The Conversation was launched in 2011 in partnership with Australian universities. Later on, the platform was extended to support partnerships with institutions in the UK, USA and Africa. So far, for all regions, the site was presented in English.

Then in 2015 the technical team began preparing to launch The Conversation in France. It was our first foray into using the i18n features of Rails to support the presentation of our site in another language.

To facilitate translations from English into French we chose to use the PhraseApp service.

This post describes how our i18n workflow makes use of PhraseApp.

Updating English phrases

Within our Rails apps we update the English phrases using the standard Rails i18n features. So, for example, in our main application, config/locales/tc/en.yml contains English phrases for various i18n keys that are used within the Rails application code. We need a reliable way of ensuring that the equivalent French phrases are in config/locales/tc/fr_fr.yml.

Pushing to PhraseApp

Having updated the English phrases within the Rails config/locales files, the next step in our workflow is to make them available to PhraseApp. Each of our Rails applications has a Rake task which makes use of some common utility code to do this.

```
rake translations:push
```

When this task is run, the phraseapp-ruby RubyGem is invoked to upload the English locale file to PhraseApp via the PhraseApp API .

Each time we deploy, this rake task is run. This ensures that updates to English phrases are provided to PhraseApp in a timely fashion. It also minimises the likelihood of English phrases appearing on French pages while we wait for translations.

Translations within PhraseApp

Once the new or updated English phrases have been uploaded to PhraseApp, our colleague in Paris translates them. Due to Leighton Walter Kille's bilingual skills and diligence together with the time difference between Australia and France, we often find the French translations available the next morning!

Pulling from PhraseApp

In a similar fashion to pushing translations, a rake task pulls translations from PhraseApp via their API.

```
rake translations:pull
```

Running this task will update the French phrases within config/locales/tc/fr_fr.yml. Naturally, this is done on a Git branch.

Checking i18n Integrity

Without careful checking, mistakes can creep into the locale yml files. So our CI build includes a step which we call i18n-hygiene. This checks whether or not an i18n key is used in the codebase, whether interpolation variables are correctly encoded and several other tests. It's an important part of our workflow which deserves a post in its own right so I won't elaborate any further now.

Deploying

Once the GitHub pull request has a clean build on Buildkite and has been approved, it is merged into the master branch and deployed. Then the translations are visible on our French site

.

Conclusion

The combination of Rails i18n features, PhraseApp and custom i18n checking has served us well so far. Hopefully the infrastructure we have developed will enable us to smoothly prepare for the next language that The Conversation supports.

Authors: Keith Pitty, Developer, The Conversation

Read more http://theconversation.com/our-i18n-workflow-64960