21/02/2023

Documentation Technique

Manuel d'utilisation – Extension WordPress DiapoManager

Guillaume PASCAIL

Table des matières

I. Introduction de la documentation technique. 2

II. Architecture de l'extension. 3

III. Le fichier de configuration. 4

IV. Les class de sous-menus. 7

1. Déclaration des class de sous-menus : class-submenu.php 7

2. Affichage du contenu des pages du sous-menu. 9

I. Introduction de la documentation technique.

Cette documentation technique à pour but de compléter la documentation utilisateur. Elle peut servir aux utilisateurs ayant un problème avec la bonne utilisation du plugin mais aussi aux développeurs souhaitant reprendre et mettre à jour toute ou partie du plugin.

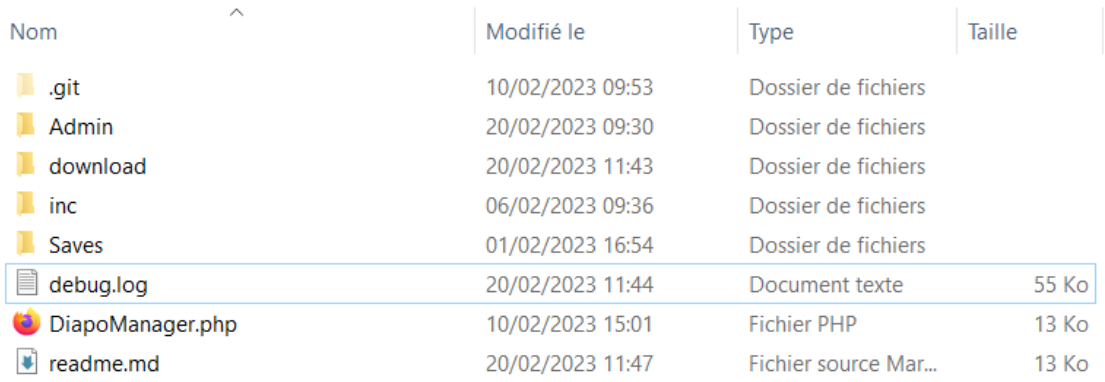
Nous palerons ici en langage technique en approchant divers aspects du code du plugin. Chaque fichier sera passé au crible 1 par 1. Vous pourrez retrouver dans le code source des commentaires vous indiquant le résultat de certaines fonctions.

Avant de passer à la suite, il vous faut savoir que ce plugin est principalement codé en langage PhP. Il contient aussi des passages en HTML ainsi que du CSS et du JavaScript (JS).

L'extension utilise également le Framework Boostrap 5.3.0 ainsi que sa bibliothèque d'image.

II. Architecture de l'extension.

Dans cette partie, nous allons nous familiariser avec l'architecture de l'extension.



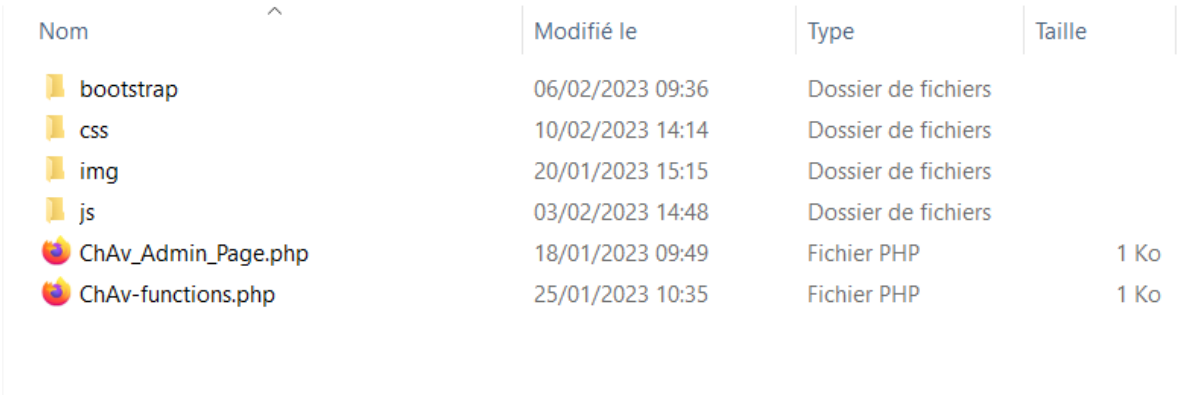
Nom	Modifié le	Type	Taille
.git	10/02/2023 09:53	Dossier de fichiers	
Admin	20/02/2023 09:30	Dossier de fichiers	
download	20/02/2023 11:43	Dossier de fichiers	
inc	06/02/2023 09:36	Dossier de fichiers	
Saves	01/02/2023 16:54	Dossier de fichiers	
debug.log	20/02/2023 11:44	Document texte	55 Ko
DiapoManager.php	10/02/2023 15:01	Fichier PHP	13 Ko
readme.md	20/02/2023 11:47	Fichier source Mar...	13 Ko

Si on oublie le dossier .git *(concerne la liaison entre le répertoire GitHub et mon répertoire sur mon PC)*, on remarque que le plugin est organisé comme la plupart des plugins WordPress : un fichier de configuration et plusieurs dossiers qui contiennent d'autres fichiers nécessaires au fonctionnement du plugin, des fichiers de téléchargements etc…

Comme le veut le nommage, le nom du fichier de configuration porte le nom de l'extension.

Ici, on remarque un fichier nommé debug.log. Il sert pour la partie développement à identifier plus facilement les erreurs. En effet, vous verrez de temps en temps, des lignes avec *error_log('texte')* : elles servent à noter une phrase à des points critique et donc détecter plus facilement les erreurs.

Remarque : les fichiers CSS et JS sont à mettre dans le dossier « inc » présent à la racine de l'extension. Attention, on parle de racine de l'extension, le dossier ou est situé le fichier de configuration. Les fichiers CSS vont dans le dossier css et les fichiers JS dans le dossier js. Le dossier boostrap concerne les fichiers CSS et JS du Framework Boostrap. Le fichier img contient toutes les potentielles images de l'extension.



Nom	Modifié le	Type	Taille
bootstrap	06/02/2023 09:36	Dossier de fichiers	
css	10/02/2023 14:14	Dossier de fichiers	
img	20/01/2023 15:15	Dossier de fichiers	
js	03/02/2023 14:48	Dossier de fichiers	
ChAv_Admin_Page.php	18/01/2023 09:49	Fichier PHP	1 Ko
ChAv-functions.php	25/01/2023 10:35	Fichier PHP	1 Ko

III. Le fichier de configuration.

Dans l'extension, le fichier de configuration porte le nom de « DiapoManager.php ». Ce fichier est indispensable à la configuration de l'extension. Il doit obligatoirement se situer à la racine de l'extension car c'est lui qui gère l'entièreté des fichiers présent ailleurs.

On retrouve au début de ce fichier, des lignes de commentaires. Ces lignes servent à définir l'affichage du plugin dans le menu « Extension » du site WordPress.

```php
<?php
/**
 * Fichier de configuration
 *
 * Ceci est le fichier principal qui sera lu par WordPress.
 *
 *
 * @wordpress-plugin
 * Plugin Name: DiapoManager
 * Plugin URI: https://github.com/Zaventurier/DiapoManagerDeveloppement
 * Description: Gérer des diaporamas n'à jamais été aussi simple ! Créer un diaporama, gérer les images et les descriptions que vous voulez et pub
 * Author: Guillaume Pascail
 * Version: 1.8.4
 * Author URI: https://github.com/Zaventurier/
 * License: No License
 * License URI:
 */

//Fichiers requis
require_once( ABSPATH . 'wp-includes/shortcodes.php' );
require_once plugin_dir_path(__FILE__) . 'inc/ChAv-functions.php';
require_once plugin_dir_path(__FILE__) . 'Admin/class-submenu.php';

if ( ! defined( 'ABSPATH' ) ) {
    die;
}
```

On a également inclus les fichiers requis pour le fonctionnement de certaines fonctions.

La condition en dessous est une mesure de sécurité : elle vérifie si la constante ABSPATH est définie. ABSPATH est une constante WordPress qui contient le chemin racine de l'installation WordPress. Si cette constante n'est pas définie, cela peut signifier que le fichier de configuration est appelé directement plutôt que via WordPress, ce qui peut entraîner des problèmes de sécurités et/ou de compatibilité. Si elle n'est pas définie, elle exécute la fonction die qui arrête l'exécution du script.

Une fois la mesure de sécurité passé, on définit des hooks. Un hook est un point d'entrées définit par WordPress qui permet au développeur de modifier et de personnaliser le comportement d'un site WordPress. Il existe 2 types de hook : les hooks d'action et les hooks de filtres. Un hook d'action permet au développeur d'exécuter son propre code à un moment précis lors de l'exécution du site et les hooks de filtres permettent de modifier les données avant qu'elles ne soit affichés à l'utilisateur.

Ici on utilise 2 hooks d'action qui vont exécuter, chacun une fonction : Create_Caroussel et Create_Slide toutes 2 contenus dans le même fichier.

```php
/**
 * Appel des fonctions qui vont s'éxécuter à l'activation du plugin
 * @since 1.1.2
 * Modifié : 1.6.0
 * Remarque : __FILE__ signifie que la fonction est présente dans le fichier.
 */

//register_activation_hook(__FILE__,'Prepare_To_Run');//On appelle la fonction Prepare_To_Run contenu dans ce fichier.
register_activation_hook(__FILE__, 'Create_Caroussel');//On appelle la fonction Create_Caroussel contenu dans ce fichier.
register_activation_hook(__FILE__, 'Create_Slide');//Appel de la fonction Create_Slide contenu dans ce fichier.
```

Ces 2 fonctions permettent de créer 2 tables dans la base de données du site WodPress.

Analysons la fonction Create_Caroussel :

On utilise la variable de type global $wpdb, une variable WordPress. Cette variable va nous permettre de créer le nom de la table en fonction du préfixe présent dans la base de données du site. On fait une vérification afin d'éviter que les 2 tables portent le même nom. On prépare la requête SQL si la table n'existe pas. L'exécution de la requête nécessite le fichier upgrade.php de WordPress. On utilise ensuite la fonction dbDelta, fonction WordPress qui permet d'exécuter des requêtes sur le SGBD du site.

L'entièreté des fonctions utilisant du SQL fonctionne dans un format similaire. Si vous avez compris le fonctionnement de cette fonction, vous comprendrez le fonctionnement des autres.

```php
/**
 * Résumé de Create_Caroussel
 * @return void
 * @since 1.1.3
 * Modifié : 1.4.1
 * Créer une deuxième table pour la gestion des Caroussels
 * Cette table à pour clé primaire idCaroussel et pour clé étrangère idImage de Image
 */

0 references
function Create_Caroussel(){
    global $wpdb;
    $nom_table = $wpdb->prefix . 'diapomanagercaroussel';
    $charset_collate = $wpdb->get_charset_collate();
    error_log('Fonction Create_Caroussel : Fonction correctement appellé !');
    /*On vérifie et on ajoute ici une seconde table pour la gestion des caroussels sur l'entièreté du site*/
    if($wpdb->get_var("SHOW TABLES LIKE '$nom_table'") != $nom_table) {
        $sql = "CREATE TABLE $nom_table (
            idCaroussel mediumint(11) NOT NULL AUTO_INCREMENT,
            nomCaroussel varchar(30) NOT NULL,
            PRIMARY KEY  (idCaroussel)
        ) $charset_collate;";
        error_log('Fonction Create_Caroussel : Requête préparé avec succés !');
        require_once( ABSPATH . 'wp-admin/includes/upgrade.php' );
        dbDelta( $sql );
        echo $wpdb->print_error();
    }
}
```

Autre cas de figure :

Il arrive que de temps en temps, la requête ne soit pas préparée *(niveau sécurité, on repassera)* mais le fonctionnement reste le même.

```php
2 references
function get_diaporama_by_id(int $idDiapo){
    global $wpdb;
    $table_name = $wpdb->prefix. 'diapomanagercaroussel';
    $resultat = $wpdb->get_results(" SELECT * FROM $table_name WHERE idCaroussel = $idDiapo ", ARRAY_A);
    return $resultat;
}
```

Les dernières fonctions permettent de définir les shortcodes (codes courts). Elles permettent d'exécuter un code.

```php
function carrousel_shortcode($atts) {
    // Récupérer l'argument du shortcode
    $atts = shortcode_atts(
        array(
            'id' => 0,
        ), $atts, 'carrousel'
    );
    $id = (int) $atts['id'];

    // Récupérer les informations sur le diaporama choisi
    $diaporama = get_diaporama_by_id($id);
    if (!$diaporama) {
        return '';
    }

    // Récupérer les images du diaporama
    $images = get_images_by_diaporama_id($id);

    //chargement du fichier CSS
    wp_register_style('bootstrap-style', 'https://cdn.jsdelivr.net/npm/bootstrap@5.3.0-alpha1/dist/css/bootstrap.min.css');
    wp_enqueue_style('bootstrap-style');

    //Chargement du fichier CSS pour modifier le style du caroussel.
    wp_register_style('DMstyle-css', '/wp-content/plugins/DiapoManager/inc/css/DMstyle.css');
    wp_enqueue_style('DMstyle-css');

    // Chargement du CDN JS de Bootstrap
    wp_register_script('bootstrap-script', 'https://cdn.jsdelivr.net/npm/bootstrap@5.3.0-alpha1/dist/js/bootstrap.bundle.min.js');
    wp_enqueue_script('bootstrap-script');

    // Initialiser le code HTML pour les diapositives
    $slides = '';
    $first = true;
    foreach ($images as $image) {
        $slides .= '<div class="carousel-item';
        if ($first) {
            $slides .= ' active';
//add_shortcode permet de définir, en tapant [carousel] d'éxécuter la fonction carrousel_shortcode et d'intégrer le code dans la page souhaité.
add_shortcode( 'DiapoA', 'carrousel_shortcode' );
```

Cette fonction permet, lorsque que l'on tape le shortcode définit dans la fonction add_shortcode. Le paramètre $atts est donc l'id du diaporama que l'ont veut afficher. Pour commencer, on récupère l'argument du shortcode pour le mettre dans une variable $id. Ensuite grâce à cette variable, on récupère les informations du diaporamas (Nom, etc…). On récupère ensuite toutes les images du diaporama en appelant une fonction et en passant en paramètre l'id du diaporama.

On charge ensuite les fichiers CSS et JS que l'on à besoin et on initialise le HTML.

Remarque : vous pouvez aller dans le code pour voir la suite de l'initialisation HTML. A savoir que l'initialisation HTML peut être personnaliser selon les besoins.

IV. Les class de sous-menus.

1. Déclaration des class de sous-menus : class-submenu.php

Pour afficher tout les sous-menus, on doit créer un dossier qui va les gérer. On le nomme « Admin ».

La totalité des sous menu est géré par le fichier class-submenu.php.

Le fichier requiert d'abord toutes les pages qui afficheront tous les sous-menus :

Remarque : Plugin_dir_path permet de récupérer le chemin absolu du fichiers dans lequel il est présent.

```php
require_once plugin_dir_path(__FILE__) . 'class-submenu-page.php';
require_once plugin_dir_path(__FILE__) . 'class-submenu-aide.php';
require_once plugin_dir_path(__FILE__) . 'class-subemnu-images.php';
require_once plugin_dir_path(__FILE__) . 'class-submenu-caroussel.php';
```

On déclare ensuite toutes les variables en privée : elles serviront à gérer les sous-menus.

```php
class Submenu {      You, last month • FirstCommit …
    /**
     * Cette class gère le menu et les sous-menu
     * C'est ici que son gérer l'affichage des différents sous-menu
     */

    /**
     * A reference the class responsible for rendering the submenu page.
     *
     * @var Submenu_Page
     * @access private
     */

    /**
     * Variables privées
     * @var mixed
     */

    //Pour la page ChasseAvenir87
    2 references
    private $submenu_page;
    //Pour le sous menu Aide
    1 reference
    private $submenu_aide;
    //Pour le sous menu Gérer les Images
    1 reference
    private $submenu_images;
    //Pour le sous menu Gérer les caroussels
    2 references
    private $submenu_caroussel;
    //Pour la maintenance des pages
    0 references
    private $maintenance;
```

On construit ensuite le constructeur. Il définit les class qui définissent chaque sous-menu.

Par exemple : le sous-menu « Aide » est définit par la class Submenu_Aide() ;

```
1 reference | 0 overrides
public function __construct( $submenu_page ) {
    $this->submenu_page = $submenu_page;
    $this->submenu_aide = new Submenu_Aide();
    $this->submenu_images = new Submenu_Images();
    $this->submenu_caroussel = new Submenu_Diapo();
}
```

Ensuite, on utilise la fonction add_option_page qui permet de définir une nouvelle page à afficher dans le menu WordPress :

```
1 reference | 0 overrides
public function add_options_page() {
    add_menu_page(
        'DiapoManager', //titre de la page d'options qui sera affiché en haut de l'écran dans le menu d'administration de WordPress.
        'DiapoManager', //nom qui apparaîtra dans le menu d'administration de WordPress.
        'manage_options',//niveau de permission requis pour accéder à cette page d'options.
        'info-page', //l'identifiant unique de la page d'options qui sera utilisé pour construire l'URL de la page.
        array( $this->submenu_page, 'render' ),//méthode de la classe qui est utilisée pour afficher le contenu de la page d'options.
        'dashicons-format-gallery',//l'icône qui apparaîtra à côté du nom de la page dans le menu d'administration de WordPress.
        100//rang de la page dans le menu d'administration de WordPress (plus le nombre est élevé, plus tard dans le menu la page apparaîtra).
    );
```

Remarque : l'identifiant unique de la page d'option, à savoir ici « info-page » est très important dans le cas ou vous souhaitez ajouter des pages enfant. Les pages enfants étant liés à la page parents, ils devront avoir son identifiant lors de leurs créations.

On ajoute ensuite un sous-menu dans notre menu de l'extension. Pour ce faire, on utilise la fonction WordPress add_submenu_page qui permet de définir cette page de sous-menu.

```
add_submenu_page(
    'info-page',//Identifiant de la page parent
    'Gérer les Diaporamas',//Nom de la page en haut
    'Gérer les Diaporamas',//Nom de la page dans le menu
    'manage_options',//Niveau de permission requis pour accéder à cette page
    'chasseavenir-caroussel',//Identifiant de la sous page
    array( $this->submenu_caroussel, 'render_diapo_page' )//Tableau contenant la méthode à appellé pour afficher la page
    );
```

Il est important de mettre l'identifiant de la page parent pour que le sous-menu s'affiche correctement avec le menu de l'extension. Si on souhaite avoir plusieurs menus pour l'extension, chaque page parents aura son propre identifiant unique et cela permettra d'ajouter les pages souhaiter à chaque menus.

2. Affichage du contenu des pages du sous-menu.

Ici, nous allons nous intéresser à l'affichage du sous-menu « Gérer les Diaporamas ». Comme vu précédemment, on à déclarer une variable $submenu_caroussel qu'on à ensuite construit avec une class Submenu_diapo(). Une fois la class appelé, on appelle la fonction contenu dans la class à savoir render_diapo_page qui contient le code à affiché dans la page.

```php
<?php       You, 4 weeks ago • Plugin Version 1.5.0 …
/**
 *Fichier permettant l'affichage de la page "Gérer les Caroussels"
 *
 * Fourni les fonctionnalité nécessaire pour le rendu de la page.
 *
 * @since 1.4.1
 * Modifié : 1.5.13
 */


//On initialise le fichier de débogage pour bébuger le code.
ini_set('error_log', dirname(__FILE__) . '/debug.log');
//On inclut le fichier qui contient la gestion du header
require_once plugin_dir_path(__FILE__) . '/class-header.php';

1 référence | 0 implementations | You, 2 weeks ago | 1 author (You)
class Submenu_Diapo
{
    /**
     * Résumé de render_caroussel_page
     * @return void
     * Cette fonction gère l'affichage du sous menu Gérer les caroussels
     * @since 1.1.3
     * Modification : 1.5.2
     */
    0 références | 0 overrides
    function render_diapo_page()
    {
        error_log('[ChasseAvenir87] > Utilisateur présent sur la page de Gestion des caroussels !');
        // Code pour afficher le contenu de la page "Gérer les caroussels"?>
        <DOCTYPE html>
            <html>
```

Le fonctionnement de cette page est classique à une autre page PhP. Je tiens à faire remarquer que le header de la page est géré en dehors de cette page. Pour ce faire, on appelle la page class-header.php. Cette page contient le code de gestion du header de notre page.

Il est géré par une class et donc une fonction qu'on appelle un peu plus loin dans notre code :

```php
function render_diapo_page()
{
    error_log('[DiapoManager] > Utilisateur présent sur la page de Gestion des caroussels !');      You, 1 second ago • Uncommitted changes
    // Code pour afficher le contenu de la page "Gérer les caroussels"?>
    <DOCTYPE html>
        <html>
            <head>
                <!-- Import du CSS de Bootsrap 5.3.0 -->
                <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0-alpha1/dist/css/bootstrap.min.css" rel="stylesheet"
                    integrity="sha384-GLhlTQ8iRABdZLl6O3oVMWSktQOp6b7In1Zl3/Jr59b6EGGoI1aFkw7cmDA6j6gD" crossorigin="anonymous">
                <!-- Import des icônes Bootsrap 1.10.3 -->
                <link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap-icons@1.10.3/font/bootstrap-icons.css">
            </head>
            <body>
                <?php
                //Appel de la fonction qui gère le header
                render_header();
                if (isset($_POST['diapo'])) {
                    $nom = isset($_POST['nom']) ? $_POST['nom'] : '';
                    $id = isset($_POST['id']) ? $_POST['id'] : '';
                } else {
                    $nom = "";
                    $id = 0;
                }
                ?>
                <hr style="border-top: 2px solid gray;">
                <div class="row">
                    <div class="col-md-2 text-center" style='border-right:2px solid gray;border-top:2px solid gray;border-bottom:2px solid gray;
                        <form class="form-floating" action="" method="post">
                            <input name="nomDiapo" type="input" class="form-control" id="floatingInputValue" style="width:100%;margin-top:10px;"
```

La plupart des fonctionnalités ont été expliqués ici. Vous êtes libre de parcourir le code pour comprendre certaines parties.

Cette documentation technique fonctionne pour la version 1.8.4 et supérieure et sera mis à jour avec l'arrivée de nouvelles fonctionnalités.

Cette documentation technique montre l'aspect technique de l'extension (le fonctionnement du code). Elle peut être compléter avec la documentation utilisateur qui explique comment se servir des différentes fonctionnalités intégrés !