

Développement

WORDPRESS, PERFECTIONNEMENT

RÉALISATION FRANCK CHATELOT

TABLE DES MATIÈRES

O1 (slide 6)

Hébergement

1. Analyse de la procédure d'installation (Différence entre installation distante et locale)
2. La base de données, le fichier wp-config.

O2 (slide 21)

Fonctionnement des thèmes

1. Les fichiers du thème.
2. Hiérarchie des templates.
3. Les fichiers inclus.
4. La boucle wordpress (loop)
5. La requête de la boucle (WP_QUERY), requêtes multiples.

O3 (slide 49)

Création de thèmes

1. Gestion des menus.
2. Gestion des widgets.
3. Modèle de page.
4. Thème enfant.

TABLE DES MATIÈRES

O4 (slide 66)

Les différents contenus

1. Rappel Articles/Pages/Liens.
2. Initiation au Codex de WordPress.
3. Les champs personnalisés.
4. Les shortcodes.
5. Les custom_post_type.
6. La taxonomie.

O5 (slide 95)

Extensions et Widgets

1. Notion de hook. Analyse du fonctionnement.
2. Les extensions. Analyse de l'extension Hello dolly.
3. Les widgets. Analyse du fonctionnement.

TABLE DES MATIÈRES

06 (slide 117)

Aspects techniques divers

1. Aperçu du fichier .htaccess.
2. Référencement (réécriture des URL, métas, sitemaps, google analytics, ...).
3. Analyse d'un thème compatible HTML5 (Bones, HTML5 Blanks...).

07 (slide 135)

Sécurité

1. Sécuriser par les extensions.
2. Sécuriser par le .htaccess.
3. Sécuriser par la gestion des utilisateurs.

08 (slide 149)

Maintenance et Migration

1. Mise à jour.
2. Maintenance.
3. Transfert de WordPress.

Wordpress, perfectionnement

HÉBERGEMENT

O1

Analyse de la procédure d'installation

INSTALLATION À DISTANCE

ELEMENTS REQUIS



Un hébergeur

La plupart des hébergeurs actuels permettent de mettre en place de sites avec le CMS Wordpress. (OVH, O2Switch)



Nom de domaine

Il est important de bien choisir son nom de domaine au moment de son achat.



Logiciel FTP

Logiciel FTP Un logiciel FTP est nécessaire au transfert des fichiers vers l'hébergeur distant (FileZilla...).

INSTALLATION EN LOCAL

ELEMENTS REQUIS



Un serveur local

Wordpress est un CMS fonctionnant avec le langage PHP. L'utilisation d'un serveur est requise (Xampp, Mamp..).

Wordpress.org

On récupère l'archive de Wordpress sur le site Wordpress.org
<https://wordpress.org/download>

Base de données

Une base de données qui va contenir les différentes tables de Wordpress (MySQL avec PHPMyAdmin)

INSTALLATION EN LOCAL

Installation de la base de données



Vous devez saisir ci-dessous les détails de connexion à votre base de données. Si vous ne les connaissez pas, contactez votre hébergeur.

Nom de la base de données	<input type="text" value="wordpress"/>	Le nom de la base de données avec laquelle vous souhaitez utiliser WordPress.
Identifiant	<input type="text" value="root"/>	Votre identifiant MySQL.
Mot de passe	<input type="password"/>	Votre mot de passe de base de données.
Adresse de la base de données	<input type="text" value="localhost"/>	Si localhost ne fonctionne pas, demandez cette information à l'hébergeur de votre site.
Préfixe des tables	<input type="text" value="wp_"/>	Si vous souhaitez faire tourner plusieurs installations de WordPress sur une même base de données, modifiez ce réglage.

Paramétrage du site

Bienvenue

Bienvenue dans la très célèbre installation en 5 minutes de WordPress ! Vous n'avez qu'à remplir les informations demandées ci-dessous et vous serez prêt à utiliser la plus extensible et puissante plateforme de publication de contenu au monde.

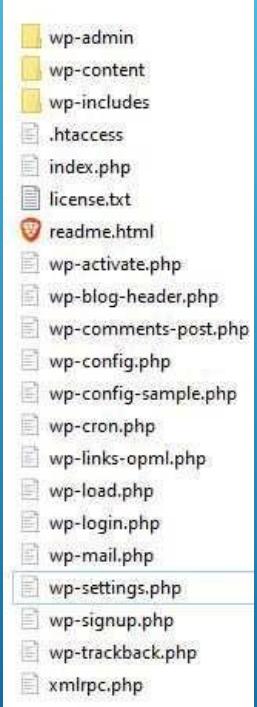
Informations nécessaires

Veuillez renseigner les informations suivantes. Ne vous inquiétez pas, vous pourrez les modifier plus tard.

Titre du site	<input type="text"/>	
Identifiant	<input type="text"/>	Les identifiants ne peuvent utiliser que des caractères alphanumériques, des espaces, des tirets bas (_), des traits d'union (-), des points et le symbole @.
Mot de passe	<input type="password" value="5E0Bbp3oliw7xPYHqB"/>  <input type="button" value="Masquer"/>	Important : Vous aurez besoin de ce mot de passe pour vous connecter. Pensez à le stocker dans un lieu sûr.
Votre e-mail	<input type="text"/>	Vérifiez bien cette adresse e-mail avant de continuer.
Visibilité par les moteurs de recherche	<input type="checkbox"/> Demander aux moteurs de recherche de ne pas indexer ce site	Certains moteurs de recherche peuvent décider de l'indexer malgré tout.

INSTALLATION EN LOCAL

DOSSIERS CONTENUS DANS L'ARCHIVE



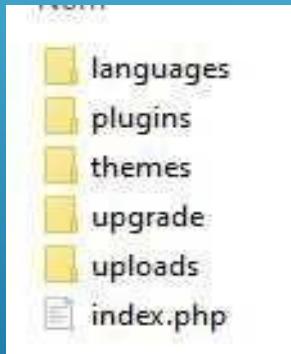
wp-admin contient toutes les classes en lien avec le back-office.

wp-content contient tout le contenu de notre site (extension, thème, media...).

wp-includes contient l'ensemble des classes qui servent au bon fonctionnement de Wordpress (le système de blocs ajax, la barre d'admin...).

INSTALLATION EN LOCAL

DOSSIERS CONTENUS DANS LE DOSSIER WP-CONTENT



languages : dossier contenant les fichiers de langues

plugins : dossier qui contient les extensions

thèmes : dossier qui contient les extensions

upgrade : dossier qui contient les fichiers temporaires lors des mises à jours

uploads : dossier qui contient tous les médias téléchargés



BASE DE DONNÉES

Chaque fonctionnalité de Wordpress possède sa propre table au sein de la base de données. Les extensions supplémentaires ajouteront de nouvelles tables lors d'installation

Table	Action
wp_commentmeta	
wp_comments	
wp_links	
wp_options	
wp_postmeta	
wp_posts	
wp_termmeta	
wp_terms	
wp_term_relationships	
wp_term_taxonomy	
wp_usermeta	
wp_users	
12 tables	Somme

BASE DE DONNÉES

wp_commentmeta

Table qui permet de stocker les informations supplémentaires sur les commentaires.

wp_links

Wp_links est une ancienne table utilisée pour les listes de liens externes. Elle existe encore pour des raisons de compatibilité avec d'anciennes versions.

Table	Action
wp_commentmeta	Parcourir Structure
wp_comments	Parcourir Structure
wp_links	Parcourir Structure
wp_options	Parcourir Structure
wp_postmeta	Parcourir Structure
wp_posts	Parcourir Structure
wp_termmeta	Parcourir Structure
wp_terms	Parcourir Structure
wp_term_relationships	Parcourir Structure
wp_term_taxonomy	Parcourir Structure
wp_usermeta	Parcourir Structure
wp_users	Parcourir Structure
12 tables	Somme

wp_comments

Table qui stocke les commentaires postés sur le site.

wp_options

Cette table est l'une des plus importantes de Wordpress. Elle contient les paramètres généraux de votre site (url, description, email...).

BASE DE DONNÉES

wp_postmeta

Table qui permet de stocker les informations supplémentaires sur les publications (posts) : article, page...

wp_post

Cette table contient l'ensemble des publications telles que les articles ou les pages.

Table	Action
wp_commentmeta	
wp_comments	
wp_links	
wp_options	
wp_postmeta	
wp_posts	
wp_termmeta	
wp_terms	
wp_term_relationships	
wp_term_taxonomy	
wp_usermeta	
wp_users	
12 tables	Somme

wp_termmeta

Table qui permet de stocker les informations supplémentaires sur les catégories et les étiquettes.

wp_term

Wp_term contient les différentes catégories et étiquettes (tags) créées sur le site.

BASE DE DONNÉES

wp_term_relationships

Cette table est une table de liaison permettant d'établir les relations entre les publications et les catégories/étiquettes (tags).

wp_term_taxonomy

Table qui permet de décrire les différents types de taxonomies.

Table	Action
wp_commentmeta	★ Parcourir Structure
wp_comments	★ Parcourir Structure
wp_links	★ Parcourir Structure
wp_options	★ Parcourir Structure
wp_postmeta	★ Parcourir Structure
wp_posts	★ Parcourir Structure
wp_termmeta	★ Parcourir Structure
wp_terms	★ Parcourir Structure
wp_term_relationships	★ Parcourir Structure
wp_term_taxonomy	★ Parcourir Structure
wp_usermeta	★ Parcourir Structure
wp_users	★ Parcourir Structure

wp_usermeta

Table qui permet de stocker les informations supplémentaires sur les utilisateurs inscrits sur le site.

wp_users

Wp_users contient les comptes utilisateurs (email, pseudo, mot de passe...).

WP-CONFIG.PHP

Le fichier wp-config.php contient les différents éléments de configuration de notre site Wordpress.
(données de connexion à la base de données, clé de salage...).

WP-CONFIG.PHP

Lors d'un transfert vers l'hébergeur, ces différentes données devront être modifiées.

DB_NAME : nom de la base de données

DB_USER : nom d'utilisateur

DB_PASSWORD : mot de passe utilisateur

DB_HOST : adresse de la base de données

```
// ** Database settings - You can get
// The name of the database for WordPress
define( 'DB_NAME', 'wordpress' );

/** Database username */
define( 'DB_USER', 'root' );

/** Database password */
define( 'DB_PASSWORD', '' );

/** Database hostname */
define( 'DB_HOST', 'localhost' );

/** Database charset to use in creating database */
define( 'DB_CHARSET', 'utf8mb4' );

/** The database collate type. Don't change it if in doubt. */
define( 'DB_COLLATE', '' );
```

WP-CONFIG.PHP

Après les paramètres de base de données, nous pouvons retrouver des clés de sécurité et des clés de salage.

Elles sont générées à l'installation de Wordpress. Il est possible d'en générer de nouvelles en suivant ce lien :

<https://api.wordpress.org/secret-key/1.1/salt/>

Ces clés ont pour rôle de crypter les cookies contenant les informations de connexion.

```
/*
define( 'AUTH_KEY',      'qX17@z|1E;*qQ}Z@S;C[=aB) BDdjw|^)^<{3]1-|canhCv4x<>GT],P6B;zFV8%*' );
define( 'SECURE_AUTH_KEY', '39E>@ fU/G=N9[ M7lzMm j1e7$cA-Wn@}=WnmH=xMY;t9.u*(m,w+KSV^+_ru50' );
define( 'LOGGED_IN_KEY',   'L,S1+P3y<<z5fzb~.+kzr[]G0( Q$@$&nt3!hf<>hYcmU4V-:7wfpu%JdD>?vZm' );
define( 'NONCE_KEY',       'z#./PR:Z]-:ON{J!Ysn>|d/u>3f~cViHN^?>-Q#7:t%>X.qV?IK|KPssv1iKh#?T' );
define( 'AUTH_SALT',        'T@XYDeoQw/H)pdmm8Si-)M!4*tPpzcMO_MGLKOL5?s-DffQmNCI`%xbL(<W34P~' );
define( 'SECURE_AUTH_SALT', ',N&C$b+[G1zX16f%F_T*kI*6,o<ZsM).JTo |d1U]b+-Z'$fGRNom3NPg_- Ofk!' );
define( 'LOGGED_IN_SALT',   '296zmwP?kCRVAX5@pK{2$0$3Z_rI$Qzg%D18s@%<~ma<cGVx>s[6VH#f=:!V!cY6' );
define( 'NONCE_SALT',       '.+Q]4z>#((<8iK-fbYO.15%!@+G]J@<DZ/Uj4=!(QV&$lo7sRE3@r#0*}hGkB6&]' );
```

```
$table_prefix = 'wp_';

/**
 * For developers: WordPress debugging mode.
 *
 * Change this to true to enable the display of errors.
 * It is strongly recommended that plugin and theme developers test
 * in their development environments.
 *
 * For information on other constants that can be defined,
 * visit the documentation.
 *
 * @link https://wordpress.org/support/article/debugging-in-wordpress/
 */

define( 'WP_DEBUG', false );
```

WP-CONFIG.PHP

`$table_prefix` : contient le préfixe de toutes les tables contenues dans la base de données

`define('WP_DEBUG', false);` on a la possibilité de définir des constantes qui s'appliqueront à notre site. `WP_DEBUG` avec la valeur "true" permet l'affichage des erreurs.

Nous pourrons définir un ensemble de constantes telles que l'activation du cache ou encore la désactivation de l'éditeur du côté back office.

<https://www.team-ever.com/constantes-de-wordpress/>

WP-CONFIG.PHP

Les derniers paramètres contenus dans le fichier wp-config.php sont les suivants :

- **ABSPATH** permet de définir le chemin absolu vers le dossier de Wordpress
- On charge le fichier **wp-settings.php** contenant le chargement de la plupart des classes utilisées par Wordpress.

```
/* That's all, stop editing! Happy publishing. */

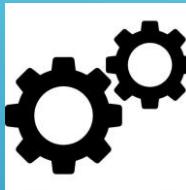
/** Absolute path to the WordPress directory. */
if ( ! defined( 'ABSPATH' ) ) {
    define( 'ABSPATH', __DIR__ . '/' );
}

/** Sets up WordPress vars and included files. */
require_once ABSPATH . 'wp-settings.php';
```

02

Fonctionnement des thèmes

LES FICHIERS DU THÈMES



style.css

Le fichier style.css contient les informations essentielles au thème.

functions.php

functions.php gère l'ensemble des fonctionnalités de notre thème.

index.php

index.php est le premier template de notre thème. Il s'agit d'un premier pas vers la hiérarchie des templates.

screenshot.png

Ce dernier fichier est optionnel. Il représente l'image de visualisation du thème.

LES FICHIERS DU THÈMES

style.css

Le fichier style.css est obligatoire pour la création d'un thème.

Il permet à Wordpress de détecter le thème.

On y précise des informations comme le nom du thème, sa version, la version de wordpress minimum.

<https://developer.wordpress.org/themes/basics/main-style-sheet-style-css/#explanations>

```
/*
Theme Name: Mon thème
Theme URI: https://montheme.com
Author: Nolan Bigot
Description: Mon premier thème !
Requires at least: WordPress 5.0
Version: 1.0
*/
```

Certaines informations sont obligatoires dans le cadre d'une publication de thème sur le catalogue officielle de Wordpress

LES FICHIERS DU THÈMES

functions.php

Le fichier functions.php contient l'ensemble des fonctionnalités de notre thème, quelques exemples :

- L'ajout de script et de feuille de style
- L'ajout de zone de menu
- L'ajout de zone de widget
- Tous les éléments supportés par notre thème.

```
<?php  
  
add_theme_support( 'automatic-feed-links' );  
  
add_theme_support( 'post-thumbnails' );  
  
add_theme_support( 'title-tag' );  
  
register_nav_menus( array(  
    'primary' => __( 'Primary Menu', 'myfirsttheme' ),  
    'secondary' => __( 'Secondary Menu', 'myfirsttheme' )  
) );
```

LES FICHIERS DU THÈMES

index.php

Ce fichier est le premier template que l'on met en place au sein de notre thème.

En effet, le système de template est l'un des points fondamentaux autour de la création de thème Wordpress.

Les différentes publications créées via le back-office emploieront ces templates comme design. Les templates permettent d'agencer les pages.

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
</head>
<body>
    <h1>Index.php</h1>
</body>
</html>
```

LES FICHIERS INCLUS

header.php

Ce fichier contient l'entête générale de nos templates. Cette entête permettra l'automatisation de plusieurs réglages de Wordpress.

footer.php

footer.php contient le pied de page général de nos templates. Certains réglages supplémentaires s'appliqueront au sein de cette zone.

LES FICHIERS INCLUS

header.php

```
<!DOCTYPE html>
<html <?php language_attributes(); ?>>
<head>
    <meta charset=<?php bloginfo( 'charset' ); ?>">
    <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">

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

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

language_attributes() :
indique la langue par défaut
employée par Wordpress

bloginfo('charset') : indique
l'encodage paramétré au
sein de Wordpress

wp_head() : permet de
déclarer la zone d'entête
dans laquelle les scripts,
feuilles de style et autres
metas devront être chargés.

LES FICHIERS INCLUS

header.php

```
<!DOCTYPE html>
<html <?php language_attributes(); ?>>
<head>
    <meta charset=<?php bloginfo( 'charset' ); ?>">
    <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no"/>

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

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

body_class() : intègre plusieurs classes dynamiques qui s'adaptent en fonction de la page visitée.

wp_body_open() : permet de déclarer la zone de début de la balise body, cela permet à certaines extensions d'intégrer des éléments de code.

LES FICHIERS INCLUS

footer.php

`wp_footer()` : permet de déclarer la zone du pied de page dans laquelle les scripts et autres codes d'extension pourront être ajoutés.

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

LES FICHIERS INCLUS

Intégrer header.php et footer.php

Il est possible d'utiliser deux fonctions pour intégrer ces fichiers :

- `get_header();`
- `get_footer();`

Automatiquement Wordpress intégrera les fichiers

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

LES FICHIERS INCLUS

Les templates parts

Les templates parts sont des petits fichiers contenant généralement du HTML et du CSS que l'on va pouvoir intégrer à plusieurs reprises.

Le fonctionnement est similaire aux fichiers header et footer.

On va simplement créer un fichier .PHP que l'on intégrera avec la fonction :

```
get_template_part('form');
```

Le paramètre est le nom du fichier sans l'extension .PHP

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

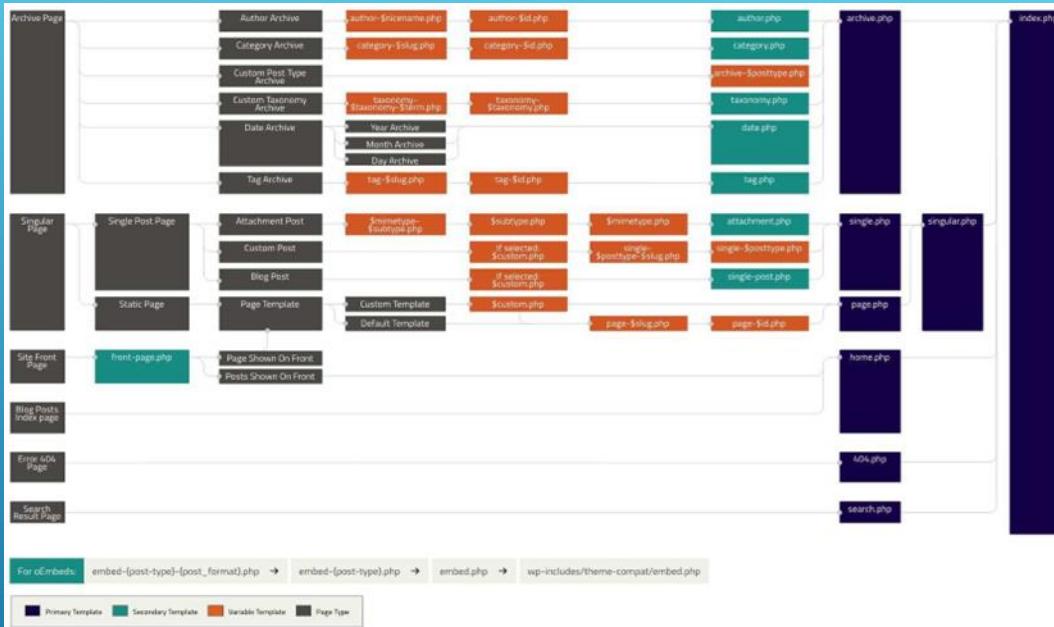
LA HIÉRARCHIE DES TEMPLATES

La hiérarchie des templates ou "Template Hierarchy" est un système de fonctionnement propre à Wordpress.

En effet, le "Template Hierarchy" va nous proposer un ensemble de template à utiliser selon des situations données.

Les templates permettent de désigner et de structurer le contenu de nos publications.

LA HIÉRARCHIE DES TEMPLATES



<https://developer.wordpress.org/files/2014/10/Screenshot-2019-01-23-00.20.04.png>

LA HIÉRARCHIE DES TEMPLATES



Généralités

Le schéma (slide précédente) se lit de gauche à droite.

On commence par les blocs gris qui représentent les types de page (page 404, page d'archive, page d'accueil, publications comme les articles...).

Il suffit de suivre la ligne attachée au bloc gris choisi. Nous aurons alors plusieurs choix de template possibles selon les utilisations.

Il existe trois types de templates.

Il existe trois types de templates :

- Les templates variables (oranges) permettent d'obtenir des templates très précis englobant des pages spécifiques (exemple : page contenant un slug ou un id précis).
- Les templates secondaires (turquoises) sont utilisés pour des pages uniques (comme la page d'accueil) ou des types de pages spécifiques.
- Les templates primaires (violets) sont les templates les plus généralistes et vont englober plusieurs types de pages.

LA HIÉRARCHIE DES TEMPLATES

Généralités

Il est très important de se référer à ce schéma pour développer son thème.

En effet, chaque nom de template proposé dans le "Template hierarchy" sont les différents noms qu'il faudra utiliser pour la création des templates au sein de notre thème.

Le "Template hierarchy" n'est pas seulement théorique, il s'agit d'un réel processus de fonctionnement au sein de Wordpress.

Si nous voulons utiliser le template "page.php" pour gérer le design des pages dites statiques, il faudra créer un fichier portant ce nom (sans faute).

LA HIÉRARCHIE DES TEMPLATES

Généralités

LA HIÉRARCHIE DES TEMPLATES

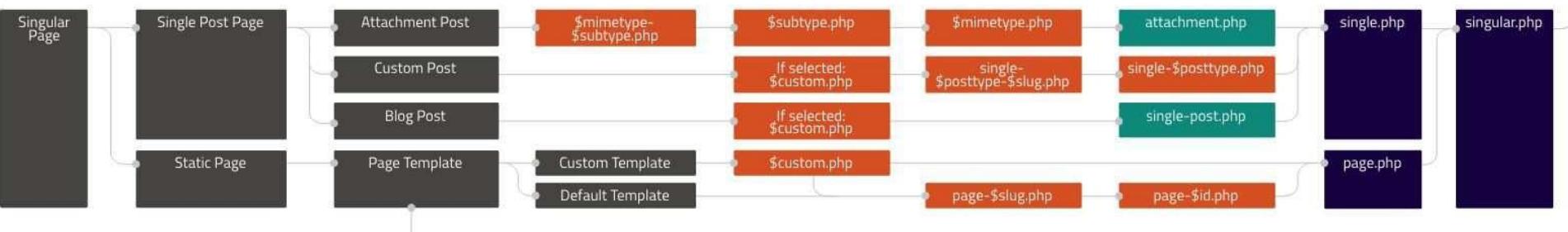


<https://developer.wordpress.org/files/2014/10/Screenshot-2019-01-23-00.20.04.png>

Il existe un template spécifique pour les pages suivantes :

- front-page.php pour la page d'accueil
- 404.php pour la page d'erreur 404
- search.php pour la page des recherches
- home.php pour la page de blog (page contenant les articles)

LA HIÉRARCHIE DES TEMPLATES



<https://developer.wordpress.org/files/2014/10/Screenshot-2019-01-23-00.20.04.png>

La gestion des types de publication va se scinder en deux groupes :

- Les "Static Page" qui représentent les "pages statistiques" (les pages que l'on crée dans les back-office)
- Les "Single Post Page" qui représentent les publications (telles que les articles ou les publications personnalisées)

On a plusieurs possibilités de template : par slug, par id, des templates personnalisés ou encore des templates généralistes.

LA HIERARCHIE DES TEMPLATES



<https://developer.wordpress.org/files/2014/10/Screenshot-2019-01-23-00.20.04.png>

Le dernier type de page que l'on peut observer dans le template hierarchy sont les pages d'archives. Les pages d'archives représentent les pages contenant des listes de publications (page contenant des articles, page contenant des produits...).

Les pages d'archives vont être divisées en plusieurs catégories. Le choix des templates est multiple : template par slug, par id, par taxonomy ou encore des templates plus généralistes.

LA BOUCLE WORDPRESS (LOOP)

Une fois que l'on a créé les différents templates correspondant à nos attentes au sein de notre thème, nous devons mettre en place la gestion du contenu.

Wordpress nous propose un système nommé la boucle. Cette boucle aura pour rôle de récupérer le contenu d'une page à l'aide de requêtes.

LA BOUCLE WORDPRESS

La syntaxe

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

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

<?php the_content(); ?>

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

La boucle s'établit à partir d'une condition qui vérifie l'existence de contenu à l'aide de la fonction **have_posts()** (cette fonction utilise la requête Wordpress).

Si un contenu est trouvé, on exécute la boucle à l'aide de l'instruction **while** qui permettra de continuer à récupérer le contenu tant qu'il y en a. Tout cela est rendu possible à l'aide des fonctions **have_posts()** et **the_post()**

LA BOUCLE WORDPRESS

Le contenu

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

Lorsqu'il existe du contenu, on va employer un ensemble de fonctions inclus nommées des "Template Tags".

Les "Templates Tags" permettent d'afficher les différentes informations administrées au sein du "back-office", y compris le contenu ajouté au sein des publications.

*Exemple : the_title() et
the_content()*

LA BOUCLE WORDPRESS

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

<article>
    <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; ?>
```

Le contenu

the_title() : affiche le titre de la publication.

the_post_thumbnail() : affiche l'image de mise en avant.

the_time() : affiche la date de parution.

the_author() : affiche l'auteur.

comments_number() : affiche le nombre de commentaires.

A BOUCLE WORDPRESS

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

<article>
    <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; ?>
```

Le contenu

`the_excerpt()` : affiche le résumé de la publication

`the_permalink()` : affiche le lien vers la publication

`the_content()` : affiche le contenu entier de la publication

https://codex.wordpress.org/Template_Tags

LA REQUÊTE WP_QUERY

Pour pouvoir fonctionner, la boucle Wordpress utilise une classe nommée "**WP_QUERY**" qui permet d'utiliser le système de requête (proposée par Wordpress).

L'intérêt autour de cette classe est de pouvoir filtrer le contenu récupéré à partir de la boucle.

Le second point d'intérêt est d'effectuer plusieurs requêtes pour travailler avec plusieurs boucles.

LA REQUÊTE WP_QUERY

WP_Query est avant tout une classe PHP.

On doit l'instancier, le constructor de la classe accepte en argument un tableau qui va permettre de filtrer le contenu récupéré.

Dans l'exemple de droite, on accepte les articles et les produits de la catégorie "music" à raison de 7 éléments.

```
$args = array(
    'post_type' => ['post', 'product'],
    'category_name' => 'music',
    'posts_per_page' => 7,
);

$the_request = new WP_Query( $args );

if( $the_request->have_posts() ) : while( $the_request->have_posts() ) : $the_request->the_post(); ?>

    <p><?php the_title(); ?></p>
    <?php the_content();?>
    <?php the_post_thumbnail();?>

<?php

endwhile; endif;

wp_reset_postdata();
```

LA REQUÊTE WP_QUERY

L'ensemble des filtres peut être trouvés en suivant ce lien :

https://developer.wordpress.org/reference/classes/wp_query/#parameters

La seconde étape consiste à utiliser l'objet instancié précédemment dans la boucle.

Exemple : \$the_request->have_posts();

La dernière étape consiste à réinitialiser la requête principale à l'aide de **wp_reset_postdata()**.
Pour pouvoir revenir à la requête initiale.

```
$args = array(
    'post_type' => ['post', 'product'],
    'category_name' => 'music',
    'posts_per_page' => 7,
);

$the_request = new WP_Query( $args );

if( $the_request->have_posts() ) : while( $the_request->have_posts() ) : $the_request->the_post(); ?>

    <p><?php the_title(); ?></p>
    <?php the_content();?>
    <?php the_post_thumbnail();?>

<?php
endif;
wp_reset_postdata();
```

LA REQUÊTE WP_QUERY

La dernière étape consiste à réinitialiser la requête principale à l'aide de `wp_reset_postdata()`.

En effet, il est possible d'exécuter plusieurs boucles différentes au sein d'un même template (*exemple : dans le cas où on veut récupérer des publications personnalisées différentes*).

Le fait de créer une nouvelle instance de `WP_Query` ne réinitialise pas la requête de la boucle, c'est pour cela qu'il est important d'utiliser `wp_reset_postdata()`.

```
$args = array(
    'post_type' => ['post', 'product'],
    'category_name' => 'music',
    'posts_per_page' => 7,
);

$the_request = new WP_Query( $args );

if( $the_request->have_posts() ) : while( $the_request->have_posts() ) : $the_request->the_post(); ?>

    <p><?php the_title(); ?></p>
    <?php the_content();?>
    <?php the_post_thumbnail();?>

<?php

endifwhile; endif;

wp_reset_postdata();
```

O3

Création de thème

LA GESTION DES MENUS

Toujours dans l'optique d'automatiser le thème à partir du back-office, il est impératif de proposer des zones de menus paramétrables.

Pour cela, nous allons travailler dans les fichiers de template et dans le fichier functions.php

LA GESTION DES MENUS

Zone de menu

Au sein du fichier functions.php, nous allons utiliser la fonction `register_nav_menus()`.

Cette fonction accepte en paramètre un tableau associatif :

- Les clés permettent d'identifier une zone de menu dans le code.
- Les valeurs sont les noms apparaissant dans l'onglet "menu" du "back-office"

```
register_nav_menus( [  
    'main' => 'Menu Principal',  
    'footer' => 'Bas de page',  
]; );
```

LA GESTION DES MENUS

Zone de menu

Lorsque l'on a déclaré les zones de menus dans le fichier **functions.php**, nous devons les intégrer dans les différents templates à l'aide la fonction **wp_nav_menu()**

La fonction accepte en paramètre un tableau associatif avec plusieurs informations :

La clé identifiant la zone de menu (**theme_location**).

La possibilité de changer le container (**container**) ou d'ajouter une classe personnalisée (**menu_class**)

```
wp_nav_menu(  
    [  
        'theme_location' => 'main',  
        'container' => 'ul',  
        'menu_class' => 'menu_class',  
    ]  
);  
  
wp_nav_menu(  
    [  
        'theme_location' => 'footer',  
        'container' => 'ul',  
        'menu_class' => 'menu_class',  
    ]  
);
```

LA GESTION DES MENUS

Conclusion

Une fois les zones de menu déclarées et intégrées dans les templates, l'utilisateur peut agencer ces menus à partir du "back-office".

Vous pouvez retrouver tous les paramètres de `wp_nav_menu()` en suivant ce lien :

https://developer.wordpress.org/reference/functions/wp_nav_menu/

```
wp_nav_menu(  
[  
    'theme_location' => 'main',  
    'container' => 'ul',  
    'menu_class' => 'menu_class',  
];  
  
wp_nav_menu(  
[  
    'theme_location' => 'footer',  
    'container' => 'ul',  
    'menu_class' => 'menu_class',  
];
```

LA GESTION DES WIDGETS

La mise en place des zones de widgets dans un thème est très similaire à celle des zones de menus.

Nous devons déclarer la zone de widget dans le fichier functions.php puis l'intégrer dans le template désiré.

LA GESTION DES WIDGETS

Zone de widgets

Au sein du fichier `functions.php`, nous allons utiliser la fonction `register_sidebar()`.

Cette fonction accepte en paramètre un tableau associatif :

- La clé "`id`" permet d'identifier la zone de widget
- La clé "`name`" permet de nommer la zone de widget dans le "back-office".

```
register_sidebar( [  
    'id' => 'site-sidebar',  
    'name' => 'Site',  
]; );
```

LA GESTION DES WIDGETS

Zone de widgets

Lorsque l'on a déclaré les zones de widget dans le fichier **functions.php**, nous devons les intégrer dans les différents templates à l'aide la fonction **dynamic_sidebar()**.

La fonction accepte en paramètre une chaîne de caractère qui correspond à l'id déclaré dans le fichier **functions.php**

```
dynamic_sidebar( 'site-sidebar' );
```

LA GESTION DES WIDGETS

Conclusion

Une fois les zones de widget déclarées et intégrées dans les templates, l'utilisateur peut agencer ces widgets à partir du "back-office" dans l'onglet "widget".

Le fonctionnement est très similaire à celui des zones de menu.

```
dynamic_sidebar( 'site-sidebar' );
```

LES MODÈLES DE PAGE

Au sein du "Template Hierarchy", il existe un type de template permettant la création de "modèle de page" dit personnalisé.

Il est possible que les templates proposés ne conviennent pas dans le cas de certaines pages où la structure et le design doivent être uniques.

Par exemple : une page de formulaire de contact.

Nous pourrons utiliser les modèles de page pour résoudre ce problème.

LES MODÈLES DE PAGE



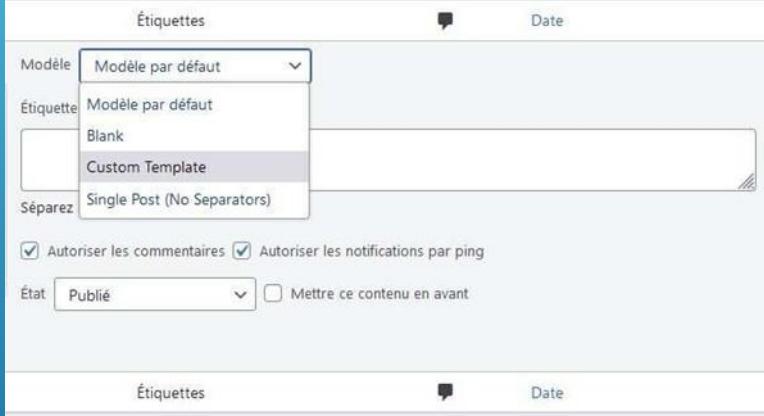
```
/*
Template Name: Custom Template
Template Post Type: post, product, page
*/
```

L'image nous montre qu'il est possible de créer des "custom template" : **\$custom.php**

Pour créer un modèle de page, on ajoute un commentaire en début de fichier avec plusieurs informations :

- Le nom du modèle de page
- Les types de publications pouvant utiliser ce modèle de page (ici les articles et les produits)

LES MODÈLES DE PAGE



Pour pouvoir utiliser le modèle de page précédemment créé, on doit se rendre sur le back-office et modifier notre publication.

On peut alors trouver notre modèle dans le menu déroulant "Modèle".

Le modèle peut être utilisé sur plusieurs publications.

LES THÈMES ENFANTS

Nous avons pu voir la création d'un thème à partir de zéro.

Au sein du CMS Wordpress, il est possible de créer ce que l'on appelle des thèmes enfants.

Les thèmes enfants ont pour objectif de se baser sur un thème parent. Le thème enfant reprend l'ensemble des fonctionnalités du thème parent.

Cependant, tous les fichiers placés dans le thème enfant prennent le dessus sur ceux du thème parent.

LES AVANTAGES DES THÈMES ENFANTS



Une base

Le premier avantage est de pouvoir créer un thème sans pour autant partir de zéro.



Les thèmes premiums

Généralement, beaucoup de thèmes premium proposent une version gratuite. Il est tout à fait possible de se baser dessus et d'ajouter ses propres fonctionnalités.



Mise à jour

Lorsque l'on ajoute ses propres fonctionnalités à un thème qui ne nous appartient pas, on risque de devoir tout remettre en place à chaque mise à jour. L'utilisation d'un thème enfant permet de répondre à ce problème.

LES THÈMES ENFANTS



Création

Pour mettre en place un thème enfant, la première étape est de créer un nouveau dossier généralement qui portera le même nom avec le suffixe "-child"

LES THÈMES ENFANTS

```
add_action( 'wp_enqueue_scripts', 'theme_enqueue_styles' );
function theme_enqueue_styles() {
    wp_enqueue_style( 'parent-style', get_template_directory_uri() . '/style.css' );
}
```

```
/*
Theme Name: Twenty Twenty Two Child
Author: Nolan Bigot
Template: Twenty Twenty-Two
Version: 0.1.0
*/
```

Création

Seulement deux fichiers sont obligatoires pour déclarer un thème enfant :

- **functions.php** qui contient la fonction **wp_enqueue_script()** permettant de charger le **style.css** du thème parent
- **style.css** qui contient une information supplémentaire : Template avec le nom du thème parent.

LES THÈMES ENFANTS

```
add_action( 'wp_enqueue_scripts', 'theme_enqueue_styles' );
function theme_enqueue_styles() {
    wp_enqueue_style( 'parent-style', get_template_directory_uri() . '/style.css' );
}
```

```
/*
Theme Name: Twenty Twenty Two Child
Author: Nolan Bigot
Template: Twenty Twenty-Two
Version: 0.1.0
*/
```

conclusion

Après la création de ces deux fichiers, tous les fichiers supplémentaires prendront le dessus sur les fichiers du thème parent.

Il ne reste plus qu'à adapter le thème à nos objectifs.

04

Les Différents Contenus

RAPPEL ARTICLES/PAGES

Il est important de rappeler qu'il existe plusieurs types de publication au sein de Wordpress.

La différence principale se situe entre les articles et les pages.

LES ARTICLES

Les articles contrairement aux pages ont principalement un intérêt social.

L'article aura tendance à susciter des réactions (avec la possibilité pour l'utilisateur de laisser des commentaires).

Ils possèdent un aspect éphémère, les pages resteront sur le site alors que les articles disparaîtront au fur et à mesure.

Généralement l'article sera écrit par un auteur, contiendra une date de publication, une catégorie et des étiquettes.

Ils apparaissent dans les flux RSS.

LES PAGES

Les pages ont un aspect plus structurant pour le site avec une hiérarchie bien spécifique.

Contrairement à l'article, leur contenu n'est pas éphémère (politique de confidentialité, mentions légales...).

Les pages vont générer moins d'interactions que les articles.

Les pages n'apparaissent pas dans les flux RSS.

INITIATION AU CODEX DE WORDPRESS. LE PORTAIL DÉVELOPPEUR.

Comme toute technologie, Wordpress possède sa propre documentation : le Codex.

Cette documentation est traduite dans plusieurs langues, y compris le français.

On y retrouve énormément d'informations telles qu'une introduction à Wordpress, le design et les structures, les fonctionnalités de Wordpress...

<https://codex.wordpress.org/fr:Accueil>

INITIATION AU CODEX DE WORDPRESS. LE PORTAIL DÉVELOPPEUR.

Généralités

Le Codex se divise en 7 catégories :

- "Démarrer avec Wordpress" contenant les guides d'installations, le dépannage autour de l'installation, la découverte du "back-office" (avec la gestion des thèmes, des publications, des extensions...).
- "Travailler avec Wordpress" contenant les fonctionnalités générales, la création de thème et le développement.

Démarrer avec WordPress »

- Démarrer avec WordPress
- Avant l'installation
- Nouveau sur WordPress - Où Commencer ?
- Héberger WordPress
- Installer WordPress
- Utiliser un Client FTP
- Mettre à Jour WordPress
- Faire Aux Questions Installation

Travailler avec WordPress »

- Travailler avec WordPress
- Fonctionnalités
- Plugins
- Gestion de Plugins

Style et Disposition (Design and Layout) »

- Style et Disposition
- Utiliser les thèmes
- Hiérarchie de Modèles
- Découvrir les Marqueurs de Modèle
- Marqueurs de Modèle
- Marqueurs Conditionnels
- La Boucle (The Loop)
- Les thèmes enfants

Sujets Avancés

- Utiliser les Champs Personnalisés
- Déplacer WordPress

Documentation pour développeurs (français/anglais)

- Description de la base de données
- Liste des fonctions de wordpress
- Les API pour plugins
- Développer un plugin
- Convention de codage wordpress

Dépannage

INITIATION AU CODEX DE WORDPRESS. LE PORTAIL DÉVELOPPEUR.

Généralités

- "Style et disposition" contenant les guides autour de l'ajout de style, de la gestion de contenu avec la boucle et l'utilisation de modèle.
- "Sujets avancés" contenant les guides autour des champs personnalisés et le transfert de Wordpress
- "Documentation pour développeurs" contenant tous les liens vers les documentations de développement
- "Dépannage" contenant la foire aux questions

Démarrer avec WordPress »

- Démarrer avec WordPress
- Avant l'installation
- Nouveau sur WordPress - Où Commencer ?
- Héberger WordPress
- Installer WordPress
- Utiliser un Client FTP
- Mettre à Jour WordPress
- Faire Aux Questions Installation

Travailler avec WordPress »

- Travailler avec WordPress
- Fonctionnalités
- Plugins
- Gestion de Plugins

Style et Disposition (Design and Layout) »

- Style et Disposition
- Utiliser les thèmes
- Hiérarchie de Modèles
- Découvrir les Marqueurs de Modèle
- Marqueurs de Modèle
- Marqueurs Conditionnels
- La Boucle (The Loop)
- Les thèmes enfants

Sujets Avancés

- Utiliser les Champs Personnalisés
- Déplacer WordPress

Documentation pour développeurs (français/anglais)

- Description de la base de données
- Liste des fonctions de wordpress
- Les API pour plugins
- Développer un plugin
- Convention de codage wordpress

Dépannage

INITIATION AU CODEX DE WORDPRESS. LE PORTAIL DÉVELOPPEUR.

Généralités

- "À Propos de Wordpress"
contenant toutes les informations en lien avec le CMS

Démarrer avec WordPress »

- Démarrer avec WordPress
- Avant l'installation
- Nouveau sur WordPress - Où Commencer ?
- Héberger WordPress
- Installer WordPress
- Utiliser un Client FTP
- Mettre à Jour WordPress
- Faire Aux Questions Installation

Travailler avec WordPress »

- Travailler avec WordPress
- Fonctionnalités
- Plugins
- Gestion de Plugins

Style et Disposition (Design and Layout) »

- Style et Disposition
- Utiliser les thèmes
- Hiérarchie de Modèles
- Découvrir les Marqueurs de Modèle
- Marqueurs de Modèle
- Marqueurs Conditionnels
- La Boucle (The Loop)
- Les thèmes enfants

Sujets Avancés

- Utiliser les Champs Personnalisés
- Déplacer WordPress

Documentation pour développeurs (français/anglais)

- Description de la base de données
- Liste des fonctions de wordpress
- Les API pour plugins
- Développer un plugin
- Convention de codage wordpress

Dépannage

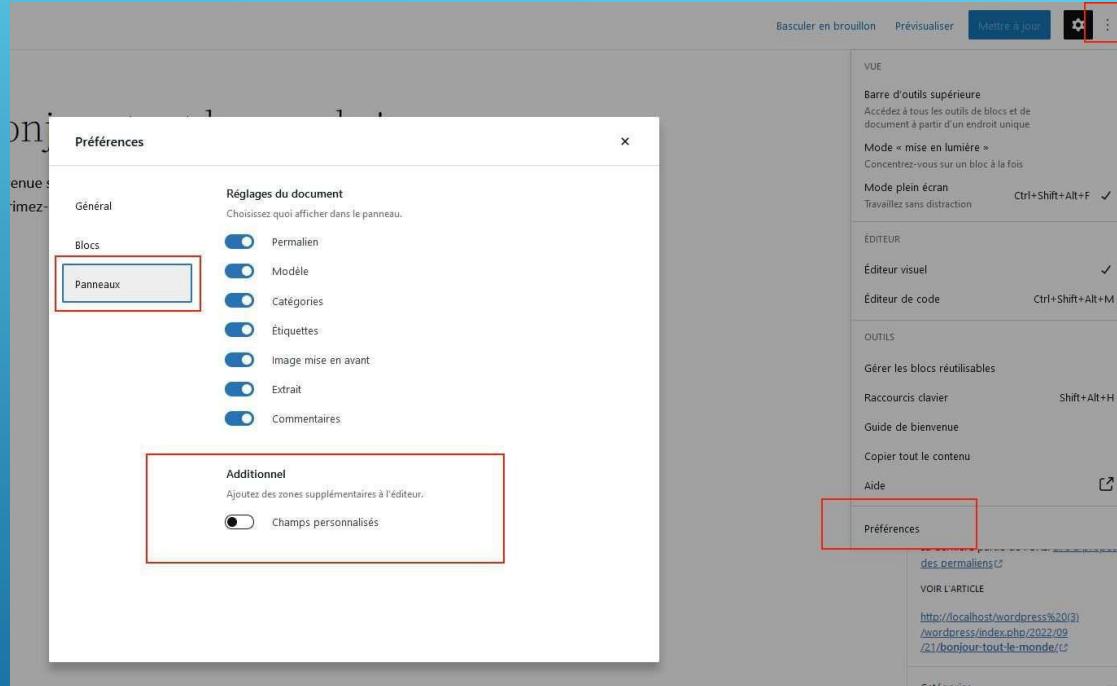
LES CHAMPS PERSONNALISÉS.

Nous avons pu mettre en place du contenu à l'aide de la boucle Wordpress.

Les champs personnalisés (ou custom fields) permettent d'ajouter des informations supplémentaires dans le contenu.

Par exemple, nous proposons des biens immobiliers sur notre site. On veut pouvoir ajouter des informations telles que la superficie, le type de bien...

LES CHAMPS PERSONNALISÉS



LES CHAMPS PERSONNALISÉS

Champs personnalisés

Ajouter un nouveau champ personnalisé :

Nom	Valeur
<input type="text"/>	<input type="text"/>

[Ajouter un champ personnalisé](#)

Les champs personnalisés peuvent être utilisés pour ajouter des métadonnées supplémentaires à une publication, que vous pouvez ensuite [utiliser dans votre thème](#).

Les deux images (slide courante et slide précédente) permettent l'activation des champs personnalisés au sein de Wordpress.

Une fois les champs personnalisés apparents, nous pouvons en ajouter autant que nous le voulons. On ajoute un nom et une valeur, le nom permet d'afficher la valeur dans le code lors de la récupération du champ.

LES CHAMPS PERSONNALISÉS

Pour afficher la valeur d'un champ personnalisé, on utilise la fonction `get_post_meta()` au sein de la boucle Wordpress.

La fonction accepte en paramètres :

- `get_the_ID()` : fonction qui permet de récupérer l'id de la publication pour l'identifier
- Le nom du champ personnalisé
- Un boolean pour dire que le champ est unique.

```
echo get_post_meta( get_the_ID(), 'superficie', true );
```

Il ne faut pas oublier l'instruction "echo"

LES CHAMPS PERSONNALISÉS

Il existe une extension nommée **ACF** : **Advanced Custom Fields** permettant d'augmenter les possibilités des champs personnalisés.

En effet, par défaut les "custom fields" se limitent à une simple chaîne de caractères.

ACF permet de créer des champs personnalisés de différents types : date, nombre, fichier, choix multiples...



LES CHAMPS PERSONNALISÉS

Ajouter un groupe de champs

#	Libellé	Nom
1	(aucun libellé)	

Modifier Dupliquer Supprimer

General Validation Présentation Logique conditionnelle

Type de champ

- Texte
- Basique**
 - Texte
 - Zone de texte
 - Nombre
 - Plage
 - E-mail
 - URL
 - Mot de Passe
- Contenu
 - Image
 - Fichier
 - Éditeur Wysiwyg
 - Contenu oEmbed
 - Galerie (Pro uniquement)
- Choix
 - Liste déroulante
 - Case à cocher
 - Bouton radio
 - Groupe de boutons

► On installe ACF à partir de l'onglet extension. Une fois installée, on peut ajouter un groupe de champs.

► Le groupe de champs contient plusieurs champs personnalisés de tout type.

► Chaque type va posséder ses propres paramètres.

► Le groupe de champ pourra être ajouté à un type de publication spécifique.

LES CHAMPS PERSONNALISÉS

Pour afficher les champs personnalisés créés à partir de l'extension ACF, nous aurons deux possibilités :

- **the_field()** : affiche la valeur.
- **get_field()** : permet de récupérer la valeur.

```
<?php the_field( 'superficie' ); ?>  
  
<?php $superficie = get_field( 'superficie' ); ?>
```

LES CODES COURTS (SHORTCODES)

Les shortcodes sont des petits morceaux de code entre crochets.

Ces codes permettent au chargement d'une page d'exécuter une fonction.

Par exemple, cette fonction peut servir à afficher des éléments comme un carrousel ou un slider.

LES CODES COURTS (SHORTCODES)

Création

Pour créer un shortcode, on utilise la fonction `add_shortcode()`.

Cette fonction doit être intégrée au sein du fichier `functions.php`.

`add_shortcode()` accepte en paramètre un nom et une fonction. Le nom sera utilisé dans le shortcode entre crochets.

Il est possible de passer des paramètres dans le shortcode.

```
function shortcode_get_email( $atts ) {  
    $atts = shortcode_atts( array(  
        'id' => get_current_user_id(),  
    ), $atts, 'email' );  
  
    $user = get_userdata( $atts['id'] );  
  
    return $user->email;  
}  
add_shortcode( 'email', 'shortcode_get_email' );
```

LES CODES COURTS (SHORTCODES)

Création

Les paramètres peuvent être récupérés à l'aide de la fonction `shortcode_atts()`

Pour utiliser le shortcode, on intégrera le code : `[email id=15]`

Les shortcodes doivent être intégrés dans le contenu d'une publication à partir de Gutenberg par exemple.

Il est cependant possible d'utiliser la fonction `apply_shortcode('email')` pour l'utiliser où vous le voulez dans le code

```
function shortcode_get_email( $atts ) {  
  
    $atts = shortcode_atts( array(  
        'id' => get_current_user_id(),  
    ), $atts, 'email' );  
  
    $user = get_userdata( $atts['id'] );  
  
    return $user->email;  
}  
add_shortcode( 'email', 'shortcode_get_email' );
```

LES CODES COURTS (SHORTCODES)

Création

Il est possible de créer des shortcodes acceptant du contenu avec une balise d'ouverture et de fermeture.

Pour cela, on utilisera le paramètre **\$content**.

Le code à utiliser sera le suivant :

[email]mon email[/email]

```
function shortcode_get_email( $atts, $content ) {  
    return '<div class="column">' . $content . '</div>' ;  
}  
add_shortcode( 'email', 'shortcode_get_email' );
```

LES TYPES DE PUBLICATION PERSONNALISÉES (CUSTOM POST TYPE)

Nous avons vu qu'il était possible de créer des champs personnalisés pour ajouter des informations à notre contenu.

Nous pouvons aller plus loin en créant des types de publication personnalisée. En effet, le type de publication par défaut sur Wordpress est l'article (historique au CMS).

Dans certains cas, on aimerait avoir des publications différentes (comme l'ajout de bien immobilier, l'ajout de recettes de cuisine...).

```
function register_post_types() {  
  
    $labels = [  
        'name' => 'Immobilier',  
        'all_items' => 'Tous les biens immobiliers',  
        'singular_name' => 'Bien immobilier',  
        'add_new_item' => 'Ajouter un bien immobilier',  
        'edit_item' => 'Modifier le bien immobilier',  
        'menu_name' => 'Immobilier'  
    ];  
  
    $args = [  
        'labels' => $labels,  
        'public' => true,  
        'show_in_rest' => true,  
        'has_archive' => true,  
        'supports' => [ 'title', 'editor', 'thumbnail', 'custom-fields' ],  
        'menu_position' => 3,  
        'menu_icon' => 'dashicons-admin-multisite',  
    ];  
  
    register_post_type( 'immobilier', $args );  
}  
  
add_action( 'init', 'register_post_types' );
```

LES TYPES DE PUBLICATION PERSONNALISÉES (CUSTOM POST TYPE)

On déclare les CPT (Custom Post Type) au sein du fichier functions.php

On va utiliser la fonction **add_action()*** pour déclarer la fonction créant le CPT.

La fonction va exécuter une autre fonction nommée **register_post_type()** acceptant le nom du "custom post type" et un tableau d'arguments.

*Cette fonction est abordée plus loin.

```
function register_post_types() {  
  
    $labels = [  
        'name' => 'Immobilier',  
        'all_items' => 'Tous les biens immobiliers',  
        'singular_name' => 'Bien immobilier',  
        'add_new_item' => 'Ajouter un bien immobilier',  
        'edit_item' => 'Modifier le bien immobilier',  
        'menu_name' => 'Immobilier'  
    ];  
  
    $args = [  
        'labels' => $labels,  
        'public' => true,  
        'show_in_rest' => true,  
        'has_archive' => true,  
        'supports' => [ 'title', 'editor', 'thumbnail', 'custom-fields' ],  
        'menu_position' => 3,  
        'menu_icon' => 'dashicons-admin-multisite',  
    ];  
  
    register_post_type( 'immobilier', $args );  
}  
  
add_action( 'init', 'register_post_types' );
```

LES TYPES DE PUBLICATION PERSONNALISÉES (CUSTOM POST TYPE)

Le tableau **\$args** contient l'ensemble des paramètres en lien avec notre CPT.

- **labels** contient l'ensemble des intitulés liés à l'onglet du CPT
- **public** permet de spécifier si le CPT est public et visible par tous.
- **show_in_rest** permet d'afficher le CPT dans l'API Rest de Wordpress
- **has_archive** permet de spécifier si le CPT doit fonctionner comme un article ou une page

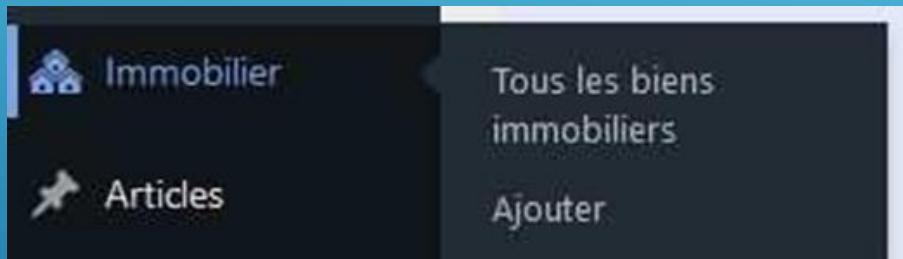
```
function register_post_types() {  
  
    $labels = [  
        'name' => 'Immobilier',  
        'all_items' => 'Tous les biens immobiliers',  
        'singular_name' => 'Bien immobilier',  
        'add_new_item' => 'Ajouter un bien immobilier',  
        'edit_item' => 'Modifier le bien immobilier',  
        'menu_name' => 'Immobilier'  
    ];  
  
    $args = [  
        'labels' => $labels,  
        'public' => true,  
        'show_in_rest' => true,  
        'has_archive' => true,  
        'supports' => [ 'title', 'editor', 'thumbnail', 'custom-fields' ],  
        'menu_position' => 3,  
        'menu_icon' => 'dashicons-admin-multisite',  
    ];  
  
    register_post_type( 'immobilier', $args );  
}  
add_action( 'init', 'register_post_types' );
```

LES TYPES DE PUBLICATION PERSONNALISÉES (CUSTOM POST TYPE)

Le tableau \$args contient l'ensemble des paramètres en lien avec notre CPT.

- **supports** permet de préciser les différentes fonctionnalités supportées par notre CPT
- **menu_position** gère la position de l'onglet du CPT
- **menu_icon** propose une icône pour l'onglet du CPT

LES TYPES DE PUBLICATION PERSONNALISÉES (CUSTOM POST TYPE)



Il est possible de retrouver l'ensemble des paramètres possibles pour les Custom Post Type en suivant ce lien :

https://developer.wordpress.org/reference/functions/register_post_type/

Lorsque l'on enregistre le fichier **functions.php**, le CPT est censé apparaître dans le 'back-office'

LES TAXONOMIES PERSONNALISÉES

Pour terminer ce module, il est important de préciser qu'il est possible de créer des taxonomies personnalisées.

Par défaut, il existe au sein de Wordpress : les catégories et les étiquettes.

Parfois, ces deux possibilités ne nous conviennent pas ou sont trop restreintes. Par exemple, j'aimerais une taxonomie pour les types de bien immobiliers.

LES TAXONOMIES PERSONNALISÉES

Les taxonomies personnalisées se déclarent de façon similaire au Custom Post Type.

On les déclare dans le fichier `functions.php`.

Cette fois, on utilise la fonction `register_taxonomy()`;

```
$labels = array(
    'name' => 'Type de bien immobilier',
    'new_item_name' => 'Nom du type de bien immobilier',
    'parent_item' => 'Type de bien immobilier parent',
);

$args = array(
    'labels' => $labels,
    'public' => true,
    'show_in_rest' => true,
    'hierarchical' => true,
);

register_taxonomy( 'type-immobilier', 'immobilier', $args );
}

add_action( 'init', 'register_post_types' );
```

LES TAXONOMIES PERSONNALISÉES

Les paramètres sont aussi similaires, si ce n'est "hierarchical" qui précise si la taxonomie doit fonctionner comme une étiquette ou une catégorie.

https://developer.wordpress.org/reference/functions/register_taxonomy/

Le deuxième paramètre de la fonction register_taxonomy correspond au Custom Post Type sur lequel la taxonomie sera disponible.

```
$labels = array(
    'name' => 'Type de bien immobilier',
    'new_item_name' => 'Nom du type de bien immobilier',
    'parent_item' => 'Type de bien immobilier parent',
);

$args = array(
    'labels' => $labels,
    'public' => true,
    'show_in_rest' => true,
    'hierarchical' => true,
);

register_taxonomy( 'type-immobilier', 'immobilier', $args );
}

add_action( 'init', 'register_post_types' );
```

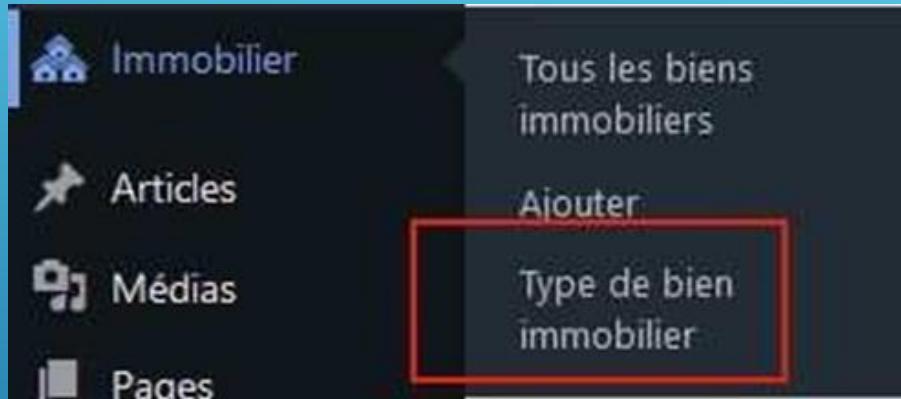
LES TAXONOMIES PERSONNALISÉES

La taxonomie apparaît ensuite du côté "back-office".

Il est possible d'intégrer les termes de la taxonomie à l'aide du code suivant :

```
<?php the_terms( get_the_ID() ,  
'type-immobilier' ); ?>
```

Le code peut être ajouté dans n'importe quel template



O5

Extensions et widgets

LES HOOKS

Les hooks font partie des points fondamentaux les plus importants autour de Wordpress.

En effet, à plusieurs reprises, nous avons employé des hooks sans forcément nous en rendre compte.

Les hooks sont des fonctions permettant d'interagir avec le cœur de Wordpress ou d'une extension.

Il est possible de retrouver une liste exhaustive des hooks en suivant ce lien (attention, tout n'est pas à jour).

https://adambrown.info/p/wp_hooks_hook

Les hooks

LES INTÉRÊTS DES HOOKS



Mise à jour

L'utilisation de hook permet de travailler avec les fichiers se trouvant dans les dossier wp-admin et wp-include sans craindre les mises à jour qui pourraient réinitialiser les modifications apportées



Bugs

Le fait de modifier directement les fichiers de ces dossiers pourrait amener à corrompre son dossier Wordpress.

Les hooks

LES TYPES DE HOOK



Action

Les hooks d'action permettent d'exécuter nos fonctions à un moment donné du chargement de Wordpress.

Ces hooks vont fonctionner avec la fonction `add_action()`.



Filtre

Les hooks de filtre permettent d'intercepter des données pour les modifier. Par exemple, des extensions SEO peuvent intercepter des données pour les analyser en lien avec le référencement.

Les hooks d'actions

LES HOOKS

L'image propose un exemple de hook d'action.

Nous ajoutons une feuille de style CSS. Pour cela, on emploie la fonction `add_action()`;

Elle accepte deux paramètres, la fonction que l'on veut exécuter en second paramètre.

Le premier paramètre correspond à l'instant où la fonction devra s'exécuter.

En l'occurrence, il s'agit du moment où les scripts et les styles sont ajoutés.

```
function add_css() {  
    wp_enqueue_style(  
        'main_css',  
        get_template_directory_uri() . '/css/main.css',  
        array(),  
        '1.0'  
    );  
}  
add_action( 'wp_enqueue_scripts', 'add_css' );
```

LES HOOKS

Les hooks d'actions

Les hooks d'action correspondent à ce premier paramètre.

Vous pouvez vous référer au lien précédent cité pour retrouver une liste de hook d'action.

Il est possible de supprimer un hook d'action à l'aide de la fonction `remove_action()`;

```
function add_css() {  
    wp_enqueue_style(  
        'main_css',  
        get_template_directory_uri() . '/css/main.css',  
        array(),  
        '1.0'  
    );  
    add_action( 'wp_enqueue_scripts', 'add_css' );  
}
```

LES HOOKS

Les hooks de filtre

```
if( ! function_exists( 'prefix_custom_excerpt_length' ) )
{
    function prefix_custom_excerpt_length( $length )
    {
        return 40;
    }
}
add_filter( 'excerpt_length', 'prefix_custom_excerpt_length', 999 );
```

L'image propose un exemple de hook de filtre. Ici, on intercepte la taille maximum du résumé d'un article pour spécifier un nombre limité.

Pour cela, on utilise la fonction **add_filter()** qui va permettre d'intercepter une donnée pour la modifier.

La fonction va accepter deux paramètres.

LES HOOKS

Les hooks de filtre

```
if( ! function_exists( 'prefix_custom_excerpt_length' ) )
{
    function prefix_custom_excerpt_length( $length )
    {
        return 40;
    }
}
add_filter( 'excerpt_length', 'prefix_custom_excerpt_length', 999 );
```

La fonction va accepter deux paramètres :

- La donnée a intercepté.
- La fonction a exécuté avec sa valeur de retour qui va modifier la donnée.

Il est possible d'utiliser `remove_filter()` pour supprimer un hook de filtre.

LES HOOKS

```
if( ! function_exists( 'prefix_custom_excerpt_length' ) )
{
    function prefix_custom_excerpt_length( $length )
    {
        return 40;
    }
}
add_filter( 'excerpt_length', 'prefix_custom_excerpt_length', 999 );
```

Conclusions

Les deux fonctions `add_action()` et `add_filter()` acceptent deux paramètres supplémentaires :

- **La priorité du hook** (chaque hook se retrouvent dans une file d'attente) avec un type `int` (plus le nombre est petit plus la priorité est haute). Exemple : 999.
- **Le nombre d'arguments acceptés par notre fonction** avec un type `int`.

LES HOOKS

Conclusions

Les hooks sont utilisés dans les thèmes, mais le sont aussi au sein des extensions.

Il faut garder à l'esprit que ce système vous permettra de mettre en place énormément de fonctionnalité lors de vos phases de développement avec Wordpress.

LES EXTENSIONS

Les extensions font partie des éléments incontournables de Wordpress.

Ils proposent de repousser les limites du CMS en intégrant un panel de fonctionnalités presque "illimitées".

Un double problème peut se poser avec ce système d'extension :

- On pourra vouloir des fonctionnalités qui n'existent pas même avec des extensions.
- Certaines extensions peuvent coûter assez cher.

La solution serait donc de développer ses propres extensions

LES EXTENSIONS

En dehors du développement de nos propres fonctionnalités.

L'intérêt supplémentaire du développement d'extension est le fait de pouvoir découper son thème.

Après plusieurs développements au sein de son thème, on peut être amené à se demander si certaines des fonctionnalités ne pourront pas faire l'objet d'extension pour plus d'organisations.

```
<?php
/**
 * @package Hello_Dolly
 * @version 1.7.2
 */
/*
Plugin Name: Hello Dolly
Plugin URI: http://wordpress.org/plugins/hello-dolly/
Description: This is not just a plugin, it symbolizes
Author: Matt Mullenweg
Version: 1.7.2
Author URI: http://ma.tt/
*/
```

ANALYSE D'UNE EXTENSION : HELLO DOLLY

La première partie à mettre en place pour créer une extension est l'ajout de commentaire comme nous avons pu le faire au sein de style.css pour le thème.

Cette fois, les informations sont un peu différentes, on utilise :

- Plugin Name : nom de l'extension
- Plugin URI : lien vers le site de l'extension

<https://developer.wordpress.org/plugins/plugin-basics/header-requirements/>

```
// This just echoes the chosen line, we'll position it later.
function hello_dolly() {
    $chosen = hello_dolly_get_lyric();
    $lang   = '';
    if ('en_' !== substr( get_user_locale(), 0, 3 ) ) {
        $lang = ' lang="en"';
    }

    printf(
        '<p id="dolly"><span class="screen-reader-text">%s </span><span dir="ltr">%s>%s</span></p>',
        __( 'Quote from Hello Dolly song, by Jerry Herman:' ),
        $lang,
        $chosen
    );
}

// Now we set that function up to execute when the admin_notices action is called.
add_action( 'admin_notices', 'hello_dolly' );
```

ANALYSE D'UNE EXTENSION : HELLO DOLLY

En seconde partie, il ne reste plus qu'à développer l'extension à l'aide des hooks.

Dans le cas de l'extension Hello Dolly, nous avons une fonction qui se déclenche au moment de "l'`admin_notices`".

L'admin notice représente les petits encadrés nous proposant des informations (par exemple : lors de la désactivation d'une extension).

```
// We need some CSS to position the paragraph
function dolly_css() {
    echo "
<style type='text/css'>
#dolly {
    float: right;
    padding: 5px 10px;
    margin: 0;
    font-size: 12px;
    line-height: 1.6666;
}
.rtl #dolly {
    float: left;
}
.block-editor-page #dolly {
    display: none;
}
@media screen and (max-width: 782px) {
    #dolly,
    .rtl #dolly {
        float: none;
        padding-left: 0;
        padding-right: 0;
    }
}
</style>
";
}

add_action( 'admin_head', 'dolly_css' );
```

ANALYSE D'UNE EXTENSION : HELLO DOLLY

Hello Dolly propose une dernière fonction ajoutant du style à l'écriture des paroles de la musique.

Pour intégrer le style, un nouveau hook d'action est employé : "admin_head". Ce hook permet d'ajouter des éléments dans toutes les en-têtes des pages d'administrations.

En conclusion, il faut retenir que le système de hook est très important pour la conception d'extension et de thème. La conception d'extension est similaire à celle des thèmes.

LES WIDGETS

Pour clore ce module, nous allons aborder la création de widget au sein de son thème Wordpress.

En effet, si nous avons la possibilité de créer des extensions, il est tout à fait possible de créer des widgets.

Nos widgets créés apparaîtront dans le "back-office".

LES WIDGETS

Wordpress nous propose une classe **WP_Widget** qui permet de créer un nouveau widget.

La première étape est d'hériter de cette classe.

Notre classe doit contenir certains éléments importants :

- **Le constructor** : il permet la création d'un ID, d'un titre et d'une description pour notre widget.

```
class mon_widget extends WP_Widget {  
  
    function __construct() {  
  
    }  
  
    public function widget( $args, $instance ) {  
  
    }  
  
    public function form( $instance ) {  
  
    }  
  
    public function update( $new_instance, $old_instance ) {  
  
    }  
}
```

LES WIDGETS

- La méthode `widget()` : cette méthode gère la valeur de sortie qui apparaîtra à l'utilisation du widget.
- La méthode `form()` : cette méthode permet de gérer la partie que l'on voit sur le "back-office". Les options de paramétrages de notre widget.
- La méthode `update()`: cette méthode permet de remplacer l'ancienne version du widget par la nouvelle en base de données, lors de modification.

```
class mon_widget extends WP_Widget {  
  
    function __construct() {  
  
    }  
  
    public function widget( $args, $instance ) {  
  
    }  
  
    public function form( $instance ) {  
  
    }  
  
    public function update( $new_instance, $old_instance ) {  
  
    }  
}
```

LES WIDGETS

On utilise le constructor du parent pour paramétrer les informations de notre widget.

La première information est l'ID de notre widget.

La seconde information est le nom du widget qui apparaît dans le "back-office".

La dernière information est la description du widget.

```
function __construct() {
    parent::__construct(
        'mon_widget',
        __('Mon Widget', 'mon_widget_domain'),
        array(
            'description' => __(
                'Simple widget pour le support', 'mon_widget_domain'
            ),
        )
    );
}
```

LES WIDGETS

La méthode `widget` représente ce qui sera affiché lors de l'utilisation du widget.

Dans notre cas présent, on affiche le titre que l'on récupère grâce au paramètre `$instance` (ce paramètre a pour rôle de récupérer les données liées à notre widget en base de données).

On affiche aussi le texte "Bonjour le monde!"

`$args` permet de définir l'emplacement de chaque élément (avant, après...)

```
public function widget( $args, $instance ) {
    $title = apply_filters( 'widget_title', $instance['title'] );

    echo $args['before_widget'];
    if ( ! empty( $title ) )
        echo $args['before_title'] . $title . $args['after_title'];

    echo __( 'Bonjour le monde!', 'mon_widget_domain' );
    echo $args['after_widget'];
}
```

LES WIDGETS

```
public function form( $instance ) {
    if ( isset( $instance[ 'title' ] ) ) {
        $title = $instance[ 'title' ];
    }
    else {
        $title = __( 'New title', 'mon_widget_domain' );
    }
    ?>
    <p>
        <label for=<?php echo $this->get_field_id( 'title' ); ?>><?php _e( 'Title:' ); ?></label>
        <input class="widefat" id=<?php echo $this->get_field_id( 'title' ); ?>" name=<?php echo $this->get_field_name( 'title' ); ?>" type="text"
            value=<?php echo esc_attr( $title ); ?>" />
    </p>
    <?php
}
```

La méthode **form()** permet l'ajout d'un formulaire du côté du back-office.

Ici, nous récupérons simplement une valeur pour l'affecter à title. Title sera affiché lors de l'utilisation du widget.

On vérifie dans un premier temps si un titre existe déjà pour l'afficher par défaut dans le champ de formulaire.

LES WIDGETS

```
public function update( $new_instance, $old_instance ) {  
    $instance = array();  
    $instance['title'] = ( ! empty( $new_instance['title'] ) ) ? strip_tags( $new_instance['title'] ) : '';  
    return $instance;  
}
```

La méthode **update()** a pour rôle d'enregistrer la dernière version du widget. On récupère le nouveau titre pour l'enregistrer en base de données.

Il est ensuite utilisé sur les fonctions précédentes.

LES WIDGETS

La dernière étape consiste à utiliser un hook d'action pour charger le widget sur notre thème.

Le widget se nomme "widgets_init", on lui propose une fonction qui va exécuter une autre fonction nommée `register_widget()`.

Cette fonction accepte comme paramètre notre classe.

```
function mon_widget() {  
    register_widget( 'mon_widget' );  
}  
add_action( 'widgets_init', 'mon_widget' );
```

O6

Aspects techniques divers

APERÇU DU FICHIER .HTACCESS

Le fichier .htaccess est un fichier de configuration lié à Apache.

On en retrouve dans tous les sites faits à partir de Wordpress et utilisant Apache.

On peut modifier ce fichier pour améliorer la sécurité, augmenter la vitesse du site, mettre en place des redirections.

Il est possible de mettre en place plusieurs .htaccess dans nos dossiers Wordpress.

LE FICHIER .HTACCESS

Aperçu

Il est important de noter qu'il vaut mieux créer une copie de notre fichier .htaccess.

Des modifications non-fonctionnelles pourraient entraîner beaucoup de problèmes au sein de votre site.

Lors de la modification des permaliens au sein de l'administration, le fichier .htaccess est modifié.

```
# BEGIN WordPress
# Les directives (lignes) entre « BEGIN WordPress » et « END WordPress » sont générées
# dynamiquement, et doivent être modifiées uniquement via les filtres WordPress.
# Toute modification des directives situées entre ces marqueurs sera surchargée.
<IfModule mod_rewrite.c>
RewriteEngine On
RewriteRule .* - [E=HTTP_AUTHORIZATION:{HTTP:Authorization}]
RewriteBase /wordpress%20(3)/wordpress/
RewriteRule ^index\.php$ - [L]
RewriteCond %{REQUEST_FILENAME} !-f
RewriteCond %{REQUEST_FILENAME} !-d
RewriteRule . /wordpress%20(3)/wordpress/index.php [L]
</IfModule>

# END WordPress
```

LE FICHIER .HTACCESS

Aperçu

<IfModule></IfModule : ces balises ne sont exécutées que pour les hébergeurs autorisant la réécriture d'URL .

mod_rewrite.c : permet de manipuler les URL .

RewriteEngine On : permet d'activer ou de désactiver le moteur de réécriture des URL. Si on précise Off, toutes les règles suivantes n'ont pas d'intérêt.

```
# BEGIN WordPress
# Les directives (lignes) entre « BEGIN WordPress » et « END WordPress » sont générées
# dynamiquement, et doivent être modifiées uniquement via les filtres WordPress.
# Toute modification des directives situées entre ces marqueurs sera surchargée.
<IfModule mod_rewrite.c>
RewriteEngine On
RewriteRule .* - [E=HTTP_AUTHORIZATION:%{HTTP:Authorization}]
RewriteBase /wordpress%20(3)/wordpress/
RewriteRule ^index\.php$ - [L]
RewriteCond %{REQUEST_FILENAME} !-f
RewriteCond %{REQUEST_FILENAME} !-d
RewriteRule . /wordpress%20(3)/wordpress/index.php [L]
</IfModule>

# END WordPress
```

LE FICHIER .HTACCESS

Aperçu

RewriteRule .* - [E=HTTP_AUTHORIZATION:%{HTTP:Authorization}] : permet l'utilisation du protocole HTTP.

RewriteBase /wordpress%20(3)/wordpress/ : cette ligne précise où se trouve la racine du dossier

```
# BEGIN WordPress
# Les directives (lignes) entre « BEGIN WordPress » et « END WordPress » sont générées
# dynamiquement, et doivent être modifiées uniquement via les filtres WordPress.
# Toute modification des directives situées entre ces marqueurs sera surchargée.
<IfModule mod_rewrite.c>
RewriteEngine On
RewriteRule .* - [E=HTTP_AUTHORIZATION:%{HTTP:Authorization}]
RewriteBase /wordpress%20(3)/wordpress/
RewriteRule ^index\.php$ - [L]
RewriteCond %{REQUEST_FILENAME} !-f
RewriteCond %{REQUEST_FILENAME} !-d
RewriteRule . /wordpress%20(3)/wordpress/index.php [L]
</IfModule>

# END WordPress
```

Aperçu

LE FICHIER .HTACCESS

```
RewriteRule ^index\.php$ - [L]
RewriteCond %{REQUEST_FILENAME} !-f
RewriteCond %{REQUEST_FILENAME}
!-d
RewriteRule .
/wordpress%20(3)/wordpress/index.ph
p [L]
```

Ces quatre lignes sont liées.

^index\.php\$ - [L] : index.php a pour rôle de rediriger tous les accès lorsque l'utilisateur consulte un fichier non-réel (PDF, image...)

La dernière ligne précise que Le «,» doit être remplacé par index.php pour tout ce qui n'est pas un fichier non-réel.

```
# BEGIN WordPress
# Les directives (lignes) entre « BEGIN WordPress » et « END WordPress » sont générées
# dynamiquement, et doivent être modifiées uniquement via les filtres WordPress.
# Toute modification des directives situées entre ces marqueurs sera surchargée.
<IfModule mod_rewrite.c>
RewriteEngine On
RewriteRule .* - [E=HTTP_AUTHORIZATION:%{HTTP:Authorization}]
RewriteBase /wordpress%20(3)/wordpress/
RewriteRule ^index\.php$ - [L]
RewriteCond %{REQUEST_FILENAME} !-f
RewriteCond %{REQUEST_FILENAME} !-d
RewriteRule . /wordpress%20(3)/wordpress/index.php [L]
</IfModule>

# END WordPress
```

LE FICHIER .HTACCESS

Optimisation

Ajoutons quelques règles pour optimiser le référencement :

```
RewriteRule ^category/(.+)$  
localhost/wordpress%20(3)/wordpress  
/$1 [R=301,L]
```

Cette règle permet de supprimer "category" des urls.

```
<Ifmodule mod_expires.c>  
<filesmatch "\.(jpg|gif|png|css|js)$">  
ExpiresActive on  
ExpiresDefault "access plus 1 year"  
</filesmatch> </ifmodule>
```

Cette règle permet l'autorisation d'utiliser le système de cache.

```
# BEGIN WordPress  
# Les directives (lignes) entre « BEGIN WordPress » et « END WordPress » sont générées  
# dynamiquement, et doivent être modifiées uniquement via les filtres WordPress.  
# Toute modification des directives situées entre ces marqueurs sera surchargée.  
<IfModule mod_rewrite.c>  
RewriteEngine On  
RewriteRule .* - [E=HTTP_AUTHORIZATION:%{HTTP:Authorization}]  
RewriteBase /wordpress%20(3)/wordpress/  
RewriteRule ^index\.php$ - [L]  
RewriteCond %{REQUEST_FILENAME} !-f  
RewriteCond %{REQUEST_FILENAME} !-d  
RewriteRule . /wordpress%20(3)/wordpress/index.php [L]  
</IfModule>  
  
# END WordPress
```

LE FICHIER .HTACCESS

Optimisation

```
RewriteEngine on
RewriteCond %{REQUEST_URI}
!/maintenance.html$
RewriteCond %{REMOTE_ADDR}
!^123\.123\.123\.123
RewriteRule $ /maintenance.html
[R=302,L]
```

Cette règle permet de rediriger les utilisateurs vers une page de maintenance.

```
# BEGIN WordPress
# Les directives (lignes) entre « BEGIN WordPress » et « END WordPress » sont générées
# dynamiquement, et doivent être modifiées uniquement via les filtres WordPress.
# Toute modification des directives situées entre ces marqueurs sera surchargée.
<IfModule mod_rewrite.c>
    RewriteEngine On
    RewriteRule .* - [E=HTTP_AUTHORIZATION:{HTTP:Authorization}]
    RewriteBase /wordpress%20(3)/wordpress/
    RewriteRule ^index\.php$ - [L]
    RewriteCond %{REQUEST_FILENAME} !-f
    RewriteCond %{REQUEST_FILENAME} !-d
    RewriteRule . /wordpress%20(3)/wordpress/index.php [L]
</IfModule>

# END WordPress
```

RÉFÉRENCEMENT (RÉÉCRITURE DES URL, MÉTAS, SITEMAPS, GOOGLE ANALYTICS, ...).

Il existe plusieurs possibilités pour améliorer son référencement au sein de son site Wordpress.

Nous allons aborder quelques points de référencement en plus de ce que nous avons pu voir précédemment avec le .htaccess

De la réécriture des URL à l'utilisation de google analytics, le CMS Wordpress aborde beaucoup de solutions possibles.

LE RÉFÉRENCEMENT

```
add_rewrite_tag( '%brand%', '([^&]+)' );
add_rewrite_tag( '%color%', '([^&]+)' );
add_rewrite_tag( '%size%' , '([^&]+)' );

add_rewrite_rule(
    'catalogue/([^/]+)/([^.]+)/([^.]+)',
    'index.php?post_type=catalogue&brand=$matches[1]&color=$matches[2]&size=$matches[3]',
    'top'
);
```

Le premier point d'intervention est la réécriture des URL.

WordPress gère automatiquement les URL, cependant on peut créer des URL sur mesure pour répondre à nos besoins.

On utilisera des expressions régulières pour gérer leur formatage.

LE RÉFÉRENCEMENT

Le second point est l'utilisation des balises métas.

Il nous est possible d'ajouter des balises métas au sein des templates de notre Thème.

```
<meta name="description" content="" />

<meta name="description" content="php if ( is_single() ) {
    single_post_title('', true);
} else {
    bloginfo('name');
    echo " - ";
    bloginfo('description');
}?" /&gt;</pre
```

On ajoute la balise `<meta name="description" />` dans le fichier header.php

La seconde balise meta a pour objectif d'afficher le contenu de l'article s'il y en a un pour ne pas à avoir à remplir chaque balise meta. Ici, vous aurez un automatisme.

On peut utiliser Yoast SEO pour gérer les métas

LE RÉFÉRENCEMENT

XML Sitemap

Generated by **YoastSEO**, this is an XML Sitemap, meant for consumption by search engines.

You can find more information about XML sitemaps on [sitemaps.org](https://www.sitemaps.org).

This XML Sitemap contains 739 URLs.

URL

<https://yoast.com/seo-blog/>
<https://yoast.com/archive-seo/>
<https://yoast.com/introducing-yoast-comment-hacks/>
<https://yoast.com/google-panda-google-core-algorithm/>
<https://yoast.com/whipping-your-hosting-into-shape/>
<https://yoast.com/google-analytics-custom-dimensions/>
<https://yoast.com/plugin-future/>
<https://yoast.com/ab-testing-your-newsletters/>
<https://yoast.com/analyze-drop-traffic/>
<https://yoast.com/ranking-your-local-business-part-7-social-signals/>
<https://yoast.com/love-optimized-website/>
<https://yoast.com/google-webmaster-guidelines-update/>
<https://yoast.com/search-seo-2018/>
<https://yoast.com/yoast-seo-for-magento2-and-typo3/>
<https://yoast.com/the-sense-and-nonsense-of-xml-sitemaps/>
<https://yoast.com/google-core-algorithm-update/>
<https://yoast.com/copywriting-mobile/>

	Images	Last Mod.
0	2021-04-29 12:31 +00:00	
2	2016-12-20 09:00 +00:00	
3	2017-03-02 12:34 +00:00	
1	2017-03-08 10:11 +00:00	
1	2017-03-29 09:00 +00:00	
5	2017-06-01 13:22 +00:00	
2	2017-08-21 14:26 +00:00	
2	2017-10-23 11:16 +00:00	
7	2018-04-30 11:30 +00:00	
8	2018-04-30 13:19 +00:00	
1	2018-05-01 08:55 +00:00	
2	2018-05-01 08:56 +00:00	
1	2018-06-22 12:28 +00:00	
1	2018-06-27 11:09 +00:00	
1	2018-07-17 12:50 +00:00	
2	2018-07-17 13:16 +00:00	
1	2018-07-25 11:25 +00:00	

Le troisième point possible est la mise en place d'une sitemap.

Une sitemap est un plan de notre site web. Ce plan permet aux robots des moteurs de recherches de se retrouver sur notre site.

Il est possible d'utiliser deux extensions pour générer une sitemap :

- Yoast SEO
- WP Sitemap Page

Il est ensuite possible de soumettre la sitemap à Google via la Google Search Console

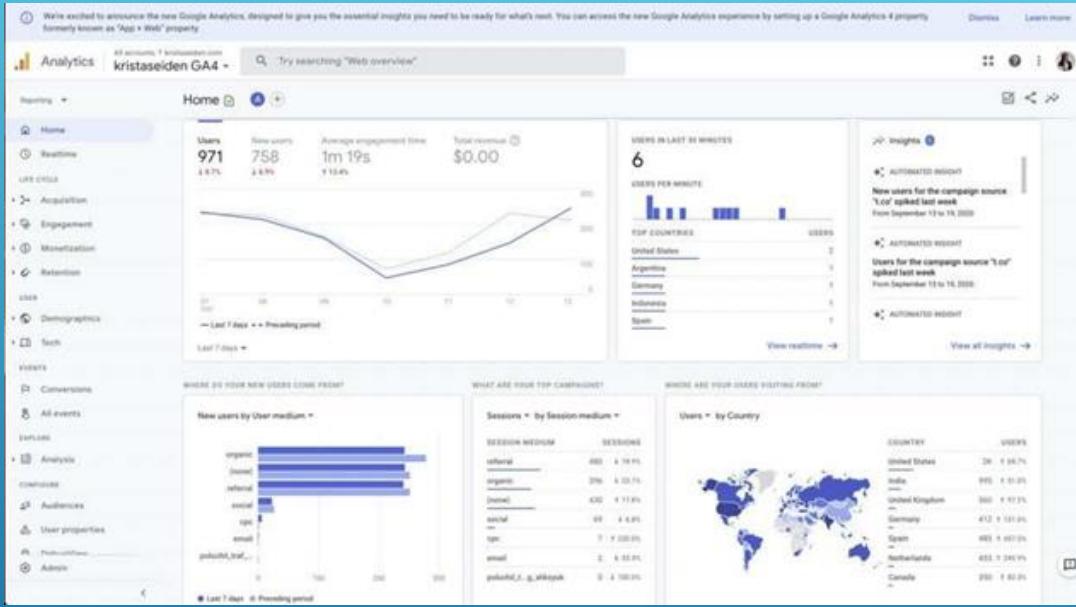
LE RÉFÉRENCEMENT

Un dernier point possible est l'utilisation de Google Analytics.

Google Analytics est un outil proposé par Google permettant d'obtenir un maximum de statistiques concernant notre site : le nombre de visiteurs, les performances de l'action marketing, des produits, du contenu...

Le lien suivant vous redirige vers un article très complet sur l'utilisation de Google Analytics :

<https://wpmarmite.com/google-analytics-wordpress/>



ANALYSE D'UN THÈME COMPATIBLE HTML5 (BONES, HTML5 BLANKS...)

Nous allons procéder à une analyse du thème Bones.

Bones est un thème de départ sur lequel on peut se baser pour réaliser rapidement un site avec Wordpress. En effet, il existe plusieurs thèmes permettant d'avoir des bases pour ne pas repartir à zéro à chaque création de thème au sein du CMS.

Ce thème correspond à plusieurs attentes que peut avoir le développeur.

Il suit tous les standards les plus récents du Web et permet par exemple d'utiliser le langage Sass.

ANALYSE DE BONES



Avantages

Bones proposent des bonnes bases pour commencer à développer son thème. Respect du HTML5, prise en compte du responsive avec un "mobile-first".

Il permet de travailler avec Less, CSS ou encore Sass

L'ensemble du code proposé par Bones est totalement commenté et compréhensible. Il permet des modifications simples et rapides.

ANALYSE DE BONES

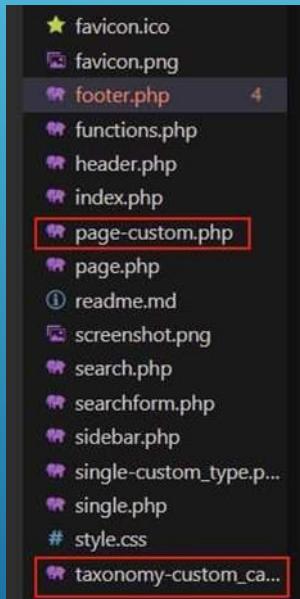
Functions.php



Au sein du fichier "functions.php" on y retrouve des fichiers intégrés qui permettent l'ajout des Custom Post Types, des sidebars ou encore la possibilité de modifier facilement l'administration.

Bones à commenter un ensemble de hook qui permettent d'apporter des suppressions rapide et simple des personnalisations apportées par la technologie.

ANALYSE DE BONES



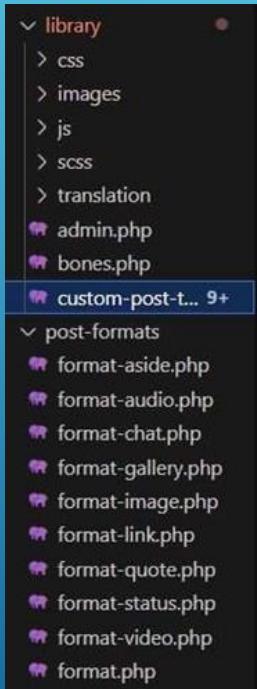
Structure

L'archive contient un ensemble de template qui peuvent être repris rapidement.

Par exemple, nous avons des templates personnalisés déjà prêt à l'utilisation pour des taxonomies ou des pages.

Il y a aussi des templates pour des barres de recherche.

ANALYSE DE BONES



Structure

Les dossiers "library" et "post-formats" contiennent tous les fichiers de dépendances sur lesquels on peut intervenir pour modifier notre thème.

On peut voir sur l'image le fichier gérant les Custom Post Type.

On peut conclure sur le fait que Bones est une aide incontestable pour démarrer un projet sans repartir de zéro.

07

Sécurité

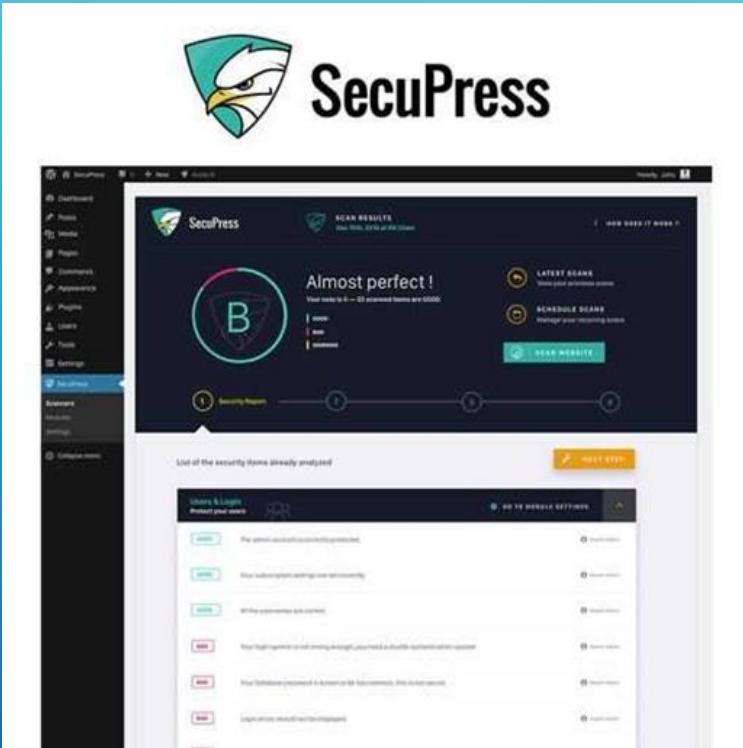
SÉCURISER PAR LES EXTENSIONS.

Il est important de considérer l'enjeu de sécurité qu'il peut y avoir autour de Wordpress.

On a tendance à penser que CMS est forcément garant de la sécurité sur nos sites.

La réalité est différente, entre les failles que peut posséder Wordpress et les failles que nous pouvons apporter en concevant nos thèmes et nos extensions.

Les extensions sont une première solution pour corriger des problèmes de sécurité.



SÉCURISER PAR LES EXTENSIONS.

UNE PREMIÈRE EXTENSION QUI PEUT ÊTRE UTILISÉE POUR LA SÉCURITÉ EST : SECUPRESS

Elle possède une version gratuite et une version payante. SecuPress va effectuer une analyse du site et proposer différents points d'amélioration.

L'analyse proposera une lettre qui augmentera selon les corrections que vous apportez.

Les différents points de sécurité sont regroupés au sein de catégorie. Une explication est proposée sur la manière de régler le problème de sécurité

<https://secupress.me/fr/>

The screenshot shows the Sucuri WP Plugin settings interface. At the top, there are tabs for General Settings, Scanner (which is highlighted with a purple arrow), Hardening, Post-Hack, Alerts, API Service Communication, and Website Info. Below the tabs, there's a section for 'API Key' with a note about its purpose and a field containing a long string of characters. Under 'Data Storage', there's a table listing files in the wp-content/uploads/sucuri directory. The table has columns for File Path, File Size, Status, and Writable. Most files are marked as existing and writable, except for a few which are listed as Does Not Exist or Not Writable.

File Path	File Size	Status	Writable
wp-content/uploads/sucuri/	1.49K	Exists	Writable
wp-content/uploads/sucuri/sucuri-auditlogs.php	1.12K	Exists	Writable
wp-content/uploads/sucuri/sucuri-auditqueue.php	0B	Does Not Exist	Not Writable
wp-content/uploads/sucuri/sucuri-blockedusers.php	0B	Does Not Exist	Not Writable
wp-content/uploads/sucuri/sucuri-failedlogins.php	0B	Does Not Exist	Not Writable
wp-content/uploads/sucuri/sucuri-honeypots.php	0B	Does Not Exist	Not Writable
wp-content/uploads/sucuri/sucuri-ignoresampling.php	0B	Exists	Writable
wp-content/uploads/sucuri/sucuri-integrity.php	94B	Exists	Writable

SÉCURISER PAR LES EXTENSIONS.

UNE SECONDE EXTENSION DE SÉCURITÉ COMPLÈTE : SUCURI SECURITY

Sucuri possède aussi une version gratuite et une version payante.

Sucuri embarque plusieurs modules de sécurité : un scanner de malware, une protection anti-DDOS, mesure des performances du site.

Cette extension reste plus complète que SecuPress.

<https://sucuri.net/>



Jetpack

SÉCURISER PAR LES EXTENSIONS.

LA DERNIÈRE EXTENSION EST :
JETPACK

Cette extension très connue est développée par des développeurs de Wordpress.com

Jetpack propose énormément d'outils pas seulement de sécurité.

Cependant, on peut noter qu'elle permet d'éviter les spams, de scanner les malware et proposer un système de backup performant.

<https://fr.jetpack.com/>

SÉCURISER AVEC LE FICHIER .HTACCESS

Le second point d'intervention pour protéger son site des attaques potentielles est la modification du fichier .htaccess

On a pu le voir précédemment ce fichier permet de réaliser plusieurs actions comme l'optimisation du référencement, mais pas seulement.

En effet, à l'aide de certaines règles, nous allons pouvoir protéger un peu plus notre site.

SÉCURISER LE FICHIER .HTACCESS

Ajout de règles

Options All -Indexes : cette règle permet de cacher les répertoires d'accès aux dossiers de notre site.

ServerSignature Off : cette règle permet de cacher les informations du serveur.

```
<Files wp-config.php>
order allow,deny
deny from all
</Files> : cette règle permet de
sécuriser le fichier wp-config.php
```

```
# BEGIN WordPress
# Les directives (lignes) entre « BEGIN WordPress » et « END WordPress » sont générées
# dynamiquement, et doivent être modifiées uniquement via les filtres WordPress.
# Toute modification des directives situées entre ces marqueurs sera surchargée.
<IfModule mod_rewrite.c>
RewriteEngine On
RewriteRule .* - [E=HTTP_AUTHORIZATION:{HTTP:Authorization}]
RewriteBase /wordpress%20(3)/wordpress/
RewriteRule ^index\.php$ - [L]
RewriteCond %{REQUEST_FILENAME} !-f
RewriteCond %{REQUEST_FILENAME} !-d
RewriteRule . /wordpress%20(3)/wordpress/index.php [L]
</IfModule>

# END WordPress
```

SÉCURISER LE FICHIER .HTACCESS

Ajout de règles

```
<Files ~ "^.*\.(Hh|Tt|AaPp)">  
order allow,deny  
deny from all  
satisfy all  
</Files> : cette règle permet de protéger le fichier .htaccess.
```

```
<IfModule mod_rewrite.c>  
RewriteCond %{QUERY_STRING}  
^author=([0-9]*) RewriteRule .* -  
[F] </IfModule> : permet de cacher l'identifiant de l'auteur d'un article.
```

```
# BEGIN WordPress  
# Les directives (lignes) entre « BEGIN WordPress » et « END WordPress » sont générées  
# dynamiquement, et doivent être modifiées uniquement via les filtres WordPress.  
# Toute modification des directives situées entre ces marqueurs sera surchargée.  
<IfModule mod_rewrite.c>  
RewriteEngine On  
RewriteRule .* - [E=HTTP_AUTHORIZATION:{HTTP:Authorization}]  
RewriteBase /wordpress%20(3)/wordpress/  
RewriteRule ^index\.php$ - [L]  
RewriteCond %{REQUEST_FILENAME} !-f  
RewriteCond %{REQUEST_FILENAME} !-d  
RewriteRule . /wordpress%20(3)/wordpress/index.php [L]  
</IfModule>  
  
# END WordPress
```

SÉCURISER LE FICHIER .HTACCESS

Ajout de règles

```
<IfModule mod_rewrite.c>
RewriteCond
%{REQUEST_METHOD} POST
RewriteCond %{REQUEST_URI}
.wp-comments-post\.\php*
RewriteCond %{HTTP_REFERER}
!.monsite.com.* [OR]
RewriteCond
%{HTTP_USER_AGENT} ^$ 
RewriteRule (.*)
^http://%\{REMOTE_ADDR\}/$
[R=301,L]
</IfModule> : cette règle permet
d'éviter le spam de commentaires.
```

```
# BEGIN WordPress
# Les directives (lignes) entre « BEGIN WordPress » et « END WordPress » sont générées
# dynamiquement, et doivent être modifiées uniquement via les filtres WordPress.
# Toute modification des directives situées entre ces marqueurs sera surchargée.
<IfModule mod_rewrite.c>
RewriteEngine On
RewriteRule .* - [E=HTTP_AUTHORIZATION:%{HTTP:Authorization}]
RewriteBase /wordpress%20(3)/wordpress/
RewriteRule ^index\.php$ - [L]
RewriteCond %{REQUEST_FILENAME} !-f
RewriteCond %{REQUEST_FILENAME} !-d
RewriteRule . /wordpress%20(3)/wordpress/index.php [L]
</IfModule>

# END WordPress
```

SÉCURISER LE FICHIER .HTACCESS

Ajout de règles

```
<Limit GET POST> order  
allow,deny deny from  
xxx.xxx.xxx.xxx allow from all  
</Limit> : il est possible de bannir  
des IP (en remplaçant les xxx).
```

```
<IfModule mod_rewrite.c>  
RewriteEngine on RewriteCond  
%{HTTP_REFERER} monsite1.com  
[NC,OR] RewriteCond  
%{HTTP_REFERER} monsite2.com  
[NC,OR] RewriteRule .* - [F]  
</ifModule> : cette règle permet  
de bloquer des visiteurs  
provenant d'un autre site.
```

```
# BEGIN WordPress  
# Les directives (lignes) entre « BEGIN WordPress » et « END WordPress » sont générées  
# dynamiquement, et doivent être modifiées uniquement via les filtres WordPress.  
# Toute modification des directives situées entre ces marqueurs sera surchargée.  
<IfModule mod_rewrite.c>  
RewriteEngine On  
RewriteRule .* - [E=HTTP_AUTHORIZATION:%{HTTP:Authorization}]  
RewriteBase /wordpress%20(3)/wordpress/  
RewriteRule ^index\.php$ - [L]  
RewriteCond %{REQUEST_FILENAME} !-f  
RewriteCond %{REQUEST_FILENAME} !-d  
RewriteRule . /wordpress%20(3)/wordpress/index.php [L]  
</IfModule>  
  
# END WordPress
```

SÉCURISER PAR LA GESTION DES UTILISATEURS

Le dernier point d'action sur lequel on peut intervenir est la gestion des utilisateurs.

En effet, les utilisateurs vont pouvoir posséder des rôles. Chaque rôle aura ses propres autorisations sur notre site Wordpress.

Moins les utilisateurs auront de droits, plus le site sera sécurisé. Il est impératif de protéger les comptes possédant les droits les plus importants.

SÉCURISER PAR LA GESTION DES UTILISATEURS

La sécurisation par la gestion utilisateur passe avant tout par la diminution des droits des utilisateurs.

Pour cela, il est possible d'utiliser une extension : **User Role Editor**

Cette extension permet de gérer en profondeur le système de rôle de Wordpress. On va pouvoir modifier les droits de chacun des rôles proposés et créer de nouveaux rôles.

L'avantage principal de l'extension est qu'elle s'adapte aux autres extensions.

SÉCURISER PAR LA GESTION DES UTILISATEURS

The screenshot shows the 'User Role Editor' interface in a WordPress admin area. The top navigation bar includes 'Well, hello, Dolly'. A dropdown menu at the top left says 'Sélectionner un rôle et changer ses permissions : Abonné (subscriber)'. Below this are two checkboxes: 'Afficher les permissions dans une forme invisible par l'homme' and 'Afficher les permissions obsolètes'. The main content area has a 'Groupe (Total/Accordé)' section with a tree view of permissions. The 'Tout (73/2)' node is expanded, showing categories like 'Cour', 'Général', 'Thèmes', 'Articles', 'Pages', 'Extensions', 'Comptes', 'Obsolète', 'Type de contenu personnalisé', 'Groupes de champs', 'Champs', 'Types personnalisés', 'Permissions personnalisées', and 'User Role Editor'. To the right of the tree view is a large list of individual permission names, many of which are checked. At the bottom of the list are buttons for 'Mettre à jour', 'Ajouter un rôle', 'Renommer le Rôle', and 'Ajouter la permission'. The bottom of the screen features a toolbar with icons for 'Tout surligner', 'Respecter la casse', 'Respecter les accents et diacritiques', 'Mots entiers', and 'Occurrence 6 sur 25'.

La méthode **update()** a pour rôle d'enregistrer la dernière version du widget. On récupère le nouveau titre pour l'enregistrer en base de données.

Il est ensuite utilisé sur les fonctions précédentes.

SÉCURISER PAR LA GESTION DES UTILISATEURS

La sécurisation par la gestion utilisateur passe dans un second temps par la sécurisation des utilisateurs possédant les rôles les plus importants (admin).

Il est important que ces utilisateurs possèdent des mots de passe renforcés.
La double authentification est une possibilité (Two Factor Authentication plugin).

Il vaut mieux supprimer les comptes nommés "admin", "moderator".



O8

Maintenance Et Migration

MISE À JOUR

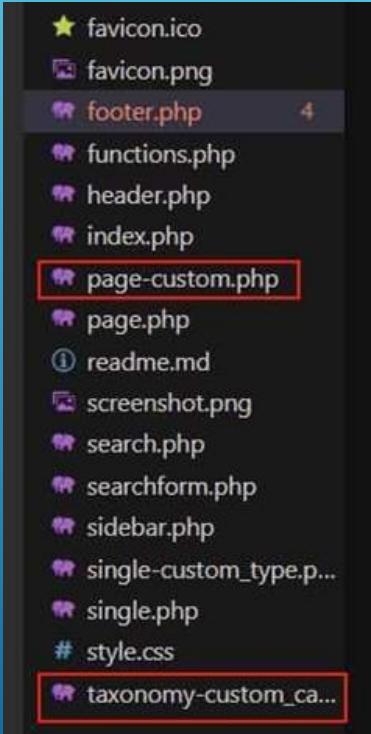
La mise à jour de Wordpress et des extensions est une action importante pour le processus de développement et de maintien de notre site.

La mise à jour permettra la résolution de bug, l'amélioration de certaines fonctionnalités et apportera une réelle amélioration de la sécurité.

En effet, évoluer sur une version ancienne de Wordpress n'est pas sans conséquence.

L'objectif de cette partie est d'établir les différentes étapes de mise à jour.

MISE À JOUR



Sauvegarde

La première étape à suivre est de sauvegarder son site Wordpress.

La sauvegarde passe par deux éléments :

- La sauvegarde la base de données.
- La copie de son dossier Wordpress.

MISE À JOUR

The screenshot shows the 'Export' page of PHPMyAdmin for the 'wordpress' database. The top navigation bar includes links for Structure, SQL, Rechercher, Requête, Exporter, Importer, Opérations, and Privilégiés. The main title is 'Exportation des tables depuis la base de données « wordpress »'. Under 'Modèles d'exportation', there is a 'Nouveau modèle' input field containing 'wordpress' with a 'Créer' button next to it. A 'Modèles existants' section shows a dropdown menu set to 'Sélectionner un modèle' with 'Mettre à jour' and 'Supprimer' buttons. Below this is the 'Méthode d'exportation' section, which has two radio buttons: 'Rapide, n'afficher qu'un minimum d'options' (selected) and 'Personnalisée, afficher toutes les options possibles'. The 'Format:' section shows a dropdown menu set to 'SQL'. The bottom of the page features a footer with a 'Rechercher' search bar and a 'Documentation' link.

Sauvegarde

Pour sauvegarder la base de données, on peut se connecter à PHPmyAdmin et exporter la base de données avec les données.

Il est possible sur certains hébergeurs de mettre en place des tâches CRON qui s'occupent de sauvegarder la base de données selon un temps défini (OVH / O2Switch).

MISE À JOUR

Mettre à jour

L'étape suivante est d'identifier les éléments à mettre à jour :

- Wordpress
- Extension
- Les langues
- Les thèmes

La mise à jour n'aura pas le même impact selon les éléments mis à jour.

The screenshot shows the WordPress dashboard with a dark sidebar. The sidebar includes links for Accueil, Mises à jour (with a red notification badge showing 3), Articles, Médias, Pages, and Commentaires. The main content area is titled 'Mises à jour de WordPress'. It displays a message: 'Vous avez la dernière version de WordPress. Les prochaines mises à...' followed by 'Si vous souhaitez réinstaller la version 4.5.2-fr_FR, vous pouvez le faire ici, ou télécharger l'an...'. Below this are three buttons: 'Ré-installer maintenant', 'Télécharger la version 4.5.2-fr_FR', and 'Masquer cette mise à jour'. A small note at the bottom states: 'Cette version localisée contient à la fois la traduction et divers correctifs liés à la localisation. Si vous souhaitez cons...'. At the top of the content area, there's a status bar with 'Dernière vérification le 17 mai 2016 à 22 h 44 min.' and a 'Vérifier à nouveau' button.

MISE À JOUR

Mettre à jour

The screenshot shows the WordPress dashboard with a dark theme. The top navigation bar includes links for WordPress, Tableau de bord, Accueil, Mises à jour (with a red notification badge), Articles, Médias, Pages, and Commentaires. The main content area is titled 'Mises à jour de WordPress'. It displays a message stating 'Vous avez la dernière version de WordPress. Les prochaines mises à jour...' (You have the latest version of WordPress. Future updates...). Below this, there are three buttons: 'Ré-installer maintenant', 'Télécharger la version 4.5.2-fr_FR', and 'Masquer cette mise à jour'. A small note at the bottom says 'Cette version localisée contient à la fois la traduction et divers correctifs liés à la localisation. Si vous souhaitez consulter les détails de la mise à jour, cliquez sur Télécharger la version 4.5.2-fr_FR.' (This localized version contains both the translation and various localization-related fixes. If you want to view the details of the update, click on 'Télécharger la version 4.5.2-fr_FR').

Une fois que l'on a identifié les éléments à mettre à jour, on peut procéder à la mise à jour :

- Depuis Wordpress 3.7, des mises à jour automatiques sont effectuées pour les mises à jour mineurs et les mises à jour de sécurité.
- Pour les autres mises à jour, on aura toujours un bouton sur lequel un clic suffit pour mettre les éléments à jour.

MISE À JOUR

Conseils

The screenshot shows the WordPress dashboard with a dark sidebar and a light main content area. The sidebar includes links for Tableau de bord, Accueil, Mises à jour (with a red notification badge showing 3), Articles, Médias, Pages, and Commentaires. The main content area is titled 'Mises à jour de WordPress'. It displays a message: 'Vous avez la dernière version de WordPress. Les prochaines mises à jour sont prévues pour le 17 mai 2016.' Below this, there's a note about the localization version. Three buttons are visible: 'Ré-installer maintenant', 'Télécharger la version 4.5.2-fr_FR', and 'Masquer cette mise à jour'.

Lors de la mise à jour d'une version de Wordpress, il est conseillé de désactiver l'ensemble des extensions pour éviter tout conflit de version.

On peut réaliser cela facilement en renommant le fichier contenant les extensions.

Il est important de vérifier les prérequis de la version (version de PHP, de MySQL...).

MISE À JOUR

The screenshot shows the WordPress dashboard with a dark sidebar. The sidebar includes links for 'Tableau de bord', 'Accueil', 'Mises à jour' (with a red notification badge), 'Articles', 'Médias', 'Pages', and 'Commentaires'. The main content area is titled 'Mises à jour de WordPress' and displays a message: 'Vous avez la dernière version de WordPress. Les prochaines mises à jour sont prévues pour le 17 mai 2016.' It also mentions a download link for 'Télécharger la version 4.5.2-fr_FR'. At the bottom, there's a note about localization.

Conseils

Il est important de bien choisir son horaire puisque le site pourrait potentiellement rester indisponible durant une durée indéterminée.

MISE À JOUR

Echec de la mise à jour

The screenshot shows the WordPress dashboard with a dark sidebar. The sidebar includes links for Tableau de bord, Accueil, Mises à jour (with a red notification badge showing 3), Articles, Médias, Pages, and Commentaires. The main content area is titled "Mises à jour de WordPress" and displays a message: "Vous avez la dernière version de WordPress. Les prochaines mises à jour sont prévues pour le 17 mai 2016 à 22 h 44 min." It also includes buttons for "Ré-installer maintenant", "Télécharger la version 4.5.2-fr_FR", and "Masquer cette mise à jour". A small note at the bottom states: "Cette version localisée contient à la fois la traduction et divers correctifs liés à la localisation. Si vous souhaitez consulter les détails de la mise à jour, veuillez accéder à l'onglet Mises à jour de l'administration." The top navigation bar has links for WordPress, Tableau de bord, 3 notifications, Créeur, and SEO.

Dans le cas où la mise à jour par clic échouerait, il faudra supprimer le fichier `.maintenance` se trouvant à la racine de notre dossier Wordpress.

Il est possible par la suite d'effectuer une mise à jour manuelle.

MISE À JOUR

Mise à jour manuelle

La documentation de Wordpress propose un suivi d'étape pour la réalisation d'une mise à jour manuelle.

Vous pouvez suivre ce lien pour le trouver :

<https://fr.wordpress.org/support/article/updating-wordpress/#etape-1-replacer-les-fichiers-wordpress>

The screenshot shows the WordPress dashboard with a dark sidebar. The sidebar includes links for Tableau de bord, Accueil, Mises à jour (with a red notification badge), Articles, Médias, Pages, and Commentaires. The main content area is titled 'Mises à jour de WordPress'. It displays a message: 'Vous avez la dernière version de WordPress. Les prochaines mises à jour...' (You have the latest version of WordPress. The next updates...). Below this, it says 'Si vous souhaitez réinstaller la version 4.5.2-fr_FR, vous pouvez le faire ici, ou télécharger l'archive.' (If you want to reinstall the 4.5.2-fr_FR version, you can do it here, or download the archive.). There are three buttons: 'Ré-installer maintenant', 'Télécharger la version 4.5.2-fr_FR', and 'Masquer cette mise à jour'.

MISE À JOUR

The screenshot shows the WordPress dashboard with a dark theme. The top navigation bar includes links for WordPress, Tableau de bord, Accueil, Mises à jour (with a red notification badge), Articles, Médias, Pages, and Commentaires. The main content area is titled "Mises à jour de WordPress". It displays a message: "Vous avez la dernière version de WordPress. Les prochaines mises à jour sont prévues pour le 17 mai 2016 à 22 h 44 min." Below this, there is a note about re-installing version 4.5.2-fr_FR and a link to download it. A small note at the bottom states: "Cette version localisée contient à la fois la traduction et divers correctifs liés à la localisation. Si vous souhaitez consulter les détails de cette mise à jour, veuillez accéder à l'interface d'administration." Buttons for "Ré-installer maintenant", "Télécharger la version 4.5.2-fr_FR", and "Masquer cette mise à jour" are visible.

conclusion

La dernière étape est de vérifier que la mise à jour d'extension ou de version Wordpress n'a pas impacté votre site y compris du "front-office".

Il est important de faire un tour de son site pour lister les éventuels problèmes liés à la mise à jour.

MAINTENANCE

Durant les mises à jour du site, il est possible de passer le site en mode "maintenance".

En effet, pour éviter tout problème lié à la visite d'utilisateur durant les mises à jour, il est important de proposer un mode "maintenance".

Il améliore l'expérience utilisateur mais aussi le référencement avec la possibilité de rediriger les utilisateurs vers une page prévue à cet effet.

MAINTENANCE



Il est possible de mettre en place une page de maintenance à l'aide du fichier .htaccess

```
RewriteEngine on RewriteCond  
%{REQUEST_URI} !/maintenance.html$  
RewriteCond %{REMOTE_ADDR}  
!^xxx\.xxx\.xxx\.xxx RewriteRule $  
/maintenance.html [R=302,L]
```

Il faut remplacer les xxx par votre ip et il est impératif de créer un fichier "maintenance.html"

MAINTENANCE



Wp Maintenance est une extension permettant d'afficher une page de maintenance.

Cette extension est très simple d'utilisation.

<https://fr.wordpress.org/plugins/wp-maintenance/>

TRANSFERT DE WORDPRESS

Lorsque nous avons créé notre thème, créé nos extensions, nos widgets.

Il est temps de passer au transfert de Wordpress du serveur local vers le serveur distant.

Plusieurs étapes sont à prendre en compte : la migration de la base de données puis la migration du dossier contenant le site Wordpress.

Fichier à importer :

Le fichier peut être compressé (gzip, bzip2, zip) ou non.
Le nom du fichier compressé doit se terminer par **[format].[compression]**. Exemple : **.sql.zip**

Parcourir les fichiers : [Parcourir...](#) Aucun fichier sélectionné. (Taille maximale : 40Mio)

Il est également possible de glisser-déposer un fichier sur n'importe quelle page.

Jeu de caractères du fichier : ▾

Importation partielle :

Permettre l'interruption de l'importation si la limite de temps configurée dans PHP est sur le point d'être atteinte. (Ceci pourrait entraîner une perte de données).

Ignorer ce nombre de requêtes (pour SQL), à partir du début : ▾

Autres options :

Activer la vérification des clés étrangères

Format :

▾

Options spécifiques au format :

Mode de compatibilité SQL : ▾

Ne pas utiliser AUTO_INCREMENT pour la valeur zéro

TRANSFERT DE WORDPRESS

Dans un premier temps, nous devons exporter la base de données de PHPMyAdmin du serveur local.

Il suffit ensuite de se connecter au panel d'administration de l'hébergeur choisi (OVH, O2Switch), généralement la plupart des hébergeurs proposent PHPMyAdmin.

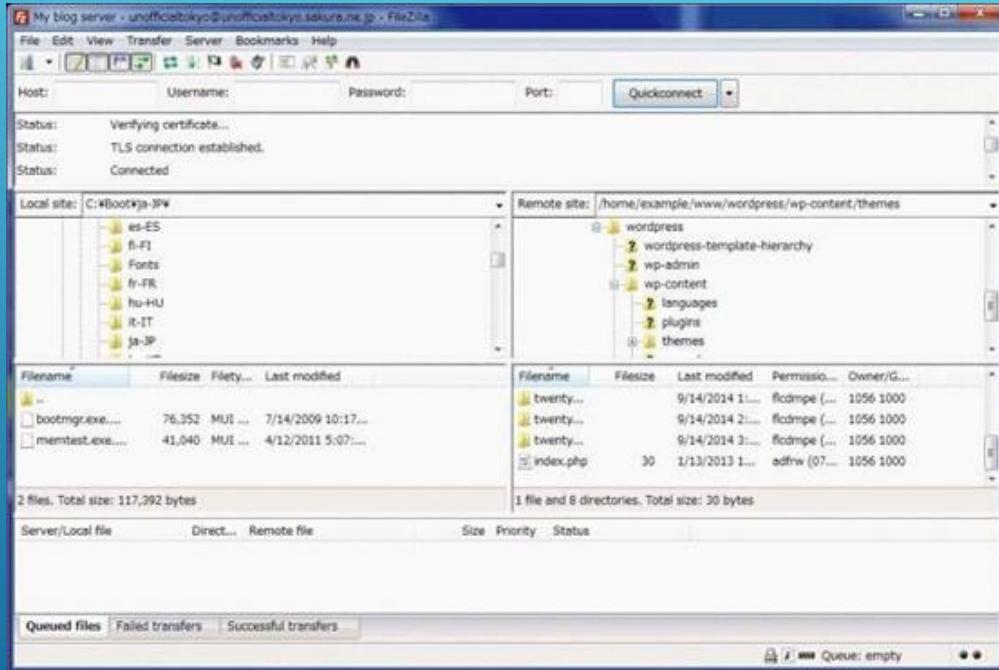
À partir de ce menu, vous pouvez importer votre base de données.

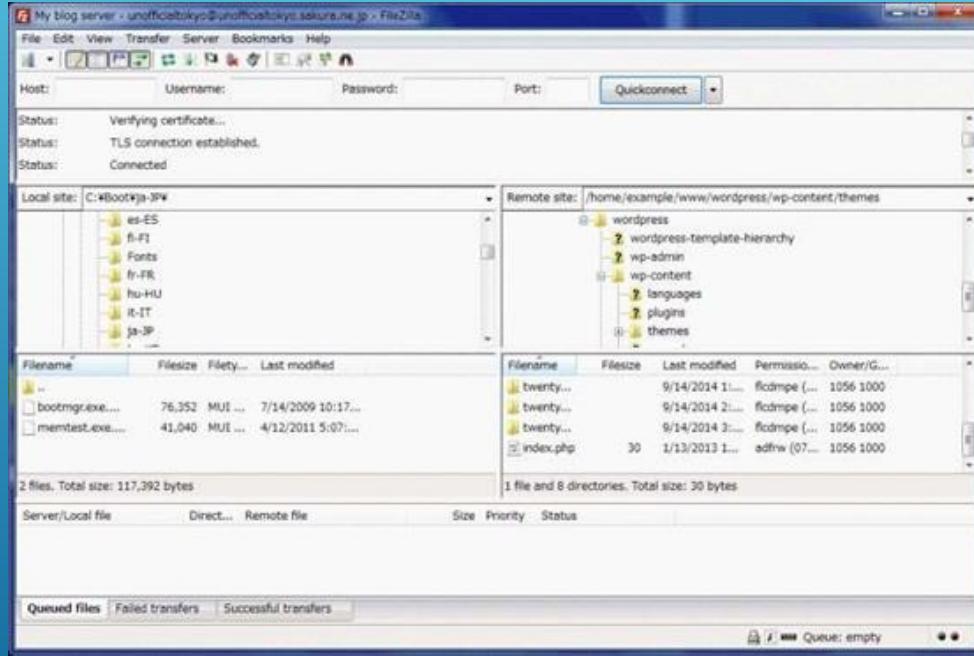
TRANSFERT DE WORDPRESS

Dans un second temps, on transfère via le logiciel FTP, le dossier contenant les fichiers de notre site Wordpress.

Plusieurs éléments sont à prendre en compte :

- Il faut changer les identifiants de connexion à la base de données dans wp-config.php
- Il faut désactiver l'affichage des erreurs dans wp-config.php
- Il faut changer tous les URL qui contiennent localhost y compris dans la base de données.





TRANSFERT DE WORDPRESS

- Il faut modifier le fichier .htaccess en fonction des données de l'hébergeur (nom de domaine, ip...)