ENI Service - WordPress - PHP

Paul Schuhmacher

Durée : 28h

Novembre 2025

Partie 4 : Les modèles de données de WordPress

Partie 4 : Les modèles de données de WordPress

Objectifs : **Créer ses propres types de données** et développer **ses interfaces pour les administrer**

- Posts, pages, users, taxonomies et **métadonnées**;
- **Custom Post Type**;
- **Custom Taxonomies**;
- Développer ses propres types de données avec le plugin ACF (indispensable);
- **Étendre les interfaces d'administration** pour la gestion des données (**logique métier**);

Les contenus WordPress

> Rappelons-nous le schéma de la base de données WordPress

- **Posts** : posts par défaut (`posts`), pages (`page`), *Custom Post Type* (CPT);
- **Taxonomies** : pour classer et organiser les posts et CPTs (catégories, tags, *Custom Taxonomy* (CT))
- **Users** : les compteurs utilisateurs et leur **roles** (permissions et capacités)
- **Métadonnées** : données supplémentaires sous forme de paires clé/valeur, parfois complexes, attachées aux posts, users ou taxonomies.

Créer ses propres types de post (Custom Post Types)

- Un post = un modèle de données générique (Un produit, un livre, une personne, etc.);
- **Un post + métadonnées** (champs personnalisés ou custom fields) = un modèle de données complet sur mesure (comme une *map*, un objet JSON, un document XML, etc.)

Custom Post Types (CPT)

> Là, WordPress devient un CMS (encore plus) intéressant !

1. **Créer** un CPT avec la fonction `register_post_type()` :

```php
function bp_register_cpt_book() {
  //Labels (affichage dans les écrans d'admin)
  $label = array(
    ...
  );
  //Options du CPT
  $args = array(
    'labels' => $label
    ...
    );
  // Enregistrement du CPT
  register_post_type('book', $args);
}
```

2. **Enregistrer** la fonction créant le CPT sur le hook `init` :

```php
add_action('init', 'bp_register_cpt_book', 0);
```

> Pensez à TOUJOURS préfixer vos fonctions par votre nom *vendor* !

> Architecture : L'enregistrement de CPT est critique au fonctionnement du site. **À placer dans un plugin ou mu-plugin**.

Noms réservés par le *core*

N'utilisez pas un des noms suivants !

```
post
page
attachment
revision
nav_menu_item
custom_css
customize_changeset
oembed_cache
user_request
wp_block
wp_global_styles
wp_navigation
wp_template
wp_template_part
```

Exemple complet

Copier/Coller dans le projet (dans `functions.php` pour le moment) et faisons le tour des options disponibles.

```php
/**
 * Description: Crée un Custom Post Type "Book" pour la bibliothèque.
 * Version: 1.0
 */

/**
 * Fonction pour enregistrer le Custom Post Type "Book"
 */
function bp_register_cpt_book() {

 // Labels affichés dans l'admin (Les générer (Plugin, IA, Site web!).
 $labels = array(
  'name'                  => _x( 'Livres', 'Post Type General Name', 'textdomain' ),
  'singular_name'         => _x( 'Livre', 'Post Type Singular Name', 'textdomain' ),
  'menu_name'             => __( 'Livres', 'textdomain' ),
  'name_admin_bar'        => __( 'Livre', 'textdomain' ),
  'archives'              => __( 'Archives des livres', 'textdomain' ),
  'attributes'            => __( 'Attributs du livre', 'textdomain' ),
  'parent_item_colon'     => __( 'Livre parent :', 'textdomain' ),
  'all_items'             => __( 'Tous les livres', 'textdomain' ),
  'add_new_item'          => __( 'Ajouter un nouveau livre', 'textdomain' ),
  'add_new'               => __( 'Ajouter', 'textdomain' ),
  'new_item'              => __( 'Nouveau livre', 'textdomain' ),
  'edit_item'             => __( 'Éditer le livre', 'textdomain' ),
  'update_item'           => __( 'Mettre à jour le livre', 'textdomain' ),
  'view_item'             => __( 'Voir le livre', 'textdomain' ),
  'view_items'            => __( 'Voir les livres', 'textdomain' ),
  'search_items'          => __( 'Rechercher un livre', 'textdomain' ),
  'not_found'             => __( 'Aucun livre trouvé', 'textdomain' ),
  'not_found_in_trash'    => __( 'Aucun livre trouvé dans la corbeille', 'textdomain' ),
  'featured_image'        => __( 'Image à la une', 'textdomain' ),
  'set_featured_image'    => __( 'Définir l’image à la une', 'textdomain' ),
  'remove_featured_image' => __( 'Supprimer l’image à la une', 'textdomain' ),
  'use_featured_image'    => __( 'Utiliser comme image à la une', 'textdomain' ),
  'insert_into_item'      => __( 'Insérer dans le livre', 'textdomain' ),
  'uploaded_to_this_item' => __( 'Téléversé sur ce livre', 'textdomain' ),
  'items_list'            => __( 'Liste de livres', 'textdomain' ),
  'items_list_navigation' => __( 'Navigation liste de livres', 'textdomain' ),
  'filter_items_list'     => __( 'Filtrer la liste de livres', 'textdomain' ),
 );

 // Options principales du CPT.
 $args = array(
  'label'                 => __( 'Livre', 'textdomain' ),
  'description'           => __( 'Custom Post Type pour les livres', 'textdomain' ),
  'labels'                => $labels,
  'supports'              => array( 'title', 'editor', 'thumbnail', 'excerpt' ), // Important : pour les options d'édition dans la page admin.
  'taxonomies'            => array(),  // Important : Enregistrer les taxonomies à utiliser par le CPT.
  'hierarchical'          => false,    // true pour type parent/enfant (comme pages).
  'public'                => true,
  'show_ui'               => true,
  'show_in_menu'          => true,
  'menu_position'         => 4,
  'menu_icon'             => 'dashicons-book', // icône dans l'interface (important pour l'UI).
  'show_in_admin_bar'     => true,
  'show_in_nav_menus'     => true,
  'register_meta_box_cb' => '',       // Fonction à appeler pour enregistrer auto des MetaBox.
  'can_export'            => true,
  'has_archive'           => true,     // Important ! Permet d'avoir template archive custom (archive-book.php).
  'exclude_from_search'   => false,
  'publicly_queryable'    => true,
  'capability_type'       => 'post',   // Important pour définir les capabilities (attachées aux roles !). Meme que sur posts par défaut.
  'show_in_rest'          => true,     // Pour Gutenberg et autres Page builder (API utilisée par les blocks) et l’API REST.

  'rewrite'               => array(
   'slug' => 'books', // ! Important ! Permet de réécrire le slug dans les urls. Mettre au pluriel.
  ),
  'delete_with_user'      => false,
 );

 // Enregistrement du CPT.
 register_post_type( 'livre', $args );
}
// Hook pour initialiser le CPT après le chargement du core.
add_action( 'init', 'bp_register_cpt_book', 0 );
```

Remarques sur les fonctions __() et _x()

WordPress fournit plusieurs fonctions d'internationalisation pour rendre les chaînes traduisibles dont les fonctions :

- `__( string $text, string $domain = 'default' ): string` : Retourne une chaîne traduite **sans contexte**.
- `_x( string $text, string $context, string $domain = 'default' ): string` : Retourne une chaîne traduite **avec un contexte**

Le **contexte** aide les traducteur·ices à choisir la bonne traduction, quand un même mot peut avoir plusieurs sens selon... le contexte.

```
// Le contexte est Post Type Singular Name, le domaine de mon theme/plugin est 'bp_domain'
_x('Livre', 'Post Type Singular Name', 'bp_domain');
```

> En savoir plus sur l'internationalisation et la localisation dans WordPress

Choisir une *dash-icon*

- Important pour l'UI, la *dash-icon* devrait identifier clairement le type de contenu dans l'admin ;
- **Utiliser** une icone du *dash-icon projet*, **homogénéité** des interfaces d'administration;
- Naturellement intégrée à WordPress, utiliser directement leur *slug*. Par exemple `dashicons-admin-post`.

> Le projet va élargir sa collection avec 36 nouvelles icônes prochainement.

Rewrite

Utiliser `rewrite` :

- `slug` : (*conseillé*) **Réécrire le slug** du *CPT*, notamment le mettre au pluriel pour homogénéiser les noms d'URL (`/books` est une collection) ou le changer pour éviter ambiguïtés/collisions avec d'autres ressources du site. Permet de **découpler le nom du CPT et son URL**.

```
'rewrite' => array(
    'slug'       => 'books',
)
```

> Remarque : Utiliser l'option `rewrite` **ne change pas le nom du posttype** (pour les templates et la *template hierarchy*! Ici, le `posttype` reste bien `book`).

Pratique : Créer nos CPT

1. **Créer les CPT** :
    i. **Book**
        a. Slug : /books
        b. A une archive
    ii. **Editor**
        a. Slug : /editors
        b. A une archive
    iii. **Author**
        a. Slug : /authors
        b. A une archive

> On réécrire les capabilities plus tard pour implémenter nos règles de gestion et notre processus éditorial.

Custom Taxonomies (CT)

Dans la même veine, **créer une CT** avec register_taxonomy().

```php
/**
 * Déclare la taxonomie "Genre" pour le post type "book".
 */
function bp_register_taxonomy_genre() {
 //Labels
 $labels = array(
  ...
 );
 //Options
 $args = array(
  ...
  ),
 );
 //Enregistrer la CT
 register_taxonomy(
  'genre', array('book'), $args
 );
}
//Ajouter l'enregistrement de la CT sur le hook init (comme CPT).
add_action( 'init', 'bp_register_taxonomy_genre' );
```

Exemple

Copier/coller dans le theme (toujours dans `functions.php` pour l'instant) et faisons le tour.

```php
/**
 * Déclare la taxonomie "Genre" pour le post type "book".
 */
function bp_register_taxonomy_genre() {

 $labels = array(
  'name'              => _x( 'Genres', 'taxonomy general name', 'textdomain' ),
  'singular_name'     => _x( 'Genre', 'taxonomy singular name', 'textdomain' ),
  'search_items'      => __( 'Rechercher un genre', 'textdomain' ),
  'all_items'         => __( 'Tous les genres', 'textdomain' ),
  'parent_item'       => __( 'Genre parent', 'textdomain' ),
  'parent_item_colon' => __( 'Genre parent :', 'textdomain' ),
  'edit_item'         => __( 'Modifier le genre', 'textdomain' ),
  'update_item'       => __( 'Mettre à jour le genre', 'textdomain' ),
  'add_new_item'      => __( 'Ajouter un nouveau genre', 'textdomain' ),
  'new_item_name'     => __( 'Nouveau nom de genre', 'textdomain' ),
  'menu_name'         => __( 'Genres', 'textdomain' ),
 );

 $args = array(
  'labels'            => $labels,

  // Taxonomie hiérarchique (true = comme catégories ; false = comme étiquettes).
  'hierarchical'      => true,

  // Permet de gérer la taxonomie dans l'admin.
  'show_ui'           => true,
  'show_admin_column' => true,

  // Disponible dans les requêtes publiques (WP_Query).
  'public'            => true,

  // Réécriture de l'URL.
  'rewrite'           => array(
   'slug'         => 'genre',   // URL de base : /genre/...
   'hierarchical' => true,      // permet /genre/roman/policier/ .
  ),
  // Personnaliser la MetaBox. Ici utilise la même que post_categories_meta_box (avoir meme UI que si hiérarchique, avec checkboxes).
  'meta_box_cb'       => 'post_categories_meta_box',
 );

 register_taxonomy(
  'genre',  // slug de la taxonomie.
  array( 'book' ), // CPT sur lequel elle s'applique.
  $args
 );
}
add_action( 'init', 'bp_register_taxonomy_genre' );
```

Pratique : Créer une CT pour les livres

1. Créer la CT `genre` pour catégoriser les livres
    i. Elle est hiérarchique;
    ii. Créer quelques termes : romans, essais, romans/policier, romans/fantastique, essai/politique, poésie, etc.;
2. Créer la CT `popularity` pour catégoriser les livres
    i. Elle n'est pas hiérarchique
    ii. Créer quelques termes : `classic`, `best-seller`
3. Générer des livres avec FakerPress appartenant à ces catégories (pour avoir du contenu)
4. Tester les différentes URL `/genre/romans`, `/genre/essais`, `/genre/romans/biographies` etc.

> On peut sûrement faire mieux comme catégorisation. Ça vaut le coup d'y réfléchir *davantage* que ce que j'ai fait en rédigeant cette page...

Créer les métadonnées

D'après notre dictionnaire de données, nos modèles doivent disposer de champs personnalisés. Custom Fields.

Pour cela, nous allons utiliser **l'indispensable plugin ACF**

ACF

- **Plugin indispensable** pour utiliser au mieux **les custom fields natifs de WordPres** (simple clés/valeurs, inutilisable en pratique par de admins)
- Développé par DeliciousBrains, qui développe également les excellents plugins WP Migrate et Better Search Replace. WordPress Core contributors. **Vendor de qualité**.
- DeliciousBrains acquis en 2022 par WP Engine (Texas)
- Si le budget le permet, **acheter la version PRO (sans hésiter une seconde)**

> Leur ancien logo

ACF en pratique : enrichir nos modèles (Book, Author, Editor)

Pratique.

Consulter l'excellente documentation du plugin (final user et dev)

Utiliser ACF Local JSON

- La manière recommandée d'utiliser **les champs ACF** est d'enregistrer les champs sous forme de fichier local JSON
- Bénéfices en **termes de performances** :
  - **Moins d'appels à la base de données, voire aucun** !
  - Fichier est chargé une fois, rapidement (cache sur disque) !
- **Versionnement** : versionner l'évolution des champs via Git;
- **Personnalisation** : des filtres ACF vous permettent de choisir où et comment enregistrer les différents champs.

Utiliser un fichier JSON local

Par défaut, ACF chercher un répertoire `acf-json` dans votre thème où sont stockés le ou les fichiers JSON. Ce dossier doit être inscriptible par le serveur web (`755`).

1. **Créer** le répertoire acf-json
2. **Donner les droits** en écriture à l'utilisateur de votre serveur web (`www-data` avec Apache par ex.)

   ```
   chown -R www-data:www-data /path/vers/montheme/acf-json
   ```

3. **Enregistrer** les champs via l'admin. Et *voilà* !

Personnaliser point de sauvegarde et chargement du JSON

```php
<?php

//Personnaliser le point de sauvegarde du JSON.
function enis_acf_json_save_point( $path ) {
    return get_stylesheet_directory() . '/my-custom-folder';
}
add_filter( 'acf/settings/save_json', 'enis_acf_json_save_point' );

//Personnaliser le point de chargement du JSON.
function enis_json_load_point( $paths ) {
    // Remove the original path (optional).
    unset($paths[0]);

    // Append the new path and return it.
    $paths[] = get_stylesheet_directory() . '/my-custom-folder';

    return $paths;
}
add_filter( 'acf/settings/load_json', 'enis_json_load_point' );
```

> Voir toutes les options possibles

Synchronisation du JSON

https://www.advancedcustomfields.com/resources/synchronized-json/

Créer les associations entre nos modèles

- Avec le plugin ACF et les *custom fields*, on peut **reproduire toutes les associations de notre MCD** (many-to-one, many-to-many, reflexive, etc.) :
  - **One-to-one** : un champ Post Object ou Relation pointant vers un unique objet;
  - **One-to-many** : un champ Relationship sur le côté *many*, ou un *Repeater field* (PRO) contenant plusieurs références;
  - **Many-to-many** : deux champs Relationship croisés;
  - **Relations réflexives** : un champ Relationship vers soi-même.

> Voir leur guide sur l’implémentation d'associations bidirectionnelles

Bonnes pratiques avec ACF

- **Utiliser ACF Local JSON**, ne stocker pas vos champs en base;
- Diviser les groupes de champ en groupes de plus petites tailles
  - Améliore les **performances**;
  - Améliore la **maintenabilité** (plus logique, plus facile à maintenir : lire et modifier)
- **Tester** l'existence du champ avant d'y **accéder** :

```php
$value = get_field( "text_field" );

if( $value ) {
    echo wp_kses_post( $value );
} else {
    echo 'empty';
}
```

- Utiliser de manière sécurisée (échappement)

Pratique : Créer les associations entre nos modèles

- Créer les associations entre données (Book-Editor, Author-Book) avec un Custom Field Relationship.

ACF Pages (ACF PRO only)

ACF PRO permet de créer des pages d'administration directement depuis ACF.

Le développement de pages d'admin peut se faire :

- Avec les primitives de WordPress `add_menu_page()` et `add_submenu_page()` (dans un plugin);
- Via la Settings API pour des pages d'options (avec formulaires). Sur mesure, natif;
- Avec ACF Pro Options Page

ACF Pro Options Page

Permet de **créer des pages d'options administrables** avec les possibilités suivantes :

- Création de **pages** et de **sous-pages** ;
- Contrôle de l'**affichage** (UI) : icones, nom des menus, labels ;
- **Contrôle des permissions** (sécurité) :
  - Donner accès à des roles (capabilities associées);
  - Stockage des données (choisir où, par défaut dans la table `options`);

> Gain de temps énorme. Très utile.

ACF Pro Options Page - Workflow

Parcourir la documentation

1. **Créer** une page;
2. **Ajouter** des champs personnalisés;
3. **Intégrer** la page à WordPress (dans un plugin !) :

```php
<?php
add_action('acf/init', function() {
  if( function_exists('acf_add_options_page') ) {
    acf_add_options_page();
  }
});
```

4. **Récupérer** les valeurs, avec les mêmes fonctions (`get_field`, `the_field`, etc.) que pour des champs ACF "classiques" (même API !). Il suffit d'ajouter en 2nd argument l'id de la page :

```php
<?php
//Fournir
$variable = get_field('field_name', 'option');
```

> 'option' est l'id par défaut d'une page d'options. Voir la documentation de acf_add_options_page([$settings]);

Exemple (issu de la doc)

```
add_action('acf/init', function() {
  //Vérifier que la plugin ACF PRO est activé
  if( function_exists('acf_add_options_page') ) {

    //id par défaut 'option'
    acf_add_options_page(array(
        'page_title'    => 'Theme General Settings',
        'menu_title'    => 'Theme Settings',
        'menu_slug'     => 'theme-general-settings',
        'capability'    => 'edit_posts',
        'redirect'      => false
    ));

    acf_add_options_sub_page(array(
        // Définir la page parent via slug
        'parent_slug'   => 'theme-general-settings',
        'page_title'    => 'Theme Header Settings',
        'menu_title'    => 'Header',
    ));

    acf_add_options_sub_page(array(
        'parent_slug'   => 'theme-general-settings',
        'page_title'    => 'Theme Footer Settings',
        'menu_title'    => 'Footer',
    ));

  }
});
```

Alternatives ACF (sans ACF Pro)

Pour les options du site (générales) on peut utiliser la page *Home* pour stocker les custom fields.

Créer le template d'un Livre (single)

MetaBox, enrichir les pages d'admin avec champs personnalisés de formulaire (natif à WordPress)

Ajouter des MetaBox custom sur n'importe quel écran d'admin (Admin site, CPT, CT, Posts, etc.) avec les fonctions `add_meta_box` et `add_meta_boxes`.

> Permet de customiser n'importe quel formulaire d'une page d'admin, **avec ou sans ACF**. C'est cette API qu'utilise ACF !

Exemple : Ajout d'une MetaBox sur le CPT *Book* : rendu et traitement custom

```php
//Ajouter metabox (champ formulaire, ou juste une info en lecture seule)
function bp_book_register_metabox() {
    add_meta_box(
        'book_details',
        'Détails du livre',
        // Callback pour le rendu de la métabox (HTML/CSS/JS)
        'bp_single_book_metabox_render',
        'book',
        'normal',
        'default'
    );
}
add_action('add_meta_boxes', 'bp_book_register_metabox');

// Fonction en charge du rendu de la metabox (markup)
function bp_single_book_metabox_render($post) {
    $author = get_post_meta($post->ID, '_book_author', true);
    // Ajouter un nonce (protection anti CSRF)
    wp_nonce_field('book_details_nonce', 'book_details_nonce_field');
    ?>
    <p>
        <label for="book_author">Auteur :</label><br>
        // Ajouter nos champs customs (input de form)
        <input type="text" id="book_author" name="book_author"
               value="<?php echo esc_attr($author); ?>" style="width:100%;">
    </p>
    <?php
}

// Traiter le formulaire
function bp_book_save_metabox($post_id) {
    // Filtrer sur le champ de la metabox
    if (!isset($_POST['book_details_nonce_field'])) return;
    // Vérifier nonce
    if (!wp_verify_nonce($_POST['book_details_nonce_field'], 'book_details_nonce')) return;
    if (defined('DOING_AUTOSAVE') && DOING_AUTOSAVE) return;
    // Validation
    if (isset($_POST['book_author'])) {
        update_post_meta($post_id, '_book_author', sanitize_text_field($_POST['book_author']));
    }
}
add_action('save_post_book', 'bp_book_save_metabox');
```

Pratique

1. Compléter les templates pour les **archives** (page listing) Editor, Author et Book;
2. Compléter les templates pour les les **single** (page détail) Editor et Author;
3. Dans l'écran d'admin `'Edit' du post type book` (listing), **ajouter une colonne personnalisé** affichant le nombre de copies disponibles à l'emprunt pour chaque livre;
4. Dans l'écran d'admin `'Edit' du post type book` (listing), permettre de **trier les livres suivant le nombre de copies disponibles** (ASC). Pour cela, **rendre la colonne crée précédemment *sortable***.
5. Sur la page d'un livre, ajouter un formulaire pour emprunter le livre s'il est **disponible** et **si l'utilisateur est connecté** (a le role `Abonné`)
6. Grâce au hook approprié, mettre à jour le nombre de copies disponibles.