DOCUMENTATION PLUGIN

Lucas MARTEAU

Table des matières

mon-plugin.php 2
admin-page.php 3
man.php 5
search.php 6
scraper.php 8
publish-post.php 9
template.php 10

mon-plugin.php

Le fichier mon-plugin.php est le point d'entrée principal du plugin "Mon Plugin". Ce plugin est conçu pour faciliter le web scraping et l'automatisation de la publication de contenu sur un site WordPress. Voici une explication détaillée des sections clés de ce fichier :

Plugin Header:

- **Plugin Name**: Le nom du plugin est "Mon Plugin".
- **Plugin URI**: L'URL où les utilisateurs peuvent trouver plus d'informations sur ce plugin.
- **Description**: Une brève description indiquant que le plugin est utilisé pour le web scraping.
- **Version**: La version actuelle du plugin.
- **Requires at least**: Version minimale de WordPress requise pour exécuter ce plugin.
- **Requires PHP**: Version minimale de PHP requise.
- **Author**: Les auteurs du plugin, LucasMarteau et HectorMorlaix.
- **Author URI**: L'URL du site Web de l'auteur.
- **License**: Le type de licence sous lequel le plugin est distribué, GPL v2 or later.
- **License URI**: L'URL où la licence complète peut être consultée.
- **Update URI**: L'URL où les mises à jour du plugin peuvent être trouvées.
- **Text Domain**: Le domaine de texte utilisé pour la traduction du plugin.
- **Domain Path**: Le chemin relatif vers les fichiers de langue.

Inclusions:

Les lignes suivantes incluent différents fichiers PHP nécessaires au fonctionnement du plugin :

- admin-page.php et man.php dans le répertoire includes/page pour les pages d'administration.
- search.php, scraper.php, publish-post.php dans le répertoire includes/fonction pour les fonctions de recherche, scraping et publication.
- template.php dans le répertoire includes/templates pour les templates de contenu.
- simple_html_dom.php, une dépendance externe utilisée pour manipuler le HTML lors du scraping.

admin-page.php

Le fichier admin-page.php contient des fonctions et des logiques pour gérer une page d'administration dans WordPress, spécifique au plugin "Mon Plugin". Voici un aperçu des fonctionnalités principales implémentées :

1. **Ajout de la page d'administration (`mon_plugin_add_admin_page()`):**
   - Utilisation de la fonction `add_menu_page()` pour ajouter une nouvelle entrée dans le menu d'administration de WordPress.
   - Le titre de la page est défini comme "Mon Plugin".
   - L'identifiant unique de la page est `mon-plugin-admin`.
   - La fonction de rappel `mon_plugin_admin_page_callback()` est spécifiée pour afficher le contenu de la page.
   - L'icône utilisée dans le menu est `dashicons-admin-generic`.
   - La position dans le menu est définie à 85.

2. **Enqueue des styles CSS de l'administration (mon_plugin_admin_enqueue_styles()):**
   - Utilisation de `wp_enqueue_style()` pour charger le fichier CSS `admin-style.css` depuis le répertoire `/assets/css/` du plugin.
   - Le style est uniquement enregistré et enqueué si le hook actuel correspond à `toplevel_page_mon-plugin-admin`.

3. Callback de la page d'administration (`mon_plugin_admin_page_callback()`):
   - Vérifie les permissions de l'utilisateur pour accéder à la page avec `current_user_can('manage_options')`.
   - Traite les soumissions de formulaires POST pour la recherche d'articles et le scraping d'URL.
   - Pour la recherche d'articles :
     - Récupère un mot-clé de recherche, un filtre de date et un nombre de résultats à afficher.
     - Utilise la fonction `search_articles_by_keyword()` pour rechercher des articles basés sur les critères donnés.
     - Affiche les résultats de la recherche.
   - Pour le scraping d'URL :
     - Récupère une URL à scraper.
     - Utilise `mon_plugin_publish_scraped_content()` pour publier le contenu scrappé (non défini dans ce code).
     - Affiche le contenu scrappé récupéré via `scrape_url_content()`.

4. Affichage des formulaires et des résultats :
    - Affiche des formulaires HTML pour entrer un mot-clé de recherche, filtrer par date, spécifier le nombre de résultats et scraper une URL.
    - Affiche le contenu scrappé et le message de publication dans des sections distinctes après les soumissions de formulaire.

Ce fichier définit essentiellement les fonctionnalités clés de l'interface d'administration pour le plugin "Mon Plugin", permettant aux utilisateurs d'effectuer des recherches d'articles par mots-clés, de scraper du contenu à partir d'URLs spécifiques, et de gérer ces actions depuis le tableau de bord de WordPress.

man.php

Le fichier man.php est destiné à gérer la page de manuel d'utilisation pour le plugin "Mon Plugin". Voici les points clés de ce fichier :

1. Plugin Header :
   - Le plugin est nommé "Manuel d'utilisation de Mon Plugin".
   - La description indique qu'il s'agit d'une page de manuel d'utilisation pour le plugin.
   - La version est spécifiée comme 1.0.
   - L'auteur du plugin est défini comme "Votre Nom".
   - Le domaine de texte est défini comme "manuel-utilisation-plugin".

2. Fonction pour afficher le contenu de la page de manuel (mon_plugin_manuel_utilisation_page()) :
   - Cette fonction génère le contenu HTML de la page de manuel.
   - Elle inclut des sections telles que l'introduction, l'installation, la configuration, les fonctionnalités et le support.
   - Chaque section est décrite avec des instructions et des détails sur l'utilisation du plugin.

3. Fonction pour ajouter une page de manuel dans le menu d'administration (`mon_plugin_add_manuel_page()`) :
   - Utilisation de `add_menu_page()` pour ajouter une nouvelle entrée de menu dans l'administration de WordPress.
   - Le titre de la page est "Manuel d'utilisation de Mon Plugin".
   - Le texte du menu est "Manuel d'utilisation".
   - La capacité requise pour voir la page est 'manage_options'.
   - Le slug de la page est 'mon-plugin-manuel'.
   - La fonction de rappel pour afficher la page est `mon_plugin_manuel_utilisation_page()`.
   - L'icône utilisée dans le menu est `dashicons-book-alt`.
   - La position dans le menu est définie à 85.

Ce fichier permet d'intégrer une documentation complète et structurée directement dans l'interface d'administration de WordPress pour guider les utilisateurs sur l'installation, la configuration et l'utilisation du plugin "Mon Plugin".

search.php

Le fichier search.php définit une fonction nommée search_articles_by_keyword() qui permet de rechercher des articles en fonction d'un mot-clé donné à l'aide de l'API Google Custom Search. Voici une analyse détaillée de son fonctionnement :

1. **Fonction search_articles_by_keyword($keyword, $date_filter = '', $num_results = 10)** :
    - **Paramètres** :
        - $keyword : Le mot-clé à rechercher dans les articles.
        - $date_filter : Optionnel. Filtre de date pour restreindre les résultats (dernière heure, dernières 24 heures, dernière semaine, dernier mois, etc.).
        - $num_results : Optionnel. Nombre maximum de résultats à retourner (par défaut, 10).
    - **Cache de résultats** :
        - Utilisation de get_transient() et set_transient() pour mettre en cache les résultats de recherche pendant une heure (3600 secondes) afin d'améliorer les performances en évitant des requêtes répétées à l'API Google.
    - **Paramètres de l'API Google Custom Search** :
        - Clé API ($api_key) : Clé d'accès à l'API Google Custom Search. À remplacer par votre propre clé.
        - ID du moteur de recherche personnalisé ($search_engine_id) : Identifiant de votre moteur de recherche personnalisé. À remplacer par votre propre ID.
    - **Recherche multi-site** :
        - Utilisation d'un tableau ($sites_to_search) qui associe des pays à des sites web spécifiques pour la recherche.
        - À chaque itération, la recherche est effectuée sur un site différent en fonction du nombre de résultats déjà récupérés.
    - **Boucle de récupération des résultats** :
        - Utilisation de boucles pour récupérer des résultats par lots de 10 jusqu'à atteindre le nombre maximum spécifié ($num_results).
    - **Traitement des réponses** :
        - Utilisation de cURL pour exécuter des requêtes HTTP vers l'API Google Custom Search.
        - Gestion des erreurs potentielles avec curl_errno() et curl_error() pour capturer et enregistrer les erreurs dans les logs.
    - **Formatage des résultats** :
        - Les résultats sont formatés sous forme de liste HTML <ul> avec des liens vers les titres des articles trouvés.
    - **Retour des résultats** :
        - Les résultats sont retournés sous forme de chaîne HTML qui inclut les titres des articles et leurs liens correspondants.

Cette fonctionnalité permet à votre plugin de rechercher et de présenter des articles pertinents en fonction des mots-clés spécifiés par l'utilisateur, en utilisant l'API Google Custom Search pour accéder à divers sites web prédéfinis.

scraper.php

Le fichier scraper.php définit une fonction nommée scrape_url_content() qui permet de récupérer et de traiter le contenu d'une URL spécifiée. Voici un examen détaillé de son fonctionnement :

1. **Fonction scrape_url_content($url)** :
    - **Paramètre** :
        - $url : L'URL à scraper pour récupérer le contenu.
2. **Initialisation des données** :
    - Création d'un tableau $data avec deux clés : post_title (titre de l'article) et post_content (contenu de l'article).
3. **Vérification de l'URL non vide** :
    - Vérifie si l'URL fournie n'est pas vide avant de continuer le processus de scraping.
4. **Chargement du contenu HTML de l'URL** :
    - Utilisation de la fonction file_get_html() qui utilise Simple HTML DOM Parser pour charger et analyser le contenu HTML de l'URL.
5. **Extraction du titre de la page** :
    - Utilisation de Simple HTML DOM Parser pour trouver et extraire le titre de la page à partir de la balise <title>.
6. **Extraction du contenu principal** :
    - Recherche des éléments de contenu principaux à partir des balises HTML comme article, .entry-content, main, etc.
    - Si ces éléments spécifiques ne sont pas trouvés, tous les paragraphes <p>, listes <ul> et <ol> sont récupérés pour former le contenu.
7. **Formatage du contenu** :
    - Les balises HTML et leur contenu sont conservés en utilisant innertext pour les éléments principaux comme article.
    - Les paragraphes, listes et leurs éléments sont formatés et concaténés dans $content.
8. **Gestion des erreurs** :
    - Si une erreur survient lors du chargement de l'URL ou si l'URL est invalide, un message d'erreur approprié est assigné à $data['post_content'].
9. **Nettoyage des ressources** :
    - Après l'extraction du contenu, la mémoire utilisée par Simple HTML DOM Parser est libérée à l'aide de clear() et unset().
10. **Retour des données** :
    - Le tableau $data contenant le titre et le contenu de l'article est retourné.

Cette fonctionnalité permet à votre plugin de scraper le contenu d'une URL spécifiée, extrayant le titre de la page et le contenu principal pour une utilisation ultérieure, par exemple, pour publier automatiquement des articles à partir des données récupérées.

publish-post.php

Le fichier publish-post.php définit une fonction nommée mon_plugin_publish_scraped_content() qui permet de publier le contenu scrappé d'une URL spécifiée en tant qu'article WordPress. Voici un examen détaillé de son fonctionnement :

1. **Fonction mon_plugin_publish_scraped_content($url) :**
   - **Paramètre** :
     - $url : L'URL à partir de laquelle le contenu doit être scrappé et publié.
2. **Vérification des capacités de l'utilisateur** :
   - Vérifie si l'utilisateur actuel a la capacité de publier des articles (publish_posts). Si non, la fonction retourne "Permission refusée."
3. **Scraper le contenu de l'URL** :
   - Utilise la fonction scrape_url_content($url) pour récupérer le titre et le contenu de l'article à partir de l'URL spécifiée.
4. **Vérification du succès du scraping** :
   - Vérifie si les données scrappées contiennent à la fois un titre (post_title) et un contenu (post_content). Si l'une de ces valeurs est vide, la fonction retourne "Erreur : Impossible de récupérer le contenu scrappé."
5. **Création d'un nouvel article WordPress** :
   - Construit un tableau $new_post contenant les détails de l'article à insérer :
     - post_title : Le titre de l'article scrappé, sans balises HTML, grâce à wp_strip_all_tags().
     - post_content : Le contenu de l'article scrappé.
     - post_status : Statut de l'article, ici défini sur "draft" pour un brouillon.
     - post_author : L'ID de l'auteur actuellement connecté via get_current_user_id().
6. **Insertion du post dans la base de données** :
   - Utilise wp_insert_post($new_post) pour insérer l'article dans la base de données de WordPress.
7. **Gestion des erreurs d'insertion** :
   - Vérifie si l'insertion du post retourne une instance de WP_Error. Si c'est le cas, enregistre le message d'erreur dans les logs et retourne "Erreur lors de la publication de l'article : [message d'erreur]".
   - Si $post_id est valide (non nul et non erreur), retourne un message de succès avec un lien vers l'article publié.
8. **Retour de résultats** :
   - Si tout se passe bien, retourne un message indiquant que le contenu scrappé a été publié avec succès en tant qu'article, avec un lien pour voir l'article.
   - Si une autre erreur non spécifiée survient, retourne "Erreur : Impossible de publier le contenu scrappé."

template.php

Le fichier template.php que vous avez fourni définit une fonction `get_scraped_content_template($content)` qui génère un template HTML stylisé pour afficher du contenu scrappé. Voici une explication détaillée de son fonctionnement :

Fonction `get_scraped_content_template($content)` :

- **Paramètre** :
  - `$content` : Le contenu à afficher dans le template.

2. **Mise en mémoire tampon de sortie (Output Buffering)** :
   - Utilise `ob_start()` pour démarrer la mise en mémoire tampon de la sortie PHP. Cela permet de capturer tout le contenu HTML généré à l'intérieur de la fonction.
3. **Structure HTML du template** :
   - Définit une structure HTML stylisée pour le contenu scrappé :

```
html
Copier le code
<div class="scraped-content">
  <div class="scraped-body">
    <?php echo wpautop(wp_kses_post($content)); ?>
  </div>
</div>
```

     - Utilise `wpautop()` pour ajouter des balises de paragraphe (`<p>`) autour des blocs de texte.
     - Utilise `wp_kses_post()` pour s'assurer que le contenu est sécurisé en tant que contenu de publication WordPress.
4. **Styles CSS intégrés** :
   - Inclut des styles CSS internes pour formater le contenu :

```
css
Copier le code
.scraped-content {
  font-family: Arial, sans-serif;
  line-height: 1.6;
  margin: 20px auto;
  max-width: 800px;
  border: 1px solid #ddd;
  border-radius: 8px;
  box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);
  background-color: #fff;
  padding: 20px;
  overflow: hidden;
}
.scraped-body {
  margin-top: 20px;
}
.scraped-body p {
  margin: 15px 0;
```

```
    font-size: 16px;
    line-height: 1.8;
}
.scraped-body ul, .scraped-body ol {
    margin: 15px 0 15px 30px;
    padding-left: 10px;
}
.scraped-body li {
    margin-bottom: 8px;
}
.scraped-body img {
    max-width: 100%;
    height: auto;
    display: block;
    margin: 15px 0;
    border-radius: 8px;
    box-shadow: 0 2px 4px rgba(0, 0, 0, 0.1);
}
```

        - Ces styles définissent la mise en page, la typographie et les marges pour le contenu affiché dans le template.

5. **Fin de la mise en mémoire tampon et retour du contenu** :
    - Utilise ob_get_clean() pour récupérer le contenu de la mémoire tampon et nettoyer la mémoire tampon de sortie sans l'afficher. Cela retourne tout le HTML généré par la fonction.
6. **Retour de la fonction** :
    - Retourne le HTML complet du template qui peut être intégré dans d'autres parties de votre plugin pour afficher du contenu scrappé de manière esthétique et cohérente.