Eskimon

Zeste de savoir

Comment (et pourquoi) j'ai intégré ZMarkdown à Pelican

15 janvier 2019

Table des matières

% COMMENT (ET POURQUOI) J'AI INTÉGRÉ ZMARKDOWN À PELICAN % Eskimon % 12 avril 2018

1. Introduction

J'expliquais récemment sur le forum que j'ai remis au goût du jour mon blog , notamment en revoyant ma chaîne de rédaction qui intègre désormais le parseur markdown de zeste de savoir : zmarkdown. Ce tuto-billet vous propose de voir la réflexion qui a accompagné ce choix, ainsi que le processus ayant permis de l'intégrer sans heurts dans le moteur de site statique Pelican.

2. Pourquoi ne pas se contenter du parseur md de base ?

C'est une bonne question ça ! Après tout, Pelican propose de base de faire de la rédaction en Markdown via le paquet python du même nom. Pourquoi ne pas s'en contenter ?

Et bien la réponse est assez simple. Je rédige depuis maintenant un moment sur ZdS et je me suis habitué à sa syntaxe *évoluée* du markdown. En effet, le markdown de Zeste de Savoir propose des choses intéressantes qui n'existe pas de base, comme l'ajout simple de vidéo directement embarqué (et non pas juste le lien), l'ajout de jsfiddle pour faire de la démo interactive de code ou encore la gestion des notes de bas de page. Bref, tout un tas de petites choses qui mise bout à bout font une grande différence.

Un autre aspect est que je souhaite pouvoir facilement partager mes rédactions entre mon blog et ZdS. Avoir la même chaîne de traitement des écrits me permet donc de gagner énormément de temps. En effet, je n'ai qu'à faire du copier/coller de mon blog vers ZdS ou bien importer directement le zip dans un outil de mon cru dans mon blog pour que l'import se fasse automagiquement. Bref, je me simplifie la vie et j'aime ça.

Enfin, ça aussi été un très bon exercice puisque grâce à ça j'ai pu plonger un peu dans le nouveau code de zmarkdown (en javascript), mais je reviens sur ce point dans une des parties de cet article.

Ah, *last but not least*, l'utilisation de zmarkdown me permettra à terme de publier aussi bien en html pour le blog qu'en pdf/epub pour proposer de la lecture hors-ligne, et ça c'est cool, pas besoin de s'embêter avec d'autres outils !

3. Mes premiers pas avec zmarkdown

Bon, on va pas se mentir, j'ai un peu galéré (IRC témoignera). Je suis assez novice en javascript et encore plus avec les outils modernes de ce dernier. Bref, j'ai appris des choses

Commençons, par le début, le dépôt se trouve à l'adresse suivante https ://github.com/zestedesavoir/zmarkdown . Du coup la première étape fût de le cloner, jusque là ça va .

3.1. Installation

Du coup première étape naturel, tentons de convertir un petit truc md en html, vu que c'est ça le but final. Alors allons-y gaiement, lisons la doc de zmd . Elle nous invite à installer node 8 et yarn. Du coup on cherche tout ca et on découvre que de base node propose de télécharger directement les binaires (https ://nodejs.org/en/download/). LoL me direz-vous. Étant sous Ubuntu j'aurais bien aimé une installation propre via mon gestionnaire de paquet. Du coup je garde mon sang-froid et découvre que plus bas dans la page on me propose un "via package manager". Ouf, je suis les liens et trouve effectivement mon bonheur : https ://nodejs.org/en/download/package-manager/#debian-and-ubuntu-based-linux-distributions . Pour Yarn c'est relativement similaire, on va sur le site et on trouve assez vite le lien qui va bien https ://yarnpkg.com/en/docs/install .

Les prérequis sont là, installons tout ca. La doc demande de cloner le dépôt, ça c'est fait, puis de faire `yarn`. Alors on s'exécute. Les paquets/dépendances s'installent, tout roule.

La doc parle ensuite de test, j'ignore ceci ne souhaitant pas tester le code lui-même.

Enfin, on nous présente une longue liste des paquets utilisés pour finir sur une note concernant la licence du code. Ok. Cool. Je fais quoi ? Petit appel à l'aide sur IRC et on relit tout pour finir par la liste qui parle du paquet *zmarkdown*, "Fully integrated package to be used in zeste de savoir website". Allons voir là dedans (Je découvrirais alors plus tard que le dossier package du dépot contient tout les paquets développés spécifiquement pour zmd, et que zmarkdown est un exemple du serveur en lui-même).

Je me retrouve alors dans le dossier zmarkdown . Le README commence bien, "zmarkdown server HTTP API", ouf ! Il attaque direct par une section "usage" (chouette) en disant de lancer la commande `npm run server`. Je ~~m~~l'exécute.

```
eskimon@eskimon-desktop:/tmp/zmarkdown/packages/zmarkdown$ npm run server

> zmarkdown@5.4.5 server /tmp/zmarkdown/packages/zmarkdown
> pm2 start -f server/index.js -i 3 --max-memory-restart 150M

[PM2] Starting /tmp/zmarkdown/packages/zmarkdown/server/index.js in cluster_mode (3 instances)
[PM2] Done.
```

App name	id	mode	pid	status	restart	uptime	cpu	mem	user	watching
index	0	cluster	28709	online	0	0s	49%	39.3 MB	eskimon	disabled
index	1	cluster	28715	online	0	0s	85%	34.6 MB	eskimon	disabled
index	2	cluster	28743	online	0	0s	75%	29.3 MB	eskimon	disabled

```
Use `pm2 show <id|name>` to get more details about an app
```

FIGURE 3. – npm run server

Ok, apparemment ça marche... Mais la doc ne précise pas sur quel port tout cela est servi !! À l'aide !! Les gentils compères sur IRC m'explique alors que le serveur tourne sur le port 27272. Allons voir...

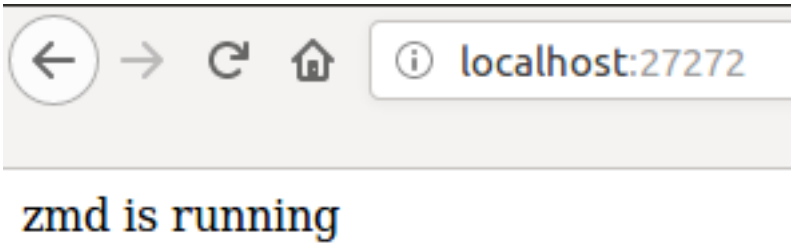

FIGURE 3. – zmd is running

Effectivement, ça marche !

3.2. Utilisation

En lisant la doc, on voit que le serveur sert plusieurs routes pour les différents types de contenu souhaité. On veut de l'html, donc on va aller faire une requête sur la route du même nom, `/html`. (cf la doc pour le contenu du json à envoyer).

```
1 $ curl -H "Content-Type: application/json" -X POST -d
      '{"md":"foo"}' http://localhost:27272/html
```

Ce à quoi le serveur nous répond gentiment (cf la doc pour mieux comprendre le contenu du json réponse),

```
1 ["<p>foo</p>",{"disableToc":true,"languages":[],"depth":1},[]]
```

Yes, on a du contenu converti !! Joie et bonheur !!

Il ne reste plus qu'à faire fonctionner ça avec pelican...

4. Intégration dans pelican

Maintenant que l'on sait convertir du markdown de ZdS (que j'appelerais ~~Quézac~~ *zmd*), il nous faut l'intégrer à Pelican pour que ce dernier sache comment l'utiliser.

4.1. Une histoire de Reader

Faut le reconnaître, Pelican s'est relativement bien foutu. Et en plus comme c'est open-source, on trouve facilement des exemples de code pour faire plein de choses. Ce fut le cas pour créer un nouveau *Reader* , c'est à dire une procédure de traitement pour avoir en entrée un texte avec des métadonnées et en sortie de l'html pour le site. Dans mon cas, le texte sera du zmd évidemment.

De base Pelican possède des readers pour plusieurs formats, comme le markdown (md) de base, le rst ou encore l'asciidoc. J'en ai donc créer un pour traiter le zmd. Histoire de gagner du temps et pour ne pas tout réinventer, je me suis inspiré de différents readers proposé par la communauté dans le dépôt pelican-plugins . Prenons par exemple multimarkdown_reader et la doc de pelican qui va bien.

On apprend tout d'abord qu'il faut hériter de `BaseReader` et avoir une variable `enable` à `True`. Ok facile. On lui renseignera aussi quelles extensions on se doit de traiter avec notre reader. Disons `zmd`. On a un début de classe comme ceci :

```
1 class ZmdReader(BaseReader):
2     enabled = True
3     file_extensions = ['zmd']
4 
5     def read(self, filename):
6         pass
```

Il ne reste "plus qu'à" coder la méthode read qui traitera un fichier en entrée puis donnera de l'html en sortie.

Pour rappel (ou pas), un fichier de contenu zmd aura la forme suivante :

```

```

On va donc séparer le traitement du fichier en deux étapes. Tout d'abord extraire les métadonnées (titre, date, description etc) puis convertir le zmd en html.

Grosso modo, voici le code de la fonction read. Le traitement du zmd se fait dans la fonction `_zmdtohtml()` ligne 17 et que l'on verra ensuite.

```python
def read(self, filename):
    with pelican_open(filename) as fp:
        text = list(fp.splitlines())

    # Split metadata from content
    metadata = {}
    for i, line in enumerate(text):
        meta_match = re.match(r'^([a-zA-Z_-]+):(.*)', line)
        if meta_match:
            name = meta_match.group(1).lower()
            value = meta_match.group(2).strip()
            metadata[name] = self.process_metadata(name, value)
        else:
            content = '\n'.join(text[i:])
            break

    output = self._zmdtohtml(content)

    return output, metadata
```

4.2. La conversion du zmd via le serveur idoine

Maintenant que l'on a isolé le texte au format zmd, il ne reste plus qu'à le donner à manger au serveur zmarkdown pour que ce dernier nous renvoi un beau contenu html. Et en python c'est archi simple !

En effet, il suffit finalement de simplement mettre le contenu du markdown dans du json...

```python
content = {'md': md}
json_content = json.dumps(content)
```

...de préparer une connection au serveur...

```python
connection = http.client.HTTPConnection('localhost', 27272)
headers = {'Content-type': 'application/json'}
```

... et d'envoyer/recevoir le tout !

```python
connection.request('POST', '/html', json_content, headers)
response = connection.getresponse()
return json.loads(response.read().decode())[0]
```

En résumé :

```
def _zmdtohtml(self, md):
    connection = http.client.HTTPConnection('localhost', 27272)
    headers = {'Content-type': 'application/json'}
    content = {'md': md}
    json_content = json.dumps(content)

    connection.request('POST', '/html', json_content, headers)

    response = connection.getresponse()
    return json.loads(response.read().decode())[0]
```

Et BOUM ! on a tout !

Enfin presque Il ne reste qu'à prévenir Pelican qu'il existe un nouveau reader en rajoutant les deux fonctions suivantes dans le fichier de notre nouvelle classe (mais pas dans la classe elle-même) :

```
def add_reader(readers):
    for ext in ZmdReader.file_extensions:
        readers.reader_classes[ext] = ZmdReader

def register():
    signals.readers_init.connect(add_reader)
```

et dire à Pelican où chercher notre reader en éditant la configuration (habituellement le fichier s'appelle pelicanconf.py) :

```
PLUGIN_PATHS = ['my-plugins']  # J'ai ajouté le fichier dans un
    dossier "my-plugins"
PLUGINS = ['zmd_reader']  # mon plugin reader s'appelle zmd_reader
```

Et voilà !!!

5. Conclusion

Une dernière chose n'est pas évoquée dans ce billet, l'étape de configuration de zmarkdown. En effet, le dossier https ://github.com/zestedesavoir/zmarkdown/tree/master/packages/zmarkdown/config possède des fichiers servant à personnaliser le rendu. J'ai donc passé pas mal de temps dedans pour obtenir les bonnes classes CSS ou les bons chemin de fichier pour les smileys par exemple. Mais à la fin tout roule une bonne fois pour toute, et ça, ça fait plaisir !

J'aimerais profiter de cet espace de conclusion pour remercier les dev de ZdS qui bossent sur zmarkdown. C'est un bel outil, un peu rude à personnaliser quand on est une quiche en javascript mais leur aide sur IRC a toujours été positive. Bref, des cœurs à eux, ils se reconnaîtront