Faculté
des Sciences
Département des Sciences Informatiques

UMONS
Université de Mons

Générateur Dynamique de Contenu Static

Projet de 1^{ere} Master réalisé par Maxime DE WOLF

Année académique 2017–2018

Directeur: Decan Alexandre

Service: Sciences Informatiques

Faculté des Sciences • Université de Mons • Place du Parc 20 • B-7000 Mons

Table des matières

1 Introduction

Dans le cadre du cours de lecture et rédactions scientifiques, il nous été demandé de réaliser un projet d'envergure relativement importante. Dans notre cas, il s'agit d'implémenter un générateur dynamique de contenu static. Ce rapport a pour but de vous présenter l'avancement de ce projet.

Dans un premier temps, nous dresserons le cahier des charges de ce projet. Cela consistera à lister les fonctionnalités nécessaires à respecter afin que ce projet soit mené à bien. Nous illustrerons également quelques cas d'utilisations afin de clarifier le cahier des charges.

Ensuite, nous listerons rapidement quelques générateurs de contenu static existants. Nous exhiberons leurs différences ainsi que leurs fonctionnalités.

Troisièmement, nous dresserons une liste de l'ensemble des fonctionnalités que devra posséder notre générateur de contenu static. Nous en profiterons également pour le comparer aux autres générateurs présenté en section 3. Cela nous permettra donc de justifier la nécessité de ce projet.

Enfin, nous parlerons des techniques mises en places pour répondre aux besoins présentés en section 4. En d'autres termes, cette section détaillera l'approche qui sera utilisée lors du développement de notre générateur de contenu static.

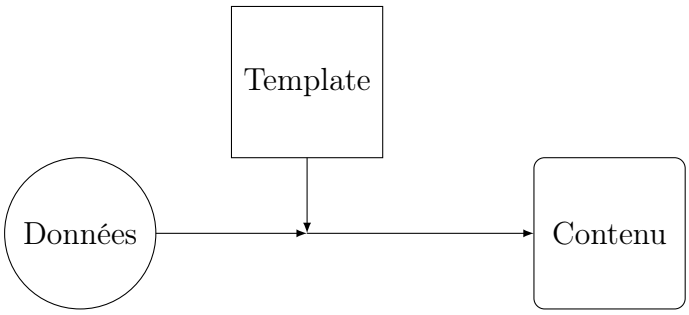

FIGURE 1 – Utilisation type d'un générateur de contenu static

2 Cahier des charges

Cette section a pour rôle de présenter le cahier des charges de ce projet. Nous introduirons également le concept de générateur de contenu static. La seconde partie de cette section listera quelques cas d'utilisation afin d'illustrer le fonctionnement souhaité par ce projet. Notez que les fonctionnalités en elles-même seront détaillées dans la section 4.

Le cahier des charges de ce projet est volontairement resté assez flou. En effet, les seules consignes données étaient de concevoir un générateur de contenu static qui donnait à l'utilisateur une grande souplesse d'utilisation. La simplicité d'utilisation du générateur est également un critère important. Afin de clarifier ce que veut dire "souplesse d'utilisation", quelques cas d'utilisations seront illustrer dans cette section.

2.1 Générateur de contenu static

Un générateur de contenu static sert -comme son nom l'indique- à générer du contenu. Il sera qualifié de "static" si ce contenu ainsi généré est prêt à l'emploi et ne nécessite plus de transformation afin d'être utile. A savoir que les générateurs de contenu static sont très utilisés pour produire des sites web statics. Celui-ci aura une vocation un peu plus générale car il permettra de générer à peu près n'importe quel type de contenu static. Leur utilisation est souvent lié au concept de *templates* qui permettent de séparer les informations de la mise en page en elle-même. La figure 1 montre une utilisation type de ce genre de générateur.

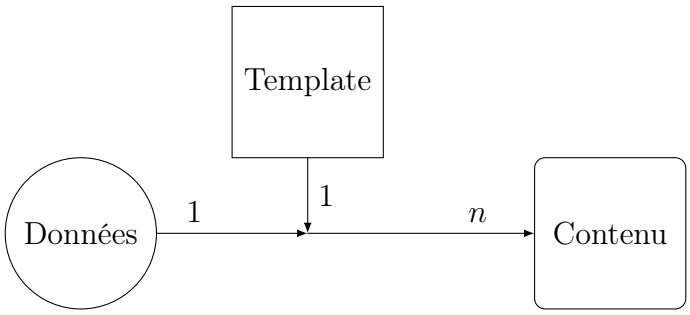

FIGURE 2 – Représentation du cas **OneToAll**.

2.2 Cas d'utilisations

Nous distinguons principalement trois cas d'utilisations propres aux générateurs de contenu statique. Chacun d'entre eux sera illustré à l'aide d'un schéma. Nous appellerons ces trois cas comme suit :

1. *OneToAll*
2. *AllToOne*
3. *AllToMany*

OneToAll

La cas *OneToAll* désigne l'utilisation d'un générateur de contenu static afin de générer plusieurs fichiers en sorties à partir d'un seul fichier en entré. Cela correspond, par exemple, à afficher un produit par page à partir d'un fichier contenant l'ensemble des produits disponibles. Cela se traduit par le schéma 2 en considérant que **n** est le nombre de produits contenus dans le fichier.

AllToOne

Le second cas, *AllToOne*, est le cas inverse de *OneToAll*. En effet, ce cas désigne l'utilisation d'un générateur de contenu static pour générer un seul fichier à partir de plusieurs. En pratique, cela consiste par exemple à afficher l'intégralité des posts d'un blog sur une seule page, chaque page étant symbolisée par un fichier. Cela donne le schéma 3 où **n** est le nombre de fichiers contenant un post.

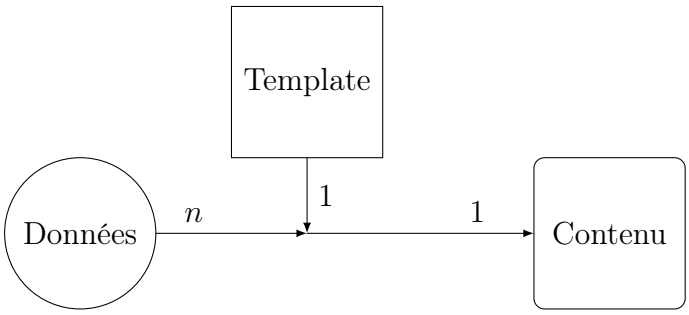

FIGURE 3 – Représentation du cas **AllToOne**.

AllToMany

3 Etat de l'art

Cette section détaille, de manière non-exhaustive, l'existence d'autres générateurs de contenu statique *similaires* à ce projet. Leurs fonctionnalités principales seront présentées afin d'être comparées avec notre propre générateur.

Pelican

Pellican [1] est un générateur de site web statique dont les principales fonctionnalités sont :

— Le support de *pages* (e.g. "Contact", ...)
— Le support d'*articles* (e.g. posts d'un blog)
— La régénération rapide de fichiers grâce à un système de caches et d'écriture sélective.
— La gestion de *thèmes* créés à partir de *templates jinja2*
— La publication d'articles dans plusieurs langues

Lektor

Lektor [2] est également un générateur de site web statique. Ses principales fonctionnalités sont les suivantes :

— Un système de construction intelligent qui ne reconstruit que le contenu qui a été modifié
— Un outil graphique qui permet la modification de pages sans toucher au code source
— Utilisation de système de *templates Jinja2* pour le rendu du contenu
— Un outil permettant la création relativement simple de site web multilingue

On peut donc en conclure que c'est un outil assez similaire à Pelican [1] sauf que Lektor [2] propose un outil graphique pour l'édition de contenu.

Jekyll

Enfin, Jekyll [3] est le plus connu des générateurs de sites web statiques *open source*. Il est écrit en *Ruby* à la différence de Pellican et Lektor, écrits en *Python*. Voici ses principales fonctionnalités :

— Utilisation de *templates Liquid*
— Support de contenu de type *pages* (e.g "Contact", "Accueil", ...) et *articles* (e.g. posts d'un blog)

4 Analyse fonctionnelle

4.1 Terminologie

- **Target :**
- **Template :**
- **Data :**
- **Output :**
- **Rule :**

5 Analyse technique

5.1 Configuration

— input dir
— output dir
— Rulebackend, TemplateBackend, ...
—

5.2 Structure

Surcharger le *RuleBackend* permet par exemple de changer la syntaxe.

Bibliographie

[1] Documentation de pellican. http://docs.getpelican.com/en/stable/.

[2] Page d'accueil de lektor. https://www.getlektor.com/.

[3] Présentation de jekyll. https://jekyllrb.com/.