Sphinx Documentation

Release 3.2.0

Georg Brandl

Jul 06, 2020

CONTENTS

CHAPTER

ONE

USING SPHINX

This guide serves to demonstrate how one can get started with Sphinx and covers everything from installing Sphinx and configuring your first Sphinx project to using some of the advanced features Sphinx provides out-of-the-box. If you are looking for guidance on extending Sphinx, refer to /development/index.

1.1 Internationalization

New in version 1.1.

Complementary to translations provided for Sphinx-generated messages such as navigation bars, Sphinx provides mechanisms facilitating *document* translations in itself. See the intl-options for details on configuration.

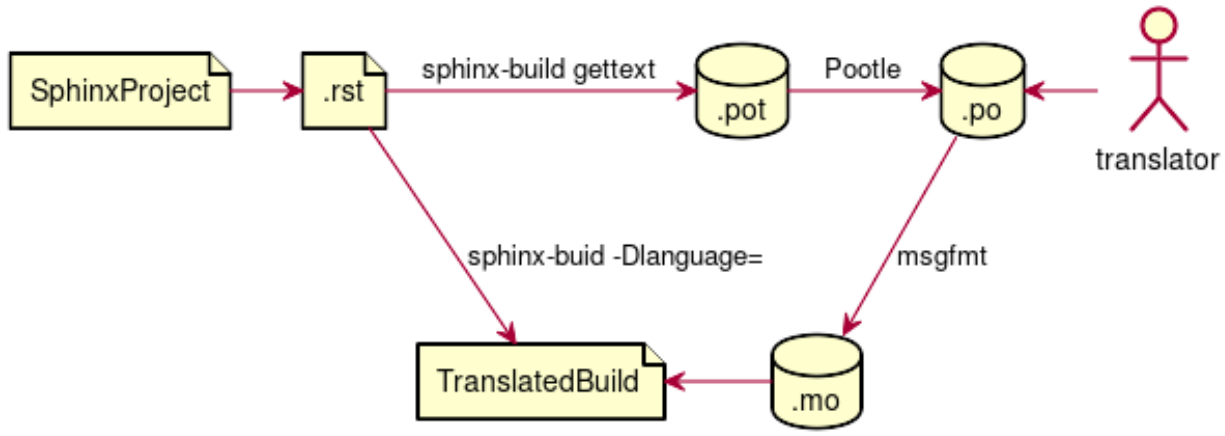

Fig. 1: Workflow visualization of translations in Sphinx. (The figure is created by plantuml[3].)

- *Sphinx internationalization details*
- *Translating with sphinx-intl*
  - *Quick guide*
  - *Translating*

[3] http://plantuml.com

1.1.1 Sphinx internationalization details

gettext[1] is an established standard for internationalization and localization. It naively maps messages in a program to a translated string. Sphinx uses these facilities to translate whole documents.

Initially project maintainers have to collect all translatable strings (also referred to as *messages*) to make them known to translators. Sphinx extracts these through invocation of `sphinx-build -b gettext`.

Every single element in the doctree will end up in a single message which results in lists being equally split into different chunks while large paragraphs will remain as coarsely-grained as they were in the original document. This grants seamless document updates while still providing a little bit of context for translators in free-text passages. It is the maintainer's task to split up paragraphs which are too large as there is no sane automated way to do that.

After Sphinx successfully ran the `MessageCatalogBuilder` you will find a collection of `.pot` files in your output directory. These are **catalog templates** and contain messages in your original language *only*.

They can be delivered to translators which will transform them to `.po` files — so called **message catalogs** — containing a mapping from the original messages to foreign-language strings.

gettext compiles them into a binary format known as **binary catalogs** through **`msgfmt`** for efficiency reasons. If you make these files discoverable with `locale_dirs` for your `language`, Sphinx will pick them up automatically.

An example: you have a document `usage.rst` in your Sphinx project. The *gettext* builder will put its messages into `usage.pot`. Imagine you have Spanish translations[2] stored in `usage.po` — for your builds to be translated you need to follow these instructions:

- Compile your message catalog to a locale directory, say `locale`, so it ends up in `./locale/es/LC_MESSAGES/usage.mo` in your source directory (where `es` is the language code for Spanish.)

```
msgfmt "usage.po" -o "locale/es/LC_MESSAGES/usage.mo"
```

- Set `locale_dirs` to `["locale/"]`.
- Set `language` to `es` (also possible via `-D`).
- Run your desired build.

[1] See the GNU gettext utilities[13] for details on that software suite.
[13] https://www.gnu.org/software/gettext/manual/gettext.html#Introduction
[2] Because nobody expects the Spanish Inquisition!

1.1.2 Translating with sphinx-intl

Quick guide

sphinx-intl[4] is a useful tool to work with Sphinx translation flow. This section describe an easy way to translate with *sphinx-intl*.

1. Install sphinx-intl[5].

```
$ pip install sphinx-intl
```

2. Add configurations to `conf.py`.

```
locale_dirs = ['locale/']   # path is example but recommended.
gettext_compact = False     # optional.
```

   This case-study assumes that BUILDDIR is set to `_build`, `locale_dirs` is set to `locale/` and `gettext_compact` is set to `False` (the Sphinx document is already configured as such).

3. Extract translatable messages into pot files.

```
$ make gettext
```

   The generated pot files will be placed in the `_build/gettext` directory.

4. Generate po files.

   We'll use the pot files generated in the above step.

```
$ sphinx-intl update -p _build/gettext -l de -l ja
```

   Once completed, the generated po files will be placed in the below directories:

   - `./locale/de/LC_MESSAGES/`
   - `./locale/ja/LC_MESSAGES/`

5. Translate po files.

   AS noted above, these are located in the `./locale/<lang>/LC_MESSAGES` directory. An example of one such file, from Sphinx, `builders.po`, is given below.

```
# a5600c3d2e3d48fc8c261ea0284db79b
#: ../../builders.rst:4
msgid "Available builders"
msgstr "<FILL HERE BY TARGET LANGUAGE>"
```

   Another case, msgid is multi-line text and contains reStructuredText syntax:

```
# 302558364e1d41c69b3277277e34b184
#: ../../builders.rst:9
msgid ""
"These are the built-in Sphinx builders. More builders can be added by "
":ref:`extensions <extensions>`."
msgstr ""
```

(continues on next page)

[4] https://pypi.org/project/sphinx-intl/
[5] https://pypi.org/project/sphinx-intl/

(continued from previous page)

```
"FILL HERE BY TARGET LANGUAGE FILL HERE BY TARGET LANGUAGE FILL HERE "
"BY TARGET LANGUAGE :ref:`EXTENSIONS <extensions>` FILL HERE."
```

Please be careful not to break reST notation. Most po-editors will help you with that.

6. Build translated document.

   You need a `language` parameter in `conf.py` or you may also specify the parameter on the command line.

   For for BSD/GNU make, run:

```
$ make -e SPHINXOPTS="-D language='de'" html
```

   For Windows **cmd.exe**, run:

```
> set SPHINXOPTS=-D language=de
> .\make.bat html
```

   For PowerShell, run:

```
> Set-Item env:SPHINXOPTS "-D language=de"
> .\make.bat html
```

Congratulations! You got the translated documentation in the `_build/html` directory.

New in version 1.3: **sphinx-build** that is invoked by make command will build po files into mo files.

If you are using 1.2.x or earlier, please invoke **sphinx-intl build** command before **make** command.

Translating

Update your po files by new pot files

If a document is updated, it is necessary to generate updated pot files and to apply differences to translated po files. In order to apply the updates from a pot file to the po file, use the **sphinx-intl update** command.

```
$ sphinx-intl update -p _build/locale
```

1.1.3 Using Transifex service for team translation

Transifex[6] is one of several services that allow collaborative translation via a web interface. It has a nifty Python-based command line client that makes it easy to fetch and push translations.

1. Install transifex-client[7].

   You need **tx** command to upload resources (pot files).

```
$ pip install transifex-client
```

[6] https://www.transifex.com/
[7] https://pypi.org/project/transifex-client/

See also:

Transifex Client documentation[8]

2. Create your transifex[9] account and create new project for your document.

   Currently, transifex does not allow for a translation project to have more than one version of the document, so you'd better include a version number in your project name.

   For example:

   Project ID `sphinx-document-test_1_0`

   Project URL `https://www.transifex.com/projects/p/sphinx-document-test_1_0/`

3. Create config files for **`tx`** command.

   This process will create `.tx/config` in the current directory, as well as a `~/.transifexrc` file that includes auth information.

```
$ tx init
Creating .tx folder...
Transifex instance [https://www.transifex.com]:
...
Please enter your transifex username: <transifex-username>
Password: <transifex-password>
...
Done.
```

4. Upload pot files to transifex service.

   Register pot files to `.tx/config` file:

```
$ cd /your/document/root
$ sphinx-intl update-txconfig-resources --pot-dir _build/locale \
  --transifex-project-name sphinx-document-test_1_0
```

   and upload pot files:

```
$ tx push -s
Pushing translations for resource sphinx-document-test_1_0.builders:
Pushing source file (locale/pot/builders.pot)
Resource does not exist.  Creating...
...
Done.
```

5. Forward the translation on transifex.

6. Pull translated po files and make translated HTML.

   Get translated catalogs and build mo files. For example, to build mo files for German (de):

```
$ cd /your/document/root
$ tx pull -l de
Pulling translations for resource sphinx-document-test_1_0.builders (...)
 -> de: locale/de/LC_MESSAGES/builders.po
```

(continues on next page)

[8] https://docs.transifex.com/client/introduction/
[9] https://www.transifex.com/

(continued from previous page)

```
...
Done.
```

Invoke **make html** (for BSD/GNU make):

```
$ make -e SPHINXOPTS="-D language='de'" html
```

That's all!

Tip: Translating locally and on Transifex

If you want to push all language's po files, you can be done by using **tx push -t** command. Watch out! This operation overwrites translations in transifex.

In other words, if you have updated each in the service and local po files, it would take much time and effort to integrate them.

1.1.4 Contributing to Sphinx reference translation

The recommended way for new contributors to translate Sphinx reference is to join the translation team on Transifex.

There is sphinx translation page[10] for Sphinx (master) documentation.

1. Login to transifex[11] service.
2. Go to sphinx translation page[12].
3. Click `Request language` and fill form.
4. Wait acceptance by transifex sphinx translation maintainers.
5. (After acceptance) Translate on transifex.

[10] https://www.transifex.com/sphinx-doc/sphinx-doc/
[11] https://www.transifex.com/
[12] https://www.transifex.com/sphinx-doc/sphinx-doc/