Comment Créer un Plugin WordPress pour Importer des Produits depuis n'importe quel Site E-commerce (Guide Complet)

Table des matières

> *Cet article est un guide technique complet destiné aux développeurs. Il explique comment concevoir et coder un plugin WordPress capable d'extraire des informations de produits depuis un site e-commerce via son URL et de les importer dans une boutique WooCommerce. Nous aborderons les défis, les outils nécessaires et les étapes de développement, de la création de l'interface à l'automatisation de la création de produits.*

Introduction : Le Défi de l'Importation Universelle

L'idée d'importer un produit depuis n';importe quel site e-commerce avec un simple lien est séduisante. Elle promet un gain de temps considérable pour les gestionnaires de boutiques, notamment dans le cadre du dropshipping ou de l';analyse concurrentielle. Cependant, la réalité technique est bien plus complexe. Il n'existe pas de format standard pour les pages produits sur le web. Chaque site, qu'il soit basé sur Shopify, Magento, PrestaShop ou une solution maison, possède sa propre structure HTML, ses propres classes CSS et charge souvent ses données de manière dynamique avec JavaScript.

Créer un plugin "universel" qui fonctionne partout sans configuration est donc un défi majeur. Ce guide a pour objectif de vous fournir les connaissances et les briques de code nécessaires pour construire un tel outil. Nous allons nous appuyer sur deux technologies clés :

- **Le Web Scraping :** Une technique d'extraction automatisée de données depuis des pages web. Nous utiliserons une bibliothèque PHP puissante pour "lire" la page produit et en extraire les informations pertinentes (nom, prix, description, images).

- **L'API de WooCommerce :** Une fois les données extraites, nous utiliserons les fonctions natives de WooCommerce pour créer un nouveau produit dans votre boutique WordPress de manière propre et structurée.

Ce tutoriel vous montrera comment construire une base solide pour votre plugin, tout en explorant les limites de cette approche et les solutions plus avancées pour tendre vers une véritable universalité.

Comprendre le Web Scraping pour l'E-commerce

Avant de coder, il est essentiel de comprendre les mécanismes et les obstacles du web scraping, particulièrement dans le contexte très compétitif de l'e-commerce.

Qu'est-ce que le Web Scraping ?

Le web scraping (ou extraction de données web) est le processus qui consiste à utiliser des programmes informatiques, appelés "scrapers" ou "bots", pour extraire automatiquement des informations publiques depuis des sites web. Au lieu de copier-coller manuellement les données, un scraper visite une page, analyse son code source (HTML) et récupère les données structurées dont il a besoin, comme le nom d'un produit, son prix ou ses avis clients. Selon ScraperAPI, c'est un outil essentiel pour l'analyse concurrentielle et la veille de marché.

Les Défis du Scraping E-commerce

Les sites e-commerce sont des cibles particulièrement difficiles pour le scraping en raison de plusieurs facteurs :

- **Contenu Dynamique (JavaScript/AJAX) :** De nombreux sites modernes chargent les informations produits (comme le prix ou la disponibilité) après le chargement initial de la page, via JavaScript. Un scraper simple qui ne lit que le HTML initial ne verra pas ces données. Des outils comme Selenium ou Puppeteer sont souvent nécessaires pour simuler un navigateur et exécuter ce JavaScript.
- **Mesures Anti-Scraping :** Pour protéger leurs données, les sites déploient des défenses comme le blocage d'adresses IP, la limitation du nombre de requêtes (rate limiting) et des systèmes de détection de bots. Ils peuvent également présenter des CAPTCHAs pour distinguer un humain d'un programme.

- **Changements de Structure Fréquents :** Les marques mettent constamment à jour le design de leur site pour des campagnes marketing, des tests A/B ou des optimisations de conversion. Un scraper programmé pour une structure HTML spécifique peut cesser de fonctionner du jour au lendemain après une simple mise à jour du site cible.

Choisir les bons outils : Pourquoi PHP et Panther ?

Puisque notre objectif est de créer un plugin WordPress, le choix de **PHP** est une évidence. C'est le langage sur lequel repose WordPress. Pour surmonter les défis du scraping, notamment le contenu dynamique, nous avons besoin d'une bibliothèque PHP capable de piloter un vrai navigateur.

C'est là qu'intervient **Symfony Panther**. Panther est une bibliothèque PHP qui utilise le protocole WebDriver pour contrôler des navigateurs comme Chrome ou Firefox.
Contrairement à des outils plus simples comme cURL ou Goutte, Panther peut :

- Exécuter le JavaScript d'une page, révélant ainsi le contenu chargé dynamiquement.
- Simuler des interactions humaines comme cliquer sur des boutons ou faire défiler la page.
- Prendre des captures d'écran pour le débogage.

En choisissant Panther, nous nous donnons les moyens de scraper une plus grande variété de sites e-commerce, y compris les plus modernes et complexes.

Étape 1 : Mettre en Place la Structure du Plugin WordPress

Commençons par créer la coquille de notre plugin. C'est la base sur laquelle nous construirons toutes nos fonctionnalités.

Créer les Fichiers et Dossiers du Plugin

La structure d'un plugin WordPress est simple. Dans le répertoire `/wp-content/plugins/` de votre installation WordPress, créez un nouveau dossier. Nommons-le `ecommerce-product-importer`. À l'intérieur de ce dossier, créez un fichier PHP principal du même nom : `ecommerce-product-importer.php`.

Suivre les bonnes pratiques de WordPress pour la structure des fichiers est essentiel pour la maintenabilité et la compatibilité.

```
wp-content/
└── plugins/
    └── ecommerce-product-importer/
        └── ecommerce-product-importer.php
```

L'En-tête du Plugin

Pour que WordPress reconnaisse votre fichier comme un plugin, il doit commencer par un bloc de commentaires spécifique, appelé "en-tête du plugin". Ouvrez `ecommerce-product-importer.php` et ajoutez le code suivant :

```
<?php
/**
 * Plugin Name:       E-commerce Product Importer
 * Plugin URI:        https://example.com/
 * Description:       Importe des produits depuis n'importe quel site e-comme
 * Version:           1.0.0
 * Author:            Votre Nom
 * Author URI:        https://your-website.com/
 * License:           GPL v2 or later
 * License URI:       https://www.gnu.org/licenses/gpl-2.0.html
 * Text Domain:       ecommerce-product-importer
 */

// Empêcher l'accès direct au fichier
if ( ! defined( 'ABSPATH' ) ) {
    exit;
}
```

Une fois ce fichier sauvegardé, vous devriez pouvoir voir et activer votre plugin depuis le menu "Extensions" de votre tableau de bord WordPress.

Créer une Page d'Administration

Notre plugin a besoin d'une interface où l'utilisateur pourra coller l'URL du produit à importer. Nous allons créer une nouvelle page dans le menu d'administration de WordPress. Pour cela, nous utilisons le hook `admin_menu` et la fonction `add_menu_page()`.

Ajoutez ce code à la fin de votre fichier `ecommerce-product-importer.php` :

```
function epi_add_admin_menu() {
    add_menu_page(
        'E-commerce Importer',          // Titre de la page
        'Product Importer',             // Titre du menu
        'manage_options',               // Capacité requise
        'ecommerce-product-importer',   // Slug du menu
        'epi_admin_page_html',          // Fonction qui affiche le contenu de
        'dashicons-download',           // Icône
        20                              // Position
    );
}
add_action('admin_menu', 'epi_add_admin_menu');

function epi_admin_page_html() {
    // Nous ajouterons le HTML de notre page ici
    if (!current_user_can('manage_options')) {
        return;
    }
    echo '<div class="wrap">';
    echo '<h1>' . esc_html(get_admin_page_title()) . '</h1>';
    echo '<p>Collez l\'URL du produit que vous souhaitez importer ci-dessous.
    // Le formulaire sera ajouté dans la prochaine étape
    echo '</div>';
}
```

La documentation de WordPress fournit des détails complets sur les paramètres de `add_menu_page()`. Après avoir ajouté ce code, un nouveau menu "Product Importer" apparaîtra dans votre administration.

Étape 2 : Construire l'Interface d'Importation

Maintenant que notre page d'administration existe, ajoutons un formulaire pour que l'utilisateur puisse soumettre une URL.

Le Formulaire HTML

Modifions notre fonction `epi_admin_page_html()` pour y inclure un formulaire. Ce formulaire enverra les données à `admin-post.php`, la méthode standard et sécurisée de WordPress pour gérer les soumissions de formulaires depuis le back-office.

```
function epi_admin_page_html() {
    if (!current_user_can('manage_options')) {
        return;
    }
    ?>
    <div class="wrap">
        <h1><?php echo esc_html(get_admin_page_title()); ?></h1>
        <p>Collez l'URL du produit que vous souhaitez importer ci-dessous.</p

        <form method="post" action="<?php echo esc_url(admin_url('admin-post.
            <input type="hidden" name="action" value="epi_import_product">
            <?php wp_nonce_field('epi_import_product_nonce', 'epi_nonce'); ?>

            <table class="form-table">
                <tr>
                    <th scope="row">
                        <label for="product_url">URL du Produit</label>
                    </th>
                    <td>
                        <input type="url" name="product_url" id="product_url"
                    </td>
                </tr>
            </table>

            <?php submit_button('Importer le Produit'); ?>
        </form>
    </div>
    <?php
}
```

Points importants dans ce code :

- `action="<?php echo esc_url(admin_url('admin-post.php')); ?>"` : Cible le gestionnaire de requêtes POST de WordPress.

- `<input type="hidden" name="action" value="epi_import_product">` : Indique à WordPress quelle action exécuter.
- `wp_nonce_field(...)` : Ajoute un champ de sécurité (nonce) pour protéger contre les attaques CSRF.

Gérer la Soumission du Formulaire avec admin-post.php

Lorsque le formulaire est soumis, WordPress cherchera un hook nommé `admin_post_{votre_action}`. Dans notre cas, c';est `admin_post_epi_import_product`. Nous devons créer une fonction qui s'accroche à ce hook pour traiter les données.

Ajoutez ce code à votre plugin :

```
function epi_handle_form_submission() {
    // 1. Vérifier le nonce pour la sécurité
    if (!isset($_POST['epi_nonce']) || !wp_verify_nonce($_POST['epi_nonce'],
        wp_die('La vérification de sécurité a échoué.');
    }

    // 2. Vérifier les permissions de l'utilisateur
    if (!current_user_can('manage_options')) {
        wp_die('Vous n\'avez pas les permissions pour effectuer cette action.
    }

    // 3. Valider et nettoyer l'URL
    if (!isset($_POST['product_url']) || empty($_POST['product_url'])) {
        wp_die('Veuillez fournir une URL de produit.');
    }
    $product_url = esc_url_raw($_POST['product_url']);

    // 4. Lancer le processus de scraping (sera détaillé ensuite)
    // Pour l'instant, affichons simplement l'URL
    echo 'Lancement du scraping pour l\'URL : ' . esc_html($product_url);
    // Ici, nous appellerons nos fonctions de scraping et de création de prod

    // 5. Rediriger l'utilisateur (optionnel mais recommandé)
    // wp_redirect(admin_url('admin.php?page=ecommerce-product-importer&statu
    // exit;
}
```

```
add_action('admin_post_epi_import_product', 'epi_handle_form_submission');
```

L'utilisation du hook `admin_post_{action}` est la méthode recommandée par WordPress pour gérer les requêtes POST dans le back-office, car elle garantit que l'environnement WordPress est entièrement chargé.

Étape 3 : Le Cœur du Plugin - Scraper les Données Produit avec Panther

C'est ici que la magie opère. Nous allons utiliser Panther pour visiter l'URL fournie et en extraire les données. Notez que pour utiliser Panther, votre plugin devra gérer des dépendances via Composer, ce qui est une pratique avancée dans l'écosystème WordPress.

> ***Note sur Composer :*** *Pour installer Panther, vous devez utiliser Composer, un gestionnaire de dépendances pour PHP. Exécutez* `composer require symfony/panther` *dans le dossier de votre plugin. Assurez-vous ensuite d'inclure le fichier d'autoload de Composer au début de votre plugin :* `require_once __DIR__ . '/vendor/autoload.php';`.

Initialiser Panther et Naviguer vers l'URL

Dans notre fonction `epi_handle_form_submission`, nous allons remplacer le simple `echo` par l'initialisation de Panther.

```
use Symfony\Component\Panther\Client;

// ... dans la fonction epi_handle_form_submission() après la validation de l

try {
    $client = Client::createChromeClient(); // Ou createFirefoxClient()
    $crawler = $client->request('GET', $product_url);

    // Le scraping se fera ici...

} catch (\Exception $e) {
    wp_die('Erreur lors du scraping : ' . $e->getMessage());
```

```
}
```

Extraire les Données Essentielles

C'est la partie la plus délicate, car elle dépend entièrement de la structure du site cible. Vous devrez utiliser les outils de développement de votre navigateur (clic droit > "Inspecter") pour trouver les sélecteurs CSS ou XPath des éléments que vous voulez extraire.

Voici un exemple générique qui tente d'extraire le nom, le prix, la description et l'URL de l'image principale. **Ces sélecteurs sont des exemples et devront être adaptés pour chaque site.**

```
// ... à l'intérieur du bloc try

// Attendre que l'élément principal du produit soit visible (utile pour les s
$client->waitForVisibility('.product-main-info'); // Sélecteur d'exemple

// Extraire le nom du produit
$name = $crawler->filter('h1.product-title')->text(); // Sélecteur d'exemple

// Extraire le prix
$price = $crawler->filter('.price .amount')->text(); // Sélecteur d'exemple
// Nettoyer le prix pour ne garder que les chiffres
$price = preg_replace('/[^0-9\.,]/', '', $price);
$price = str_replace(',', '.', $price);

// Extraire la description
$description = $crawler->filter('#product-description')->html(); // Sélecteur

// Extraire l'URL de l'image principale
$image_url = $crawler->filter('.product-image-main img')->attr('src'); // Sél

$product_data = [
    'name'        => trim($name),
    'price'       => floatval($price),
    'description' => trim($description),
    'image_url'   => trim($image_url),
];

// Pour le débogage
echo '<pre>';
print_r($product_data);
```

```
echo '</pre>';

// Ensuite, nous passerons $product_data à notre fonction de création de prod
```

Gérer le Contenu Dynamique et les Interactions

L'un des grands avantages de Panther est sa capacité à attendre que des éléments apparaissent. Comme l'explique ZenRows dans son tutoriel sur Panther, des méthodes comme `$client->waitFor('.selector')` sont cruciales pour les sites qui chargent du contenu via AJAX. Si un prix n'apparaît qu'après une seconde, cette fonction mettra le script en pause jusqu'à ce qu'il soit disponible, rendant le scraping beaucoup plus fiable.

WooCommerce fournit des objets PHP pour manipuler les produits par programmation

Étape 4 : Créer le Produit dans WooCommerce

Une fois les données extraites, l'étape finale consiste à les utiliser pour créer un produit dans votre boutique WooCommerce.

Utiliser les Objets CRUD de WooCommerce

Depuis la version 3.0, WooCommerce a introduit des objets CRUD (Create, Read, Update, Delete) pour la gestion des données. C'est la méthode recommandée et la plus sûre pour interagir avec les produits. Il est fortement déconseillé d'utiliser les fonctions WordPress de

base comme `wp_insert_post()` car cela pourrait causer des incohérences dans les données de WooCommerce.

Nous utiliserons la classe `WC_Product_Simple` pour créer un produit simple.

Gérer les Images du Produit

Importer une image depuis une URL externe dans WordPress nécessite quelques étapes. Heureusement, WordPress fournit une fonction très pratique pour cela : `media_sideload_image()`. Cette fonction télécharge une image depuis une URL, la sauvegarde dans la médiathèque et l'attache à un article (ou produit).

Nous allons créer une fonction d'aide pour gérer ce processus et définir l'image téléchargée comme "image mise en avant" du produit.

```
function epi_attach_image_from_url($image_url, $product_id) {
    require_once(ABSPATH . 'wp-admin/includes/media.php');
    require_once(ABSPATH . 'wp-admin/includes/file.php');
    require_once(ABSPATH . 'wp-admin/includes/image.php');

    // Télécharge l'image et l'attache à la médiathèque
    $attachment_id = media_sideload_image($image_url, $product_id, 'Product i

    // Si le téléchargement réussit, définir comme image mise en avant
    if (!is_wp_error($attachment_id)) {
        set_post_thumbnail($product_id, $attachment_id);
        return $attachment_id;
    }

    return false;
}
```

La documentation officielle de `media_sideload_image` confirme qu'elle peut retourner l'ID de l'attachement, ce qui est exactement ce dont nous avons besoin.

Assembler le Code de Création de Produit

Créons maintenant la fonction principale qui prendra notre tableau `$product_data` et créera le produit.

```
function epi_create_woocommerce_product($data) {
    if (!class_exists('WC_Product_Simple')) {
        return new WP_Error('woocommerce_not_active', 'WooCommerce n\'est pas
    }

    $product = new WC_Product_Simple();

    // Définir les données de base
    $product->set_name($data['name']);
    $product->set_regular_price($data['price']);
    $product->set_description($data['description']);
    $product->set_short_description(''); // Peut être rempli si scrapé

    // Définir le statut sur "publié"
    $product->set_status('publish');
    $product->set_catalog_visibility('visible');

    // Gérer le stock (exemple simple)
    $product->set_manage_stock(false);
    $product->set_stock_status('instock');

    // Sauvegarder le produit pour obtenir un ID
    $product_id = $product->save();

    if ($product_id > 0 && !empty($data['image_url'])) {
        // Attacher l'image
        epi_attach_image_from_url($data['image_url'], $product_id);
    }

    return $product_id;
}
```

Il suffit maintenant d'appeler cette fonction à la fin de notre processus de scraping dans `epi_handle_form_submission()` :

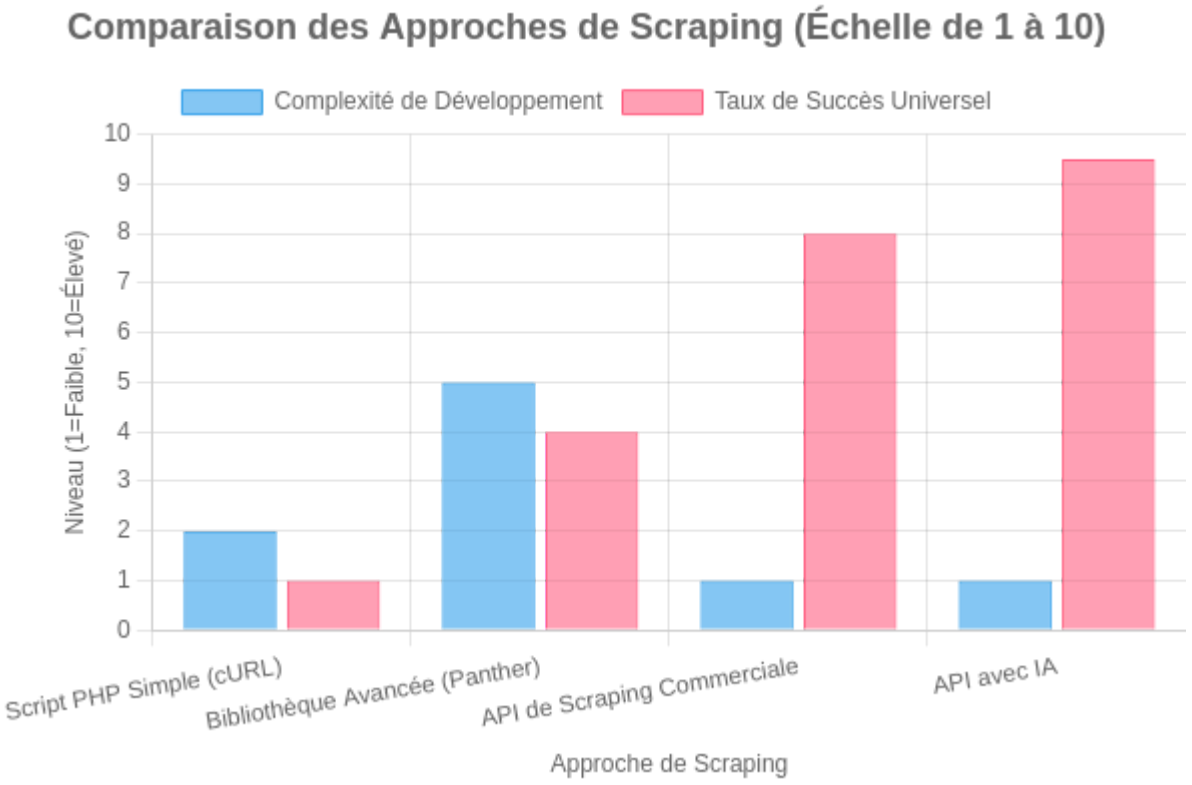
```
// ... après avoir rempli le tableau $product_data
$new_product_id = epi_create_woocommerce_product($product_data);

if (is_wp_error($new_product_id)) {
    wp_die($new_product_id->get_error_message());
} elseif ($new_product_id) {
    // Rediriger avec un message de succès
```

```
    wp_redirect(admin_url('post.php?post=' . $new_product_id . '&action=edit'
    exit;
} else {
    wp_die('La création du produit a échoué.');
}
```

Vers une Solution plus Robuste et Universelle

Notre plugin de base fonctionne, mais il a une faiblesse majeure : il dépend de sélecteurs CSS fixes. Pour le rendre vraiment puissant, il faut envisager des approches plus avancées.



Le Problème des Sélecteurs Statiques

Comme nous l'avons vu, si Amazon change la classe CSS de son titre de `.product-title` à `.main-product-title`, notre scraper échoue. Maintenir des sélecteurs pour des dizaines de sites est un travail à plein temps. C'est pourquoi les solutions professionnelles adoptent d'autres stratégies.

Piste 1 : Les API de Web Scraping Commerciales

Des services comme Bright Data, Oxylabs, ou ScraperAPI offrent des API spécialisées. Au lieu de scraper vous-même, vous envoyez l'URL à leur service, et ils vous retournent les données déjà structurées en format JSON. Ces services gèrent pour vous :

- La rotation de millions d'adresses IP pour éviter les blocages.
- La résolution automatique des CAPTCHAs.
- Le rendu JavaScript.
- La maintenance des "parsers" (les logiques d'extraction) pour les sites les plus populaires.

Intégrer une telle API dans notre plugin simplifierait énormément le code de scraping. Il suffirait de faire un appel HTTP à leur service et de traiter la réponse JSON.

Piste 2 : L'Intelligence Artificielle pour l'Extraction de Données

La nouvelle frontière du scraping est l'utilisation de l'IA. Des services avancés, comme ceux mentionnés par Zyte ou Oxylabs avec son "Adaptive AI-based parser", utilisent des modèles d'IA pour "comprendre" une page produit. Au lieu de chercher un sélecteur précis comme `h1.product-title`, l'IA identifie l'élément qui est sémantiquement le titre du produit, même si sa balise ou sa classe change. Cette approche est la plus proche d'une solution véritablement universelle.

Bonnes Pratiques et Considérations Finales

Sécurité et Validation

La sécurité est primordiale dans un plugin WordPress. Assurez-vous de toujours :

- **Valider et nettoyer toutes les entrées :** L'URL fournie par l'utilisateur doit être validée avec `esc_url_raw()`.
- **Nettoyer les sorties :** Les données scrapées avant de les insérer dans la base de données (ex: `sanitize_text_field` pour le nom) et avant de les afficher à l'écran (`esc_html`).
- **Utiliser des nonces :** Comme nous l'avons fait pour protéger nos formulaires.

Performance

Le scraping avec Panther peut être lent et gourmand en ressources, car il lance un véritable navigateur en arrière-plan. Pour un usage intensif, il serait préférable de déporter ces tâches en arrière-plan (via WP-Cron ou une file d'attente) pour ne pas bloquer l'interface utilisateur.

Éthique et Légalité

Le web scraping évolue dans une zone grise juridique. Il est crucial de :

- **Respecter le fichier `robots.txt`** du site cible, qui indique les parties du site que les robots ne sont pas autorisés à visiter.
- **Ne pas surcharger les serveurs** en envoyant trop de requêtes en peu de temps. Introduisez des délais entre vos requêtes.
- **Consulter les conditions d'utilisation** du site cible. Beaucoup interdisent explicitement le scraping.

Ces considérations légales et éthiques sont fondamentales pour éviter des problèmes.

Conclusion

Créer un plugin WordPress pour importer des produits depuis une URL est un projet de développement ambitieux mais réalisable. Ce guide vous a montré comment construire une base fonctionnelle en utilisant PHP, WordPress, et la puissante bibliothèque Symfony Panther pour le scraping.

Vous avez appris que le principal obstacle à une solution "universelle" réside dans la diversité et l'évolution constante des structures des sites e-commerce. La clé du succès est de comprendre que les sélecteurs d'extraction devront être adaptés ou, pour une solution plus robuste, de s'appuyer sur des API de scraping commerciales qui mutualisent cet effort de maintenance et intègrent de plus en plus d'intelligence artificielle.

En suivant les étapes et les bonnes pratiques décrites ici, vous disposez de tous les éléments pour développer un outil d'importation personnalisé et puissant, adapté à vos besoins spécifiques.

Documentation de référence

[1] Optimiser ses données produits pour son activité E-commerce

https://www.ecommerce-nation.fr/optimiser-donnees-produits/

[2] Web Scraping E-commerce Product Data to Build Your Own E ...
https://www.realdataapi.com/ecommerce-web-scraping-building-ecommerce-scraper.php

[3] Guide to Scraping E-commerce Websites - ScrapingBee
https://www.scrapingbee.com/blog/guide-to-scraping-e-commerce-websites/

[4] Panther PHP Web Scraping: Step-By-Step Tutorial - ZenRows
https://www.zenrows.com/blog/panther-web-scraping

[5] How to Scrape E-Commerce Websites: A Complete Guide
https://www.unwrangle.com/blog/web-scraping-ecommerce-websites/

[6] Mastering data extraction from complex websites with web ...
https://zyte.com/blog/data-extraction-from-complex-websites/

[7] Web Scraping With PHP 101 - Scrapfly
https://scrapfly.io/blog/web-scraping-with-php-101/

[8] 8 Best PHP Web Scraping Libraries in 2025 - Medium
https://medium.com/@datajournal/8-best-php-web-scraping-libraries-in-2025-60b9c88336e2

[9] Create Attributes in WooCommerce Programmatically - Webkul
https://webkul.com/blog/create-attributes-woocommerce-programmatically/

[10] Administration Menus - WordPress Codex
https://codex.wordpress.org/Administration_Menus

[11] Creating a Wordpress Plugin Part 1: Adding the Admin Page
https://blog.idrsolutions.com/wordpress-plugin-part-1/

[12] Handle WordPress Form Submissions with Admin-Post & Admin-Ajax
https://wpmudev.com/blog/handling-form-submissions/

[13] admin_post_{$action} – Hook - WordPress Developer Resources
https://developer.wordpress.org/reference/hooks/admin_post_action/

[14] Importation/Exportation de produits au format CSV
https://woocommerce.com/document/importation-exportation-de-produits-au-format-csv/

[15] Create Product Programmatically in WooCommerce
https://rudrastyh.com/woocommerce/create-product-programmatically.html

[16] 7 meilleurs plugins d'importation et d'exportation pour ...
https://www.isitwp.com/fr/best-wordpress-import-export-plugins/

[17] Ajout et gestion de produits Documentation
https://woocommerce.com/document/ajout-et-gestion-de-produits/

[18] Données structurées pour le commerce électronique : votre guide SEO
https://www.clickrank.ai/fr/seo-academy/ecommerce-seo/structured-data-ecom/

[19] symfony/panther: A browser testing and web crawling library for PHP ...
https://github.com/symfony/panther

[20] Web Scraping With PHP - Complete Guide - Bright Data
https://brightdata.com/blog/how-tos/web-scraping-php

[21] add_menu_page() – Function - WordPress Developer Resources
https://developer.wordpress.org/reference/functions/add_menu_page/

[22] media_sideload_image() – Function | Developer.WordPress.org
https://developer.wordpress.org/reference/functions/media_sideload_image/

[23] Top 7 Web Scraping APIs for 2025 - Tested & Reviewed
https://medium.com/@datajournal/best-web-scraping-apis-fbbdcf7b88f4

[24] Best Practices – Plugin Handbook - WordPress Developer Resources
https://developer.wordpress.org/plugins/plugin-basics/best-practices/

[25] What is Web Scraping? The Complete Guide for 2025
https://www.scraperapi.com/web-scraping/