Merge translations into the application

To merge the completed translations into your project, complete the following actions

1. Use the Angular CLI to build a copy of the distributable files of your project
2. Use the `"localize"` option to replace all of the i18n messages with the valid translations and build a localized variant application. A variant application is a complete a copy of the distributable files of your application translated for a single locale.

After you merge the translations, serve each distributable copy of the application using server-side language detection or different subdirectories.

For more information about how to serve each distributable copy of the application, see deploying multiple locales.

For a compile time translation of the application, the build process uses ahead-of-time (AOT) compilation to produce a small, fast, ready-to-run application.

For a detailed explanation of the build process, see Building and serving Angular apps. The build process works for translation files in the `.xlf` format or in another format that Angular understands, such as `.xtb`. For more information about translation file formats used by Angular, see Change the source language file format

To build a separate distributable copy of the application for each locale, define the locales in the build configuration in the `angular.json` workspace build configuration file of your project.

This method shortens the build process by removing the requirement to perform a full application build for each locale.

To generate application variants for each locale, use the `"localize"` option in the `angular.json` workspace build configuration file. Also, to build from the command line, use the `build` Angular CLI command with the `--localize` option.

Optionally, apply specific build options for just one locale for a custom locale configuration.

Define locales in the build configuration

Use the `i18n` project option in the `angular.json` workspace build configuration file of your project to define locales for a project.

The following sub-options identify the source language and tell the compiler where to find supported translations for the project.

Suboption	Details
`sourceLocale`	The locale you use within the application source code (`en-US` by default)
`locales`	A map of locale identifiers to translation files

`angular.json` for `en-US` and `fr` example

For example, the following excerpt of an `angular.json` workspace build configuration file sets the source locale to `en-US` and provides the path to the French (`fr`) locale translation file.

Generate application variants for each locale

To use your locale definition in the build configuration, use the `"localize"` option in the `angular.json` workspace build configuration file to tell the CLI which locales to generate for the build configuration.

- Set `"localize"` to `true` for all the locales previously defined in the build configuration.
- Set `"localize"` to an array of a subset of the previously defined locale identifiers to build only those locale versions.
- Set `"localize"` to `false` to disable localization and not generate any locale-specific versions.

NOTE: Ahead-of-time (AOT) compilation is required to localize component templates.

If you changed this setting, set `"aot"` to `true` in order to use AOT.

Due to the deployment complexities of i18n and the need to minimize rebuild time, the development server only supports localizing a single locale at a time. If you set the `"localize"` option to `true`, define more than one locale, and use `ng serve`; then an error occurs. If you want to develop against a specific locale, set the `"localize"` option to a specific locale. For example, for French (`fr`), specify `"localize": ["fr"]`.

The CLI loads and registers the locale data, places each generated version in a locale-specific directory to keep it separate from other locale versions, and puts the directories within the configured `outputPath` for the project. For each application variant the `lang` attribute of the `html` element is set to the locale. The CLI also adjusts the HTML base HREF for each version of the application by adding the locale to the configured `baseHref`.

Set the `"localize"` property as a shared configuration to effectively inherit for all the configurations. Also, set the property to override other configurations.

`angular.json` include all locales from build example

The following example displays the `"localize"` option set to `true` in the `angular.json` workspace build configuration file, so that all locales defined in the build configuration are built.

Build from the command line

Also, use the `--localize` option with the `ng build` command and your existing `production` configuration. The CLI builds all locales defined in the build configuration. If you set the locales in build configuration, it is similar to when you set the `"localize"` option to `true`.

For more information about how to set the locales, see Generate application variants for each locale.

Apply specific build options for just one locale

To apply specific build options to only one locale, specify a single locale to create a custom locale-specific configuration.

Use the Angular CLI development server (`ng serve`) with only a single locale.

build for French example

The following example displays a custom locale-specific configuration using a single locale.

Pass this configuration to the `ng serve` or `ng build` commands. The following code example displays how to serve the French language file.

For production builds, use configuration composition to run both configurations.

Report missing translations

When a translation is missing, the build succeeds but generates a warning such as `Missing translation for message "{translation_text}"`. To configure the level of warning that is generated by the Angular compiler, specify one of the following levels.

Warning level	Details	Output
`error`	Throw an error and the build fails	n/a
`ignore`	Do nothing	n/a
`warning`	Displays the default warning in the console or shell	`Missing translation for message "{translation_text}"`

Specify the warning level in the `options` section for the `build` target of your `angular.json` workspace build configuration file.

`angular.json` `error` warning example

The following example displays how to set the warning level to `error`.

When you compile your Angular project into an Angular application, the instances of the `i18n` attribute are replaced with instances of the `$localize` tagged message string. This means that your Angular application is translated after compilation. This also means that you can create localized versions of your Angular application without re-compiling your entire Angular project for each locale.

When you translate your Angular application, the *translation transformation* replaces and reorders the parts (static strings and expressions) of the template literal string with strings from a collection of translations. For more information, see `$localize`.

tldr;

Compile once, then translate for each locale.

What's next

- Deploy multiple locales

@reviewed 2021-10-13