Construire un thème pour Pelican

Sources :

- Sur le site de **Pelican** : Documentation officielle
- Sur le site de **Jinja** : Documentation officielle **Jinja**

Fonctionnement général :

A l'exécution de la commande d'élaboration du contenu, **Pelican** va parcourir et analyse l'ensemble des fichiers **Markdown** afin d'instancier un certain nombres d'objets.

Une fois le parcours des fichiers **Markdown** terminé, **Pelican** va élaborer l'ensemble des pages à l'aide du moteur de template **Jinja**.

Jinja reçoit pour chaque page du thème une collection des objets élaborés par **Pelican**. **Jinja** construit l'ensemble des page `html`.

Structure des dossiers :

Pelican s'attend à trouver un certain nombre de fichiers afin de construire les pages `html` avec **Jinja**.

Chaque thème doit comporter 2 dossiers : `static` et `template`

- `static` : Contient tous les fichiers dits *statiques* tel que les images, les fichiers *css*, etc.
- `templates` : Contient tous les fichiers *template* au format *html* qui seront utilisés pour générer l'ensemble des pages du site.

```
├── static
│   ├── css
│   └── images
└── templates
    ├── archives.html         // Affichage des archives
    ├── article.html          // Page pour la génération de chaque article
    ├── author.html           // Pour la génération de la page de chaque auteur
(contiendra tous les articles de l'auteur)
    ├── authors.html          // Génération de la page qui liste tous les auteurs
    ├── categories.html       // Génération de la page qui liste toutes les
catégories
    ├── category.html         // Génération d'une page par catégorie et qui permet
d'accéder à chaque article associée à la catégorie
    ├── index.html            // Page d'accueil
    ├── page.html             // processed for each page
    ├── period_archives.html  // Page d'affichage des périodes d'archivage
    ├── tag.html              // Page générée pour chaque tag
    └── tags.html             // Page qui listera tous les tags utilisés
```

Les variables communes :

Tous les fichiers du template peuvent accéder aux variables suivantes :

VARIABLE	DESCRIPTION
`output_file`	Nom du fichier courant, ex: `index.html` si **Pelican** génère la page d'accueil.
`articles`	Liste des articles, triés par l'ordre décroissant de leur date.
`dates`	La même liste d'articles triés par date.
`hidden_articles`	Liste des articles cachés. (un article est caché par son tag : `hidden`).
`drafts`	Liste des articles en brouillon (article avec le tage : `draft`).
`period_archives`	Un dictionnaire qui contient tous les éléments relatif à une période.
`authors`	Une liste contenant des tuples (auteur, articles).
`categories`	Une liste de tuples (categorie, articles).
`tags`	Une liste de tupes (tag, articles).
`pages`	La liste des pages.
`hidden_pages`	La liste des pages cachées.
`draft_pages`	La liste des pages en brouillon.

Variables par page du template :

Variables de `index.html` :

Une unique page `index.html` est généré. Par contre si la pagination est activée, alors il y a plusieurs pas d'index numérotée selon la convention : `index{number}.html`.

VARIABLE	DESCRIPTION
`article_paginator`	Un objet de pagination qui liste les articles.
`articles_page`	La page courant de l'article.
`articles_previous_page`	La page précédente de l'article courant, retourne `none` si la page n'existe pas.
`articles_next_page`	La page suivante de l'article suivante, retourne `none` si la page n'existe pas.
`dates_paginator`	Un objet de pagination qui liste les articles par ordre de date croissante.
`dates_page`	La page courant des articles, classé par odre croissant des dates.
`dates_previous_page`	La page suivante des articles, classé dans l'ordre croissant des dates.
`dates_next_page`	La page précédente des articles, classé dans l'odre croissant des dates.
`page_name`	Nom de la page.

Variables de `author.html` :

Une page est générée pour chaque auteur selon le schéma déclaré dans la constante `AUTHOR_SAVE_AS`, par défaut : `author/{slug}.html`. Si la pagination est activée, alors le schéma devient : `author/{slug}{number}.html`.

VARIABLE	DESCRIPTION
`authors`	Le nom de l'auteur courant.
`articles`	Les articles de l'auteur courant.
`dates`	Les articles de l'auteur courant mais dans l'odre croissant des dates.
`article_paginator`	Un objet de pagination qui liste les articles.
`articles_page`	La page courant de l'article.
`articles_previous_page`	La page précédente de l'article courant, retourne `none` si la page n'existe pas.
`articles_next_page`	La page suivante de l'article suivante, retourne `none` si la page n'existe pas.
`dates_paginator`	Un objet de pagination qui liste les articles par ordre de date croissante.
`dates_page`	La page courant des articles, classé par odre croissant des dates.
`dates_previous_page`	La page suivante des articles, classé dans l'ordre croissant des dates.
`dates_next_page`	La page précédente des articles, classé dans l'odre croissant des dates.
`page_name`	Nom de la page.

Variables de `category.html` :

Une page est généré pour chaque catégorie. Le schéma est défini par la constante : `CATEGORY_SAVE_AS`, par défaut : `category/{slug}.html`. Si la pagination est active le schéma devient : `category/{slug}{number}.html`

VARIABLE	DESCRIPTION
`category`	Le nom de la catégorie courante.
`articles`	Les articles de l'auteur courant.
`dates`	Les articles de l'auteur courant mais dans l'odre croissant des dates.
`article_paginator`	Un objet de pagination qui liste les articles.
`articles_page`	La page courant de l'article.
`articles_previous_page`	La page précédente de l'article courant, retourne `none` si la page n'existe pas.
`articles_next_page`	La page suivante de l'article suivante, retourne `none` si la page n'existe pas.
`dates_paginator`	Un objet de pagination qui liste les articles par ordre de date croissante.

VARIABLE	DESCRIPTION
`dates_page`	La page courant des articles, classé par odre croissant des dates.
`dates_previous_page`	La page suivante des articles, classé dans l'ordre croissant des dates.
`dates_next_page`	La page précédente des articles, classé dans l'odre croissant des dates.
`page_name`	Nom de la page.

Variables de `article.html`

Ce template est appliqué pour chaque article. La sortie est générée selon le schéma déclaré dans la constante : `ARTICLE_SAVE_AS`. Par défaut le schéma est : `{slug}.html)`.

Toutes les métédonnées déclarées dans l'article sont disponibles dans le template. Le nom de la catégorie est formaté au format lowercase.

L'article dispose de 2 variables :

VARIABLE	DESCRIPTION
`article`	L'objet article concerné.
`category`	Le nom de la catégorie de l'article courant.

Variables de `page.html` :

Le template est appliqué pour chaque page selon le schéma de nommage de la constante : `PAGE_SAVE_AS`. Par défaut le schéma est : `pages/{slug}.html`.

VARIABLE	DESCRIPTION
`page`	L'objet page affiché. Nous pouvons accéder à son titre, slug et au contenu.

Variables de `tag.html` :

Ce template est appliqué pour chaque tag. Le nom correspond au schéma de la constante `TAG_SAVE_AS`. Le schéma par défaut : `tag/{slug}.html`. Si la pagination est activée, le schéma devient : `tag/{slug}{number}.html`.

VARIABLE	DESCRIPTION
`tag`	Le nom du tag concerné.
`articles`	Les articles de l'auteur courant.
`dates`	Les articles de l'auteur courant mais dans l'odre croissant des dates.
`article_paginator`	Un objet de pagination qui liste les articles.
`articles_page`	La page courant de l'article.

VARIABLE	DESCRIPTION
`articles_previous_page`	La page précédente de l'article courant, retourne `none` si la page n'existe pas.
`articles_next_page`	La page suivante de l'article suivante, retourne `none` si la page n'existe pas.
`dates_paginator`	Un objet de pagination qui liste les articles par ordre de date croissante.
`dates_page`	La page courant des articles, classé par odre croissant des dates.
`dates_previous_page`	La page suivante des articles, classé dans l'ordre croissant des dates.
`dates_next_page`	La page précédente des articles, classé dans l'odre croissant des dates.
`page_name`	Nom de la page.

Variables de `period_archives.html` :

Ce template est appliqué pour chaque année si la constante `YEAR_ARCHIVE_SAVE_AS` est définie. Chaque mois si la constante `MONTH_ARCHIVE_SAVE_AS` est définie et de même chaque jours pour la constante : `DAY_ARCHIVE_SAVE_AS`.

VARIABLE	DESCRIPTION
`period`	Un tuple de la forme (année, mois, jour) qui indique la période concernée.
	Les valeurs sont de strings.
`period_num`	Un tuple comme `period` mais avec des valeurs de type entier.

Lister et atteindre une période d'archives :

La variable `period_archives` est utilisée pour générer une liste de liens pour des lots de périodes d'archivages. Cette variable commune peut-être utilisée pour générer l'affichage d'éléments de navigation.

VARIABLE	DESCRIPTION
`period`	Un tuple de la forme (année, mois, jour) qui indique la période concernée.
	Les valeurs sont de strings.
`period_num`	Un tuple comme `period` mais avec des valeurs de type entier.
`url`	L'URL de la période d'archivage de la page. Défini par `*_ARCHIVE_URL`.
`save_as`	Le chemin de la page correspondant à la période d'archivage.
`articles`	Une liste des objets Articles correspondant à la période.
`dates`	La même liste d'articles mais classés selon la constante `NEWEST_FIRST_ARCHIVES`.

Description des objets :

Les objets de la classe `Article` :

ATTRIBUT	DESCRIPTION
`author`	Objet `Author` de l'article.
`authors`	Une liste des objets `Authors` de cette article.
`category`	L'objet `Category` de l'article.
`content`	Le rendu du contenu de l'article.
`date`	Objet `Datetime` qui représente la date de l'article.
`date_format`	Format du format par défaut ou formalt de la date locale.
`default_template`	Nom du template par défaut.
`in_default_lang`	Booléen représentant si l'article est écrit ou pas dans le langage par défaut.
`lang`	Langue de l'article.
`locale_date`	Date formaté par `date_format`.
`metadata`	Dictionnaire des données d'entête de l'article.
`save_as`	Localisation de la page de l'article.
`slug`	Slug de la page.
`source_path`	Chemin complet de l'article source.
`relative_source_path`	Chemin relatif de l'article source.
`status`	Le statut de l'article : `published` ou `draft`.
`summary`	Rendu du sommaire de l'article.
`tags`	Liste des objets `Tag`.
`template`	Nom du template utilisé pour le rendu.
`title`	Titre de l'article.
`translations`	Liste des traductions de l'objet `Article`.
`url`	URL de la page de l'article.

Les objets des classes `Author, Category & Tag` :

La représentation en String des objets est l'attribut `name`.

ATTRIBUT	DESCRIPTION
`name`	Nom de l'objet concerné.
`page_name`	Nom de la page de l'auteur.
`save_as`	Localisation de la page.
`slug`	Slug de la page.

ATTRIBUT	DESCRIPTION
`url`	URL d'accès à la page.

Les objets de la classe `Page` :

ATTRIBUT	DESCRIPTION
`author`	Objet `Author` de la page.
`content`	Le rendu du contenu de la page.
`date`	Objet `Datetime` qui représente la date de l'article.
`date_format`	Format du format par défaut ou formalt de la date locale.
`default_template`	Nom du template par défaut.
`in_default_lang`	Booléen représentant si l'article est écrit ou pas dans le langage par défaut.
`lang`	Langue de la page.
`locale_date`	Date formaté par `date_format`.
`metadata`	Dictionnaire des données d'entête.
`save_as`	Localisation du fichier de la page.
`slug`	Slug de la page.
`source_path`	Chemin complet de la page source.
`relative_source_path`	Chemin relatif de la page source.
`status`	Le statut de la page : `published`, `draft` ou `hidden`.
`summary`	Rendu du sommaire du contenu.
`tags`	Liste des objets `Tag`.
`template`	Nom du template utilisé pour le rendu.
`title`	Titre de la page.
`translations`	Liste des traductions de l'objet `Article`.
`url`	URL de la page de la page.