationalize your app’s strings, images, and other resource types to prepare for the translation process.

- Using generated localizable symbols in your code

  Add keys directly to your string catalog that you can reference in your code using Xcode generated localizable symbols.

- {} Localizing Landmarks

  Add localizations to the Landmarks sample code project.