WordPress
Architecture et développement modulaire

N° de la lecture individuelle :	6
Étudiant	DAVID Guillaume, 802_1F
Sujet	Travail lecture et de recherche

Choix du sujet

J'ai choisi de réaliser une lecture individuelle sur le sujet de l'architecture et du développement de module sur la plateforme CMS WordPress. J'ai choisi de traiter ce sujet pour diverses raisons. Tout d'abord, dans le cadre du projet « CASs », j'ai réalisé une première LI sur le fonctionnement et la sélection d'un CMS. Cette deuxième partie va traiter avec plus de profondeur la partie développement de modules. De plus, c'est un sujet auquel on se retrouve très souvent confronté dans notre vie d'informaticien. Enfin, le développement de module Wordpress était un thème, des compétences que je souhaitai développer dans ma formation HES ; ce qui me permet d'étudier ce thème.

Table des matières

Résumé du travail de recherche

Introduction

Qu'est-ce qu'un CMS (rappel) ?

Un CMS (Content Management System) est un système de gestion de contenu qui permet de créer, gérer et publier **facilement** du contenu sur un site web sans avoir besoin de connaissances techniques avancées en programmation. Il fournit une interface conviviale pour la gestion du contenu, permettant aux utilisateurs de créer et de modifier des pages web, de télécharger des fichiers multimédias tels que des images et des vidéos, de gérer des utilisateurs et des groupes, de créer des menus de navigation et d'autres fonctionnalités de base pour la gestion d'un site web.

Qu'est-ce que WordPress

WordPress est un CMS open-source très populaire qui permet de créer et de gérer facilement des sites web sans avoir besoin de connaissances en programmation.

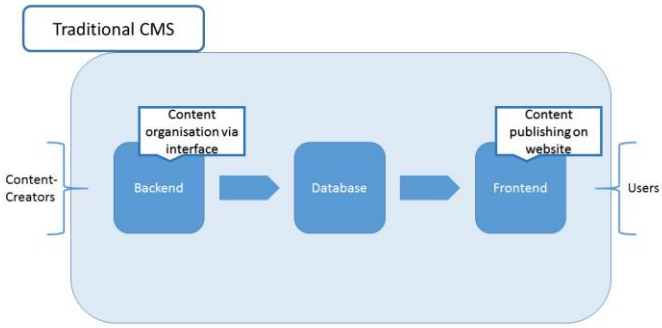
WordPress est extrêmement flexible et personnalisable, avec des milliers de thèmes et de plugins disponibles pour ajouter des fonctionnalités et personnaliser l'apparence de votre site. Il est utilisé pour tout, des blogs personnels aux sites web d'entreprise, en passant par les boutiques en ligne et les sites web de médias.

WordPress est également facile à utiliser, avec une interface conviviale et intuitive qui permet aux utilisateurs de créer et de publier du contenu rapidement et facilement. Il est soutenu par une vaste communauté de développeurs et d'utilisateurs qui contribuent régulièrement à son développement et à son amélioration.

Figure 1 - Logo de WordPress

Architecture générale (rappel)

Dans l'architecture de WordPress (logique des CMS), le backend gère les requêtes des utilisateurs et la base de données stocke les données, tandis que le frontend s'occupe de l'interface utilisateur et de l'affichage du contenu. On confond parfois « Backend » et « Frontend »



Backend	Partie « coté » serveur où est gèré les fonctionnalités qui ne sont pas directement liées à l'interface utilisateur, comme les traitements, la gestion des bases de données, la sécurité, les autorisations d'accès, etc.	C'est aussi le nom communément donné à la partie d'administration du site.
Frontend	Partie « coté » client où est affiché le contenu du site.	

Les plugins

Qu'est-ce qu'un plugin

Un plugin est une extension qui permet d'ajouter des fonctionnalités supplémentaires à un site WordPress. Les plugins sont des fichiers de code que l'on peut installer sur WordPress pour modifier le fonctionnement de base du site et ajouter de nouvelles fonctionnalités.

Ils peuvent être utilisés pour une variété de tâches, comme la création de formulaires de contact, la mise en place d'une boutique en ligne, l'optimisation de la sécurité du site, l'ajout de fonctionnalités de réseaux sociaux, …

*WordPress propose également d'autres types d'extensions, comme **les thèmes** qui sont des ensembles de fichiers qui déterminent l'apparence visuelle d'un site WordPress. Les thèmes permettent de personnaliser l'apparence d'un site sans avoir à modifier le code du site.*

*Il existe également des **widgets**, qui sont des petites « applications » que l'on peut ajouter à la barre latérale ou au pied de page d'un site. Les widgets peuvent être utilisés pour afficher des informations telles que les derniers articles, les commentaires récents, les liens de réseaux sociaux, ….*

Développer son propre plugin

Il peut arriver que l'on ne soit pas satisfait ou que l'on est d'autres besoins dont les plugins proposés n'y répondent pas. Dans ce cas, il peut être intéressant de développer son propre plugin WordPress.

Les situations suivantes amènent à un questionnement de développement personnalisé :

1. S'il n'existe pas de plugin sur le marché qui répond à vos besoins spécifiques : Si vous avez besoin d'une fonctionnalité très spécifique qui n'est pas disponible dans les plugins existants, vous pouvez envisager de créer votre propre plugin pour répondre à ce besoin.

2. Si l'on souhaite personnaliser une fonctionnalité existante : Si l'on utilise un plugin existant, mais que l'on souhaite apporter des modifications spécifiques à la fonctionnalité de ce plugin. On peut envisager de créer notre propre version personnalisée de ce plugin.
3. Si l'on souhaite améliorer les performances de notre site : Si l'on utilise plusieurs plugins pour ajouter des fonctionnalités à notre site WordPress, cela peut entraîner une charge supplémentaire sur les performances du site. En développant notre propre plugin, on peut créer une solution plus légère et plus performante qui répond spécifiquement à nos besoins. En effet, l'empilage de plugin peut devenir problématique (versions, …)

Il est important de se rappeler que le développement d'un plugin peut prendre du temps et nécessite des compétences en programmation. Il est donc recommandé d'évaluer soigneusement si la création de d'un nouveau plugin est la meilleure option pour répondre à vos besoins spécifiques.

Avantages et inconvénients d'un développement spécifique

	Inconvénients
Le plugin est personnel et répond exactement à nos besoins.	Le développement prend du temps et nécessite des compétences.
Le plugin est plus léger car spécifique et donc plus performant.	L'engagement d'un développeur ou des ressources de développements coûte.
On dispose d'un contrôle total sur le plugin notamment du code.	Il faut gérer soi-même la maintenance du plugin.
Contribution à la communauté lors du développement.	

Architecture de WordPress

Technologies utilisées

Langage de programmation	PHP (backend) , HTML, CSS, JavaScript
SGBD	MySQL ou MariaDB
Framework	jQuery, Underscore.js

Modèle de conception

Le modèle de conception utilisé par WordPress est généralement considéré comme un modèle de type Modèle-Vue-Contrôleur (MVC), bien qu'il inclue également des éléments du présentateur (Presenter).

Dans l'architecture **MVC**, le **modèle** représente les données et les règles de l'application, la **vue** affiche les données au format souhaité et le **contrôleur** coordonne les interactions entre le modèle et la vue.

Dans l'architecture de WordPress, le modèle correspond à la base de données et aux fonctions qui gèrent les données, la vue correspond à la présentation de l'interface utilisateur, et le contrôleur correspond aux fonctions qui traitent les entrées utilisateur et coordonnent les interactions entre le modèle et la vue.

Cependant, WordPress utilise également des éléments de **présentateur** (Presenter), qui permettent de séparer la logique métier de la présentation de l'interface utilisateur. Les présentateurs sont des classes PHP qui fournissent des données formatées aux vues, ce qui permet de réduire la complexité du code et de faciliter la maintenance.

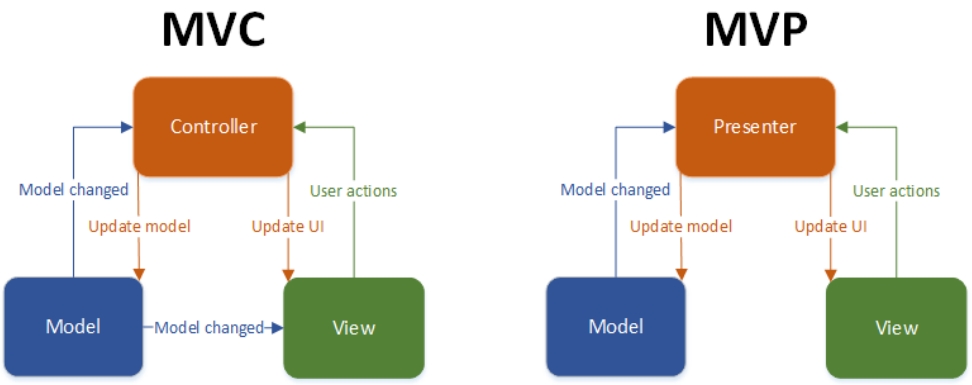


Figure 2 - Modèles MVC / MVP

Mais attention ! Il n'adhère pas strictement à ces modèles, Au lieu de cela, il adopte une approche plus souple où les différents composants (modèle, vue, contrôleur) sont souvent mélangés et interagissent de différentes manières.

Son développement est basé sur des fichiers de modèle (templates) pour la vue, des fichiers de thème et des extensions pour la personnalisation, et une base de données pour le stockage des données. Il utilise également des hooks (actions et filtres) pour permettre l'extension et la modification du comportement du système.

WordPress a été conçu pour être accessible et facile à utiliser pour les utilisateurs finaux, ce qui a influencé son architecture pour offrir une expérience simplifiée. Cela peut se traduire par une certaine souplesse dans la façon dont les composants sont organisés et interagissent.

Développement orienté objet

PHP prend en charge la programmation orientée objet (POO) depuis la version 5.0, publiée en 2004.

WordPress, qui est écrit en PHP, utilise la POO pour sa structure interne et fournit une API orientée objet pour les développeurs d'extensions et de thèmes. On peut étendre les fonctionnalités de WordPress en créant des classes personnalisées, en héritant des classes existantes et en utilisant des concepts de la POO. Bien que WordPress utilise des concepts de la programmation orientée objet, il inclut également des éléments de programmation procédurale, en particulier dans les parties du code qui ont été développées au fil des années et qui conservent une certaine rétrocompatibilité. Cela est dû à l'évolution progressive de WordPress depuis ses débuts comme plateforme de blogging jusqu'à son statut actuel de système de gestion de contenu (CMS) polyvalent.

WordPress combine une approche POO et procédural.

Wordpress core

Le cœur « core » de WordPress est le code source principal de WordPress. Il est maintenu et développé par la communauté et l'équipe de développement de WordPress. Il gère les fonctionnalités de base du CMS, telles que la gestion des utilisateurs, la gestion des contenus, les thèmes, les plugins et les fonctionnalités principales du site.

Lorsque que l'on créé un plugin, il est important de noter que le cœur de WordPress ne sera pas modifié. Il ne faut pas apporter des modifications directes au code source principal de WordPress, car cela pourrait entraîner des problèmes de compatibilité et rendre les mises à jour de WordPress difficiles voire impossible.

Pour cela, on a la possibilité de créer notre propre plugin en utilisant les API et les points d'extension fournis par WordPress. En effet, WordPress fournit de nombreux hooks (crochets) et filtres que l'on peut utiliser pour étendre les fonctionnalités de base de WordPress et ajouter nos propres fonctionnalités.

Lorsque que l'on activera le plugin, WordPress chargera le code et exécutera les fonctions aux endroits spécifiés, en fonction des « hooks » que nous aurions utilisés. Cela permet d'ajouter des fonctionnalités personnalisées à WordPress sans modifier son cœur.

La modularité de WordPress est l'un de ses points forts, car elle permet aux développeurs de créer des plugins indépendants qui peuvent être activés ou désactivés sans affecter le fonctionnement du cœur de WordPress lui-même.

Processus de chargement de WordPress

Le chargement du site WordPress depuis l'appel de l'URL jusqu'au chargement de la page complète passe par ce processus. Le processus est tout de même simplifié car bien plus complexe.

1. **wp-config.php** : Ce fichier contient les informations de configuration de WordPress, telles que les paramètres de la base de données, les clés d'authentification, etc. Il est inclus en premier lors du chargement de WordPress.

2. **wp-settings.php** : Ce fichier est inclus après wp-config.php et charge les fichiers principaux de WordPress, initialise les constantes, les paramètres de configuration et les fonctions de base. Il prépare l'environnement pour le chargement du cœur de WordPress.

3. **Fichiers du cœur de WordPress** : Une fois que wp-settings.php est chargé, les fichiers du cœur de WordPress sont inclus. Cela comprend les fichiers de base tels que wp-load.php, wp-includes/template-loader.php, ...

4. **Fonction wp()** : La fonction wp() est appelée pour initialiser le processus de chargement de WordPress. Elle effectue des tâches telles que la vérification de la requête en cours, la configuration des variables globales et l'initialisation des objets WordPress.

5. **Actions et hooks** : À ce stade, WordPress exécute une série d'actions et de hooks prédéfinis, permettant aux développeurs d'intercepter et de modifier le flux de chargement. Par exemple, l'action init est déclenchée, ce qui permet aux plugins et aux thèmes d'effectuer des initialisations supplémentaires. **Le plugin rajoute des actions et des hooks au fonctionnement du cœur.**

6. **Routage et résolution des URL** : WordPress détermine quelle page ou quel contenu est demandé en analysant l'URL de la requête. Il utilise les règles de permaliens et les routes définies pour résoudre l'URL et l'amener vers l'action demandée.

7. **Chargement du contenu** : WordPress charge le contenu spécifique demandé, qu'il s'agisse d'un article, d'une page ou d'un autre type de contenu personnalisé. Il récupère les données de la base de données, les traite et les prépare pour l'affichage.

8. **Modèles et thème** : WordPress sélectionne le thème actif et charge les fichiers de modèle correspondants (tels que header.php, footer.php, etc.). Ces fichiers définissent la structure et le style de la page.

9. **Affichage de la page** : Une fois que toutes les étapes précédentes sont terminées, WordPress génère le code HTML final et l'envoie au navigateur du visiteur. Cela inclut le contenu, les en-têtes, les pieds de page et les éléments visuels définis par le thème et les plugins.

Développement du plugin

Dans une installation standard de WordPress, le dossier des plugins se trouve dans le répertoire wp-content/plugins. Chaque plugin installé est généralement placé dans son propre sous-répertoire portant le nom du plugin selon la convention « mon-plugin ».

Structure et arborescence du répertoire

La racine du répertoire du plugin doit présenter un seul fichier PHP « index.php » qui sera le point d'entrée du plugin.

L'image ci-dessus présente une structure de base de l'arborescence du plugin où nous trouvons les différents éléments en accord avec le fonctionnement de WordPress :

index.php	Fichier d'entrée du plugin contenant également les informations spécifiques à la reconnaissance du plugin par WordPress.
public/	Dossier contenant les styles, scripts et fichiers de langues pour l'affichage sur le navigateur.
ressources/	Ressources supplémentaires propres au plugin (library,…)
src/	Code du plugin
src/admin	Partie backend (dashboard)
src/config	Configuration du plugin
src/database	Code et scripts de la BDD

A noter que lorsqu'un utilisateur désinstalle le plugin via l'interface d'administration de WordPress, le fichier « uninstall.php » est exécuté automatiquement avant que le plugin ne soit complètement supprimé. Il offre une opportunité de nettoyer les données et les paramètres liés au plugin qui ne sont plus nécessaires après la désinstallation tels certaines actions requises, les métadonnées restantes, des fichiers et dossiers, des tables,…

Les plugins WordPress peuvent se trouver dans différents états selon leur activation, désactivation ou désinstallation. Voici les principaux états d'un plugin :

1. **Activé** : Lorsqu'un plugin est activé, il est pleinement opérationnel sur le site WordPress. Ses fonctionnalités sont disponibles et peuvent être utilisées par les utilisateurs du site. **Les hooks et les actions du plugin sont déclenchés et son code est exécuté lors des événements appropriés.**

2. **Désactivé** : Lorsqu'un plugin est désactivé, il est toujours présent sur le site, mais ses fonctionnalités ne sont pas actives. **Le code du plugin n'est pas exécuté et ses hooks et actions ne sont pas déclenchés**. Les paramètres et les données spécifiques au plugin sont conservés dans la base de données et peuvent être récupérés lors de la réactivation ultérieure du plugin.

3. **Désinstallé** : Lorsqu'un plugin est désinstallé, il est entièrement supprimé du site WordPress. Cela peut se produire lorsque l'utilisateur choisit de désinstaller le plugin via l'interface d'administration de WordPress. La désinstallation supprime généralement toutes les données, les tables de la base de données et les fichiers associés au plugin. Certains plugins peuvent laisser des options de configuration ou des données sauvegardées même après la désinstallation, selon la manière dont ils sont conçus.

Il est possible de publier un plugin personnalisé sur le répertoire officiel des plugins de WordPress. C'est une plateforme où les développeurs peuvent soumettre leurs plugins pour qu'ils soient examinés et éventuellement répertoriés pour une utilisation par la communauté WordPress.

Les crochets «Hooks »

Dans le développement de plugins WordPress, les crochets « Hooks » sont des points d'entrée prédéfinis dans le code de WordPress qui permettent aux développeurs d'interagir avec le fonctionnement du cœur de WordPress, ainsi qu'avec les thèmes et les autres plugins.

Les « hooks » sont utilisés pour ajouter, modifier ou supprimer des fonctionnalités, des actions et des filtres dans WordPress, sans modifier directement le code source principal. Ils permettent aux développeurs de personnaliser le comportement de WordPress en ajoutant leur propre code à des emplacements spécifiques dans le processus de chargement.

Il existe deux types de hooks dans WordPress :

1. **Actions** : Les actions permettent aux développeurs d'ajouter leur propre code à des moments spécifiques dans le processus de chargement de WordPress. Ils sont utilisés pour effectuer des tâches spécifiques à des moments précis, tels que l'ajout de contenu à une page, l'enregistrement de données, l'envoi d'e-mails, …. Les actions sont déclenchées à l'aide de la fonction do_action() et enregistré grâce à la fonction add_action().

2. **Filtres** : Les filtres permettent aux développeurs de modifier les données avant qu'elles ne soient affichées ou utilisées par WordPress. Ils sont utilisés pour modifier les contenus, les paramètres, les requêtes de base de données, …. Les filtres sont appliqués à l'aide de la fonction apply_filters() et enregistré grâce à à la fonction add_filter().

Les hooks sont essentiels pour étendre WordPress et ajouter des fonctionnalités personnalisées sans modifier directement les fichiers principaux de WordPress ou des thèmes. Ils facilitent également la compatibilité entre les plugins et les thèmes, car ils permettent aux différents composants de s'interconnecter et de s'interagir de manière prévisible.

```php
<?php

1 reference | 0 implementations
class CASS_Dashboard_Menu
{
    1 reference | 0 overrides
    public static function execute()
    {

        //Create the dashboard menus
        add_action('admin_menu', 'wpcass_add_menu_page');
        0 references
        function wpcass_add_menu_page()
        {
            //Options
            global $cass_admin_group_list;

            add_options_page(
                'Configuration du Club Alpin Suisse de Sion',
                'CASs',
                'manage_options',
                'cass_plugin_options',
                'cass_plugin_options_page'
            );

            //Create CASs top-level menu
            add_menu_page(
                'Club Alpin Suisse de Sion',
                'CASs',
                'manage_options',
                'cass-settings',
                'page_callback_function',
                'dashicons-smiley',
                99
            );
```

Figure 3 - Exemple d'un crochet pour ajouter une page d'administration

La base de données

Lors de l'installation par défaut de WordPress, les tables suivantes sont créées :

wp_posts	Stocke les articles, les pages, les types de contenu personnalisés et d'autres éléments du contenu principal de votre site.
wp_postmeta	Elle est utilisée pour stocker les métadonnées associées aux articles, aux pages et aux types de contenu personnalisés. Elle permet de stocker des informations supplémentaires sur ces éléments, telles que les valeurs de champs personnalisés.
wp_users	Contient les informations sur les utilisateurs enregistrés sur votre site, y compris les administrateurs, les auteurs et les abonnés.
wp_usermeta	Stocke les métadonnées associées aux utilisateurs, telles que les informations de profil, les préférences, etc.
wp_comments	Elle est utilisée pour stocker les commentaires laissés par les utilisateurs sur votre site.
wp_commentmeta	Stocke les métadonnées associées aux commentaires, comme les informations supplémentaires liées à un commentaire spécifique.
wp_terms	Stocke les termes (mots-clés, catégories, tags) utilisés pour organiser et classifier le contenu de votre site.
wp_term_taxonomy	Relie les termes (stockés dans wp_terms) aux taxonomies correspondantes, telles que les catégories et les tags.
wp_term_relationships	Établit les relations entre les termes et les objets de contenu, tels que les articles et les types de contenu personnalisés.
wp_links	Elle est utilisée par le composant "Liens" de WordPress, qui permet de gérer les liens vers d'autres sites web.

wp_options	Stocke les options et les réglages généraux du site

Très important ! Les **métadonnées** « meta » sont utilisées pour ajouter des informations supplémentaires aux utilisateurs, publications (articles, pages, types de contenu personnalisés) ou autres… Elles sont stockées dans les tables **wp_usermeta** pour les utilisateurs et **wp_postmeta** pour les publications.

Figure 4 - Tables WordPress "wp_"

Les avantages d'utiliser les métadonnées plutôt que d'ajouter directement des colonnes supplémentaires aux tables **wp_users** et **wp_posts** sont les suivants :

1. **Flexibilité et extensibilité** : Les métadonnées permettent d'ajouter des informations personnalisées sans modifier directement la structure des tables de base de données de WordPress. Cela offre une plus grande flexibilité pour étendre les fonctionnalités et les informations associées aux utilisateurs et aux publications, sans risque de conflits avec les mises à jour futures de WordPress.

2. **Facilité de gestion** : Les métadonnées sont gérées par des clés-valeurs, ce qui facilite l'ajout, la récupération et la mise à jour des informations. Les clés peuvent être des chaînes de texte permettant d'identifier spécifiquement les informations que vous souhaitez stocker, tandis que les valeurs peuvent être de divers types de données (texte, nombre, tableau, etc.).

3. **Performances** : En utilisant des métadonnées, les informations supplémentaires sont stockées de manière séparée dans des tables dédiées (wp_usermeta et wp_postmeta), ce qui peut améliorer les performances en évitant une surcharge de la table principale. Cela est particulièrement bénéfique lorsque vous avez de nombreuses métadonnées ou que vous effectuez des requêtes spécifiques basées sur ces informations.

4. **Compatibilité avec les plugins et les thèmes** : Les métadonnées sont une approche standardisée utilisée par de nombreux plugins et thèmes de WordPress pour étendre les fonctionnalités de base. En utilisant les métadonnées, vous pouvez intégrer plus facilement votre code personnalisé avec d'autres extensions existantes, car elles peuvent également interagir avec les métadonnées.

```php
$cassMemberNo = get_user_meta( $user->ID, 'cassMemberNo', true );
$cassLevel = get_user_meta( $user->ID, 'cassLevel', true );
```

Figure 5 - Accéder à une métadonnée

Interactions – Les « capabilities »

WordPress met à disposition tout un environnement d'exécution avec lequel nous pouvons interagir facilement lors du développement de notre plugin grâce à diverses fonctios.

Les « capabilities » font référence aux autorisations ou aux droits d'accès qui sont attribués aux utilisateurs pour effectuer certaines actions ou accéder à certaines fonctionnalités. Les « capabilities » définissent les permissions et les niveaux d'accès des utilisateurs dans le contexte de WordPress.

Chaque utilisateur dans WordPress possède un ensemble de capabilities qui détermine ce qu'il est autorisé à faire. Par exemple, les capabilities peuvent inclure la possibilité de créer, éditer, publier ou supprimer des publications, la capacité à gérer des options ou des réglages spécifiques, ou encore la permission d'administrer le site dans son ensemble.

Les « capabilities » sont gérées par le système de rôles et de permissions de WordPress. Les rôles prédéfinis tels que "Administrateur", "Éditeur", "Auteur", "Contributeur" et "Abonné" sont associés à des « capabilities » spécifiques. Vous pouvez également créer vos propres rôles personnalisés et leur attribuer des « capabilities » spécifiques.

```php
add_role( "CASS_RL", "Chef de course");
add_role( "CASS_MEMBER", "Membre");
```

Figure 6 - Ajout de rôles dans WordPress

```php
add_action('admin_menu', 'wpcass_add_menu_page');
0 references
function wpcass_add_menu_page()
{
    //Options
    global $cass_admin_group_list;

    add_options_page(
        'Configuration du Club Alpin Suisse de Sion',
        'CASs',
        'manage_options',
        'cass_plugin_options',
        'cass_plugin_options_page'
    );
```

Figure 7 - Ajout d'un hook avec capabilities sur "manage_options"

Interactions – L'utilisateur courant

Pour récupérer l'utilisateur courant dans WordPress, on peut utiliser la fonction **wp_get_current_user().** Cette fonction retourne un objet de l'utilisateur courant représenté par la classe **WP_User**. On peut par exemple accéder à son id, son nom,…

Interactions – Exemple

Catégorie	Fonctions
Gestion des hooks/actions	add_action(), do_action(), remove_action()
Gestion des filtres	add_filter(), apply_filters(), remove_filter()

Requête de base de données	wpdb, get_results(), insert(), update(), delete()
Requêtes HTTP	wp_remote_get(), wp_remote_post(), wp_remote_request()
Gestion des options	get_option(), update_option(), delete_option()
Manipulation des fichiers/médias	wp_upload_dir(), wp_handle_upload(), wp_get_attachment_image_src()

Les formulaires

La création, modification et suppression des données fonctionne au moyen des formulaires traditionnels HTML <form> en lien ave PHP. L'environnement WordPress facilite leur intégration et allège la charge de développement au moyen de sa couche.

1. **Création de données** :
    - L'utilisateur accède à un formulaire où il peut saisir les informations nécessaires pour créer un nouvel enregistrement (par exemple, un nouvel article, une nouvelle page, un nouvel utilisateur, etc.).
    - Le formulaire soumis en utilisant la méthode POST principalement.
    - Lors de la soumission du formulaire, les données sont envoyées à une page PHP qui gère le traitement des données.
    - Les valeurs saisies par l'utilisateur sont récupéré à l'aide des variables $_POST ou $_GET.
    - Les données saisies peuvent être validées et filtrées pour s'assurer qu'elles sont correctes et sécurisées. On parle de sanitization et validation.
    - Ensuite, les fonctions telles que celles de la base de données (wpdb) permettent d'insérer les données dans la base de données.

2. **Modification de données** :
    - De manière similaire à la création, l'utilisateur accède à un formulaire prérempli qui a été récupéré au travers de paramètres, avec les données existantes qu'il souhaite modifier.
    - Le formulaire est soumis avec les modifications apportées par l'utilisateur.
    - Les données sont envoyées à la page PHP de traitement des données.
    - Les valeurs saisies/modifiées sont récupérées à partir des variables $_POST ou $_GET.
    - Les données sont validées.
    - Les données modifiées sont ensuite mises à jour dans la base de données.

3. **Suppression de données** :
    - L'utilisateur peut déclencher la suppression d'une entrée de données spécifique en cliquant sur un bouton ou en effectuant une action.

- La demande de suppression est envoyée à une page PHP de traitement.
- Les informations nécessaires pour identifier l'enregistrement à supprimer (par exemple, l'id de l'article, l'id de l'utilisateur, etc.) sont transmises via les variables $ _GET ou $_POST.
- Les données sont validées pour s'assurer que la suppression est autorisée.
- Ensuite, les fonctions de la base de données sont utilisées pour supprimer l'enregistrement.

Conclusion personnelle

La recherche et le développement de ce sujet m'a permis de comprendre plus en profondeur le fonctionnement du développement de plugin sur WordPress. C'est un livre conséquent qui traite de l'ensemble du processus de création, des fonctions. Seul bémol, la partie traitant de la WP_LIST_Table est un peu légère mais pourrait sans autre concerner un livre à elle-même.

Avant de chercher un livre, j'ai essayé de compiler de la recherche notamment au travers d'exemples de code, de recherches sur la documentation officiel et de tuto. Intéressant, cependant, un apport indéniable que le livre apport c'est un apport progressif mais surtout cohérent et intègre qui se révèle fondamental dans la facilitation d'un apprentissage.