



SQUARESPACE



PrestaShop

Panorama des CMS

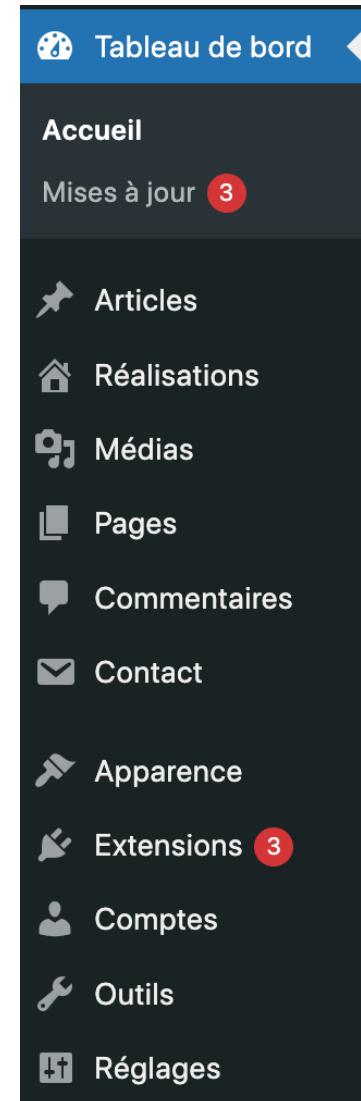
Edouard Sombié
ESGI 2024

Le principe

Content Management System

I/Fonctionnalités du cœur

- Edition de contenu (CRUD)
- Hiérarchisation des données
- Gestion des assets media
- Gestion des utilisateurs



Le principe

2/Interface d'édition

- Editeur WYSIWYG
- Gestionnaire de widgets, composants...

3/Modularité

- Thèmes graphiques
- Modules fonctionnels



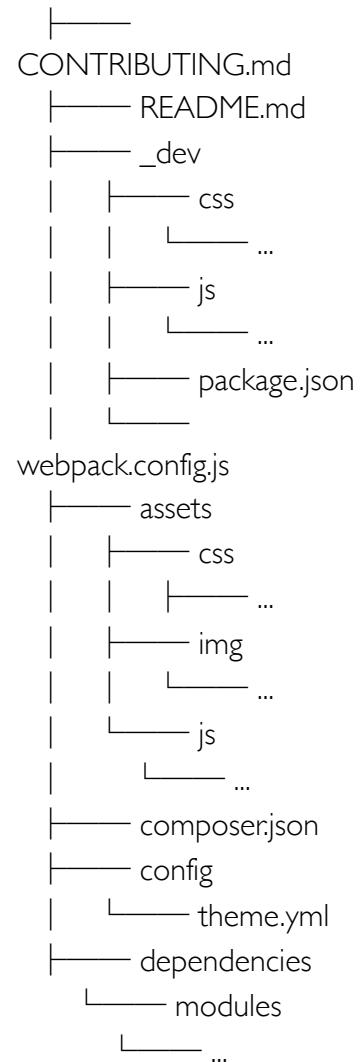
4/Sécurité

- Gestion des versions et des mises à jours
- Suivi des coeurs et modules (patches)

Le principe

Une structure définie

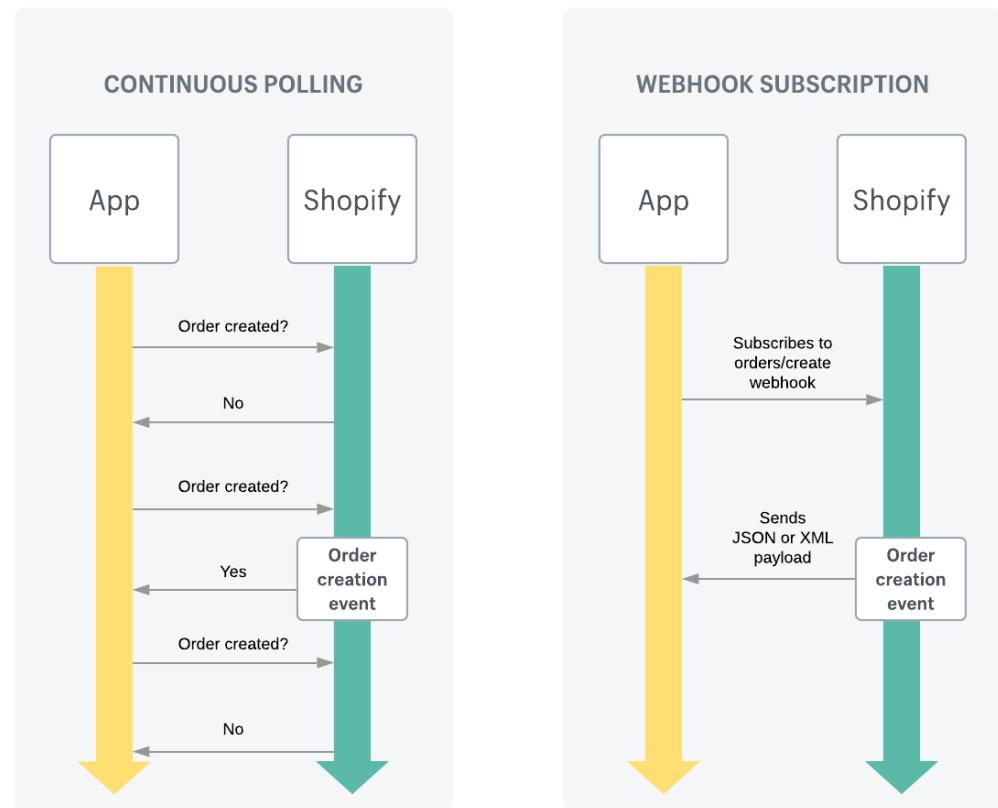
- Structure des dossiers (modules, thèmes)
- Utilisation des méthodes et des classes du CMS (CMF)



Le principe

Des webhooks

Déclenchent l'affichage de composants ou le lancement de fonctions



Les CMS les plus utilisés

<https://w3techs.com/>

Content Management Systems

Most popular content management systems

© W3Techs.com	usage	change since 1 March 2024	market share	change since 1 March 2024
1. WordPress	43.3%	+0.1%	62.8%	
2. Shopify	4.4%	+0.1%	6.3%	+0.1%
3. Wix	2.7%	+0.1%	3.8%	
4. Squarespace	2.1%		3.0%	
5. Joomla	1.7%		2.5%	

percentages of sites

Fastest growing content management systems since 1 March 2024

© W3Techs.com	sites
1. Elementor	82.6
2. WordPress	19.2
3. Shopify	17.6

daily increase of number of sites
per million

Find more details in the [content management surveys](#)

CMS hébergés

Solutions non open-source

Abonnement mensuel

Pas d'accès direct au serveur

Faciles à mettre en œuvre, peu propices à la *customisation*.

Sécurisés (revues)



SQUARESPACE



CMS hébergés



<https://www.shopify.com/>

Orienté e-commerce

<https://shopify.dev/>

Développement de modules (app vendue sur le marketplace)

<https://shopify.dev/app-developers>

Shopify CLI, GraphQL API ou REST Admin API (json)

Composants REACT

Librairies Node, PHP, Python, Ruby

CMS hébergés

Développement de thèmes

<https://shopify.dev/themes/getting-started/create>

Shopify CLI, Dawn (thème de base), Shopify GitHub (enregistrement)

JSON + Liquid

<https://shopify.dev/themes/architecture>

Processus de développement plutôt lourd.

Accès direct au CMS via l'API admin

Système graphique de base : Polaris

https://polaris.shopify.com/?itcat=partner_blog&itterm=how_to_build_a_shopify_app

CMS hébergés



SQUARESPACE

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

website builder intégré

fonction e-commerce native

<https://developers.squarespace.com/>

<https://developers.squarespace.com/template-overview>

JSON Template

LESS CSS

support GIT

language de template JSON-T

<https://developers.squarespace.com/what-is-json-t/>

Pas de développement de module...
Pas d'accès direct au CMS

CMS hébergés



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

<https://dev.wix.com/>

Website apps, Business management apps

Wix APIs

<https://dev.wix.com/api/rest/getting-started/api-query-language>

UI Guidelines

<https://devforum.wix.com/kb/en/article/uiux-guidelines>

Template de base + **Wix Editor**

<https://support.wix.com/en/article/wix-editor-getting-started-with-the-wix-editor>

Développement de module et thème très limité.

CMS hébergés



<https://www.hubspot.fr/products/cms>

CMS en lien avec le CRM Hubspot
(contenu en fonction du client)

HubL Syntax

CMS open source

Installation gratuite.

Développement libre.



CMS open source



CMS ecommerce



- Multilingue natif, installation rapide
- Modèle MVC



- Marketplace onéreux et peu surveillé. Beaucoup de modules sont buggés (y compris ceux de Prestashop...). De nombreuses fonctionnalités de base s'avèrent payantes.
- Documentation quasi inexistante et communauté trop restreinte.
- Structure du CMS anormalement complexe (données et fichiers) / 250 tables.
- Obligation de passer par les outils Prestashop (Starter theme + webpack config)
- Système de cache défaillant, fonctionnement parfois aléatoire.

CMS open source

Thèmes

- Système de layouts (full-width, 1 colonne...) + blocs
- Langage Smarty (ancêtre de Twig)
- Structure de base + manifest et assets (compilés via webpack)

Modules

- Système de hooks pour accrocher les modules
- Modules sous forme de classe controller + templates (.tpl)

Système d'override des classes natives

<https://www.prestashop.com/fr/exemples>

CMS open source



Joomla! CMS Market Share
Over a 10-Year Period



TRUELIST

Source: W3Techs



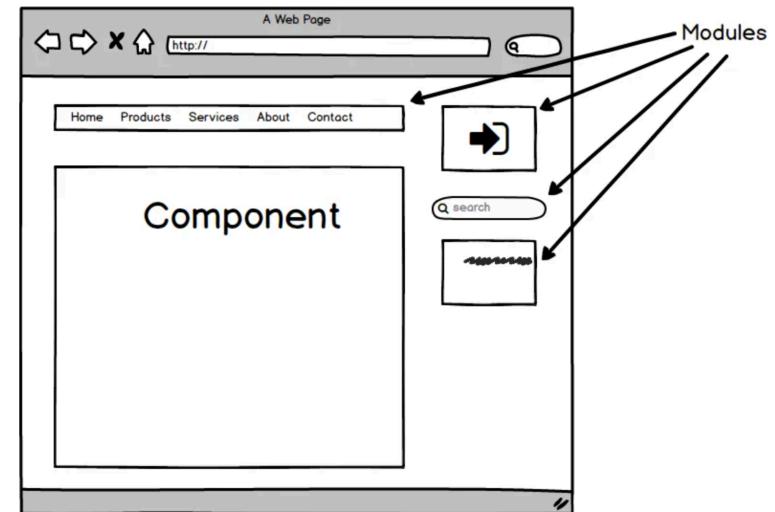
- Multilingue
- MVC
- Installation rapide



- Un peu vieillot...
- Catalogue réduit de thèmes et de modules
- Documentation pas très fournie et communauté réduite

CMS open source

- Structure complexe module / component / plugin...



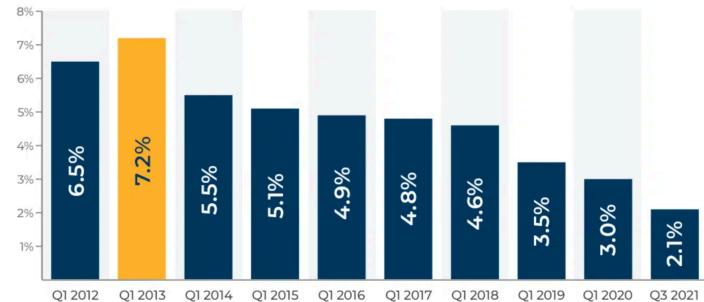
<https://dj-extensions.com/blog/general/20-most-popular-websites-using-joomla>

CMS open source



Drupal CMS Market Share
Over a 10-Year Period

TRUELIST



- Modèle de données personnalisable
- Install via composer, development tools (VSCode, SublimeText...)
- Framework Symfony
- Réputé pour sa robustesse
- Catalogue de thèmes et de modules assez fourni
- Positionnement plutôt haut de gamme, "organisations"



- Connue pour la complexité de son administration
- A priori gourmand en ressources

<https://www.vardot.com/en-us/ideas/blog/top-10-drupal-websites-world-updated>

CMS open source



CMS ecommerce

- 👍
- Version gratuite et payante
 - Modèle MVC
 - Connue pour sa robustesse et sa capacité à supporter la charge



- 👎
- Marketplace onéreux
 - Pas adapté à l'hébergement mutualisé
 - Développement un peu rigide

<https://www.tigren.com/big-brands-using-magento-websites/>

CMS open source



43% des sites dans le monde (36% du top 1M)

65% des CMS...

CMS open source



- Catalogue de thèmes et de modules très riches
- Modules "de base" gratuits et performants
- Communauté très active et large
- Excellente documentation
- Structure légère (fichiers et données)
- Module ecommerce performant (28% de pdm)
- Support incontournable pour les acteurs tiers
- Développement plutôt léger
- Mise à jour auto du cœur et des modules

<https://www.wpbeginner.com/showcase/40-most-notable-big-name-brands-that-are-using-wordpress/>

CMS open source



- Pas de modèle MVC
- Pas de type de contenu *customisable* nativement

Wordpress - modèle

Très peu de types de données (12 tables)

- wp_posts (articles, pages, attachments (images...), items de menu)
- wp_options (réglages généraux de WP)
- wp_users (utilisateurs du CMS)
- wp_comments (commentaires postés sur les articles et pages)
- wp_terms (catégories d'articles, étiquettes)

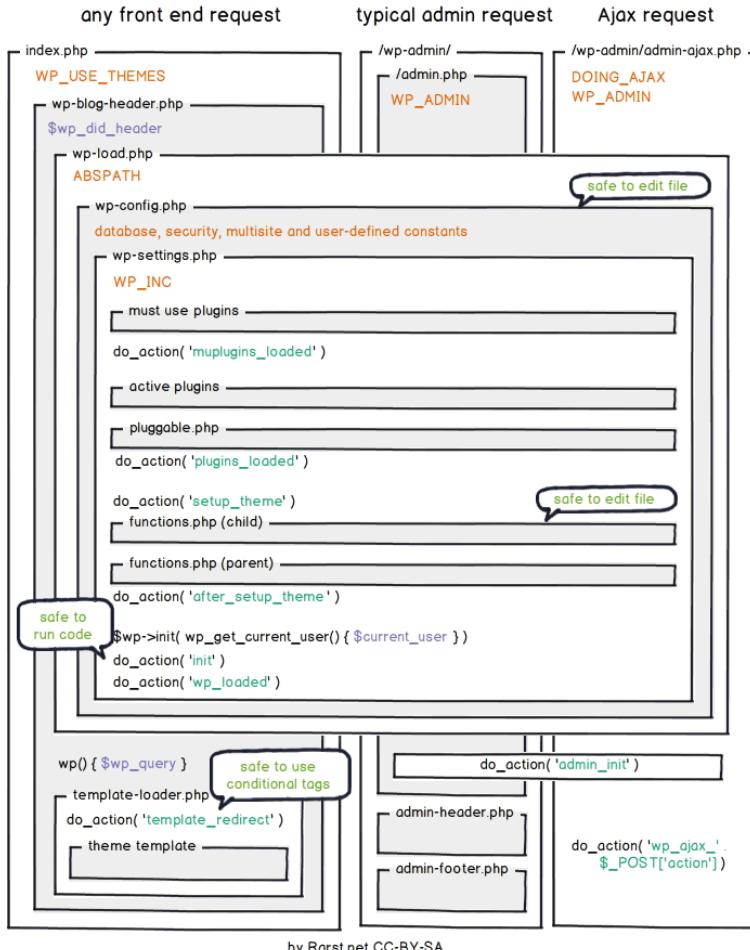
Wordpress - modèle

... enrichies par des **métadonnées**

- wp_postmeta
- wp_commentmeta
- wp_termmeta
- wp_usermeta

Wordpress - séquence de chargement

Make sense of WP core load



<https://wp-kama.com/handbook/wordpress/loading#wpcore-load>

WORDPRESS LOADING SEQUENCE



Wordpress - *La Loop*

La Loop WordPress

<https://developer.wordpress.org/themes/basics/the-loop/>

Mécanisme de base par lequel WP récupère et envoie le contenu (*Post Objects*) vers le template.

```
<?php
if ( have_posts() ) :
    while ( have_posts() ) : the_post();
        // Display post content
    endwhile;
endif;
?>
```

Wordpress - Les requêtes

WP_Query

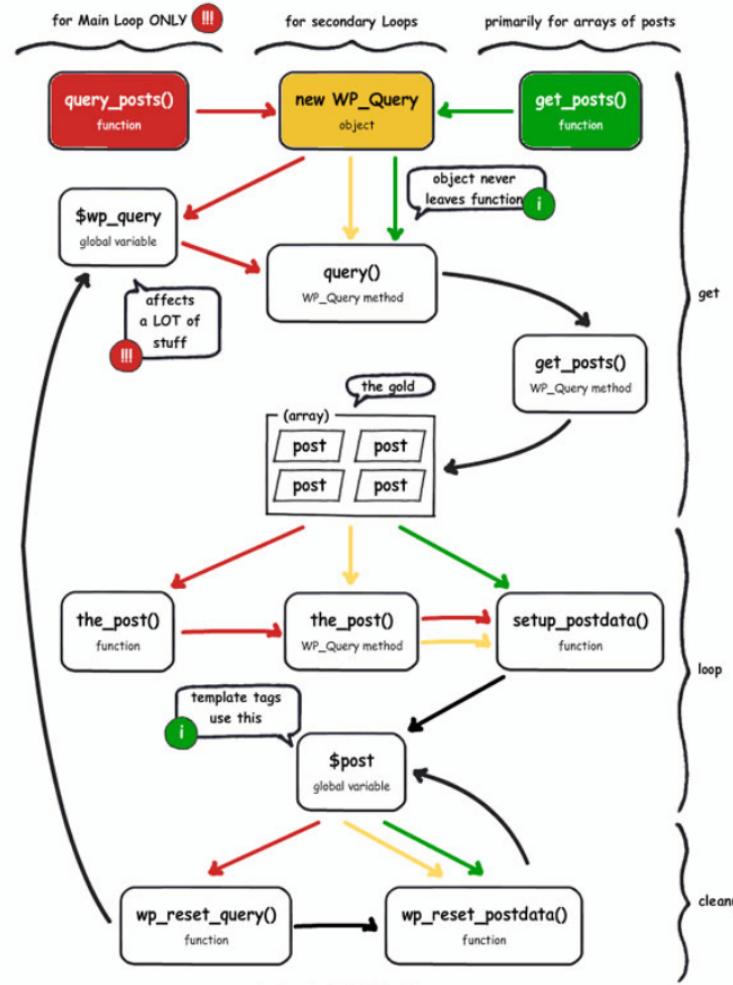
https://developer.wordpress.org/reference/classes/wp_query/

Objet utilisé pour récupérer les publications (*Post Objects*) dans la base de données. C'est sur sa méthode **have_posts()** que repose la *Loop* WordPress.

```
<?php
// the query
$the_query = new WP_Query( $args ); ?>
<?php if ( $the_query->have_posts() ) : ?>
    <!-- the loop -->
    <?php while ( $the_query->have_posts() ) : $the_query->the_post(); ?>
        <h2><?php the_title(); ?></h2>
    <?php endwhile; ?>
    <!-- end of the loop -->
<?php else : ?>
    <p><?php _e( 'Sorry, no posts matched your criteria.' ); ?></p>
<?php endif; ?>
```

Wordpress - Les requêtes

Make Sense of WP Query Functions



Wordpress - L'objet Post

WP_Post

https://developer.wordpress.org/reference/classes/wp_post/

L'entité de base du modèle wordpress.

```
$post = get_post();  
echo $post->post_title
```

On le récupère via une WP_Query, qui peut-être utilisée via la fonction **get_posts(\$args)**

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

Wordpress - Développement de thème

<https://developer.wordpress.org/themes/>

Licence GPL

<https://developer.wordpress.org/themes/getting-started/wordpress-licensing-the-gpl/>

Standards de développement WP :

<https://developer.wordpress.org/coding-standards/wordpress-coding-standards/>

Wordpress - développement de thème vs builder

Développement de thème vs *page Builder*

De nombreux thèmes commerciaux WP sont accompagnés de fonctionnalités de type *page builder* (siteOrigin, Elementor...).

Nous déconseillons l'usage de ce type de modules car ils créent souvent des structures de pages inutilement complexes et amènent les administrateurs à devoir éditer des blocs imbriqués les uns dans les autres.

A cela s'ajoute l'arrivée d'une nouvelle génération de thèmes autorisant nativement l'édition des blocs de page à partir de l'interface d'administration, les *block themes*

Wordpress - Deux types de thèmes

Depuis la version 5.9 de WordPress, ont été introduits les ***blocks themes***, caractéristiques d'une évolution du CMS vers une interface js (basée sur ajax).

Classic theme

- Ensemble de fichiers php (scripts, templates)
- Assets css
- Assets js
- Fichiers de traductions du thèmes (.mo, .po)
- Textes d'information (licence, readme...)

Block theme

- Ensemble de fichiers html (templates)
- Fichiers json
- Assets css
- Assets js
- Fichiers de traductions du thèmes (.mo, .po)
- Textes d'information (licence, readme...)

Wordpress - Block theme

Structure de base

<https://developer.wordpress.org/block-editor/how-to-guides/themes/block-theme-overview/>

```
theme
|__ style.css
|__ theme.json
|__ functions.php
|__ templates
    |__ index.html
    |__ single.html
    |__ archive.html
    |__ ...
|__ parts
    |__ header.html
    |__ footer.html
    |__ sidebar.html
    |__ ...
|__ styles
    |__ red.json
    |__ blue.json
    |__ ...
```

Ce type de thème est certainement l'avenir du CMS et augure de l'évolution de WP, mais force est de constater que l'interface d'édition en back office n'est pas encore totalement ergonomique.

Aussi, **nous privilégierons (pour l'instant) le développement de theme classique** au détriment des block themes.

Wordpress - Thème classique

Classic theme

https://codex.wordpress.org/Theme_Development

- Repose sur une **structure pré-établie** de templates php (page.php, single.php, archive.php...)
- Mais **seulement quatre fichiers obligatoires** : style.css, index.php, header.php et footer.php
- Le fichier style.css doit contenir un entête du type :

```
/*
Theme Name:Thème ESGI
Theme URI: https://esgi.fr
Author: the ESGI team
...
...
```

- Les fonctionnalités du thème (inclusion d'assets, définition des menus, fonctions diverses utilisées dans les templates) sont réunies dans un fichier **functions.php**
- Repose sur un principe de **hiérarchie des templates** : en fonction du contenu demandé, le cms détermine le template à utiliser. Si celui-ci n'est pas présent, le cms en utilise un autre.
Ex : pour afficher un article seul, WP utilise le template single.php, à défaut, il utilisera singular.php et, encore à défaut, il utilisera index.php

Wordpress - Fichiers de bases d'un thème classique

Fichiers de base

style.css

The main stylesheet. This **must** be included with your Theme, and it must contain the information header for your Theme.

rtl.css

The rtl stylesheet. This will be included **automatically** if the website's text direction is right-to-left. This can be generated using the [RTLer](#) plugin.

index.php

The main template. If your Theme provides its own templates, *index.php* must be present.

comments.php

The comments template.

front-page.php

The front page template.

home.php

The home page template, which is the front page by default. If you use a [static front page](#) this is the template for the page with the latest posts.

single.php

The single post template. Used when a single post is queried. For this and all other query templates, *index.php* is used if the query template is not present.

single-{post-type}.php

The single post template used when a single post from a custom post type is queried. For example, *single-book.php* would be used for displaying single posts from the custom post type named "book". *index.php* is used if the query template for the custom post type is not present.

page.php

The page template. Used when an individual [Page](#) is queried.

category.php

The [category template](#). Used when a category is queried.

tag.php

The [tag template](#). Used when a tag is queried.

taxonomy.php

The term template. Used when a term in a custom taxonomy is queried.

author.php

The [author template](#). Used when an author is queried.

date.php

The date/time template. Used when a date or time is queried. Year, month, day, hour, minute, second.

archive.php

The archive template. Used when a category, author, or date is queried. Note that this template will be overridden by *category.php*, *author.php*, and *date.php* for their respective query types.

search.php

The search results template. Used when a search is performed.

attachment.php

Attachment template. Used when viewing a single attachment.

image.php

Image attachment template. Used when viewing a single image attachment. If not present, *attachment.php* will be used.

404.php

The [404 Not Found](#) template. Used when WordPress cannot find a post or page that matches the query.

header.php

footer.php

sidebar.php

Wordpress - Fichiers de bases d'un thème classique



For oEmbeds: embed-{post-type}-{post_format}.php → embed-{post-type}.php → embed.php → wp-includes/theme-compat/embed.php

■ Primary Template
 ■ Secondary Template
 ■ Variable Template
 ■ Page Type

<https://wphierarchy.com/>

Wordpress - Fonctions de template

https://codex.wordpress.org/Template_Tags

Fonctions d'inclusion de template

- get_header()
- get_sidebar()
- get_footer()
- get_search_form().
- get_template_part()

Autres fonctions importantes

- blog_info()
- body_class()
- wp_enqueue_style()
- wp_enqueue_script()
- wp_head()
- wp_footer();
- wp_nav_menu()
- is_front_page()
- is_single()...

Wordpress - Modèle de page personnalisé

Création d'un modèle de page personnalisé.

Par défaut, WordPress affiche les pages avec le template **page.php**

Toutefois, il est possible de créer un modèle de page personnalisé.

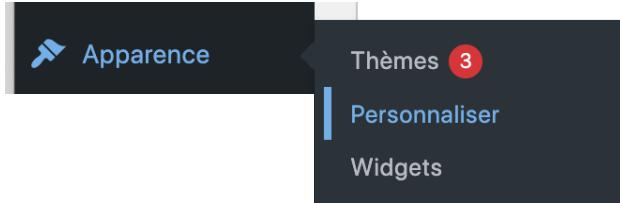
Pour cela, il suffit d'ajouter l'entête suivant au-début du template de page :

```
<?php  
/*  
Template Name: Mon modèle custom  
*/  
?>
```

Une fois ce template ajouté au thème, il apparaît dans le back office dans la liste du choix de modèle de page :

The screenshot shows the WordPress admin interface for managing a page. At the top, there are tabs for 'Page' (which is selected), 'Bloc', and a close button 'X'. Below the tabs, there's a summary section labeled 'Récapitulatif' with a collapse arrow. Underneath, there are fields for 'Visibilité' (set to 'Publique'), 'Publier' (set to 'Aujourd'hui à 12 h 17'), and 'URL' (set to 'wp-classe-janvier.test/nlle-page/'). A red box highlights the 'Modèle' dropdown menu, which is currently set to 'Modèle par défaut'. Another red box highlights the 'Mettre à la corbeille' (Move to trash) button located at the bottom right of the page settings area.

Wordpress - Paramètres de thème



<https://developer.wordpress.org/themes/customize-api/customizer-objects/>

WP permet nativement de créer des paramètres de personnalisation du thème à partir du back office.

Cette personnalisation repose sur la classe **WP_Customize_Manager**

Lorsqu'on définit un réglage (**setting**), on peut déterminer s'il sera enregistré comme *option* ou comme *theme modification*.

En principe, tous les réglages seront de type **theme_mod**.

Pour que le setting soit modifiable, on lui adjoint un **control** (de type input, color picker ou autre).

Wordpress - Paramètres de thème

Le *customizer* de thème offre une interface qui affiche en temps réel les réglages (via ajax).

The screenshot shows the WordPress theme customizer interface on the left and a preview of the ESGI theme on the right.

Customizer Left Panel:

- Top bar: "Publié" (Published) button.
- Left sidebar:
 - "Personnalisation" (Personalization) button.
 - "Personnalisation du thème" (Theme Personalization) link.
 - Section: "Paramètres du thème".
 - Couleur principale**: A color swatch set to green, with a "Sélectionner une couleur" (Select a color) button.
 - Thème sombre: *Activer la version sombre du thème.*
 - Afficher la barre latérale: *Afficher la barre latérale sur les pages d'article.*

Theme Preview Right Panel:

- Header navigation: "Accueil" (Home) is highlighted in green, while "A propos" (About) and "Blog" are in grey.
- Content area:
 - Image: A circular profile picture of Totoro from My Neighbor Totoro.
 - Title: "ESGI"
 - Text: "Another WordPress Theme"
 - Social links: Twitter, Facebook, Google+, LinkedIn icons.
- Footer: "WP-ESGI theme by YoThemes"

Wordpress - Les hooks

https://codex.wordpress.org/Plugin_API

<https://developer.wordpress.org/plugins/hooks/>

<https://developer.wordpress.org/reference/hooks/>

Les hooks permettent d'ajouter du code à certains moments de l'exécution du CMS.

Dans WordPress, il en existe deux types : les **actions** et les **filters**.

- une **action** interrompt le flux de code pour faire quelque chose, puis revient au flux normal sans rien modifier ;
- un **filtre** est utilisé pour modifier quelque chose d'une manière spécifique afin que la modification soit ensuite utilisée par le code.

Pour utiliser l'un ou l'autre, il faut écrire une fonction personnalisée appelée **Callback**, puis l'enregistrer avec un hook WordPress pour une action ou un filtre spécifique.

Ex : `add_action('nom_de_l'action', 'callbackFunction')`

Wordpress - Les actions

<https://developer.wordpress.org/plugins/hooks/actions/>

Permettent de déclencher une fonction lorsqu'un "événement" du CMS survient (initialisation, chargement des assets, affichage du header...);

```
function esgi_callback() {  
    // do something  
}  
add_action( 'init', 'esgi_callback' );  
  
do_action(), remove_action(), remove_all_actions()...
```

Liste complète des actions

https://codex.wordpress.org/Plugin_API/Action_Reference

Wordpress - Les filtres

<https://developer.wordpress.org/plugins/hooks/filters/>

Dans les fonctions du cœur de WordPress, les valeurs passent par des filtres qui les modifient. En ajoutant nos propres fonctions à ces filtres, il est possible de **surcharger le fonctionnement natif** du CMS.

Tout comme pour les actions, il est possible de définir des filtres personnalisés.

```
function esgi_filter_title( $title ) {  
    return 'The ' . $title . ' was filtered';  
}  
add_filter( 'the_title', 'esgi_filter_title' );  
  
apply_filters(), remove_filter()...
```

Liste complète des filtres

https://codex.wordpress.org/Plugin_API/Filter_Reference

Wordpress - Ajax

<https://developer.wordpress.org/plugins/javascript/ajax/>

https://codex.wordpress.org/AJAX_in_Plugins

Pour que WordPress réponde à un call ajax, celui-ci doit être authentifié, c'est à dire qu'il doit exister un callback pour l'action `wp_ajax_{$action}` où \$action correspond au paramètre 'action' envoyé par le call ajax.

```
do_action( "wp_ajax_{$action}" )
```

```
do_action( "wp_ajax_nopriv_{$action}" )
```

Toutes les requêtes ajax sont envoyées à **wp-admin/admin-ajax.php**.

Afin d'envoyer l'url de cette page de PHP vers le script js, on utilise la fonction **wp_localize_script()**

Wordpress : le développement de modules

<https://developer.wordpress.org/plugins/intro/>

Comme tous les CMS, WordPress offre un système d'enrichissement de ses fonctionnalités via l'ajout de modules fonctionnels (*plugins*).

"Don't hack core"

Il ne faut absolument jamais modifier les fichiers du cœur du CMS.

Ces modifications seraient perdues lors d'une mise à jour et pourraient interférer avec d'autres éléments du code.

Pour autoriser l'ajout de nouvelles fonctionnalités sans danger, WordPress propose un pattern du type *mediator pattern*, que l'on retrouve dans le Plugin API.

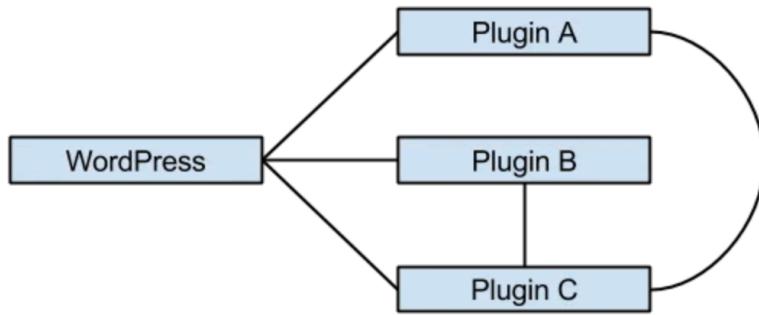
Il est primordial d'accorder beaucoup d'importance à la **sécurité des plugins**, sachant qu'ils peuvent constituer la principale faille de WordPress.

<https://developer.wordpress.org/plugins/security/>

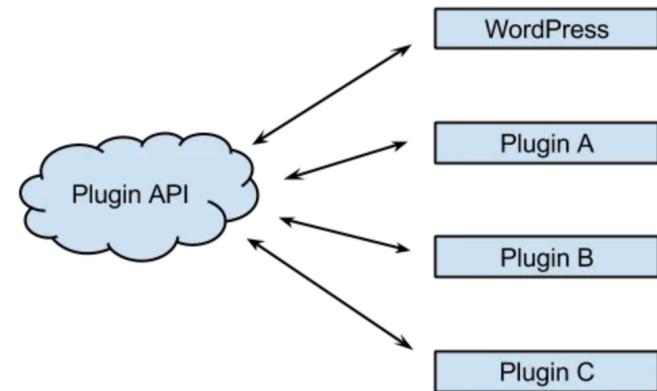
Wordpress : le Plugin API

https://codex.wordpress.org/Plugin_API

L'interaction entre WordPress et les modules externes et les thèmes repose sur le **Plugin API**



sans Plugin API



avec Plugin API

La clé de voûte de cette API sont les **filters** et les **actions**.

Wordpress - Hooks de modules

Les hooks de base du développement de module

Un certain nombre de méthodes peuvent être lancées lors de l'activation, la désactivation ou la suppression d'un module. Pour ce faire, il faut déclarer les hooks correspondants pour le module.

[register_activation_hook\(\)](#)

[register_deactivation_hook\(\)](#)

[register_uninstall_hook\(\)](#)

Wordpress - l'API REST

WordPress propose nativement une **API REST**

<https://developer.wordpress.org/rest-api/>

L'API REST WordPress fournit des *endpoints REST* (URL) représentant les publications, les pages, les taxonomies et d'autres types de données. Une application (ou plugin) peut envoyer et recevoir des données JSON à ces *endpoints* pour interroger, modifier et créer du contenu sur votre site.

C'est sur cette API que repose le système de blocs Gutenberg de WordPress. Ainsi, cet outil permettrait de développer, par exemple, un *front* ou un *back office* en "*full js*".

Note : **L'API REST est à favoriser par rapport aux appels ajax** lancés à admin-ajax.php car elle est considérée comme une méthode de récupération des données plus structurée, extensible et simple.

Wordpress - l'API REST

Liste de tous les endpoints actifs :

[https://monsite.com/**wp-json**](https://monsite.com/wp-json)

Accès au endpoint :

[https://monsite.com/wp-json/wp/v2/{**endPoint**}](https://monsite.com/wp-json/wp/v2/{endPoint})

Wordpress - toutes les APIs

https://codex.wordpress.org/WordPress_APIs

- [Dashboard Widgets API](#)
- [Database API](#)
- [HTTP API](#)
- [REST API](#)
- [File Header API](#)
- [Filesystem API](#)
- [Metadata API](#)
- [Options API](#)
- [Plugin API](#)
- [Quicktags API](#)
- [Rewrite API](#)
- [Settings API](#)
- [Shortcode API](#)
- [Theme Modification API](#)
- [Theme Customization API](#)
- [Transients API](#)
- [Widgets API](#)
- [XML-RPC WordPress API](#)

Wordpress - Pour continuer...

En plus de l'excellent codex de WordPress, de très nombreux sites sont consacrés au développement pour ce CMS.

OOP

<https://carlalexander.ca/approaching-object-oriented-programming-wordpress/>

<https://wpmudev.com/blog/advanced-wordpress-development-writing-object-oriented-plugins/>

Tutoriels / blog

<https://wpmarmite.com/>

<https://www.wpbeginner.com/>

<https://wpmudev.com/blog/>

<https://www.dev-wp.fr/>

Wordpress - Quelques modules de base

De très nombreuses fonctionnalités sont proposées par les modules gratuits et payants WordPress (SEO, mise en cache, flux RSS, e-commerce...)

Par défaut, WordPress est installé sans modules, ce sera ensuite à nous de définir quelles fonctionnalités nous développerons et quels modules existants nous utiliserons.

Note : Il faut absolument **éviter l'accumulation de modules** car elle mène vers un ralentissement du site et de possibles conflits.

Wordpress - Quelques modules de base

Mais pour certaines fonctionnalités, il est conseillé de se reposer sur des solutions existantes. Nous pouvons ainsi citer les modules suivants (gratuits) :

- WP Super cache (mise en cache, développé par Automattic)
- Yoast SEO (Référencement naturel)
- Updraft + (Sauvegarde des données et fichiers)
- Advanced Custom Fields (Gestion de champs personnalisés)
- Contact Form 7 (Formulaire de contact avec gestion des actions)
- MetaSlider (Carousel js)
- WPML (Traduction du contenu)
- WooCommerce (Structure e-commerce, développé par Automattic)
- SVG support...