SPHINX: Documentation de projet avec Sphinx

Table of Contents

Installation

Installer Sphinx

```
pip install sphinx
```

Intégration de Markdown

Markdown est un langage de balisage léger avec une syntaxe simple pour la mise en forme du texte brut. Il existe de nombreuses variantes syntaxiques. Pour prendre en charge la documentation basée sur Markdown, installer **recommonmark** (un pont entre Docutils et **CommonMark-py** , un package Python permettant d'analyser la version CommonMark Markdown):

```
pip install -- upgrade recommonmark
```

Création d'une documentation

Une fois Sphinx installé, on peut configurer un projet Sphinx. Pour faciliter le processus de mise en route, Sphinx fournit un outil, **sphinx-quickstart** , qui générera un répertoire source de la documentation et le remplira de valeurs par défaut.

Configurer les sources de la documentation

Le répertoire racine d'une collection Sphinx de sources de documents est appelé le répertoire source . Ce répertoire contient également le fichier de configuration Sphinx conf.py , dans lequel on peut configurer tous les aspects de la façon dont Sphinx lit ces sources et construit une documentation.

Sphinx est livré avec un script appelé sphinx-quickstart qui configure un répertoire source et crée un conf.py par défaut avec les valeurs de configuration les plus utiles parmi quelques questions qu'il vous pose.

Pour commencer, cd dans le répertoire racine de la documentation et taper:

```
sphinx-quickstart
```

Entrer des valeurs pour les paramètres suivants (appuyer simplement sur Entrée pour accepter une valeur par défaut, indiqué entre parenthèses):

Prompt	**Choice**
Root path for the documentation [.]:	<ENTER>
Separate source and build directories (y/N) [n]:	y
Name prefix for templates and static dir [*]: | <ENTER> | | Project name: | an*example**pypi*project	
Author name(s):	Andrew Carter
Project version:	0.0.1
Project release [0.0.1]:	<ENTER>
Source file suffix [.rst]:	<ENTER>
Name of your master document (without suffix) [index]:	<ENTER>
autodoc: automatically insert docstrings from modules (y/N) [n]:	y
doctest: automatically test code snippets in doctest blocks (y/N) [n]:	n
intersphinx: link between Sphinx documentation of different projects (y/N) [n]:	y
todo: write "todo" entries that can be shown or hidden on build (y/N) [n]:	n
coverage: checks for documentation coverage (y/N) [n]:	n
pngmath: include math, rendered as PNG images (y/N) [n]:	n
jsmath: include math, rendered in the browser by JSMath (y/N) [n]:	n
ifconfig: conditional inclusion of content based on config values (y/N) [n]:	y
Create Makefile? (Y/n) [y]:	n
Create Windows command file? (Y/n) [y]:	n

Définir la structure du document

Sphinx-quickstart a créé un répertoire source avec **conf.py** et un document maître, **index.rst** (si on a accepté la valeur **Source file suffix** par défaut). La fonction principale du document maître est de servir de page d'accueil et de contenir la racine de «l'arborescence de la table des matières» (ou toctree). C'est l'une des principales choses que Sphinx ajoute, un moyen de connecter plusieurs fichiers à une même hiérarchie de documents.

Le “document maître” (désigné par **master_doc** dans le fichier conf.py) est la “racine” de la hiérarchie de la table des matières. Afin d'utiliser la directive **toctree** pour construire la table des matières, ce document doit être au format restructuredtext.

La directive toctree

La directive toctree est initialement vide et ressemble à ceci:

```
..toctree::
  :maxdepth: 2
```

Ajouter des documents en les énumérant dans le contenu de la directive:

```
..toctree::
  :maxdepth: 2

  intro
  Didacticiel
  ...
```

C’est exactement à quoi ressemble l’arborescence de cette documentation. Les documents à inclure sont indiqués sous la forme de noms de documents , ce qui signifie qu'on doit laisser l’extension de nom de fichier et utiliser des barres obliques comme séparateurs de répertoires.

Vous pouvez maintenant créer les fichiers que vous avez répertoriés dans toctree et ajouter du contenu. Les titres de leurs sections seront insérés (jusqu'au niveau «maxdepth») à l'emplacement où la directive toctree est placée. De plus, Sphinx connaît maintenant l’ordre et la hiérarchie des documents. (Ils peuvent contenir eux-mêmes des directives toctree , ce qui signifie que l'on peut créer des hiérarchies profondément imbriquées si nécessaire.)

Prise en charge de Markdown

Pour configurer un projet Sphinx pour la prise en charge de Markdown, procéder comme suit:

Ajouter recommonmark à config.py Sphinx

```
# for Sphinx-1.4 or newer
extensions = [ 'recommonmark' ]


# for Sphinx-1.3
from recommonmark.parser import CommonMarkParser
source_parsers = {
    '.md' : CommonMarkParser ,
}
```

Pour utiliser des fichiers Markdown avec des extensions autres que .md , ajuster la variable source_suffix . L'exemple suivant configure Sphinx pour analyser tous les fichiers avec les extensions .md et .txt en tant que Markdown:

```
source_suffix = {
    '.rst' : 'restructuredtext' ,
    '.txt' : 'markdown' ,
    '.md' : 'markdown' ,
}
```

Cela permet d'écrire les fichiers .md et .rst dans le même projet.

AutoStructify

AutoStructify permet d’écrire de convertir automatiquement une documentation Markdown en rST au moment de la compilation.

Pour utiliser la transformation avancée de Mardown en Rst, ajouter **AutoStructify** au fichier Sphinx conf.py.

```
# En haut de conf.py (avec les autres instructions d'import)
import recommonmark
from recommonmark.transform import AutoStructify

# En bas de conf.py
def setup ( app ):
    app . add_config_value ( 'recommonmark_config' , {
        'url_resolver' : lambda url : github_doc_root + url ,
        'auto_toc_tree_section' : 'Contents' ,
        }, True )
    app . add_transform ( AutoStructify )
```

AutoStructify est livré avec les options suivantes:

- **enable_auto_toc_tree**: active la fonctionnalité Auto Toc Tree.
- **auto_toc_tree_section**: lorsque la valeur est True, l’arborescence Auto Toc ne sera activée que sur la section qui correspond au titre.

- **enable_auto_doc_ref**: active la fonctionnalité de réf. auto doc. Obsolète
- **enable_math**: active la fonctionnalité de formule mathématique.
- **enable_inline_math**: active la fonctionnalité mathématique en ligne.
- **enable_eval_rst**: active l'évaluation de la fonctionnalité intégrée reStructuredText.
- **url_resolver**: une fonction qui mappe une position relative existante dans le document sur un lien http

Ajouter du contenu

Placer les fichiers sources de documentation dans le répertoire source de la documentation.

Lancer la construction

Maintenant qu'on a ajouté des fichiers et du contenu, on peut lancer une première construction de la documentation. Une compilation est lancée avec le programme sphinx-build , appelé ainsi:

```
sphinx-build -b html sourcedir builddir
```

où **sourcedir** est le répertoire source et **builddir** est le répertoire dans lequel on veut placer la documentation générée . L'option -b sélectionne un générateur. Dans cet exemple, Sphinx construira des fichiers HTML.

Cependant, le script sphinx-quickstart crée un fichier Makefile qui permet également de compiler la documentation, il suffit de lancer

```
make html
```

pour construire des documents HTML dans le répertoire de construction que l'on a choisi. Exécuter make sans argument pour voir quelles cibles sont disponibles.

L'option -b buildername

Est la plus importante, elle sélectionne un constructeur. Les constructeurs les plus courants sont:

- **html**: Construire des pages HTML. C'est le constructeur par défaut.
- **dirhtml**: Construire des pages HTML, mais avec un seul répertoire par document. Crée des URLs plus jolies (pas de .html) si elles sont servies depuis un serveur Web.
- **singlehtml**: Construire un seul HTML avec tout le contenu.
- **htmlhelp , qthelp , devhelp , epub:** Construire des fichiers HTML avec des informations supplémentaires pour créer une collection de documentation dans l’un de ces formats.
- **applehelp**: Construire un manuel d’aide Apple. Requiert hiutil et codesign , qui ne sont pas Open Source et ne sont actuellement disponibles que sur Mac OS X 10.6 et versions ultérieures.
- **latex**: Construire des sources LaTeX pouvant être compilées dans un document PDF à l'aide de pdflatex.
- **man**: Construire des pages de manuel au format groff pour les systèmes UNIX.

- **texinfo**: Construire des fichiers Texinfo qui peuvent être transformés en fichiers Info en utilisant makeinfo.
- **text**: Construire des fichiers de texte brut.
- **gettext**: Construire des catalogues de messages de style gettext (fichiers .pot).
- **doctest**: Exécuter tous les doctests de la documentation, si l'extension doctest est activée.
- **linkcheck**: Vérifier l'intégrité de tous les liens externes.
- **xml**: Construire des fichiers XML natifs de Docutils.
- **pseudoxml**: Créer des fichiers «pseudo-XML» compacts et joliment imprimés affichant la structure interne des arborescences de documents intermédiaires.

Publication dans jekyll

Par défaut, Jekyll ne construit aucun fichier ni répertoire qui

- sont cachés ou utilisés pour la sauvegarde (indiqués par des noms commençant par . ou # , ou se terminant par ~);
- contiennent le contenu du site (indiqué par des noms commençant par _); ou
- sont exclus dans la configuration du site .

Le dossier **_static** contenant le thème sphinx est donc considéré par Jekyll les considère comme des ressources spéciales et il ne le copie donc pas sur le site final.

Pour contourner ce problème, utiliser la directive **include de _config.yml** pour spécifier les fichiers à ne pas ignorer.

```
include: ['_pages', '_static']
```