Work with translation files

After you prepare a component for translation, use the `extract-i18n` Angular CLI command to extract the marked text in the component into a *source language* file.

The marked text includes text marked with `i18n`, attributes marked with `i18n-`*attribute*, and text tagged with `$localize` as described in Prepare templates for translations.

Complete the following steps to create and update translation files for your project.

1. Extract the source language file
    1. Optionally, change the location, format, and name
2. Copy the source language file to create a translation file for each language
3. Translate each translation file
4. Translate plurals and alternate expressions separately
    1. Translate plurals
    2. Translate alternate expressions
    3. Translate nested expressions

Extract the source language file

To extract the source language file, complete the following actions.

1. Open a terminal window
2. Change to the root directory of your project
3. Run the following CLI command

The `extract-i18n` command creates a source language file named `messages.xlf` in the root directory of your project. For more information about the XML Localization Interchange File Format (XLIFF, version 1.2), see XLIFF.

Use the following `extract-i18n` command options to change the source language file location, format, and file name.

Command option	Details
`--format`	Set the format of the output file
`--outFile`	Set the name of the output file
`--output-path`	Set the path of the output directory

Change the source language file location

To create a file in the `src/locale` directory, specify the output path as an option.

`extract-18n --output-path` example

The following example specifies the output path as an option.

Change the source language file format

The `extract-i18n` command creates files in the following translation formats.

Translation format	Details	file extension
ARB	Application Resource Bundle	`.arb`
JSON	JavaScript Object Notation	`.json`
XLIFF 1.2	XML Localization Interchange File Format, version 1.2	`.xlf`
XLIFF 2	XML Localization Interchange File Format, version 2	`.xlf`
XMB	XML Message Bundle	`.xmb` (`.xtb`)

Specify the translation format explicitly with the `--format` command option.

The XMB format generates `.xmb` source language files, but uses `.xtb` translation files.

`extract-18n --format` example

The following example demonstrates several translation formats.

Change the source language file name

To change the name of the source language file generated by the extraction tool, use the `--outFile` command option.

`extract-18n --out-file` example

The following example demonstrates naming the output file.

Create a translation file for each language

To create a translation file for a locale or language, complete the following actions.

1. Extract the source language file
2. Make a copy of the source language file to create a *translation* file for each language
3. Rename the *translation* file to add the locale

   messages.xlf --> message.{locale}.xlf
4. Create a new directory at your project root named `locale`

   src/locale
5. Move the *translation* file to the new directory
6. Send the *translation* file to your translator
7. Repeat the above steps for each language you want to add to your application

`extract-i18n` example for French

For example, to create a French translation file, complete the following actions.

1. Run the `extract-18n` command
2. Make a copy of the `messages.xlf` source language file

3. Rename the copy to `messages.fr.xlf` for the French language (`fr`) translation.
4. Move the `fr` translation file to the `src/locale` directory.
5. Send the `fr` translation file to the translator.

Translate each translation file

Unless you are fluent in the language and have the time to edit translations, you will likely complete the following steps.

1. Send each translation file to a translator
2. The translator uses an XLIFF file editor complete the following actions
    1. Create the translation
    2. Edit the translation

Translation process example for French

To demonstrate the process, review the `messages.fr.xlf` file in the Example Angular Internationalization application. The Example Angular Internationalization application includes a French translation for you to edit without a special XLIFF editor or knowledge of French.

The following actions describe the translation process for French.

1. Open `messages.fr.xlf` and find the first `<trans-unit>` element. This is a *translation unit*, also known as a *text node*, that represents the translation of the `<h1>` greeting tag that was previously marked with the `i18n` attribute.

    > *The `id="introductionHeader"` is a custom ID, but without the `@@` prefix required in the source HTML.*

2. Duplicate the `<source>...</source>` element in the text node, rename it to `target`, and then replace the content with the French text.

    > *In a more complex translation, the information and context in the description and meaning elements help you choose the right words for translation.*

3. Translate the other text nodes. The following example displays the way to translate.

    Don't change the IDs for translation units. Each `id` attribute is generated by Angular and depends on the content of the component text and the assigned meaning. If you change either the text or the meaning, then the `id` attribute changes. For more about managing text updates and IDs, see custom IDs.

Translate plurals

Add or remove plural cases as needed for each language.

For language plural rules, see CLDR plural rules.

`minute` `plural` example

To translate a `plural`, translate the ICU format match values.

- `just now`
- `one minute ago`
- `<x id="INTERPOLATION" equiv-text="{{minutes}}"/> minutes ago`

The following example displays the way to translate.

Translate alternate expressions

Angular also extracts alternate `select` ICU expressions as separate translation units.

`gender` `select` example

The following example displays a `select` ICU expression in the component template.

In this example, Angular extracts the expression into two translation units. The first contains the text outside of the `select` clause, and uses a placeholder for `select` (`<x id="ICU">`):

When you translate the text, move the placeholder if necessary, but don't remove it. If you remove the placeholder, the ICU expression is removed from your translated application.

The following example displays the second translation unit that contains the `select` clause.

The following example displays both translation units after translation is complete.

Translate nested expressions

Angular treats a nested expression in the same manner as an alternate expression. Angular extracts the expression into two translation units.

Nested `plural` example

The following example displays the first translation unit that contains the text outside of the nested expression.

The following example displays the second translation unit that contains the complete nested expression.

The following example displays both translation units after translating.

What's next

- Merge translations into the app

@reviewed 2021-10-13