Carpentries Internationalisation Handbook

The Carpentrie's i18n team

Apr 10, 2021

TRANSLATOR GUIDE

This guide documents the infrastructure to translate the Carpentries's lessons. In here we include a translator guide using Transifex and a maintainer guide to keep everything running.

And remember! also here you should follow our Code of Conduct.

CHAPTER

ONE

TRANSLATOR GUIDE

1.1 Getting started

Do you speak any other language besides English? Would you like to use that skill to translate any of the Carpentries' material? Then you are in the right place!

1.2 Format

Before getting into the translation in itself, please pay attention to the format from the source message.

1.2.1 Links

If the link is to an external page, and there's a version of that resource in the translated language, then change it to that one. For example, a link to a book

```
Deans for Impact: "[The Science of Learning]({{ page.root }}/files/papers/science-of-
↪learning-2015.pdf)"
```

that exists in Spanish would be:

```
Decanos por el Impacto: ["La Ciencia del Aprendizaje"](https://deansforimpact.org/wp-
↪content/uploads/2019/03/LA-CIENCIA-DEL-APRENDIZAJE_FINAL-DFI_1.pdf)
```

otherwise leave it to the original language.

If the link is different, Transifex will show a warning as you've translated a link. That's OK.

Internal links

Todo: Find how to "translate" these so it works in the rendered lesson.

1.2.2 New lines

Though not essential, try to keep the translation breaking the lines as the original text. However, it's important that you don't add a new line at the end of the translation if the source doesn't include one. Note that Transifex won't let you add the translation if you do so.

1.2.3 Spaces

Most of the material being translated in the Carpentries are written in markdown. Blank spaces at the beginning of the line are important! They mean that are part of a previous paragraph or section. Please, try to keep them as in the source language.

```
____please work in <https://github.com/carpentries/workshop-template>.
```

Should be translated into Spanish as:

```
____por favor trabaja en <https://github.com/carpentries-es/workshop-template-es>.
```

instead of:

```
por favor trabaja en <https://github.com/carpentries-es/workshop-template-es>.
```

1.2.4 YAML front matter block

These refer to the top blocks on Jekyll pages

```
---
layout: lesson
root: .
permalink: index.html
---
```

By default these should be kept without changes, except the content of tags that will be rendered like `title`, `questions`, ... but not the tag themselves. For example:

```
---
title: "Date and Time"
teaching: 10
exercises: 10
questions:
- "How can I know what day is today?"
objectives:
- "Write a program that calculate the number of hours between two dates."
keypoints:
- "Use `timedelta` to store find differences."
---
```

Should be translated in Spanish as:

```
---
title: "Fecha y Hora"
Teaching: 10
Exercises: 10
questions:
```

(continues on next page)

(continued from previous page)

```
- "¿Cómo puedo saber que día es hoy?"
objectives:
- "Escribir un programa que calcule el número de horas entre dos fechas."
keypoints:
- "Usar `timedelta` para encontrar diferencias."
---
```

1.2.5 Carpentries formatting tags

Many sections are fenced with tags to provide formatting on the rendered lessons, these tags should not be translated. For example:

```
{: .prereq}
```

Note that all of these tags uses are followed by blocks of text or code that are prepended by one or more > symbols. Keep the same alignment as in the original text.

1.2.6 Variables on code samples

Code samples and their references should use meaningful variable names (e.g., `age`, `temperature`) and when working on an international environment we should use a language that's common between all the collaborators. However, when learning we should remove the barriers we can to make it easier for learners, so when possible, we suggest variables are translated to the given language.

```
age = 42
first_name = 'Ahmed'
```

Translated into Spanish would be:

```
edad = 42
hombre = 'Ahmed'
```

Functions from libraries should be kept as is, however, if we are creating a function as part of the lesson they should be translated:

```
def calculate_area(radius):
    return np.pi * radius ** 2

plt.plot(heights, ages)
```

In Spanish would look like:

```
def calcular_area(radio):
    return np.pi * radio ** 2

plt.plot(alturas, edades)
```

1.3 Tools

1.3.1 Transifex

Transifex is a collaborative translation platform. You need an account to be able to contribute to the translations.

Depending of the role you've been assigned you can translate, revise, invite new contributors, etc.

Their documentation on how to use the web editor is short but very complete. You should take a look to familiarise with the platform and all its advantages.

We are keeping glossaries for different languages that can be used across different lessons to keep consistency. If you find a common term that's not there, add it!

If there's a sentence that you are not sure how to translate, or the meaning is unclear, use the comments and issues tool. We don't have enough experience yet to define when to create an issue or a comment, so feel free to experiment.

You can also download one or more files to *translate them offline* or from your favourite tool (some specified below). However, be aware that you won't see if someone is translating these files at the same time nor use the other tools that Transifex offers. These and other problems that they say in the guide are not going to happen too often if the lesson files are distributed between the translators.

1.3.2 POEDIT

POEDIT is a multi-platform desktop software. It has support to integrate with **`Crowdin`_** (a platform similar to Transifex). One of the advantages of using a software locally is that you can see some metadata that Transifex is not showing (e.g., comments made by the po4gitbook tool), however, comments you add in the file are not uploaded to the platform.

1.3.3 Gtranslator

Gtranslator is an open source desktop application that runs on GNOME (Linux) and is very simple but powerful.

1.3.4 Lokalize

Lokalize is the KDE cousin of Gtranslator.

1.3.5 OmegaT

OmegaT is a very powerful open source desktop application, it runs over different OSs and understand more than 30 file formats. It has many features and tools, for example it can interact with Google Translate, has support for right-to-left languages and much more.

CHAPTER

TWO

MAINTAINER GUIDE

2.1 Getting started

To be the maintainer of this infrastructure doesn't require that you speak other thing than English! However, you should be fluent with few technologies: `git`, `jekyll`, `sphinx`, `python` and `gettext`.

Are you interested to help? Keep reading the guide and introduce yourself on the internationalisation slack room (or matrix) and search for `DavidPS`.

2.2 Understanding the structure

At the moment there are translation support for two types of sources:

- carpentries lessons using jekyll (project 1)
- carpentries documentation using sphinx (project 4)

The other type of material is the lessons based on `rmarkdown`. There is the project 2 with issues to tackle the implementation of that type of lessons.

The goal is that everything is translated using gettext system. This system converts the source product into a `PO` file format that many translation software understand and it is therefore easier to update when the source changes and find what bits are new. All the issues related with the tooling and infrastructure is at the project 3.

2.2.1 The Carpentries-i18n organisation

This organisation includes, for now forks of all the lessons and other repositories with code needed to convert, automate and visualise the lessons and their translations.

Note: At the moment the material and all the repositories in this organisation is not officially under The Carpentries umbrella. However, the people behind this effort are part of the Carpentries' community.

The mains repositories are:

- i18n-handbook - The source of this documentation.
- i18n - The machinery to support the `jekyll` lessons.
- carp-theme - A Jekyll theme to render the lessons with multilingual support
- handbook-translations - Translations repository for the Carpentries handbook that uses sphinx.

2.3 Working with Jekyll lessons

Note: Some of the steps below needs to be run by someone with write access to the carpentries-i18n organisation.

2.3.1 Prepare the lesson

To prepare a jekyll-based lesson for translation we need few steps:

1. Fork the repository under the carpentries-i18n organisation.
2. Set-it up to use the Jekyll-theme (i.e., remove everything that's not lesson content).
3. Create it as a submodule within the i18n repository.
4. Generate the *po* files using po4gitbook software.
5. Since *po4gitbook* generates the whole lesson as a single file, break it into episodes with `splitpot.py`.
6. Create the project on Transifex and push the source files.

The first three steps can be done automatically using `lesson2theme.py` from i18n. You'll need to create an access token on GitHub first.

```
$ cd i18n
i18n (git)-[master]$ export gh_access_token=xxxxxxxx
i18n (git)-[master]$ python helpers/lesson2theme.py swcarpentry/python-novice-
↪gapminder
i18n (git)-[python-novice-gapminder]$
```

That command:

1. forks the repository,
2. fixes any recognised formatting issues that will affect the conversion to PO (e.g., commit 52cacd8),
3. removes everything that's provided by the theme (e.g., commit 682a376), and
4. updates `_config.yml` so it accepts multiple languages (e.g., commit f4e6e3b).
5. Then adds the forked repo as a submodule to i18n repository in a branch with the lessons name.

The output of the above command gives you the link of the forked repository so the result can be checked manually with a list of the following steps that are to be done manually.

The first of these steps is to run `po4gitbook` to update/generate the `po` files.

Warning: If the command gets stuck for more than a couple of seconds that's due to some formatting issues as the once fixed on the second step above.

Note: The `update.sh` goes through all the md files, finds whether there's been an update on the sources and propagates to new files, if they've changed tries to merge them and marks the blocks as fuzzy if the source has changed.

Now you should have a new file under the `po` directory:

```
i18n (git)-[python-novice-gapminder]$ po4gitbook/update.sh
...
i18n (git)-[python-novice-gapminder]$ ls po
...
python-novice-gapminder.pot
...
```

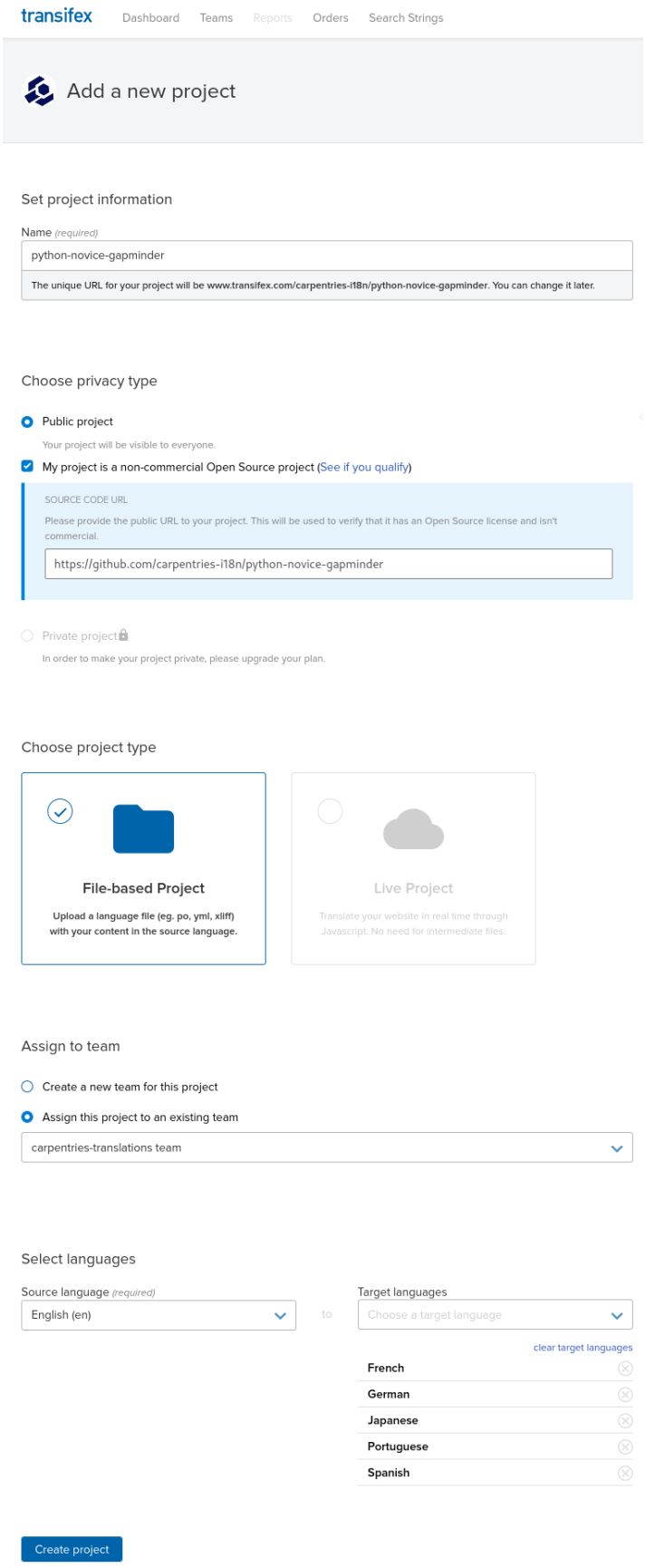
That `pot` file is the template that we will use to break it up into chunks first and then send these to transifex.

```
i18n (git)-[python-novice-gapminder]$ python helpers/splitpot.py po/python-novice-
↪gapminder.pot
...
i18n (git)-[python-novice-gapminder]$ ls transifex/python-novice-gapminder/pot
00__CODE_OF_CONDUCT.md.pot  06__03-types-conversion.md.pot  12__09-plotting.md.pot
↪       18__15-coffee.md.pot             24__about.md.pot      30__aio.md.pot
01__CONTRIBUTING.md.pot     07__04-built-in.md.pot           13__10-lunch.md.pot
↪       19__16-writing-functions.md.pot  25__design.md.pot     31__index.md.pot
...
```

Then we need to create the target language directory we want (e.g., `es` for Spanish), and let Transifex's command line tool (`tx`) to prepare the files

```
i18n (git)-[python-novice-gapminder]$ mkdir -p transifex/python-novice-gapminder/es
i18n (git)-[python-novice-gapminder]$ cd transifex/python-novice-gapminder
python-novice-gapminder (git)-[python-novice-gapminder]$ cd transifex/python-novice-
↪gapminder
python-novice-gapminder (git)-[python-novice-gapminder]$ tx config mapping-bulk -p␣
↪python-novice-gapminder --source-language en --type PO -f '.pot' \
               --source-file-dir pot --expression "<lang>/{filename}.po" --execute
```

The last command generates a `config` file under a hidden `.tx` directory. We need then to add the project in Transifex where we need to input a name (same as the lesson), select that's a public project, add the url of the project, select that's a file-based project, assign it to the `carpentries-translation` team and select the target languages (by default it adds all that we've used before).


transifex
Dashboard
Teams
Reports
Orders
Search Strings
Add a new project
Set project information
Name (required)
python-novice-gapminder
The unique URL for your project will be www.transifex.com/carpentries-i18n/python-novice-gapminder. You can change it later.
Choose privacy type
Public project
Your project will be visible to everyone.
My project is a non-commercial Open Source project (See if you qualify)
SOURCE CODE URL
Please provide the public URL to your project. This will be used to verify that it has an Open Source license and isn't commercial.
https://github.com/carpentries-i18n/python-novice-gapminder
Private project
In order to make your project private, please upgrade your plan.
Choose project type
File-based Project
Upload a language file (eg. po, yml, xliff) with your content in the source language.
Live Project
Translate your website in real time through Javascript. No need for intermediate files.
Assign to team
Create a new team for this project
Assign this project to an existing team
carpentries-translations team
Select languages
Source language (required)
English (en)
to
Target languages
Choose a target language
clear target languages
French
German
Japanese
Portuguese
Spanish
Create project

Once the project is created in transifex we can push the project using `tx`:

```
python-novice-gapminder (git)-[python-novice-gapminder]$  tx push -s --parallel
```

Once the upload has been completed, you should see the resources available in the project page in Transifex (e.g., python-novice-gapminder)

Finally, add the `<lesson>.pot` file to the repository and push it to GH for review and merge with `master`.

Warning: Some times (e.g., po4gitbook#6) you may encounter that something fails in the process. If you encounter a similar problem, please add it as an issue to the right repository. If you don't know which one is the correct one, then add it to i18n.

2.3.2 Bring the translations to the rendered page

Once a lesson has been translated on Transifex (or you want to see its progress), we can pull the files as:

```
i18n (git)-[python-novice-gapminder]$ language="es"
i18n (git)-[python-novice-gapminder]$ lesson="python-novice-gapminder"
python-novice-gapminder (git)-[python-novice-gapminder]$ pushd transifex/${lesson}
python-novice-gapminder (git)-[python-novice-gapminder]$ ls -F
es/  pot/
python-novice-gapminder (git)-[python-novice-gapminder]$ tx pull -l ${language} --
↪parallel
...
[########################] 100% (34/34)
tx INFO: Done.
```

The `tx` command line offers multiple options as to define a minimum percentage of translations (`--minimum-perc`), chose only files that has been reviewed (`--mode`). Check them with `tx pull --help`.

Once we have all the translated files downloaded we need to merge them to a single file for the whole lesson (i.e., the opposite of what we did *previously*).

```
python-novice-gapminder (git)-[python-novice-gapminder]$ popd
i18d (git)-[python-novice-gapminder]$ python helpers/splitpot.py po/${lesson}.pot --
↪join transifex/${lesson} --lang ${language}
```

Now we've got all the chunks into a single file named: `{lesson}.{language}.po` under the `po` directory.

Next we need to generate the markdown files. To do so we use the *po4gitbook* tool.

```
i18d (git)-[python-novice-gapminder]$ ../po4gitbook/compile.sh
```

Note: The `compile.sh` will throw errors if the format of the `po` file is not right (e.g., if there's not a period after each year in the translators list).

This command will generate all the markdown files of the lesson in a new directory under the `locale/{language}/lesson` directory.

The next step is integrate these new files with the original lesson so the source lesson and its translations get rendered under the same page. This requires the following steps:

1. move the new files to its own repository: `{lesson}-{language}` and upload it into github

1. Add that new repository as a sub-module of the source lesson: `{lesson}/_locale/{language}` 1. The website should now show both languages in the same page

This helper tool automates these steps

```
i18d (git)-[python-novice-gapminder]$ python helpers/trans2lesson.py locale/es/python-
↪novice-gapminder python-novice-gapminder
Repository https://github.com/carpentries-i18n/python-novice-gapminder.git updated␣
↪with es.
```

Todo: Find the way to Credit translators

CHAPTER

THREE

CONTRIBUTING TO THIS GUIDE

3.1 Download and build

You want to improve this guide? Let's start by building the documentation locally on your machine!

We will use, GitHub, git, Python, Python's venv module, and Sphinx.

Note: Some of the steps below assumes some level of familiarity with the tools. If you don't know where to start exactly, do not panic! Contact me and we can go through this together and find out what to add to make this guide even better.

3.1.1 Fork and clone the repository

When contributing to a project hosted on GitHub is a good practice to fork the project first under your username and clone your fork locally.

```
git clone https://github.com/<YOUR-USER-NAME>/i18n-handbook.git
cd i18n-handbook
```

It's also a good practice to keep an eye what happens `upstream` (source repository) by configuring a remote on your clone.

```
git remote add upstream https://github.com/carpentries-i18n/i18n-handbook.git
```

3.1.2 Creating an environment

To build the documentation locally it's recommended that you create an environment with the dependencies. For example on Linux or OSX you can do as:

```
python -m venv ~/.venv/i18n-sphinx
source ~/.venv/i18n-sphinx/bin/activate
pip install -r requirements.txt
```

refer to the venv tutorial to see how this is done on Windows.

3.1.3 Build the documentation

Once you've got your environment setup you can go ahead and build the documentation locally with:

```
sphinx-build -b html . _build
```

To see what you've built then use:

```
python -m http.server --directory _build/html
```

That will serve the page on your machine under http://localhost:8000 and you can see from your browser.

CHAPTER

FOUR

TODO LIST

This is a list of things that still need attention.

Todo: Find the way to Credit translators

original entry

Todo: Find how to “translate” these so it works in the rendered lesson.

original entry