

Générateur Dynamique de Contenu Static

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

Année académique 2017–2018

Directeur: Decan Alexandre

Service: Sciences Informatiques

Table des matières

1	Introduction	2
2	Cahier des charges	3
	2.1 Générateur de contenu static	3
	2.2 Cas d'utilisations	4
3	Etat de l'art	6
4	Analyse fonctionnelle	8
	4.1 Langage de programmation	8
	4.2 Programmation modulaire	8
	4.3 Règles	8
	4.4 Système de fichiers <i>logs</i>	8
5	Analyse technique	10
	5.1 Règles	10
	5.2 Configuration	10
	5.3 Structure	10

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.

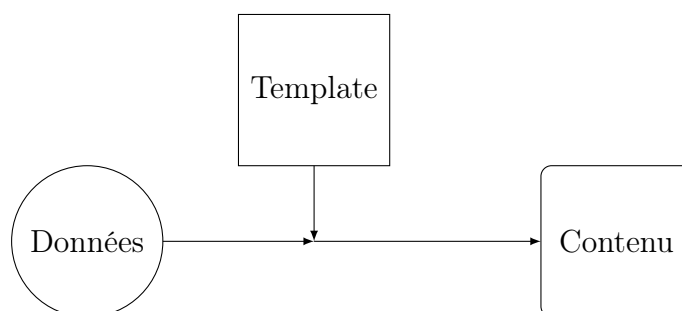


FIGURE 1 – Utilisation type d'un générateur de contenu static

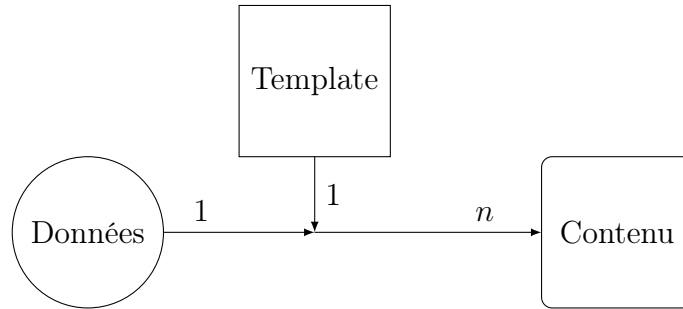
2 Cahier des charges

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-mêmes 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 : *OneToAll*, *AllToOne* et *AllToMany*.

OneToAll

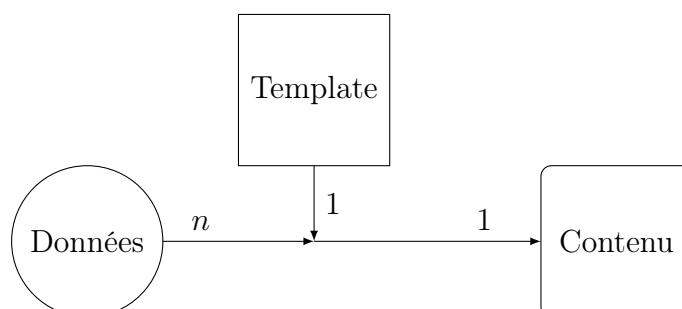
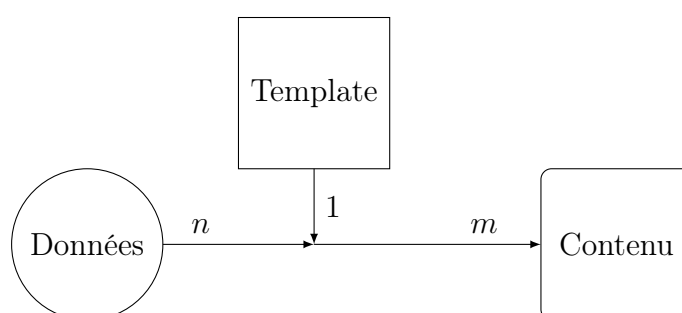
La cas *OneToAll* désigne l'utilisation d'un générateur de contenu static afin de générer plusieurs fichiers en sorties à partir d'un seul fichier en entré. Cela correspond, par exemple, à afficher un produit par page à partir d'un fichier contenant l'ensemble des produits disponibles. Cela se traduit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

AllToMany

Le dernier cas est celui que nous appellerons *AllToMany*. Ce cas est en fait une généralisation des deux cas précédant. Effectivement, *AllToMany* dé-

FIGURE 3 – Représentation du cas **AllToOne**.FIGURE 4 – Représentation du cas **AllToMany**.

signe l'utilisation d'un générateur de contenu static afin de générer plusieurs fichiers en sortis à partir de plusieurs fichier d'entrés. Mis en contexte, il désigne par exemple la génération de fichiers contenant chacun plusieurs posts à partir de fichiers contenant chacun un post. Le schéma 4 permet une vision plus claire de ce cas où **n** désigne le nombre de fichier contenant chacun un post et **m** le nombre de pages (resp. fichiers) contenant un certains nombre de posts.

Remarque 2.1:

Nous pourrions également discuter des cas d'utilisation où plusieurs templates seraient utilisés. Nous n'avons pas cru bon de le lister ce-dessus car nous n'avons trouvé aucun cas pratique correspondant à la production de fichiers contenant plusieurs fois les mêmes données mais agencées selon des templates différents. Nous discuterons tout de même de ce cas dans la section 4.

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

Pelican [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 sauf que Lektor 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 Pelican 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)

- Lancement d'un serveur local pour observer le rendu graphique des fichiers générés

Malgré leurs différences, chacun de ces générateurs de sites web statiques fonctionnent selon le cas d'utilisation *AllToOne* (voir figure 3). Ce cas est très utilisé dans le cas de sites web de type blog d'où sa popularité.

4 Analyse fonctionnelle

Cette section est destinée à justifier les choix de conception de ce projet afin remplir le cahier des charges. Certains choix seront inspirés directement de solution existantes listées dans la section précédente.

La section suivante, sera directement liée à celle-ci car elle expliquera comment ces choix seront techniquement mis en œuvres lors de la phase d'implémentation.

4.1 Langage de programmation

Afin de garantir la facilité d'utilisation de notre générateur de contenu static, notre de choix de langage de programmation s'est tourné vers *Python 3*. En effet, ce langage possède une syntaxe suffisamment flexible pour nous permettre de proposer à notre tour une syntaxe intuitive pour l'utilisateur.

4.2 Programmation modulaire

Pour ce qui est de la souplesse d'utilisation, nous appliquerons les concepts de la programmation modulaire. Cela permettra à l'utilisateur de modifier ou de créer facilement de nouvelles fonctionnalités pour notre générateur de contenu static. Nous fournirons tout de même les fonctionnalités de bases nécessaire au bon fonctionnement du projet.

4.3 Règles

Pour ce qui est de l'utilisation en elle-même de notre générateur de contenu static, nous avons pensé à un système de règles relativement simple. Ces règles proposeront un système de variables pour permettre à l'utilisateur de réutiliser la même valeur dans plusieurs champs de la même règle. Ces règles seront contenues dans un fichier qui sera passé en paramètre à notre programme qui les exécutera une par une. Chacune de ces règles symbolisera un des trois cas d'utilisation exposé en section 2 : *AllToOne*, *OneToAll* et *AlltoMany*. De plus amples informations sur la syntaxe de ces règles seront détaillées dans la section suivante.

4.4 Système de fichiers *logs*

Un système de fichiers *logs* permettrait de faciliter la correction de certaines règles lorsque les fichiers en sorties ne correspondent pas à ce que l'utilisateur attendait. Ces fichiers *logs* n'auront comme utilité que de lister

l'historique des règles exécutées pas à pas. Ce système viserait à atteindre l'objectif de simplicité d'utilisation -et d'aide à l'utilisation dans ce cas-ci- mentionné dans le cahier des charges en section 2.

5 Analyse technique

5.1 Règles

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

5.2 Configuration

- input dir
- output dir
- Rulebackend, TemplateBackend, ...
-

5.3 Structure

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

Bibliographie

- [1] Documentation de pelican. <http://docs.getpelican.com/en/stable/>.
- [2] Page d'accueil de lektor. <https://www.getlektor.com/>.
- [3] Présentation de jekyll. <https://jekyllrb.com/>.