WordPress avancé 🚀

Ajouter le support multi-langues à votre site :

Pour ajouter le support de plusieurs langues sur votre site (ici Français et Anglais), installez et configurez l'extension Polylang https://fr.wordpress.org/plugins/polylang/

Pensez à traduire vos pages et contenus 😉

Pour intégrer le menu qui permet de passer d'une langue à l'autre vous pouvez vous servir de la méthode pll_the_languages(), pour rappel si vous souhaitez accéder à des données depuis vos templates, vous pouvez le faire à l'intérieur de la méthode add_to_context() dans le fichier functions.php.

Enfin si vous souhaitez obtenir la langue courante de la page que vous consultez, vous pouvez utiliser la méthode pll_current_language().

Documentation de Polylang : https://polylang.pro/doc/function-reference/

WP-CLI : WordPress Command Line Interface

WP-CLI vous permet d'effectuer toutes actions sur votre site WordPress en ligne de commande, sans la nécessité de vous connecter à l'interface d'administration.

Vous pouvez par exemple, lister, créer, modifier, supprimer un utilisateur, une page, un article, un thème, une extension…etc

Installer et tester le, par exemple avec la commande qui permet de lister les posts :

```
wp post list
```

Nous pouvons aussi générer du code avec la commande :

```
wp scaffold
```

Custom Post Types :

Nous allons créer un nouveau type de contenu (post-type) : témoignage à l'aide de la commande :

```
wp scaffold post-type slug —parametre1=xxx —parametre2=xxx
```

On peut lui passer plusieurs paramètres dont :

- **slug** : identifiant unique, je propose **testimony**)
- **label** : l'intitulé de votre contenu dans l'interface d'administration
- **textdomain** : un identifiant qui vous permet de traduire les textes de votre CPT (Custom post type)
- **dashicon** : le choix d'une icône
- **theme** : l'identifiant de votre thème, il génèrera le code dans le thème

Rendez-vous ensuite dans votre thème pour découvrir le code généré et inclure le code dans votre thème : lib/custom-post-types.php

```
require_once('votre-custom-post-type.php');
```

Votre nouveau contenu de type témoignage devrait être accessible dans le back-office.

Block avec un rendu dynamique :

Créez un block destiné à afficher les témoignages, sa méthode save() retournera null, nous gérerons le traitement de l'affichage côté serveur.

Éditez ensuite le fichier **in-block/classes/in_block.php**

Ajouter cette inclusion juste après la déclaration du namespace en remplaçant par **example** par le **nom de votre block** :

```
require_once( plugin_dir_path( __FILE__ ) . "../src/example/render.php" );
```

Ajouter dans la partie __construct()

```
add_action('init', array( $this, 'initRenderCallback' ) );
```

Puis créer la fonction ci-dessous, en remplaçant **example** par le nom de votre block et **renderExample** par **renderVotreBlock** :

```
public function initRenderCallback() {
    register_block_type( 'in-block/example', array(
        'render_callback' => 'renderExample',
    ));
 }
```

Enfin dans le dossier de votre block, créer 2 fichiers : **render.php et render.twig**

render.twig contiendra le rendu de vos témoignages
render.php chargera les témoignages et utilisera la template render.twig

Déclarer dans render.php la méthode **renderExample**() en remplaçant à nouveau **example** par le nom de votre block 😉

C'est ici que nous chargerons nos témoignages et les utiliserons dans render.twig.

Pour charger un custom post type, nous pouvons utiliser WP_Query en lui précisant le **post_type** dans le tableau d'arguments que nous lui passons en paramètre.

Timber nous facilite ici le travail en nous proposant sa propre méthode :

```
$posts = Timber\PostQuery( $args )
```

Les paramètres que nous lui passons sont les mêmes que pour WP_Query.
Documentation : https://developer.wordpress.org/reference/classes/wp_query/