Les thèmes WordPress

Là aussi pour le choix du thème faites attention d'en trouver un souvent mis à jour et bien noté. Il ne faudrait pas qu'une faille de sécurité soit présente dans votre thème.

Ici on va voir comment créer son thème depuis 0, mais il faut savoir qu'il existe de nombreux outils pour aider à la création de thème wordpress sans passer par du code.

Créer son propre thème

Pour créer son propre thème nous allons nous rendre dans le dossier `wp-content/themes`, ici se trouvent tout les thèmes installés.

Minimum requis

Créons un nouveau dossier qui accueillera les fichiers suivants :

- functions.php
- index.php
- style.css
- screenshot.png

Pour le screenshot, on en fera un veritable une fois le thème terminé, mais en attendant n'importe quelle image en `880/660` fera l'affaire.

`index.php` va commencer par accueillir un html de base :

```
<!DOCTYPE html>
<html>
<head></head>
<body>
    <h1>Coucou !</h1>
</body>
</html>
```

Pour le `style.css` on va y placer pour l'instant un unique commentaire, mais ce commentaire aura son importance pour wordpress.

```
/*
Theme Name: DwwmTheme
Theme URI: https://github.com/NolwennWM/dwwm07
Author: Nolwenn WEBER-MARQUISET
Author URI: http://www.marquiset.fr/
Description: Apprenons à faire un thème
Requires at least: WordPress 6.0
Version: 0.1
*/
```

Attention de ne pas mettre d'espace avant ":". Wordpress va venir lire ces informations pour les afficher dans la liste des thèmes.

Enfin `functions.php` va accueillir le fonctionnement de notre thème :

```
// Ajouter la prise en charge des images mises en avant
add_theme_support('post-thumbnails');

// Ajouter automatiquement le titre du site dans l'en-tête du site
add_theme_support('title-tag');
```

Une fois cela fait, nous pouvons nous rendre dans notre interface wordpress et activer notre thème.

Header et Footer

Maintenant que nous avons notre minimum requis, nous allons créer deux nouveaux fichiers :

- header.php
- footer.php

Le header et le footer ne change peu ou pas d'une page à l'autre, donc pour éviter de nous répéter, nous allons les placer dans des fichiers séparés qui seront appelé dans chacune de nos pages.

Pour le header :

```
<!DOCTYPE html>
<html <?php language_attributes(); ?>>
<head>
    <meta charset="<?php bloginfo('charset'); ?>">
    <meta name="viewport" content="width=device-width, initial-scale=1"/>

    <?php wp_head(); ?>
</head>

<body <?php body_class(); ?>>

    <?php wp_body_open(); ?>
```

et pour le footer :

```
<?php wp_footer(); ?>
</body>
</html>
```

Enfin revenons à notre `index.php` et changeons tout par :

```php
<?php get_header(); ?>

<h1>Coucou</h1>

<?php get_footer(); ?>
```

Les fonctions ajoutés ici sont assez parlante, et ce sont celles que nous utiliserons à chaque nouvelle page.

Si nous regardons notre site wordpress nous devrions maintenant avoir la barre de wordpress qui apparaît en haut ainsi qu'un head bien plus fourni si nous l'inspectons.

Maintenant revenons sur les fonctions que l'on a ajouté :

- `language_attributes()` Celle ci permet d'afficher l'attribut de langue pour le document paramétré dans `Réglage/Général/Langue du site`;
- `bloginfo('charset')` Va indiquer l'encodage du site. (par défaut UTF-8) Mais elle peut aussi servir à aller chercher d'autres informations utiles. La liste est disponible dans la documentation : https://developer.wordpress.org/reference/functions/bloginfo/
- `wp_head()` Celle ci est capitale, elle va permettre d'afficher toute les balises styles et script entre autres choses. Donc c'est grâces à elle que vos extensions et thèmes vont pouvoir fonctionner.
- `body_class()` Défini les classes du body, on y retrouvera :
  - `home` pour la page d'accueil
  - `blog` pour afficher les articles
  - `logged-in` pour un utilisateur connecté
  - `admin-bar` pour afficher la barre d'administration
  - `no-customize-support` pour que la barre ne soit pas touché par notre thème.
- `wp_body_open()` Donne accès au début du body pour les extensions qui souhaites inclure des balises ici.
- `wp_footer()` Le même effet que wp_head() mais pour le bas de page.

Header

Profitons en pour ajouter un petit header à notre site, pour cela ajoutons un dossier `img` dans lequel on ira placer un fichier `logo.svg`, puis ajoutons le code suivant à notre `header.php`:

```php
<header class="header">
    <a href="<?php echo home_url( '/' ); ?>">
        <img src="<?php echo get_template_directory_uri(); ?>/img/logo.svg"
alt="Logo">
    </a>
</header>
```

Dans ce header, deux nouvelles fonctions apparaissent:

- `home_url()` Permettant de créer un lien vers notre page d'accueil.
- `get_template_directory_uri()` Pour obtenir l'adresse absolue vers le dossier où est stocké notre template.

Page et contenu

Pour pouvoir développer notre thème, il serait plus simple d'avoir quelques pages et du contenu à afficher.

Pour cela allons créer quelques catégories comme "sport, voyage, cuisine, cinéma...", voir `Articles/Catégories`.

Puis allons créer 3 pages via `Pages/Ajouter`;

- Accueil
- Blog
- Contact

Pour chacune écrivons un titre et un paragraphe. On pourra aussi supprimer la page d'exemple déjà présente. Mais on gardera la page sur la politique de confidentialité.

Par défaut wordpress affiche les articles sur la page d'accueil, nous allons pouvoir changer cela dans `Réglages/Lecture` en indiquant quel page sert de page d'accueil et laquelle affiche les articles.

Pour ce qui est des faux articles, nous n'allons pas créer tout à la main, mais installer une extension faite pour cela nommée `FakerPress`.

Une fois installé et activé, un menu apparaît sur la gauche, il va nous permettre d'ajouter de nouveaux articles.

Je vais lui demander de me créer 20 articles entre il y a un an et aujourd'hui et pour le reste, laisser les valeurs par défaut.

Maintenant nous pouvons voir ces articles créé dans notre liste d'article.

Il ne nous reste plus qu'à retourner travailler sur notre thème :

Template Hierarchy

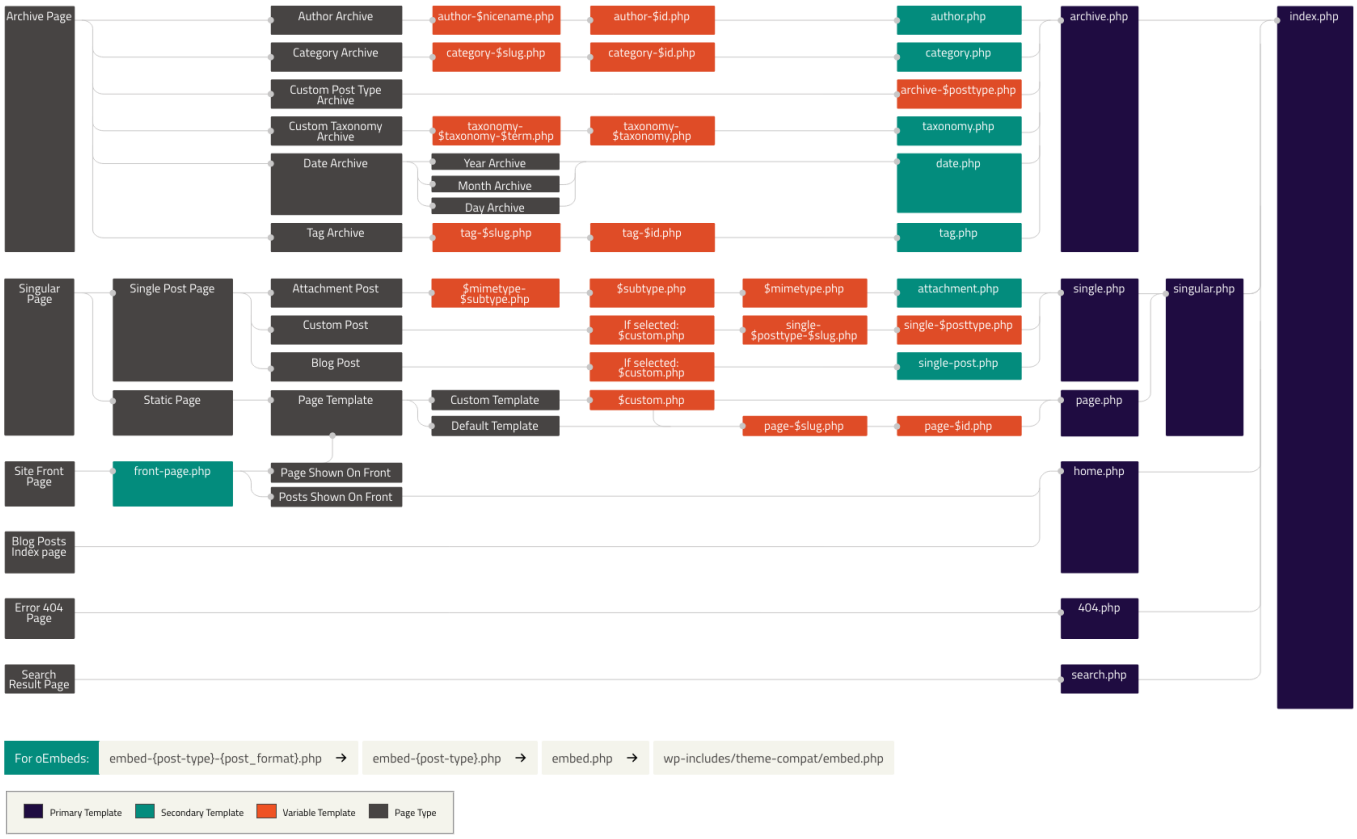
Wordpress divise les différentes pages d'un site en plusieurs modèles de page :

- `page` qui représente les pages standards
- `archive` qui listent les articles de blog (par catégorie ou non)
- `single` qui affiche un article seul.

Il va donc selon la page sélectionnée charger un modèle ou l'autre. Wordpress va suivre cet ordre :

1. Quelle page est chargée.
2. Il y a t'il un modèle spécifique à charger pour cette page.
3. Sinon il chargera le modèle correspondant au type de la page.
4. Si il n'a pas de modèle pour cette page, il chargera `index.php`

C'est d'ailleurs ce que fais actuellement notre site, il charge `index.php`. On peut retrouver dans la doc de wordpress l'image suivante :



En gris on voit le type de la page, en orange et en vert des cas plus spécifique et en bleu les cas généraux, les plus utilisé.

Et bien les noms données sur ce schema ne sont pas dû au hasard, il faut les prendre littéralement.

Concentrons nous sur les cas généraux en bleu, créons les fichiers suivants :

- archive.php
- front-page.php
- home.php
- page.php
- single.php

Et pour chacun d'entre eux, ajoutons le code suivant :

```
<?php get_header(); ?>
    <h1>Titre de la page</h1>
<?php get_footer(); ?>
```

Jusqu'ici notre page d'accueil était `index.php` mais maintenant c'est devenu `front-page.php`, cette dernière est d'ailleurs optionnelle, si elle n'est pas là, selon où vous affichez votre blog, l'accueil sera soit `home.php` soit `page.php`;

Si on inspecte notre page, on verra que d'une page à l'autre les classes de la balise body changent. Cela permet de gérer des cas spécifiques à tel ou tel type de page.

On notera que l'on utilise cela que ce soit pour une page classique autant que pour une page de blog. L'une affichera le titre de la page et son contenu uniquement et l'autre la totalité des articles.

Boucle wordpress

Comme en JS ou PHP, il nous arrive de devoir boucler sur des données. Wordpress va souvent avoir besoin de cela pour la liste des articles.

On utilisera la boucle wordpress pour gérer cela, venons modifier `front-page.php` ainsi que `page.php`, et ajoutons entre le header et le footer ceci :

```
<?php if( have_posts() ) : while( have_posts() ) : the_post(); ?>
    <h1><?php the_title(); ?></h1>
    <?php the_content(); ?>
<?php endwhile; endif; ?>
```

Revenons sur ces nouvelles fonctions :

- `have_posts()` retourne un Boolean indiquant si il reste des post à afficher.
- `the_post()` Prépare les données pour les templates tag en vue de l'affichage.
- `the_title()` Affiche le titre du post
- `the_content()` Affiche le contenu du post

Les templates tag sont des fonctions qui permettent d'afficher une partie d'un article. Il y en a tout un tas mais on en a déjà vu deux avec `the_title()` et `the_content()`.

Archive et Home

Rentrons un peu plus dans les détails avec `archive.php` et `home.php` où l'on va insérer entre le header et le footer le code suivant :

```
<h1>Archive/Home du blog</h1>

<?php if( have_posts() ) : while( have_posts() ) : the_post(); ?>

    <article class="post">
        <h2><?php the_title(); ?></h2>

        <?php the_post_thumbnail(); ?>

        <p class="post__meta">
            Publié le <?php the_time( get_option( 'date_format' ) ); ?>
            par <?php the_author(); ?> • <?php comments_number(); ?>
        </p>

        <?php the_excerpt(); ?>

        <p>
            <a href="<?php the_permalink(); ?>" class="post__link">Lire la
```

```
suite</a>
        </p>
    </article>

<?php endwhile; endif; ?>
```

On a poussé ici les choses un peu plus loin, voyons ce que l'on retrouve ici :

- `the_post_thumbnail()` Affiche l'image de mise en avant. Fonctionnalité qu'on a activé dans `functions.php` avec `add_theme_support('post-thumbnails')`
- `the_time( )` Affiche la date selon le format (Date PHP) en paramètre.
- `get_option( 'date_format' )` Récupère une option de wordpress et plus précisément, le format de date paramétré. Notons l'existence de `the_date()` qui affiche la date seulement si elle est différente de l'affichage précédent.
- `the_author()` Affiche le nom de l'auteur.
- `comments_number()` Affiche le nombre de commentaire. On peut venir lui donner optionnellement 3 arguments pour afficher un message différent selon si :
    1. Il n'y a pas de commentaire
    2. Il y a 1 commentaire
    3. Il y a plusieurs commentaires (on peut utiliser "%" pour indiquer le nombre)
- `the_excerpt()` Affiche un extrait de l'article. l'extrait est choisi selon si :
    1. Si un extrait a été fourni.
    2. Sinon si la balise "lire la suite" a été utilisé.
    3. Sinon il coupera après environ 55 mots. N'utilisez pas de balise p autour car il sera sûrement déjà dans une balise p.
- `the_permalink()` Affiche le lien de l'article. Utilisons le dans un href.

Single

Les articles en entier sont affiché sur `single.php`. Allons donc sur ce fichier et ajoutons lui :

```
<?php if( have_posts() ) : while( have_posts() ) : the_post(); ?>

    <article class="post">
        <?php the_post_thumbnail(); ?>

        <h1><?php the_title(); ?></h1>

        <div class="post__meta">
            <?php echo get_avatar( get_the_author_meta( 'ID' ), 40 ); ?>
            <p>
            Publié le <?php the_date(); ?>
            par <?php the_author(); ?>
            Dans la catégorie <?php the_category(); ?>
            Avec les étiquettes <?php the_tags(); ?>
            </p>
        </div>

        <div class="post__content">
```

```
            <?php the_content(); ?>
        </div>
    </article>

<?php endwhile; endif; ?>
```

Encore une fois de nouvelles fonctions font leur apparition :

- `get_avatar()` Récupère l'avatar de l'utilisateur dont l'id est donné en premier paramètre, et l'affiche avec une taille en pixel donnée en second paramètre.
- `get_the_author_meta()` Récupère l'information de l'auteur donnée en paramètre.
- `the_category()` affiche la catégorie de l'article.
- `the_tags()` affiche les étiquettes de l'article.

Notons que si vous souhaitez récupérer les informations sans les afficher directement, vous pouvez ajouter `get_` devant chaque fonction en `the_`;

Templates Parts

Soyons un peu DRY (Don't Repeat Yourself).

On avait mit le même code dans `archive.php` et `home.php`. Ce n'est pas très professionnel de se répéter ainsi, allons modifier `home.php`;

```
<?php get_template_part( 'archive' ); ?>
```

On a remplacé toute notre page par ceci. Cette fonction est un équivalent wordpress à "include" et "require". Les principales différences sont qu'on ne doit pas indiquer l'extension (pas de ".php"). Et que si aucun fichier ne correspond, il n'y aura pas d'erreur.

On peut tout à fait se servir de cette fonction pour appeler un template optionnel qui n'existe pas à tout les coups. Ou alors diviser des parties de code pour les réutiliser ailleurs ou juste pour alléger un fichier.

On notera qu'on peut créer des dossiers et aller chercher les fichiers dedans pour ne pas surcharger notre thème.

Déclarer des Styles et Scripts

Avec wordpress il vaut mieux évite d'insérer simplement les link et scripts directement dans le head.

On va préférer se rendre dans `functions.php` et y ajouter ce qui suis :

```
add_action( 'wp_enqueue_scripts', 'dwwm_register_assets' );

function dwwm_register_assets()
{
    wp_enqueue_style( "dwwm", get_stylesheet_uri());
}
```

Voyons ces nouvelles fonctions :

- `add_action()` Lors du crochet en premier argument, déclenche la fonction en second argument.
- `wp_enqueue_style()` Ajoute le fichier css en second argument au nom donné en premier argument. (chaque nom doit être unique)
- `get_stylesheet_uri()` Récupère le chemin du fichier "style.css" de base du thème.

On notera aussi l'existence de `wp_enqueue_script()` qui fonctionne comme son équivalent pour le style. Ainsi que `get_template_directory_uri()` qui permet de récupérer le chemin vers la racine du thème auquel on pourra concaténer le chemin vers les possibles autres fichiers css ou js que l'on souhaite ajouter.

On peut ajouter en troisième argument des fonctions `wp_enqueue_` un tableau indiquant les noms des fichiers après lesquels ils doivent être chargé si jamais l'ordre est important.

En quatrième argument on peut ajouter un string sous forme de numéro de version, il permet de passer outre le cache du navigateur car étant ajouté à l'adresse pour charger les fichiers, si il change, le navigateur pensera qu'il doit charger un nouveau fichier.

Dans le cas d'un thème enfant, nous remplaceront `get_template_directory_uri()` par `get_stylesheet_directory_uri()`;

Les thèmes enfants

La bonne pratique lorsque l'on veut modifier un thème existant, c'est de créer un thème enfant.
Ce thème héritera de tous son parent, et c'est celui ci que vous modifierez.
De cette manière, si vous mettez à jour le thème original, cela n'effacera pas vos modifications.

Pour créer un thème enfant, deux solutions s'offrent à vous :

- Utiliser une extension comme "**Child Theme Configurator**".
- Créer le thème enfant manuellement comme on va le faire ici.

Pour cela dirigeons nous vers le dossier des thèmes "**/wordpress/wp-content/themes/**".

Nous trouverons ici les dossiers de tous les thèmes actuellement installé.

1. Créer le dossier : Créons ensuite un nouveau dossier que nous nommerons par Convention comme le thème parent mais avec le suffixe "-child" ou "-enfant".

2. Créer la feuille de style : Créons dans ce nouveau répertoire un fichier "**style.css**":

```
/*
    Theme Name:   nom du thème enfant (obligatoire)
    Theme URI:    adresse de téléchargement du thème
    Description:  Description de mon thème enfant
    Author:       nom de l'auteur
    Author URI:   site de l'auteur
    Template:     nom du dossier du thème parent (obligatoire)
    Version:      0.1.0 version du thème
*/
```

3. Créer le fichier "**functions.php**":

```php
<?php
// J'ajoute la fonction wpm_enqueue_styles sur le hook wp_enqueue_scripts.
add_action( 'wp_enqueue_scripts', 'perso_enqueue_styles' );
function perso_enqueue_styles()
{
    /*
        On indique que lors du chargement du style de wordpress,
        il faut aussi charger le style du parent.
    */
    wp_enqueue_style( 'parent-style', get_template_directory_uri() .
'/style.css' );
    /*
        J'ajoute à la file le fichier de notre thème enfant.
    */
    wp_enqueue_style( 'child-style', get_template_stylesheet_uri() .
'/style.css' );
}
?>
```

4. Enfin il ne nous reste plus qu'à activer notre nouveau thème.