eBook Gratuit

APPRENEZ WordPress

eBook gratuit non affilié créé à partir des **contributeurs de Stack Overflow.**

#wordpress

Table des matières

À propos

You can share this PDF with anyone you feel could benefit from it, downloaded the latest version from: wordpress

It is an unofficial and free WordPress ebook created for educational purposes. All the content is extracted from Stack Overflow Documentation, which is written by many hardworking individuals at Stack Overflow. It is neither affiliated with Stack Overflow nor official WordPress.

Use the content presented in this book at your own risk; it is not guaranteed to be correct nor accurate, please send your feedback and corrections to info@zzzprojects.com

Chapitre 1: Démarrer avec WordPress

Remarques

WordPress est un système de gestion de contenu (CMS) open source utilisé pour créer et gérer des sites Web. WordPress est le système de gestion de contenu le plus populaire sur Internet par pays, alimentant environ la moitié des sites Web de CMS au moment de la rédaction et environ un quart de tous les sites Web sur Internet.

WordPress a commencé sa vie en tant que plateforme de blogging mais a évolué au fil des années pour convenir à la plupart des types de sites Web. L'interface peut être utilisée sans connaissances de codage, ce qui la rend populaire pour les débutants et les développeurs qui souhaitent permettre à leurs clients de gérer leur propre site Web.

Un autre facteur important dans la popularité de WordPress est sa flexibilité, principalement due aux plug-ins et aux systèmes thématiques du noyau. Le système de plug-in facilite l'extension de la fonctionnalité principale sans modifier le code principal. De la même manière, le système de thème facilite la modification de la présentation et de l'esthétique du site Web. Il existe maintenant des milliers de plugins et de thèmes WordPress gratuits et premium disponibles. Beaucoup d'entre eux se trouvent respectivement dans le référentiel de plug-ins wordpress.org et dans le référentiel de thèmes .

WordPress est développé par sa propre communauté, mais est fortement associé à la société Automattic , qui emploie un grand nombre de développeurs WordPress.

Code

WordPress repose sur le langage de script PHP Server et le langage de requête MySQL . WordPress utilise MySQL en tant que banque de données pour le contenu et la configuration des utilisateurs. Le PHP confronte les données de contenu dans une page Web HTML avec tous les atouts nécessaires.

wordpress.com vs wordpress.org

Vous pouvez utiliser WordPress en vous inscrivant au service wordpress.com d' Automattic et en hébergeant votre site Web sur leurs serveurs. Vous pouvez également télécharger le logiciel WordPress sur wordpress.org et héberger votre site Web sur un serveur sous votre contrôle. La première option est simple mais vous ne pouvez modifier aucun code de site. Vous ne pouvez apporter que des modifications via l'interface WordPress. La deuxième option nécessite plus de travail mais vous donne la possibilité de faire ce que vous voulez avec le code de votre site Web. Si vous êtes un utilisateur StackOverflow, vous utiliserez probablement la deuxième option.

Open source

WordPress est un logiciel open source, ce qui signifie qu'il est gratuit et que tout le monde peut consulter le code source et y contribuer. Les contributeurs potentiels peuvent commencer par lire la page Contribution du codex WordPress. .

Les bogues peuvent être signalés en soumettant un bug sur le tracker de billets WordPress .

Documentation

WordPress est officiellement documenté dans le Codex WordPress sur WordPress.org . Les développeurs travaillant avec WordPress seront particulièrement intéressés par la section Developer Codex et la section Developer Reference de wordpress.org .

Versions

Version	Date de sortie
1.0	2004-01-03
1.2	2004-05-22
1,5	2005-02-17
2.0	2005-12-26
2.1	2007-01-22
2.2	2007-05-16
2.3	2007-09-24

Version	Date de sortie
2,5	2008-03-29
2.6	2008-07-15
2.7	2008-12-10
2.8	2009-06-10
2.9	2009-12-18
3.0	2010-06-17
3.1	2011-02-23
3.2	2011-07-04
3.3	2011-12-12
3.4	2012-06-13
3.5	2012-12-11
3.6	2013-08-01
3.7	2013-10-24
3.8	2013-12-12
3,9	2014-04-16
4.0	2014-09-04
4.1	2014-12-17
4.2	2015-04-23
4.3	2015-08-18
4.4	2015-12-08
4.5	2016-04-12
4.6	2016-08-16
4.7	2016-12-06
4.8	2017-06-08

Examples

Introduction à WordPress

WordPress [WP] est un système de gestion de contenu open source permettant de créer des applications, des sites Web et des blogs. WP est écrit en PHP et utilise MySQL comme magasin de données pour le contenu et la configuration de l'utilisateur. Il possède un riche écosystème de plugins et de thèmes et bénéficie d'une communauté open source dynamique, d'une bonne documentation et de faibles barrières à l'entrée. La convivialité et la documentation du développeur peuvent être trouvées dans le Codex WP .

Une partie de WordPress qui la différencie de la plupart des autres produits CMS est sa programmation par événements . Il s'agit d'une méthode de programmation et de représentation logique différente de l'architecture MVC (Model View Controller) utilisée par la plupart des systèmes CMS. WordPress utilise les concepts d'actions et de filtres. Ils forment une file d'attente d'événements permettant aux plugins et aux thèmes d'insérer, de modifier ou même de supprimer des parties de la page d'application finale et / ou des parties. Un concept similaire est la compilation JIT ou Just-In-Time.

Alors qu'historiquement, WordPress était connu comme une plate-forme de blogs, et qu'il ne pourrait jamais perdre cette stigmatisation, la priorité de l'équipe WordPress principale a clairement changé. Avec l' état de la vérité 2016 , par le fondateur Matthew Mullenweg , nous constatons un net changement dans les objectifs, la vision et les efforts. En 2016, nous avons constaté des progrès incroyables lorsque le noyau WordPress a adopté la majorité du très populaire plug-in API REST . Cela était clairement une intention de l'équipe de base dès le début, alors qu'ils entamaient un effort audacieux de création d'un panel d'administrateur JavaScript JavaScript frontal, qui rompt avec la norme d'or que nous connaissons depuis tant d'années. ils l'ont appelé Calpyso .

Thèmes WordPress

Mapper des URL à des modèles spécifiques

Pour bien saisir les thèmes WordPress, vous devez comprendre deux concepts principaux:

1. Lien permanent
2. La hiérarchie des modèles

Un permalien est une URL permanente non changeante (ou un lien vers une ressource spécifique. Par exemple:

- example.com/about-us/ (une page dans WP)
- example.com/services/ (une liste de plusieurs éléments, également appelée "archive" dans le jargon WP)
- example.com/services/we-can-do-that-for-you/ (un élément individuel)

Lorsqu'un utilisateur demande une URL, WordPress effectue une ingénierie inverse du permalien pour déterminer quel modèle doit contrôler sa mise en page. WordPress recherche les différents fichiers de modèles qui *pourraient* contrôler ce contenu particulier et, en fin de compte, donne la préférence aux fichiers les plus spécifiques. C'est ce qu'on appelle la hiérarchie de modèles.

Une fois que WP a trouvé le modèle de vue correspondant dans la hiérarchie, il utilise ce fichier pour traiter et rendre la page.

Par exemple: `index.php` (le modèle par défaut "catch-all") sera remplacé par `archive.php` (le modèle par défaut pour le contenu basé sur des listes), qui sera à son tour remplacé par `archive-services.php` (un modèle fichier spécifique pour l'archive nommée "services").

Voici une excellente référence visuelle pour la hiérarchie des modèles

Structure de base du répertoire thématique

Un thème simple ressemble à ceci:

```
// Theme CSS
style.css

// Custom functionality for your theme
functions.php

// Partials to include in subsequent theme files
header.php
footer.php
sidebar.php
comments.php

// "Archives", (listing views that contain multiple posts)
archive.php
author.php
date.php
taxonomy.php
tag.php
category.php

// Individual content pages
// Note that home and frontpage templates are not recommended
// and they should be replaced by page templates
singular.php
single.php
page.php
front-page.php
home.php

// Misc. Utility Pages
index.php (a catch-all if nothing else matches)
search.php
attachment.php
image.php
404.php
```

Exemple de "Single" (modèle pour un article individuel)

```
<?php get_header(); ?>

<?php if ( have_posts() ) while ( have_posts() ) : the_post(); ?>
    <h1><?php the_title(); ?></h1>
    <?php the_content(); ?>
    <?php comments_template( '', true ); ?>
<?php endwhile; ?>

<?php get_sidebar(); ?>
<?php get_footer(); ?>
```

Qu'est-ce qu'il se passe ici? D'abord, il charge `header.php` (similaire à un include ou require PHP), configure The Loop, affiche `the_title` et `the_content` , puis inclut `comments.php` , `sidebar.php` et `footer.php` . La boucle soulève le problème en configurant un objet `Post` , qui contient toutes les informations relatives au contenu actuellement affiché.

Exemple de "Archive" (modèle pour une liste de plusieurs messages)

```
<?php get_header(); ?>

<?php if ( have_posts() ) while ( have_posts() ) : the_post(); ?>
    <a href="<?php the_permalink(); ?>"<?php the_title(); ?></a>
    <?php the_excerpt(); ?>
<?php endwhile; ?>

<?php
    next_posts_link( 'Older Entries', $the_query->max_num_pages );
    previous_posts_link( 'Newer Entries' );
?>

<?php get_sidebar(); ?>
<?php get_footer(); ?>
```

Tout d'abord, il inclut `header.php` , configure la boucle et inclut `sidebar.php` et `footer.php` . Mais dans ce cas, il y a plusieurs messages dans la boucle, de sorte qu'un extrait est affiché avec un lien vers le message individuel. `next_posts_link` et `previous_posts_link` sont également inclus afin que l'archive puisse paginer les résultats.

Posts, Pages, Types de publication personnalisés et champs personnalisés

WordPress prend en charge deux types de contenu: les `Posts` et les `Pages` . Les publications sont généralement utilisées pour les contenus non hiérarchiques tels que les articles de blog. Les pages sont utilisées pour du contenu statique et autonome, tel qu'une page À propos de nous ou une page de services d'une entreprise avec des sous-pages imbriquées en dessous.

A partir de la version 3.0, les développeurs peuvent définir leurs propres types de publication pour étendre les fonctionnalités de WordPress au-delà des bases. En plus des types de publication personnalisés, vous pouvez également créer vos propres champs personnalisés à associer à vos messages / pages / types de publication personnalisés, ce qui vous permet de structurer l'ajout et l'accès aux métadonnées dans vos modèles. Voir: Champs personnalisés avancés .

Lire Démarrer avec WordPress en ligne: https://riptutorial.com/fr/wordpress/topic/304/demarrer-avec-wordpress

Chapitre 2: Actions et filtres

Syntaxe

- add_action (tag, function_to_call, priority, num_of_args);
- add_filter (tag, function_to_call, priority, num_of_args);

Paramètres

Paramètre	Explication
$ tag	(string) (Obligatoire) Nom de l'action à laquelle la fonction $ est connectée.
fonction $	(callable) (Obligatoire) Requiert une chaîne contenant le nom de la fonction ou la fonction anonyme. Voir des exemples pour l'ajout de fonctions dans les classes.
$ priorité	(int) default = 10. Les fonctions attachées aux hooks / filters s'exécuteront dans la priorité assignée. Vous pouvez avoir une situation où vous voulez travailler avec du code avant toute autre action, définir la priorité = 1 ou après toutes les autres fonctions attachées priorité = 100 etc. Comme avec toutes les fonctions php, vous pouvez utiliser la fonction sans transmettre de valeur à une variable où une valeur par défaut a été définie, mais si vous souhaitez modifier le nombre de paramètres renvoyés, vous devez spécifier!
paramètres $	(int) default = 1. Le nombre de paramètres renvoyés à votre fonction attachée. Les paramètres renvoyés dépendront du numéro attaché où le hook a été créé. Voir `apply_filters()` et `do_action()` pour plus de détails.

Remarques

Crochets Wordpress

Quelque chose qui confond souvent les développeurs lorsqu'ils commencent à travailler avec WordPress est l'utilisation de `apply_filters()` et `add_action()` . Vous verrez souvent des plugins / thèmes utiliser ces derniers dans le code et si vous ne comprenez pas le concept, vous aurez du mal à travailler avec eux.

En bref (très bref, rechercher l'organigramme de chargement de WordPress pour un processus détaillé), WordPress se charge de la manière suivante:

1. wp-load.php - fonctions etc
2. mu-plugins - tous les fichiers trouvés dans le dossier mu-plugins - souvent utilisés pour servir des objets en cache

3. Plugins - aucun ordre particulier, les plugins installés et activés ne seront chargés
4. Thème enfant / parent actif
5. init - reste des données
6. modèle

Si vous êtes développeur et travaillez avec un fichier de fonctions, vous pouvez voir que les deux sont chargés plus tôt dans le processus que les fichiers que vous utilisez. Cela signifie que vous ne pouvez pas modifier les processus (notez que vous ne pouvez pas écraser les fonctions) ou les variables qui s'exécutent plus tard ou qui n'ont pas encore été définies. Les développeurs de thèmes peuvent également placer des crochets dans leur code pour permettre aux plug-ins de se connecter ou les plug-ins peuvent autoriser d'autres plug-ins à remplacer leurs variables. Maintenant, cela peut être déroutant jusqu'à présent, mais restez là.

Pour comprendre `add_filter()` et `add_action()` nous devons examiner comment les hooks sont créés en premier lieu.

```
$arga= 'hello';
do_action('im_a_hook', $arga );
```

Lorsque vous rencontrez ce qui précède dans WordPress, il appelle toutes les fonctions attachées au hook `im_a_hook` (recherchez `$wp_filter` pour plus d'informations sur le processus). Dans votre fonction associée, `$arga` sera disponible pour que la fonction associée fonctionne avec.

```
add_action('im_a_hook', 'attached_function');

function attached_function($arga){
     echo $arga;
}
```

Cela ouvre de nouvelles possibilités puissantes pour modifier les variables à certains points du processus de chargement. Rappelez-vous que nous avons dit plus tôt que les modèles sont chargés après les plugins / thèmes? Un plugin commun est WooCommerce qui crée des écrans plus tard dans le processus, je ne vais pas documenter comment mais un exemple de `do_action` peut être trouvé dans le plugin.

```
do_action( 'woocommerce_after_add_to_cart_button' );
```

Nous avons ici un hook créé qui ne renvoie aucune variable, mais nous pouvons toujours nous amuser avec:

```
add_action( 'woocommerce_after_add_to_cart_button', 'special_offer');

function special_offer(){
    echo '<h1>Special Offer!</h1>;
}
```

L' `add_action` ci-dessus fera `echo` un en-tête d'offre spéciale où `do_action('woocommerce_after_add_to_cart_button')` se trouve lors de la création d'un écran WooCommerce. Nous pouvons donc utiliser ce crochet pour insérer du HTML. D'autres

utilisations peuvent inclure la redirection vers un autre écran, etc.

Plusieurs variables peuvent également être transmises à la fonction. Essayez ceci dans vos fonctions thématiques. Notez le dernier paramètre que nous définissons sur 3, car nous voulons travailler avec les 3 paramètres disponibles. Si nous changions cela en 2, seuls 2 seraient renvoyés et nous aurions une erreur indéfinie.

```
add_action('custom_hook', 'attached_function', 10, 3);

function attached_function($a,$b,$c){

    var_dump($a);
    var_dump($b);
    var_dump($c);

}


$arga = 1;
$argb = 2;
$argc = 3;

do_action('custom_hook', $arga, $argb, $argc);
exit;
```

Il existe un autre type de crochet WP appelé filtre. Un filtre est différent d'une action dans son utilisation, une action ne peut recevoir que des variables, de toute évidence ces variables font partie de la portée des fonctions (vous devez savoir ce qu'est la portée PHP, si ce n'est google). Les filtres renvoient les données renvoyées afin que vous puissiez modifier les variables.

```
$filter_me= apply_filters('im_a_filter', $variable_to_filter);
```

Lorsque vous voyez ce qui précède, vous pouvez modifier la valeur de `$filter_me` car toutes les données que vous retournez seront celles stockées dans la variable. Donc, par exemple (notez que nous `$filter_me` `$variable_to_filter` en `$filter_me` dans l'exemple):

```
add_filter('im_a_filter', 'attached_function', 100);

function attached_function($filter_me){

    $filter_me= 'ray';

    return $filter_me;

}


$filter_me = 'bob';
$filter_me= apply_filters('im_a_filter', $filter_me);
```

La variable `$filter_me` contiendra désormais *«ray»* plutôt que *«bob»* , nous avons défini une priorité de 100, nous sommes donc raisonnablement convaincus que personne ne change la valeur après utilisation (il peut y avoir plusieurs filtres exécutés sur le même crochet). peut

maintenant changer les variables utilisées plus tard dans le processus si `apply_filters()` est présent.

Vous pouvez également transmettre plusieurs paramètres, mais vous ne pouvez modifier que la valeur d'un. Vous devez également retourner une valeur, sinon votre variable ne contiendra rien. Si vous comprenez comment vous utilisez php pour attribuer des valeurs / tableaux / objets à des variables, cela vous semblera évident, par exemple:

```
add_filter('im_a_filter', 'attached_function', 100, 3);

function attached_function($filter_me, $arga, $argb){

    $filter_me= 'ray'.$arga.$argb;

    $arga= 'you fool';

    return $filter_me;

}

$filter_me = 'bob';

$arga = ' middlename';
$argb = ' surname';

$filter_me= apply_filters('im_a_filter', $filter_me, $arga, $argb);
```

La variable `$filter_me` contient maintenant *"ray middlename surname"* . Mais qu'en est-il de `$arga` ? Cela contient toujours *"middlename"* , changer un `$arga` à *"you fool"* dans notre fonction n'a aucun effet sur la valeur définie en dehors de son champ d'application (il y a des manières, Google globals etc.)

add_action ($ hook_name, $ function, $ priority, $ paramètres)

add_filter ($ hook_name, $ function, $ priority, $ paramètres);

Examples

add_action - init

```
add_action('init', 'process_post');

function process_post(){
   if($_POST)
     var_dump($_POST);
}
```

add_action - init - fonction anonyme

```
add_action('init' , function(){
    echo 'i did something';
});
```

add_action - init - dans l'objet de classe

```
class sample{

    public function __construct(){
        add_action('init', array($this, 'samp') );
    }


    public function samp(){ // must be public!!
        echo 'i did something';
    }
}

new sample();
```

add_action - init - dans la classe statique

```
class sample{

    public static function add_action_func(){
        //note __CLASS__ will also include any namespacing
        add_action('init', array(__CLASS__, 'samp') );
    }

    public static function samp(){
        echo 'i did something';
    }

}

sample::add_action_func();
```

Lire Actions et filtres en ligne: https://riptutorial.com/fr/wordpress/topic/2692/actions-et-filtres

Chapitre 3: add_action ()

Syntaxe

- add_action ($ tag, $ function_to_add)
- add_action ($ tag, $ function_to_add, $ priority)
- add_action ($ tag, $ function_to_add, $ priority, $ accepted_args)

Paramètres

Paramètre	Détails
$ tag	*(string)* Nom de l'action à laquelle la procédure `$function_to_add` sera connectée.
$ function_to_add	*(callable)* La fonction / procédure appelable à appeler.
$ priorité	*(int)* Niveau de priorité auquel `$function_to_add` sera exécuté. Optionnel. Par défaut 10.
$ accepted_args	*(int)* Le nombre d'arguments que la fonction appelable `$function_to_add` accepte. Optionnel. Par défaut 1.

Examples

Fonction de rappel direct

```
add_action( 'init', function() {
    // do something here
} );
```

Utiliser un bloc fonction pour capturer un ensemble d'instructions. Avec le hook `init` , le jeu d'instructions sera exécuté juste après que wordpress aura fini de charger les composants nécessaires.

Rappel de référence de nom de fonction

```
function my_init_function() {
    // do something here
}

add_action( 'init', 'my_init_function' );
```

Utiliser le nom de la fonction pour associer un ensemble d'instructions. Avec le hook `init` , le jeu

d'instructions sera exécuté juste après que wordpress aura fini de charger les composants nécessaires.

Rappel de méthode statique de classe

```
class MyClass {
    static function my_init_method() {
        // do something here
    }
}

add_action( 'init', array( 'MyClass', 'my_init_method' ) );
```

Utiliser une méthode statique d'une classe pour capturer un ensemble d'instructions. Avec le hook `init` , le jeu d'instructions sera exécuté juste après que wordpress aura fini de charger les composants nécessaires.

Rappel de méthode objet

```
class MyClass {
    function my_init_method() {
        // do something here
    }
}

$obj = new MyClass();

add_action( 'init', array( $obj, 'my_init_method' ) );
```

Utiliser une méthode d'un objet pour capturer un ensemble d'instructions. Avec le hook `init` , le jeu d'instructions sera exécuté juste après que wordpress aura fini de charger les composants nécessaires.

Lire add_action () en ligne: https://riptutorial.com/fr/wordpress/topic/6264/add-action---

Chapitre 4: add_editor_style ()

Introduction

La fonction permet à l'utilisateur de charger des feuilles de style pour l'éditeur TinyMCE

Syntaxe

- add_editor_style ($ stylesheet)

Paramètres

Paramètre	Détails
$ stylesheet	(array ou string) (Facultatif) Nom de la feuille de style ou tableau correspondant, correspondant à la racine du thème. La valeur par défaut est "editor-style.css"

Examples

Chargement d'un fichier css unique

Code

```
function add_new_style() {
    add_editor_style( 'file-name-here.css' );
}
add_action( 'admin_init', 'add_new_style' );
```

Explication

Dans le code ci-dessus, nous avons utilisé add_editor_style pour charger le fichier css. Nous avons également utilisé add_action pour nous assurer que WordPress exécute notre fonction.

Lire add_editor_style () en ligne: https://riptutorial.com/fr/wordpress/topic/9215/add-editor-style---

Chapitre 5: add_menu_page ()

Introduction

Cette fonction consiste à ajouter un élément dans la barre de navigation du panneau d'administration.

Syntaxe

- add_menu_page ($ page_title, $ menu_title, $ capacite, $ menu_slug, $ function, $ icon_url, $ position)

Paramètres

Paramètre	Détails
$ page_title	(string) Le texte à afficher dans les balises de titre de la page lorsque le menu est sélectionné.
$ menu_title	(string) Le texte à utiliser pour le menu.
capacité $	(string) La capacité requise pour que ce menu soit affiché à l'utilisateur.
$ menu_slug	(string) Le nom du slug à faire référence à ce menu par (devrait être unique pour ce menu).
fonction $	(callable) (facultatif) Fonction à appeler pour afficher le contenu de cette page.
$ icon_url	(chaîne) (facultatif) L'URL de l'icône à utiliser pour ce menu.
position $	(int) (facultatif) La position dans l'ordre du menu devrait apparaître.

Remarques

Voici une liste des positions par défaut (pour la position $)

- 2 - Tableau de bord
- 4 - Séparateur
- 5 - Messages
- 10 - Médias
- 15 - Liens
- 20 - Pages
- 25 - Commentaires
- 59 - Séparateur

- 60 - Apparence
- 65 - Plugins
- 70 - Utilisateurs
- 75 - Outils
- 80 - Paramètres
- 99 - Séparateur

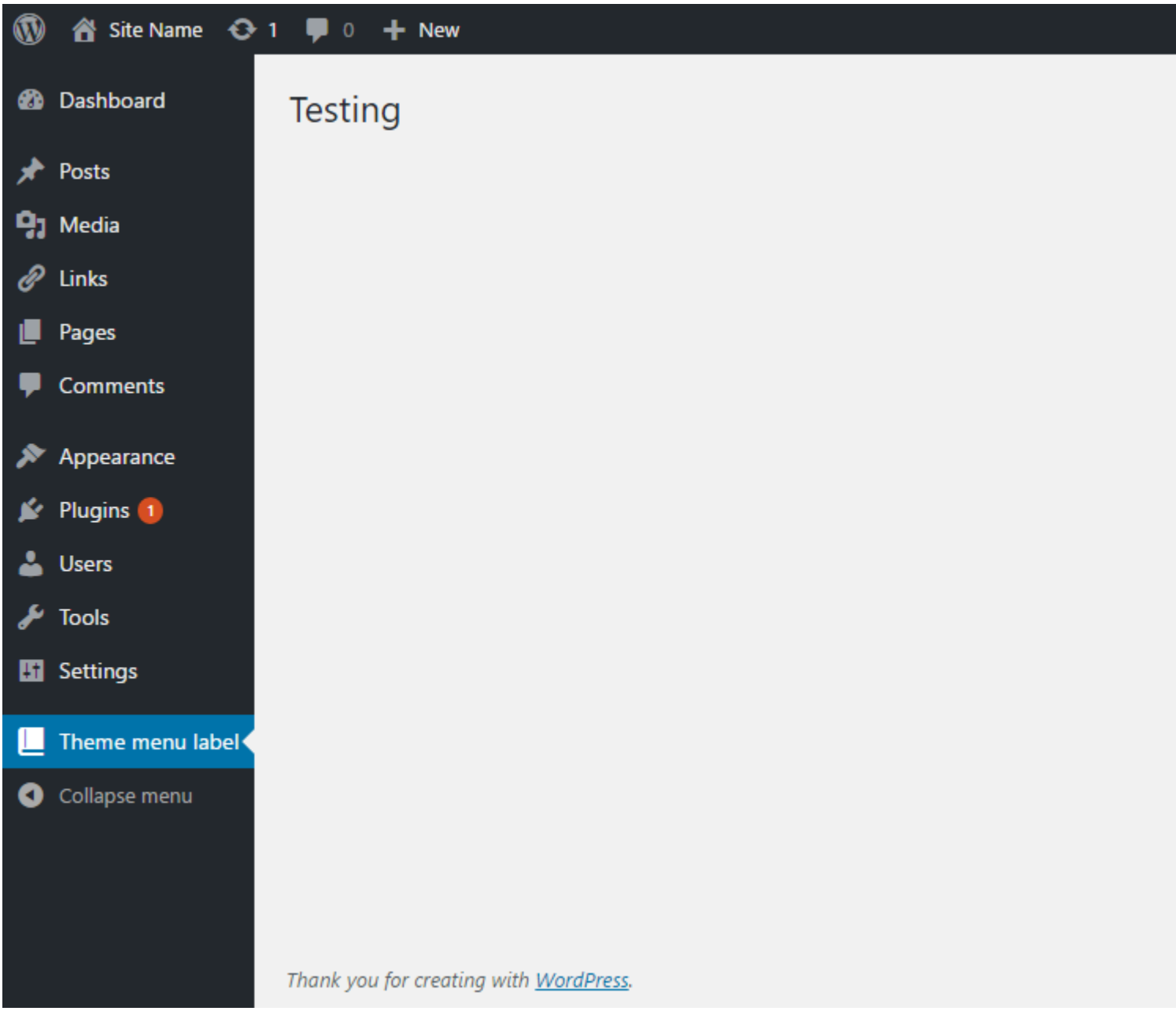
Examples

Ajout de l'élément "Titre de la page de thème" à la barre de navigation

Code

```
function add_the_theme_page(){
    add_menu_page('Theme page title', 'Theme menu label', 'manage_options', 'theme-options',
'page_content', 'dashicons-book-alt');
}
add_action('admin_menu', 'add_the_theme_page');
function page_content(){
    echo '<div class="wrap"><h2>Testing</h2></div>';
}
```

Sortie



Explication

Dans le code, nous avons créé une fonction nommée `add_the_theme_page` et nous avons utilisé `add_menu_page` pour ajouter l'élément à la barre de navigation. Veuillez vérifier la partie paramètres de cette page pour connaître les arguments que nous avons transmis. Nous avons ensuite utilisé `add_action` pour exécuter notre fonction `add_the_theme_page` . Enfin, nous avons créé la fonction `page_content` pour afficher le contenu de la page.

POO et comment charger des scripts / styles sur la page de menu

```
<?php
/*
 *  Plugin Name: Custom Admin Menu
 */

class SO_WP_Menu {
```

```
    private $plugin_url;

    public function __construct() {
        $this->plugin_url = plugins_url( '/', __FILE__ );
        add_action( 'plugins_loaded', array( $this, 'init' ) );
    }

    public function init() {
        add_action( 'admin_menu', array( $this, 'add_menu' ) );
    }

    public function add_menu() {
        $hook = add_menu_page(
            'My Menu',                  // Title, html meta tag
            '<span style="color:#e57300;">My Menu</span>', // Menu title, hardcoded style
            'edit_pages',               // capability
            'dummy-page-slug',          // URL
            array( $this, 'content' ), // output
            null,                       // icon, uses default
            1                           // position, showing on top of all others
        );
        add_action( "admin_print_scripts-$hook", array( $this, 'scripts' ) );
        add_action( "admin_print_styles-$hook", array( $this, 'styles' ) );
    }

    public function content() {
        ?>
            <div id="icon-post" class="icon32"></div>
            <h2>Dummy Page</h2>
            <p> Lorem ipsum</p>
        <?php
    }

    # Printing directly, could be wp_enqueue_script
    public function scripts() {
        ?><script>alert('My page');</script><?php
    }

    # Enqueing from a CSS file on plugin directory
    public function styles() {
        wp_enqueue_style( 'my-menu', $this->plugin_url . 'my-menu.css' );
    }
}

new SO_WP_Menu();
```

Ce qui est important de noter dans cet exemple, c'est que, lorsque vous utilisez `add_menu_page()` , il retourne un hook qui peut être utilisé pour cibler notre page exacte et y charger les styles et les scripts.
Une erreur courante consiste à mettre en file d'attente sans cibler et à renverser les scripts et les styles sur `/wp-admin` .
En utilisant la POO, nous pouvons stocker des variables communes à utiliser parmi les méthodes internes.

Lire add_menu_page () en ligne: https://riptutorial.com/fr/wordpress/topic/9189/add-menu-page---

Chapitre 6: add_submenu_page ()

Introduction

Cette fonction consiste à ajouter un sous-élément à un élément existant dans la barre de navigation des panneaux d'administration.

Syntaxe

- add_submenu_page ($ parent_slug, $ page_title, $ menu_title, $ capacity, $ menu_slug, $ function)

Paramètres

Paramètre	Détails
$ parent_slug	(chaîne) Nom du bloc pour le menu parent (ou le nom de fichier d'une page d'administration WordPress standard).
$ page_title	(string) Le texte à afficher dans les balises de titre de la page lorsque le menu est sélectionné.
$ menu_title	(string) Le texte à utiliser pour le menu.
capacité $	(string) La capacité requise pour que ce menu soit affiché à l'utilisateur.
$ menu_slug	(string) Le nom du slug à faire référence à ce menu par (devrait être unique pour ce menu).
fonction $	(callable) (Facultatif) La fonction à appeler pour générer le contenu de cette page.

Remarques

Voici une liste de slugs pour $ parent_slug

- Tableau de bord: 'index.php'
- Messages: 'edit.php'
- Media: 'upload.php'
- Pages: 'edit.php? Post_type = page'
- Commentaires: 'edit-comments.php'
- Types de messages personnalisés: 'edit.php? Post_type = your_post_type'
- Apparence: 'themes.php'
- Plugins: 'plugins.php'

- Utilisateurs: 'users.php'
- Outils: 'tools.php'
- Paramètres: 'options-general.php'
- Paramètres réseau: 'settings.php'

Examples

Ajout de la page "Sous-menu" en tant que sous-page "Outils" à la barre de navigation

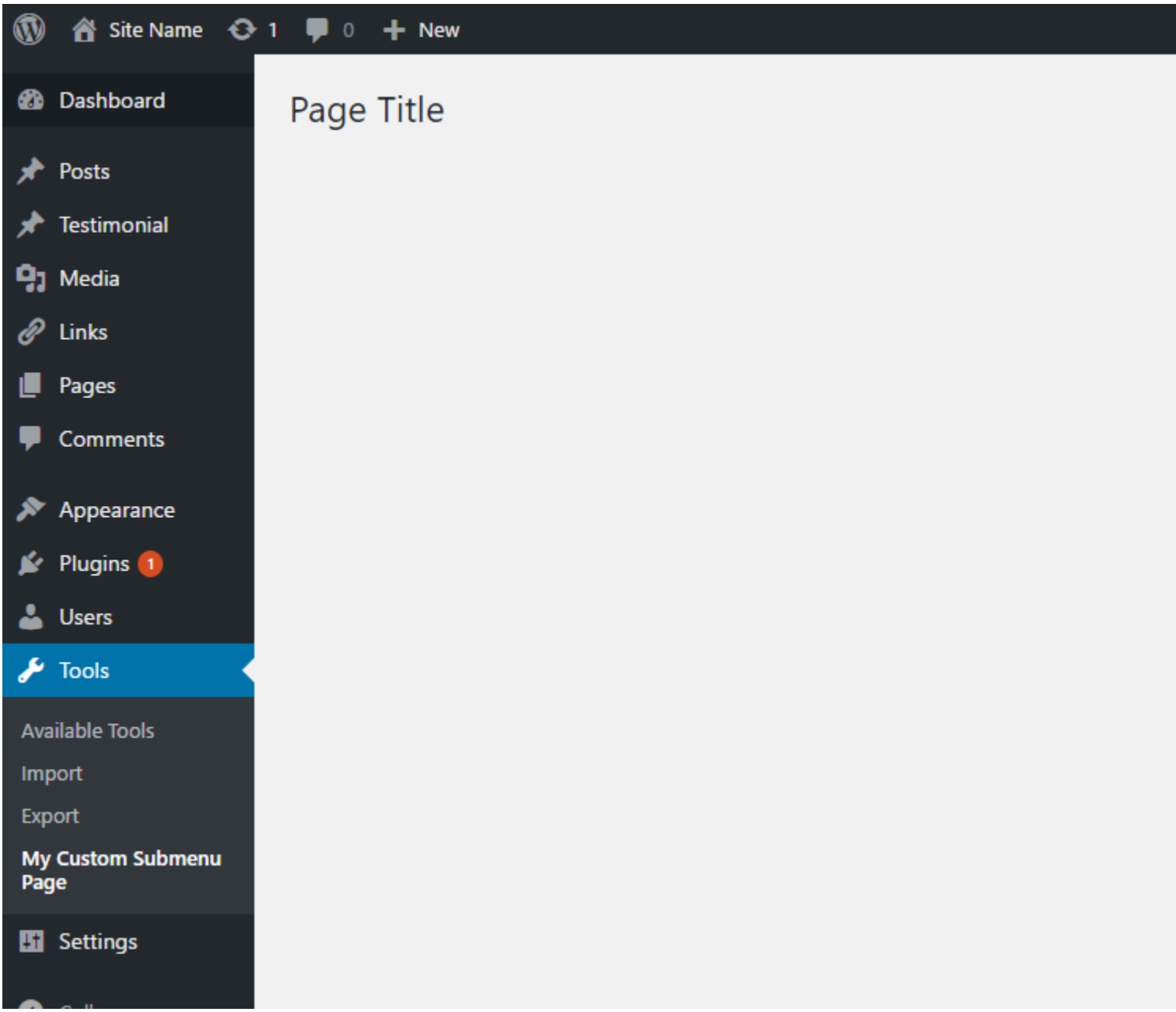
Code

```
add_action('admin_menu', 'register_my_custom_submenu_page');

function register_my_custom_submenu_page() {
    add_submenu_page(
        'tools.php',
        'Submenu Page',
        'My Custom Submenu Page',
        'manage_options',
        'my-custom-submenu-page',
        'my_custom_submenu_page_content' );
}

function my_custom_submenu_page_content() {
    echo '<div class="wrap">';
        echo '<h2>Page Title</h2>';
    echo '</div>';
}
```

Sortie



Explication

Dans le code, nous avons créé une fonction nommée `register_my_custom_submenu_page` et nous avons utilisé `add_submenu_page` pour ajouter l'élément à la barre de navigation en tant qu'enfant de tools.php, qui est la page Outils.

Veuillez vérifier la partie paramètres de cette page pour connaître les arguments que nous avons passés. Nous avons ensuite utilisé `add_action` pour exécuter notre fonction `register_my_custom_submenu_page` . Enfin, nous avons créé la fonction `my_custom_submenu_page_content` pour afficher le contenu de la page.

Lire add_submenu_page () en ligne: https://riptutorial.com/fr/wordpress/topic/9193/add-submenu-page---

Chapitre 7: add_theme_support ()

Introduction

Cette fonction enregistre les fonctionnalités prises en charge par le thème.

Syntaxe

- add_theme_support ($ feature)

Paramètres

Paramètre	Détails
$ fonctionnalité	(chaîne) La fonctionnalité ajoutée.

Remarques

Liste des fonctionnalités à utiliser dans $ feature:

- 'post-formats'
- 'post-vignettes'
- 'html5'
- 'logo personnalisé'
- 'uploads d'en-tête personnalisés'
- 'en-tête personnalisé'
- 'fond personnalisé'
- 'titre-tag'
- «contenu de base»

Examples

Ajout du support de thème pour les formats de publication

```
add_theme_support( 'post-formats', array( 'formatone', 'formattwo' ) );
```

Ajout du support de thème pour les vignettes de publication aux publications

```
add_theme_support( 'post-thumbnails', array( 'post' ) );
```

Le code ci-dessus n'autorise que les messages postaux sur tous les messages. Pour activer la fonctionnalité sur tous les types de publication, procédez comme suit:

```
add_theme_support( 'post-thumbnails' );
```

Lire add_theme_support () en ligne: https://riptutorial.com/fr/wordpress/topic/9216/add-theme-support---

Chapitre 8: AJAX

Examples

Demande AJAX avec une réponse JSON

functions.php:

```
// We add the action twice, once for logged in users and once for non logged in users.
add_action( 'wp_ajax_my_action', 'my_action_callback' );
add_action( 'wp_ajax_nopriv_my_action', 'my_action_callback' );

// Enqueue the script on the front end.
add_action( 'wp_enqueue_scripts', 'enqueue_my_action_script' );
// Enqueue the script on the back end (wp-admin)
add_action( 'admin_enqueue_scripts', 'enqueue_my_action_script' );

function my_action_callback() {
    $json = array();

    if ( isset( $_REQUEST['field2'] ) ) {
        $json['message'] = 'Success!';
        wp_send_json_success( $json );
    } else {
        $json['message'] = 'Field 2 was not set!';
        wp_send_json_error( $json );
    }
}

function enqueue_my_action_script() {
    wp_enqueue_script( 'my-action-script', 'path/to/my-action-script.js', array( 'jquery' ),
null, true );
    wp_localize_script( 'my-action-script', 'my_action_data', array(
        'ajaxurl' => admin_url( 'admin-ajax.php' ),
    ) );
}
```

my-action-script.js:

```
(function( $ ) {
    'use strict';

    $( document ).on( 'ready', function() {
        var data = {
            action: 'my_action',
            field2: 'Hello World',
            field3: 3
        };

        $.getJSON( my_action_data.ajaxurl, data, function( json ) {
            if ( json.success ) {
                alert( 'yes!' );
            } else {
                alert( json.data.message );
            }
```

```
        } );
    } );

})( jQuery );
```

AJAX avec .ajax () et WordPress Nonce

fonctions.php

```
//Localize the AJAX URL and Nonce
add_action('wp_enqueue_scripts', 'example_localize_ajax');
function example_localize_ajax(){
    wp_localize_script('jquery', 'ajax', array(
        'url' => admin_url('admin-ajax.php'),
        'nonce' => wp_create_nonce('example_ajax_nonce'),
    ));
}

//Example AJAX Function
add_action('wp_ajax_example_function', 'example_function');
add_action('wp_ajax_nopriv_example_function', 'example_function');
function example_function(){
    if ( !wp_verify_nonce($_POST['nonce'], 'example_ajax_nonce') ){
        die('Permission Denied.');
    }

    $firstname = sanitize_text_field($_POST['data']['firstname']);
    $lastname = sanitize_text_field($_POST['data']['lastname']);

    //Do something with data here
    echo $firstname . ' ' . $lastname; //Echo for response
    wp_die(); // this is required to terminate immediately and return a proper response:-
https://codex.wordpress.org/AJAX_in_Plugins
}
```

example.js

```
jQuery(document).on('click touch tap', '.example-selector', function(){
    jQuery.ajax({
        type: "POST",
        url: ajax.url,
        data: {
            nonce: ajax.nonce,
            action: 'example_function',
            data: {
                firstname: 'John',
                lastname: 'Doe'
            },
        },
        success: function(response){
            //Success
        },
        error: function(XMLHttpRequest, textStatus, errorThrown){
            //Error
        },
        timeout: 60000
    });
```

```
    return false;
});
```

wp_ajax - Fonctionnalité de base + _wpnonce check

functions.php :

```
function rm_init_js() {
    wp_enqueue_script( 'custom-ajax-script', get_template_directory_uri() . '/js/ajax.js',
array( 'jquery', 'wp-util' ), '1.0', true );
    // pass custom variables to JS
    wp_localize_script( 'custom-ajax-script', 'BEJS', array(
        'action' => 'custom_action',
        'nonce'  => wp_create_nonce( 'test-nonce' )
    ) );
}

add_action( 'wp_enqueue_scripts', 'rm_init_js' );

function rm_ajax_handler() {
    check_ajax_referer( 'test-nonce' );

    extract( $_POST );
    $data = compact( 'first_name', 'last_name', 'email' );

    foreach ( $data as $name => $value ) {
        switch ( $name ) {
            case 'first_name':
            case 'last_name':
                $data[ $name ] = ucfirst( sanitize_user( $value ) );
                break;
            case 'email':
                $data[ $name ] = sanitize_email( $value );
                break;
        }
    }

    $userID = email_exists( $data['email'] );

    if ( ! $userID ) {
        wp_send_json_error( sprintf( __( 'Something went wrong! %s try again!', 'textdomain'
), $data['first_name'] . ' ' . $data['last_name'] ) );
    }

    wp_update_user( array(
        'ID'           => $userID,
        'display_name' => $data['first_name'] . ' ' . $data['last_name'],
        'first_name'   => $data['first_name'],
        'last_name'    => $data['last_name'],
    ) );

    wp_send_json_success( sprintf( __( 'Welcome Back %s', 'textdomain' ), $data['first_name']
. ' ' . $data['last_name'] ) );
}

add_action( 'wp_ajax_custom_action', 'rm_ajax_handler' );
add_action( 'wp_ajax_nopriv_custom_action', 'rm_ajax_handler' );
```

ajax.js

```
;(function() {
    wp.ajax.post(BEJS.action, {
        first_name: 'john',
        last_name: '%65doe',
        email: 'john.doe@example.com',
        _ajax_nonce: BEJS.nonce
    }).done( function( response ) {
        alert(`Success: ${response}`);
    }).fail( function( response ) {
        alert(`Error: ${response}`);
    });
})();
```

OOP ajax soumission utilisant une classe simple avec nonce

Vous pouvez copier et coller tout ce plugin pour l'essayer. Le squelette de classe est utilisé à partir d' ici .

class-oop-ajax.cpp

```
<?php

/**
 * The plugin bootstrap file
 *
 * This file is read by WordPress to generate the plugin information in the plugin
 * Dashboard. This file defines a function that starts the plugin.
 *
 * @wordpress-plugin
 * Plugin Name:         Oop Ajax
 * Plugin URI:          http://
 * Description:         A simple example of using OOP principles to submit a form from the
front end.
 * Version:             1.0.0
 * Author:              Digvijay Naruka
 * Author URI:          http://
 * License:             GPL-2.0+
 * License URI:         http://www.gnu.org/licenses/gpl-2.0.txt
 * Text Domain:         oop-ajax
 * Domain Path:         /languages
 */

// If this file is called directly, abort.
if ( ! defined( 'WPINC' ) ) {
    die;
}

class Oop_Ajax {

  // Put all your add_action, add_shortcode, add_filter functions in __construct()
  // For the callback name, use this: array($this,'<function name>')
```

```
  // <function name> is the name of the function within this class, so need not be globally
unique
  // Some sample commonly used functions are included below
    public function __construct() {

        // Add Javascript and CSS for front-end display
        add_action('wp_enqueue_scripts', array($this,'enqueue'));

        // Add the shortcode for front-end form display
        add_action( 'init', array( $this, 'add_form_shortcode' ) );
        // Add ajax function that will receive the call back for logged in users
        add_action( 'wp_ajax_my_action', array( $this, 'my_action_callback') );
        // Add ajax function that will receive the call back for guest or not logged in users
        add_action( 'wp_ajax_nopriv_my_action', array( $this, 'my_action_callback') );

    }

    // This is an example of enqueuing a JavaScript file and a CSS file for use on the front
end display
    public function enqueue() {
        // Actual enqueues, note the files are in the js and css folders
        // For scripts, make sure you are including the relevant dependencies (jquery in this
case)
        wp_enqueue_script('my-ajax-script', plugins_url('js/oop-ajax.js', __FILE__),
array('jquery'), '1.0', true);

        // Sometimes you want to have access to data on the front end in your Javascript file
        // Getting that requires this call. Always go ahead and include ajaxurl. Any other
variables,
        // add to the array.
        // Then in the Javascript file, you can refer to it like this:
my_php_variables.ajaxurl
        wp_localize_script( 'my-ajax-script', 'my_php_variables', array(
            'ajaxurl' => admin_url('admin-ajax.php'),
            'nonce' => wp_create_nonce('_wpnonce')
        ));

    }

    /**
     * Registers the shortcode for the form.
     */
    public function add_form_shortcode() {

        add_shortcode( "oop-ajax-add-form", array( $this, "add_form_front_end" ) );

    }

    /**
     * Processes shortcode oop-ajax-add-form
     *
     * @param   array    $atts        The attributes from the shortcode
     *
     * @return    mixed    $output        Output of the buffer
     */
    function add_form_front_end($atts, $content) {

        echo "<form id='my_form'>";

            echo "<label for='name'>Name: </label>";
            echo "<input id='name' type='text' name='name' ";
```

```
            echo "<br>" ;

            echo "<label id='email' for='email'>Email: </label>" ;
            echo "<input type='text' name='email'>";

            echo "<br>" ;

            echo "<input type='hidden' name='action' value='my_action' >" ;
            echo "<input id='submit_btn' type='submit' name='submit' value='submit'> ";

        echo "</form><br><br>";
        echo "<div id='response'>ajax responce will be here</div> ";
    }

     /**
     * Callback function for the my_action used in the form.
     *
     * Processses the data recieved from the form, and you can do whatever you want with it.
     *
     * @return    echo   response string about the completion of the ajax call.
     */
    function my_action_callback() {
        // echo wp_die('<pre>' . print_r($_REQUEST) . "<pre>");

        check_ajax_referer( '_wpnonce', 'security');

        if( ! empty( $_POST )){

            if ( isset( $_POST['name'] ) ) {

                $name = sanitize_text_field( $_POST['name'] ) ;
            }

            if( isset( $_POST['email'] ) ) {

                $email = sanitize_text_field( $_POST['email'] ) ;
            }

            ////////////////////////////////////////////
            // do stuff with values
            // example : validate and save in database
            //          process and output
            ////////////////////////////////////////////

            $response = "Wow <strong style= 'color:red'>". $name . "!</style></strong> you
rock, you just made ajax work with oop.";
            //this will send data back to the js function:
            echo $response;

        } else {

            echo "Uh oh! It seems I didn't eat today";
        }

        wp_die(); // required to terminate the call so, otherwise wordpress initiates the
termination and outputs weird '0' at the end.

    }

}
```

```
//initialize our plugin
global $plugin;

// Create an instance of our class to kick off the whole thing
$plugin = new Oop_Ajax();
```

oop-ajax.js

Placez le fichier js dans le répertoire js, à savoir oop-ajax / js / oop-ajax.js

```
(function($) {
    'use strict';

    $("#submit_btn").on('click', function() {
        // set the data
        var data = {
             action: 'my_action',
             security: my_php_variables.nonce,
             name: $("#name").val(),
             email: $("#email").val()
        }

        $.ajax({
            type: 'post',
            url: my_php_variables.ajaxurl,
            data: data,
            success: function(response) {
                //output the response on success
                $("#response").html(response);

            },
            error: function(err) {
                console.log(err);
            }
        });

        return false;
    });
})(jQuery);
```

Lire AJAX en ligne: https://riptutorial.com/fr/wordpress/topic/2335/ajax

Chapitre 9: Ajouter / supprimer des informations de contact pour les utilisateurs avec le hook de filtre user_contactmethods

Examples

Permettre la plupart des réseaux sociaux populaires

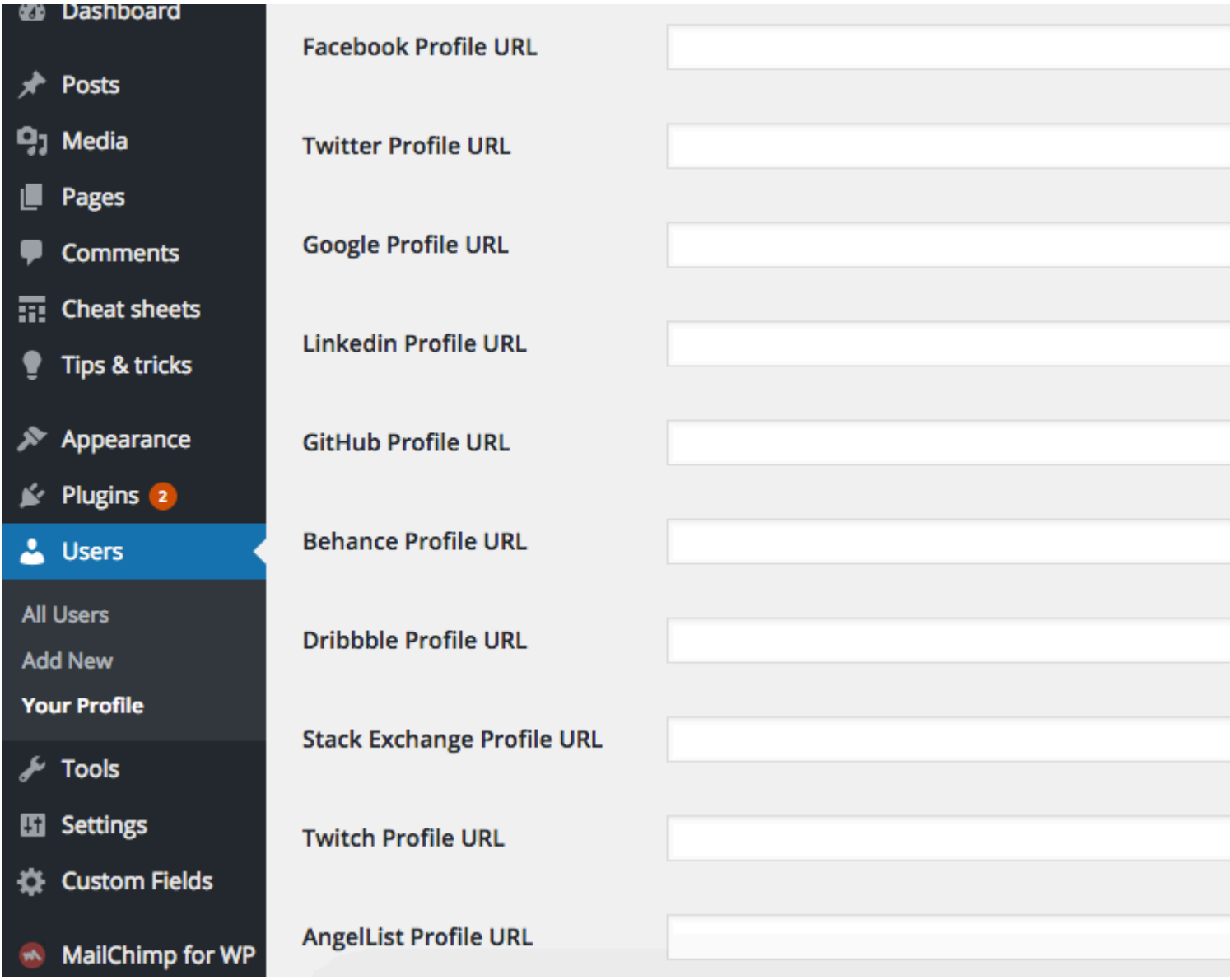
```
function social_profiles( $contactmethods ) {

    $contactmethods['facebook_profile']  = 'Facebook Profile URL';
    $contactmethods['twitter_profile']   = 'Twitter Profile URL';
    $contactmethods['google_profile']    = 'Google Profile URL';
    $contactmethods['linkedin_profile']  = 'Linkedin Profile URL';
    $contactmethods['github_profile']    = 'GitHub Profile URL';
    $contactmethods['behance_profile']   = 'Behance Profile URL';
    $contactmethods['dribbble_profile']  = 'Dribbble Profile URL';
    $contactmethods['stack_profile']     = 'Stack Exchange Profile URL';
    $contactmethods['twitch_profile']    = 'Twitch Profile URL';
    $contactmethods['angellist_profile'] = 'AngelList Profile URL';

    return $contactmethods;
}

add_filter( 'user_contactmethods', 'social_profiles', 10, 1);
```

Vous obtiendrez ce fichier dans votre tableau de bord:



Et voici comment vous le récupérez dans le code

```
<?php $user_stack_exchange = get_the_author_meta( 'stack_profile' ); ?>
```

Supprimer la méthode de contact

```
function remove_contact_methods( $contactmethods ) {

    unset($contactmethods['facebook_profile']);
    unset($contactmethods['twitter_profile']);

    return $contactmethods;
}

add_filter( 'user_contactmethods', 'remove_contact_methods', 10, 1);
```

Lire Ajouter / supprimer des informations de contact pour les utilisateurs avec le hook de filtre user_contactmethods en ligne: https://riptutorial.com/fr/wordpress/topic/2694/ajouter---supprimer-des-informations-de-contact-pour-les-utilisateurs-avec-le-hook-de-filtre-user-contactmethods

Chapitre 10: Ajouter un code court

Syntaxe

- `add_shortcode( $tag , $func );`

Paramètres

Paramètre	Détails
$ tag	*(chaîne) (obligatoire)* Balise de code abrégé à rechercher dans le contenu de l'article
$ func	*(appelable) (obligatoire)* Crochet pour s'exécuter lorsque le shortcode est trouvé

Remarques

- Le callback shortcode sera passé trois arguments: les attributs shortcode, le contenu shortcode (le cas échéant) et le nom du shortcode.
- Il ne peut y avoir qu'un seul crochet pour chaque shortcode. Ce qui signifie que si un autre plugin a un shortcode similaire, il remplacera le vôtre ou le vôtre remplacera le leur selon l'ordre dans lequel les plugins sont inclus et / ou exécutés.
- Les noms d'attributs de shortcode sont toujours convertis en minuscules avant d'être transmis à la fonction de gestionnaire. Les valeurs sont intactes.
- Notez que la fonction appelée par le shortcode ne doit jamais produire de sortie d'aucune sorte. Les fonctions de code abrégé doivent renvoyer le texte à utiliser pour remplacer le shortcode. La production directe de la production entraînera des résultats inattendus. Ceci est similaire à la manière dont les fonctions de filtrage doivent se comporter, en ce sens qu'elles ne doivent pas produire d'effets secondaires attendus de l'appel, car vous ne pouvez pas contrôler quand et où elles sont appelées.

Examples

Shortcode simple pour post récent

`add_shortcode` est le mot clé wp.

```
// recent-posts is going to be our shortcode.
add_shortcode('recent-posts', 'recent_posts_function');

// This function is taking action when recent post shortcode is called.
function recent_posts_function() {
   query_posts(array('orderby' => 'date', 'order' => 'DESC' , 'showposts' => 1));
   if (have_posts()) :
      while (have_posts()) : the_post();
```

```
            $return_string = '<a href="'.get_permalink().'">'.get_the_title().'</a>';
        endwhile;
    endif;
    wp_reset_query();
    return $return_string;
}
```

Cet extrait peut être placé dans votre theme `functions.php` .

`[recent-posts]` Ceci est un shortcode pour le post récent. Nous pouvons appliquer ce shortcode au backend (pages, post, widgets).

Nous pouvons également utiliser le même shortcode dans notre code. avec l'aide de `do_shortcode` .
Par exemple. `echo do_shortcode( '[recent-posts]' );`

Shortcode avancé pour les messages récents

Cette fonction prend le paramètre pour combien de messages récents que vous souhaitez afficher.

Ex: vous ne souhaitez afficher que cinq publications récentes. Juste passé les arguments avec posts = "5" (vous pouvez passer n'importe quel nombre de messages récents que vous souhaitez afficher).

Fonction ne récupère que cinq publications récentes de la base de données.

```
// recent-posts is going to be our shortcode.
add_shortcode('recent-posts', 'recent_posts_function');

// Functions takes parameter such as posts="5".
function recent_posts_function($atts){
    extract(shortcode_atts(array(
        'posts' => 1,
    ), $atts));

    $return_string = '<ul>';
    query_posts(array('orderby' => 'date', 'order' => 'DESC' , 'showposts' => $posts));
    if (have_posts()) :
        while (have_posts()) : the_post();
            $return_string .= '<li><a href="'.get_permalink().'">'.get_the_title().'</a></li>';
        endwhile;
    endif;
    $return_string .= '</ul>';

    wp_reset_query();
    return $return_string;
}
```

Par exemple. `echo do_shortcode( '[recent-posts posts="5"]' );`

Lire Ajouter un code court en ligne: https://riptutorial.com/fr/wordpress/topic/6580/ajouter-un-code-court

Chapitre 11: API REST

Introduction

L'API WordPress REST fournit des points de terminaison API pour les types de données WordPress qui permettent aux développeurs d'interagir avec des sites à distance en envoyant et en recevant des objets JSON (JavaScript Object Notation).

Lorsque vous envoyez du contenu à ou faites une demande à l'API, la réponse sera renvoyée dans JSON. Cela permet aux développeurs de créer, lire et mettre à jour le contenu WordPress à partir de JavaScript côté client ou d'applications externes, même celles écrites dans des langages autres que PHP.

Remarques

Pour que cet exemple simple de WordPress REST API fonctionne pour vous, vous devez apprendre comment cela fonctionne plus en détail. La documentation officielle recommande d'apprendre sur:

1. Routes / points de terminaison - qui sont des mappages de méthodes HTTP individuelles sur des itinéraires appelés "points de terminaison" - vous le faites en utilisant la fonction register_rest_route () , et vous trouverez ici plus d'informations sur les routes et les points de terminaison.

2. Requêtes - L'API WordPress REST définit la classe `WP_REST_Request` qui permet de stocker et d'extraire des informations pour la requête en cours. `WP_REST_Request` objets `WP_REST_Request` sont automatiquement générés à chaque fois que vous effectuez une requête HTTP sur un itinéraire enregistré. Les données spécifiées dans la demande détermineront la réponse que vous récupérerez de l'API. Ici, vous pouvez en apprendre plus sur la classe WP_REST_Request .

3. Réponses - sont les données que vous recevez de l'API. `WP_REST_Response` permet d'interagir avec les données de réponse renvoyées par les points de terminaison. Dans votre définition de noeud final, vous nommez la fonction de rappel (réponse) pour servir votre interaction. Ici, vous en apprendrez plus sur la classe WP_REST_Response .

4. Schéma - Chaque noeud final requiert et fournit des structures de données légèrement différentes, et ces structures sont définies dans le schéma de l'API. Si vous voulez des points de terminaison maintenables, détectables et facilement extensibles, il est recommandé d'utiliser le schéma. Ici vous pouvez en apprendre plus sur le schéma .

5. Classes de contrôleurs - elles rassemblent tous les éléments en un seul endroit. Avec une classe de contrôleur, vous pouvez gérer l'enregistrement des itinéraires et des points de terminaison, gérer les demandes, utiliser un schéma et générer des réponses API. Vous avez déjà `WP_REST_Request` parler de deux classes de contrôleurs: `WP_REST_Request` et `WP_REST_Response` . Ici vous pouvez en apprendre plus sur les classes de contrôleur

Note: Certaines de ces informations proviennent du manuel officiel Wordpress REST APi

Examples

Exemple de travail complet

```
add_action('rest_api_init', 'my_rest_validate_endpoint' );
function my_rest_validate_endpoint() {

    // Declare our namespace
    $namespace = 'myrest/v1';

    // Register the route
    // Example URL matching this route:
    // http://yourdomain/wp-json/myrest/v1/guides/tag=europe/price=29
    register_rest_route($namespace,
                        // Using regular expressions we can initially validate the input
                        '/guides/tag=(?P<tag>[a-zA-Z0-9-]+)/price=(?P<price>[0-9]+)',
                        // We can have multiple endpoints for one route
                        array(
                            array(
                                'methods'   => 'GET',
                                'callback'  => 'my_get_guides_handler'
                            )
                        ),
                        // We can register our schema callback
                        // 'schema' => 'my_get_guide_schema',

        );

    // You can register another route here the same way

}


// The callback handler for the endpoint
function my_get_guides_handler(WP_REST_Request $request) {

    // Get the parameters:
    $tag = $request->get_param('tag');
    $price = $request->get_param('price');

    // Do something with the parameters
    // for instance: get matching guides from the DB into an array - $results
    // ...

    // Prepare the response
    $message = "We've found " . count($results) . " guides ";
    $message .= "(searching for a tag: " . $tag . ", with a price tag: " . $price . ")";

    // The response gets automatically converted into a JSON format
    return new WP_REST_Response(
        array(
            'results' => $results,
            'message' => $message,
            'status' => 'OK'
        ),
        200 );
```

```
}
```

Lire API REST en ligne: https://riptutorial.com/fr/wordpress/topic/10645/api-rest

Chapitre 12: Barres latérales

Syntaxe

- register_sidebar ($ args)
- get_sidebar (string $ name = null)

Paramètres

Paramètre	Détails
$ args	*(string | array) (Facultatif)* Construit une barre latérale basée sur les vvalues `name` et `id`
$ nom	* (string) (Facultatif) Nom de la barre latérale spécialisée. Valeur par défaut: null

Remarques

Les options d'argument sont:

- **name** - **Nom de la** barre latérale *(par défaut: "Sidebar" et ID numérique localisés)* .
- **id** - **Identifiant de la** barre latérale - Doit être tout en minuscule, sans espaces *(par défaut: un ID numérique auto-incrémenté)* . Si vous ne définissez pas la valeur de l'argument id, vous obtiendrez des messages `E_USER_NOTICE` en mode débogage, à partir de la version 4.2.
- **description** - **Description** textuelle de quoi / où la barre latérale est. Montré sur l'écran de gestion des widgets. (Depuis 2.9) *(par défaut: vide)*
- **class** - **Classe** CSS à affecter à la barre latérale de la page d'administration Apparence -> Widget. Cette classe n'apparaîtra que dans la source de la page d'administration de WordPress Widget. Il ne sera pas inclus dans la partie frontale de votre site Web. **Remarque** : La `sidebar` value sera ajoutée à la valeur de la classe. Par exemple, une classe de `tal` entraînera une valeur de classe de `sidebar-tal` . *(par défaut: vide)* .
- **before_widget** - HTML à placer avant chaque widget *(par défaut:* `<li id="%1$s" class="widget %2$s">` *)* **Remarque** : utilise `sprintf` pour la substitution de variable
- **after_widget** - HTML à placer après chaque widget *(par défaut:* `</li>\n` *)* .
- **before_title** - HTML à placer avant chaque titre *(par défaut:* `<h2 class="widgettitle">` *)* .
- **after_title** - HTML à placer après chaque titre *(par défaut:* `</h2>\n` *)* .

Examples

Enregistrement des barres latérales

Dans votre `functions.php` vous pouvez enregistrer de nouvelles barres latérales avec ce code

```
/**
 * Registers sidebars
 *
 * @param array Array with default or specified array values
 * @since       1.0.0
 */
if ( function_exists( 'register_sidebar' ) ) {
    register_sidebar( array (
        'name'          => esc_html__( 'Primary Sidebar', 'mytheme'),
        'id'            => 'primary-widget-area',
        'description'   => esc_html__( 'The Primary Widget Area', 'mytheme'),
        'before_widget' => '<div id="%1$s" class="widget %2$s">',
        'after_widget'  => '</div>',
        'before_title'  => '<div class="sidebar-widget-heading"><h3>',
        'after_title'   => '</h3></div>',
    ) );

    register_sidebar( array (
        'name'          => esc_html__( 'Secondary Sidebar', 'mytheme'),
        'id'            => 'secondary-widget-area',
        'description'   => esc_html__( 'The Secondary Widget Area', 'mytheme'),
        'before_widget' => '<div id="%1$s" class="widget %2$s">',
        'after_widget'  => '</div>',
        'before_title'  => '<div class="sidebar-widget-heading"><h3>',
        'after_title'   => '</h3></div>',
    ) );
}
```

Vous pouvez ajouter autant de barres latérales que vous le souhaitez.

Obtenir la barre latérale

Vous pouvez également créer votre propre fichier de barre latérale dans le thème pour l'appeler sur différents modèles. Copiez et collez sidebar.php du thème en cours et changez le nom (ie sidebar-book.php)

Dans le modèle, vous pouvez appeler cette barre latérale en utilisant `get_sidebar('book')` . En utilisant cela, vous pouvez appeler différentes barres latérales sur différentes pages.

Lire Barres latérales en ligne: https://riptutorial.com/fr/wordpress/topic/6293/barres-laterales

Chapitre 13: Boucle principale en alternance (filtre pre_get_posts)

Syntaxe

- add_action ('pre_get_posts', 'callback_function_name');
- function callback_function_name ($ query) {}
- // pour PHP 5.3.0 ou supérieur
- add_action ('pre_get_posts', fonction ($ query) {});

Paramètres

Paramètre	Détails
$ query	*(WP_Query)* Objet en boucle

Remarques

Si vous utilisez PHP 5.3.0 ou supérieur, vous pouvez utiliser des fermetures (fonctions anonymes)

```
add_action( 'pre_get_posts', function( $query ) {
    if( !$query->is_main_query() || is_admin() ) return;

    // this code will run only if
    // - this query is main query
    // - and this is not admin screen
});
```

Examples

Ciblage en boucle encore plus spécifique

Disons que nous voulons changer *la boucle principale* , uniquement pour une taxonomie spécifique ou un type de publication.

Cibler la boucle principale uniquement sur `book` la page type d'archive post.

```
add_action( 'pre_get_posts', 'my_callback_function' );

function my_callback_function( $query ) {
    if( !$query->is_main_query() || is_admin() ) return;
    if( !is_post_type_archive( 'book' ) ) return;

    // this code will run only if
```

```
    // - this query is main query
    // - and this is not admin screen
    // - and we are on 'book' post type archive page
}
```

Vous pouvez également vérifier la page d'archive de catégorie, de tag ou de taxonomie personnalisée à l'aide de `is_category()` , `is_tag()` et `is_tax()` .

Vous pouvez utiliser n'importe quelle *balise conditionnelle* disponible dans WordPress.

Afficher les messages d'une seule catégorie

```
add_action( 'pre_get_posts', 'single_category' );

function single_category( $query ) {
    if( !$query->is_main_query() || is_admin() ) return;

    $query->set( 'cat', '1' );
    return;
}
```

Pré-publier des messages filtrer l'utilisation de base

Parfois, vous souhaitez modifier la requête WordPress principale.

Le filtre `pre_get_posts` est la voie à suivre.

Par exemple, en utilisant `pre_get_posts` vous pouvez indiquer que *la boucle principale* n'affiche que 5 messages. Ou pour afficher des articles d'une seule catégorie, ou exclure une catégorie, etc.

```
add_action( 'pre_get_posts', 'my_callback_function' );

function my_callback_function( $query ) {
    // here goes logic of your filter
}
```

Comme vous pouvez le voir, nous passons l'objet de requête de *boucle principale* dans notre argument de fonction de rappel.

Note importante ici: **nous passons en argument comme référence** . Cela signifie que nous n'avons pas besoin de renvoyer une requête ou de définir des globales pour la faire fonctionner. Comme `$query` est une référence à l'objet de requête principal, toutes les modifications que nous apportons à notre objet sont immédiatement reflétées dans l'objet de boucle principal.

Exclure la catégorie de la liste de diffusion

```
add_action( 'pre_get_posts', 'single_category_exclude' );

function single_category_exclude( $query ) {
    if( !$query->is_main_query() || is_admin() ) return;
```

```
    $query->set( 'cat', '-1' );
    return;
}
```

Changer posts_per_page pour la boucle principale

Il suffit d'utiliser la méthode `set()` de l'objet `$query` .

Il faut deux arguments, en premier lieu ce que nous voulons définir et en second lieu la valeur à définir.

```
add_action( 'pre_get_posts', 'change_posts_per_page' );

function change_posts_per_page( $query ) {
    if( !$query->is_main_query() || is_admin() ) return;

    $query->set( 'posts_per_page', 5 );
    return;
}
```

Ciblage uniquement de la boucle WordPress principale

WordPress applique le filtre `pre_get_posts` à toute boucle générée. Cela signifie que toutes les modifications apportées à notre fonction de rappel sont appliquées à toutes les boucles existantes.

De toute évidence, ce n'est pas ce que nous voulons dans la plupart des scénarios.

Dans la plupart des cas, nous aimerions cibler uniquement *la boucle principale* et uniquement les écrans non-admin.

Nous pouvons utiliser `is_main_query()` méthode et `is_admin()` fonction globale pour vérifier si nous sommes au bon endroit.

```
add_action( 'pre_get_posts', 'my_callback_function' );

function my_callback_function( $query ) {
    if( !$query->is_main_query() || is_admin() ) return;

    // this code will run only if
    // - this query is main query
    // - and this is not admin screen
}
```

Lire Boucle principale en alternance (filtre pre_get_posts) en ligne:
https://riptutorial.com/fr/wordpress/topic/4418/boucle-principale-en-alternance--filtre-pre-get-posts-

Chapitre 14: Comment puis-je intégrer l'éditeur Markdown à l'add-on de répéteur Advance Custom Field?

Examples

Ajouter un éditeur MarkDown

J'ai trouvé la solution. Veuillez considérer ci-dessous les étapes de mention.

Installez le plugin wp Markdown Editor .

Ensuite, installez " acf-wp-wysiwyg " pour le champ répéteur. Maintenant, vous devez mettre à jour dans certains fichiers. Ouvrez ce fichier et allez au numéro de ligne "180" ou allez à la fonction "create_field"

```
echo '<script> var simplemde = new SimpleMDE({element:
document.getElementById("'.$id.'")});jQuery(".quicktags-
toolbar").css("display","none");</script>';
```

Maintenant, sous le plug-in "acf-repeater", ouvrez le fichier "input.js", numéro de ligne "142"

remplacer

```
new_field_html = this.$el.find('> table > tbody > tr.row-clone').html().replace(/(=["]*[\w-
\[\]]*?)(acfcloneindex)/g, '$1' + new_id),
```

Avec

```
new_field_html = this.$el.find('> table > tbody > tr.row-clone').html().replace(/(["]*[\w-
\[\]]*?)(acfcloneindex)/g, '$1' + new_id),
```

Lire Comment puis-je intégrer l'éditeur Markdown à l'add-on de répéteur Advance Custom Field? en ligne: https://riptutorial.com/fr/wordpress/topic/6602/comment-puis-je-integrer-l-editeur-markdown-a-l-add-on-de-repeteur-advance-custom-field-

Chapitre 15: Création de plugins WordPress

Introduction

Les plugins WordPress doivent se concentrer sur la logique du serveur et / ou les parties admin de votre application Web. Les bons plugins sont comme de bonnes applications, ils font vraiment une chose. Ils sont destinés à améliorer et à automatiser des parties du CMS de manière modulaire, car vous pouvez les activer et les désactiver. Les bons plugins utilisent les actions principales de WordPress, les filtres et les frameworks javascript et css existants.

Examples

Configuration minimale d'un dossier de plugin et de fichiers

La première étape de la création d'un plug-in consiste à créer le dossier et le fichier à partir desquels le plug-in sera chargé.

Les plugins sont situés dans `/wp-content/plugins/` .

Le standard WordPress consiste à créer un dossier et un nom de fichier qui se reflètent comme suit:

```
/wp-content/plugins/myplugin/
/wp-content/plugins/myplugin/myplugin.php
```

Après avoir créé votre fichier de plug-in, vous devez démarrer votre plug-in avec un en- `Plugin Header` in. Cela permet à WordPress d'analyser votre fichier de plug-in et de stocker les métadonnées sur le plug-in, et de permettre aux utilisateurs de l'utiliser et de déterminer s'ils souhaitent que votre plug-in soit actif ou inactif. Copiez ce modèle dans la partie supérieure de votre fichier de plug-in principal que vous avez créé et modifiez-le selon vos besoins:

```
<?php
/**
 * Plugin Name: PLUGIN-NAME
 * Plugin URI: HTTP-LINK-TO-WEBSITE-PLUGIN-PAGE-OR-REPO
 * Description: BREIF DESCRIPTION - KEEP IT SHORT
 * Author: WORDPRESS-DOT-ORG-USERNAME
 * Version: 0.0.1
 * Author URI: HTTP-LINK-TO-MAINTAINER
 * License: GNU General Public License v2 or later
 * License URI: http://www.gnu.org/licenses/gpl-2.0.html
 * Text Domain: short_prefix
 */

// Begin custom PHP WordPress plugin work
```

Notez que les plugins WordPress doivent généralement être sous licence GPL. Les licences ne doivent cependant pas être abordées dans ce sujet.

À ce stade, vous devriez déjà pouvoir voir votre nouveau plug-in dans la zone d'administration de WordPress. Dans une configuration standard, vous trouverez cette zone dans `/wp-admin/plugins.php` . Allez-y et activez votre plugin, et vous êtes prêt à passer aux prochaines étapes de la construction de votre plugin!

Juste pour terminer notre exemple sur quelque chose qui peut être actionné, vous pouvez maintenant ajouter ce qui suit au bas de votre fichier de plugin:

```
die('My custom plugin is loaded : '. __FILE__);
```

Si vous actualisez votre site après cette modification, toutes les pages devraient imprimer ce texte. *Ne faites jamais cela dans des sites de production (live), et n'oubliez jamais de le retirer avant de continuer.*

Lire Création de plugins WordPress en ligne:
https://riptutorial.com/fr/wordpress/topic/9420/creation-de-plugins-wordpress

Chapitre 16: Créer un message par programme

Syntaxe

- wp_insert_post (array $ args, bool $ wp_error);

Paramètres

Paramètre	La description
$ args (Array requis)	Une valeur clé Tableau des éléments ci-dessous.
$ wp_error (Boolean Facultatif)	Renvoie un WP_Error en cas d'échec.

Remarques

Arguments

Le tableau suivant vous montre une liste d'éléments que vous pouvez utiliser dans le premier paramètre (Array).

Paramètre	La description
ID	(Int) L'identifiant de poste. Si elle est égale à autre chose que 0, le message avec cet identifiant sera mis à jour. 0 par défaut
post_author	(Int) ID de l'utilisateur qui a ajouté le message. La valeur par défaut est l'ID utilisateur actuel.
postdater	(String) La date du post. La valeur par défaut est l'heure actuelle.
post_date_gmt	(Chaîne) Date du message dans le fuseau horaire GMT. La valeur par défaut est la valeur de $ post_date.
Publier un contenu	(Mixte) Le contenu de la publication. Par défaut vide
post_content_filtered	(String) Le contenu du message filtré. Par défaut vide
titre de l'article	(String) Le titre du post. Par défaut vide

Paramètre	La description
post_category	(Array) Tableau de valeurs de catégorie post.
post_excerpt (String) L'extrait de poste. Par défaut vide	
post_status	(String) Le statut de la publication. Projet par défaut
Type de poste	(String) Le type de publication. Poste par défaut
comment_status	(Chaîne) Indique si la publication peut accepter des commentaires. Accepte ouvert ou fermé. La valeur par défaut est la valeur de l'option default_comment_status.
ping_status	(Chaîne) Indique si la publication peut accepter des pings. Accepte ouvert ou fermé. La valeur par défaut est la valeur de l'option default_ping_status.
post_password	(String) Le mot de passe pour accéder à la publication. Par défaut vide
après le nom	(Chaîne) Nom de la publication ou slug. Par défaut est le titre de publication assaini lors de la création d'une nouvelle publication.
to_ping	(String) Espace ou liste de liens séparés par un retour chariot à envoyer à ping. Par défaut vide
cinglé	(String) Espace ou liste des adresses séparées par un retour chariot. Par défaut vide
post_modified	(Chaîne) Date à laquelle le message a été modifié pour la dernière fois. La valeur par défaut est l'heure actuelle.
post_modified_gmt	(Chaîne) Date à laquelle le message a été modifié pour la dernière fois dans le fuseau horaire GMT. La valeur par défaut est l'heure actuelle.
post_parent	(Int) Définissez cette option pour le message auquel elle appartient, le cas échéant. 0 par défaut
menu_order	(Int) L'ordre dans lequel le message doit être affiché. Par défaut 0.
post_mime_type	(String) Le type MIME du message. Par défaut vide
guid	(Chaîne) ID unique global pour référencer l'article. Par défaut vide

Paramètre	La description
tax_input	(Array) Tableau de termes de taxonomie indexés par leur nom de taxonomie. Par défaut vide
meta_input	(Array) Tableau de méta-valeurs postées par leur méta-clé post. Par défaut vide

Éviter les messages dupliqués

Lorsque vous exécutez cette fonction, vous pourriez probablement obtenir un message en double, du moins cela m'est arrivé. (Vous pouvez le vérifier dans la section Post WordPress)

J'ai trouvé une solution :

```
if( !get_page_by_title( $title, 'OBJECT', 'post' ) ){
    $my_post = array('post_title' => $title,
        'post_content' => 'Content',
        'tags_input' => $tags,
        'post_category' => array(2),
        'post_status' => 'publish'
    );

    $result = wp_insert_post( $my_post );
}
```

Explication

Avant d'enregistrer un nouveau message, validez si le nouvel article existe déjà en utilisant le titre du post comme paramètre, s'il n'y a pas de titre de publication, vous pouvez enregistrer votre nouvelle publication.

Vérifiez la documentation de get_page_by_title ici .

Examples

introduction

Parfois, nous avons un autre éditeur ailleurs que TinyMCE (Wordpress Default Editor). Cela se produit lorsque nous créons notre propre thème, plug-in ou quelque chose de spécifique; et nous devons écrire et manipuler un type de message et l'enregistrer dans notre base de données WP.

Donc, si vous êtes dans cette situation, vous pouvez utiliser une fonction Wordpress appelée:

```
wp_insert_post( array $args, bool $wp_error );
```

Créer un message de base

```
$basic_post_args = array(
    'post_title' => 'My Basic Post',
    'post_content' => 'This is a basic content',
    'post_status' => 'publish',
    'post_author' => 1,
    'post_category' => array(8, 59)
);

wp_insert_post( $basic_post_args );
```

Créer une page de base

```
$basic_page_args = array(
    'post_title' => 'My Basic Page',
    'post_content' => 'This is a basic content',
    'post_type' => 'page',
    'post_status' => 'publish',
    'post_author' => 1
);

wp_insert_post( $basic_page_args );
```

Lire Créer un message par programme en ligne:
https://riptutorial.com/fr/wordpress/topic/5860/creer-un-message-par-programme

Chapitre 17: Créer un modèle personnalisé

Examples

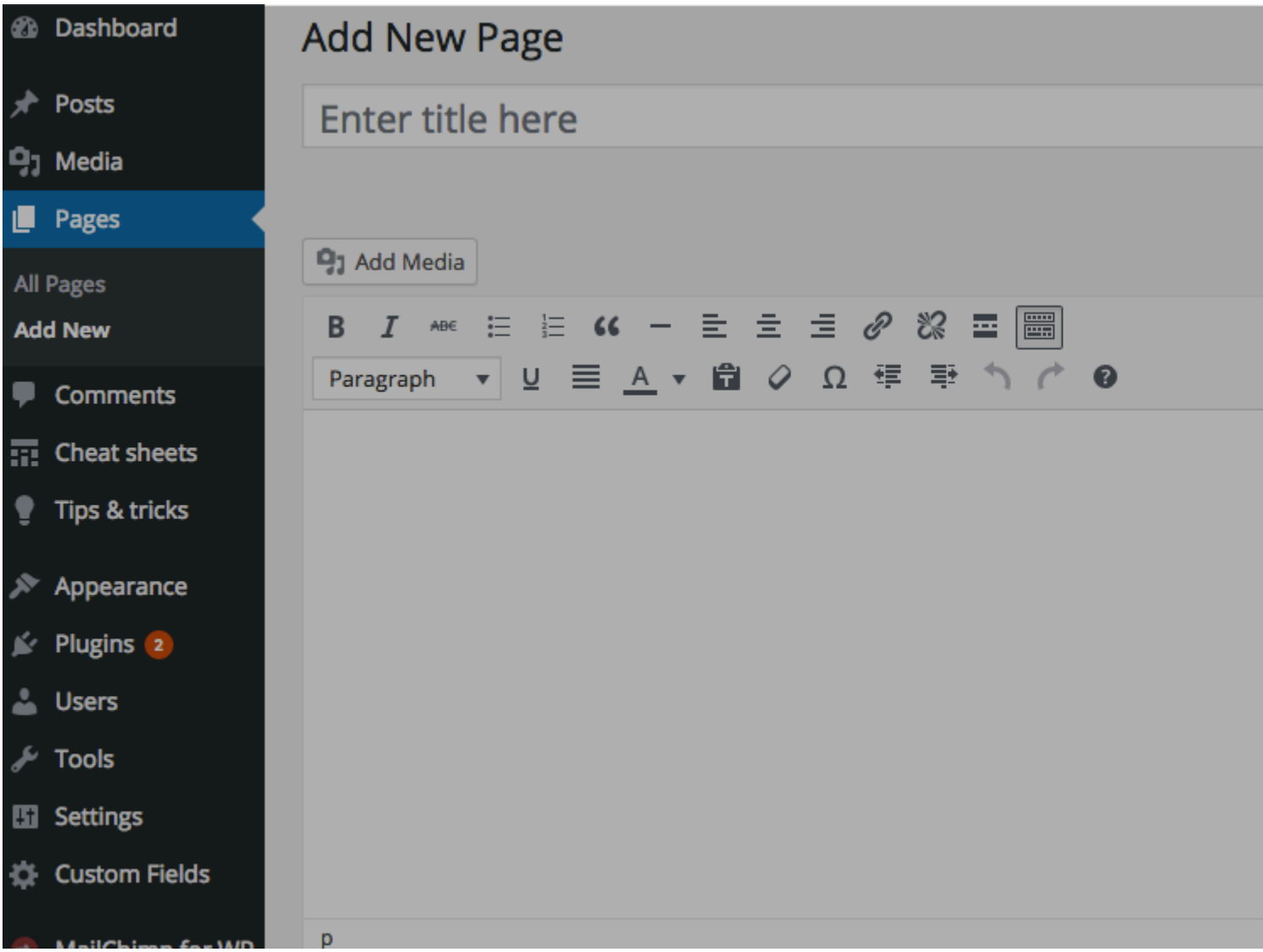
Création d'un modèle vierge de base

Pour créer un modèle personnalisé, nous devons d'abord créer un fichier php dans un répertoire de thème. Vous pouvez le nommer presque comme vous le souhaitez. Pour cet exemple, nous allons créer **example.php**

Une seule chose à définir dans notre exemple.php, à reconnaître par WordPress comme modèle, est le nom du modèle. Nous faisons cela acheter des commentaires spéciaux en haut d'un fichier, comme ceci:

```
<?php
/*
Template Name: Example
*/
?>
```

Et maintenant, lorsque nous devrions voir notre modèle répertorié dans la **liste déroulante Modèle** dans la zone **Attributs de page**



Y compris en-tête et pied de page dans notre modèle

Étendons notre modèle ci-dessus et incluons le contenu de **header.php** et **footer.php**

Y compris en-tête:

Nous allons inclure l'en-tête juste après le **commentaire du nom du modèle**

Il existe deux façons courantes de le faire. Les deux ont raison et fonctionnent de la même manière, il s'agit juste de votre style et de l'apparence du code

Premier moyen:

```
<?php
/*
Template Name: Example
*/
get_header();
?>
```

Deuxième voie:

```
<?php
/*
Template Name: Example
*/
?>
<?php get_header(); ?>
```

Y compris le pied de page:

Inclure le pied de page fonctionne de la même manière, il n'y a qu'une seule chose à laquelle nous devons nous intéresser, à savoir que nous incluons le pied de page après avoir inclus l'en-tête. Le modèle final devrait donc ressembler à ceci.

```
<?php
/*
Template Name: Example
*/
get_header();
?>

<?php get_footer(); ?>
```

Modèle personnalisé avec contenu

Nous allons étendre notre modèle et inclure le titre de la page et un contenu

```
<?php
/*
Template Name: Example
*/
get_header();

the_title();
the_content();

get_footer();
```

Et si vous voulez, vous pouvez les envelopper avec des éléments HTML comme celui-ci

```
<?php
/*
Template Name: Example
*/
get_header();

echo '<h1>' . the_title() . '</h1>';
echo '<section> . 'the_content() . '</section>';

get_footer();
```

Ou si vous préférez travailler comme un fichier HTML normal, sans utiliser d'écho

```
<?php
/*
```

```
Template Name: Example
*/
get_header();
?>

<h1><?php the_title(); ?></h1>
<section><?php the_content(); ?></section>

<?php get_footer(); ?>
```

Lire Créer un modèle personnalisé en ligne: https://riptutorial.com/fr/wordpress/topic/2791/creer-un-modele-personnalise

Chapitre 18: Créer un modèle pour un type de message personnalisé

Examples

Création d'un modèle personnalisé pour le livre de type Custom Post

Pour créer un modèle pour les publications uniques de notre type de publication personnalisé, nous devons créer un fichier appelé **nom_type_post unique** .php où **nom_type_post** est le nom de notre type de publication personnalisé.

Par exemple, si notre type de publication personnalisé s'appelle «Livres», nous devons créer un fichier PHP appelé .php à **livre** unique. Notez que nous avons utilisé le nom singulier de notre type de poste personnalisé.

Copiez le contenu du fichier single.php du dossier des thèmes et collez-le dans le nouveau modèle et enregistrez-le, puis le modèle sera appliqué à la page individuelle du type de publication personnalisée.

Modèles de type de message personnalisé

Archive de type de message personnalisé:

Pour créer un modèle d'archive pour un type de publication personnalisé, vous devez définir l'argument `has_archive` égal à `true` dans votre fonction `register_post_type()` . Dans l'exemple ci-dessous, un type de publication personnalisé est créé pour un type de publication d'événement.

```
add_action( 'init', 'create_events_post_type' );
function create_events_post_type() {
  register_post_type( 'event',
    array(
      'labels' => array(
        'name' => __( 'Events' ),
        'singular_name' => __( 'Event' )
      ),
      'public' => true,
      'has_archive' => true,
    )
  );
}
```

Pour créer un modèle pour les nouveaux types de publication personnalisés, vous devrez créer un nouveau fichier de modèle. Pour créer un modèle pour les pages de publication uniques, vous devez le nommer `single-{post_type}.php` et `archive-{post_type}.php` pour l'archive.

Le nom de fichier de notre modèle d'archive sera `archive-event.php` et pour la page de l'événement, il `single-event.php` . Les deux fichiers doivent être dans le répertoire racine de vos thèmes.

Un exemple de modèle d'archive ressemblerait à ceci. Tiré du thème des vingt ans .

```
<?php
/**
 * The template for displaying archive pages
 *
 * @link https://codex.wordpress.org/Template_Hierarchy
 *
 * @package WordPress
 * @subpackage Twenty_Seventeen
 * @since 1.0
 * @version 1.0
 */

get_header(); ?>

<div class="wrap">

    <?php if ( have_posts() ) : ?>
        <header class="page-header">
            <?php
                the_archive_title( '<h1 class="page-title">', '</h1>' );
                the_archive_description( '<div class="taxonomy-description">', '</div>' );
            ?>
        </header><!-- .page-header -->
    <?php endif; ?>

    <div id="primary" class="content-area">
        <main id="main" class="site-main" role="main">

        <?php
        if ( have_posts() ) : ?>
            <?php
            /* Start the Loop */
            while ( have_posts() ) : the_post();

                /*
                 * Include the Post-Format-specific template for the content.
                 * If you want to override this in a child theme, then include a file
                 * called content-___.php (where ___ is the Post Format name) and that will be
used instead.
                 */
                get_template_part( 'template-parts/post/content', get_post_format() );

            endwhile;

            the_posts_pagination( array(
                'prev_text' => twentyseventeen_get_svg( array( 'icon' => 'arrow-left' ) ) .
'<span class="screen-reader-text">' . __( 'Previous page', 'twentyseventeen' ) . '</span>',
                'next_text' => '<span class="screen-reader-text">' . __( 'Next page',
'twentyseventeen' ) . '</span>' . twentyseventeen_get_svg( array( 'icon' => 'arrow-right' ) ),
                'before_page_number' => '<span class="meta-nav screen-reader-text">' . __(
'Page', 'twentyseventeen' ) . ' </span>',
            ) );

        else :
```

```
            get_template_part( 'template-parts/post/content', 'none' );

        endif; ?>

        </main><!-- #main -->
    </div><!-- #primary -->
    <?php get_sidebar(); ?>
</div><!-- .wrap -->

<?php get_footer();
```

Type de poste personnalisé Modèle unique:

Voici un exemple d'un modèle unique. Tiré du thème des vingt ans .

```
<?php
/**
 * The template for displaying all single posts
 *
 * @link https://developer.wordpress.org/themes/basics/template-hierarchy/#single-post
 *
 * @package WordPress
 * @subpackage Twenty_Seventeen
 * @since 1.0
 * @version 1.0
 */

get_header(); ?>

<div class="wrap">
    <div id="primary" class="content-area">
        <main id="main" class="site-main" role="main">

            <?php
                /* Start the Loop */
                while ( have_posts() ) : the_post();

                    get_template_part( 'template-parts/post/content', get_post_format() );

                    // If comments are open or we have at least one comment, load up the
comment template.
                    if ( comments_open() || get_comments_number() ) :
                        comments_template();
                    endif;

                    the_post_navigation( array(
                        'prev_text' => '<span class="screen-reader-text">' . __( 'Previous
Post', 'twentyseventeen' ) . '</span><span aria-hidden="true" class="nav-subtitle">' . __(
'Previous', 'twentyseventeen' ) . '</span> <span class="nav-title"><span class="nav-title-
icon-wrapper">' . twentyseventeen_get_svg( array( 'icon' => 'arrow-left' ) ) .
'</span>%title</span>',
                        'next_text' => '<span class="screen-reader-text">' . __( 'Next Post',
'twentyseventeen' ) . '</span><span aria-hidden="true" class="nav-subtitle">' . __( 'Next',
'twentyseventeen' ) . '</span> <span class="nav-title">%title<span class="nav-title-icon-
wrapper">' . twentyseventeen_get_svg( array( 'icon' => 'arrow-right' ) ) . '</span></span>',
                    ) );
```

```
                endwhile; // End of the loop.
            ?>

        </main><!-- #main -->
    </div><!-- #primary -->
    <?php get_sidebar(); ?>
</div><!-- .wrap -->

<?php get_footer();
```

Les deux exemples de modèles tirent des partiels pour afficher le contenu interne.

Si votre thème enfant / parent a un modèle unique / archive, vous devez utiliser ce code comme modèle pour vos nouveaux modèles.

Lire Créer un modèle pour un type de message personnalisé en ligne: https://riptutorial.com/fr/wordpress/topic/6390/creer-un-modele-pour-un-type-de-message-personnalise

Chapitre 19: Customizer Bonjour tout le monde

Paramètres

Paramètre	Détails
mon thème	Un identifiant unique pour votre thème (ou thème enfant). Cela peut être votre slug de thème

Examples

Bonjour Monde Exemple

Le concept fondamental du personnaliseur est que les administrateurs peuvent prévisualiser en direct les modifications apportées à leur site, puis les ajouter définitivement.

Les éléments suivants peuvent être copiés et collés dans le fichier `functions.php` un thème.

- Ajouter une section de personnalisation appelée `My First Section`
- Ajouter un paramètre de personnalisation appelé `Hello World Color` permettant à l'administrateur de choisir une couleur
- Ajoutez une règle de CSS pour `.hello-world` qui corresponde à la couleur choisie et par défaut à `#000000` si rien n'est choisi. Le paramètre sera placé dans une `<style>` à la fin du `<head>` .

```
function mytheme_customize_register( $wp_customize ) {

    $wp_customize->add_section( 'my_first_section_id' , array(
        'title'      => __( 'My First Section', 'mytheme' ),
        'priority'   => 30,
    ) );

    $wp_customize->add_setting( 'hello_world_color' , array(
        'default'     => '#000000',
        'transport'   => 'refresh',
    ) );

    $wp_customize->add_control( new WP_Customize_Color_Control( $wp_customize, 'link_color',
array(
        'label'        => __( 'Hello World Color', 'mytheme' ),
        'section'    => 'my_first_section_id',
        'settings'   => 'hello_world_color',
    ) ) );


}
```

```
add_action( 'customize_register', 'mytheme_customize_register' );


function mytheme_customize_css()
{
    ?>
    <style type="text/css">
        .hello-world { color: #<?php echo get_theme_mod('hello_world_color', '000000'); ?>; }
    </style>
    <?php
}
add_action( 'wp_head', 'mytheme_customize_css');
```

Lire Customizer Bonjour tout le monde en ligne:
https://riptutorial.com/fr/wordpress/topic/2875/customizer-bonjour-tout-le-monde

Chapitre 20: Des thèmes

Introduction

Les thèmes WordPress sont le front-end de votre site Web. Ils sont ce que les gens voient quand ils visitent le site. Il existe des milliers de thèmes à choisir, des versions payantes et gratuites. Vous pouvez même créer votre propre thème personnalisé avec juste quelques fichiers nécessaires.

Examples

Thèmes WordPress

Comment choisir un thème

Chaque installation WordPress est livrée avec un thème pré-installé. Vous gérez vos thèmes à partir du tableau de bord. Accédez à Apparence> Thèmes pour installer, prévisualiser, supprimer, activer et mettre à jour des thèmes. Le thème actuel se trouve dans le coin supérieur gauche de ce menu.

Survolez l'image du thème pour afficher un bouton "Détails du thème". Ce bouton fournit des informations sur le thème, telles que la version et la description. En cliquant sur l'image du thème actuel, vous accédez à des paramètres de thème spécifiques, tels que le titre.

Mise à jour disponible

Si des mises à jour sont disponibles pour les thèmes installés, vous trouverez un message vous informant qu'une nouvelle version est disponible. Vous devriez pouvoir voir les détails de la nouvelle version ou mettre à jour maintenant.

- Afficher les détails de la version

  En cliquant sur le lien de détails de la version, vous accédez à une page du répertoire des thèmes WordPress. Vous trouverez ici les détails de la version de mise à niveau.

- Mettre à jour maintenant

  Cliquez sur le lien Mettre à jour maintenant pour installer la mise à niveau du thème. Les thèmes peuvent également être mis à niveau à partir de l'écran Administration> Tableau de bord> Mises à jour.

En plus de votre thème actuel, l'écran Gérer les thèmes affiche également les autres thèmes installés mais actuellement inactifs. Chaque thème est représenté par une petite capture d'écran. En survolant ces images, les boutons "Détails du thème", "Activer" et "Aperçu en direct" s'affichent. Vous pourrez également mettre à niveau ou supprimer des thèmes inactifs à partir de

cette page. Chaque page de cet écran affiche jusqu'à 15 captures d'écran de thème à la fois.

- Activer

  Cliquer sur ce lien en fait le thème actuel.

- Aperçu en direct

  En cliquant sur ce lien, vous affichez un aperçu du blog avec cette version de thème spécifique.

- Effacer

  En cliquant sur ce lien, vous supprimez complètement ce thème, incluez tous les fichiers et dossiers de thème. Tout ce qui n'est pas sauvegardé sera perdu pour toujours.

- Mise à jour disponible

  Reportez-vous à la section Mise à jour disponible ci-dessus.

Installer des thèmes

Vous trouverez ci-dessous plusieurs façons d’installer des thèmes:

- Installateur de thème automatisé

  Cela peut être utilisé pour installer des thèmes à partir du répertoire de thèmes WordPress. Accédez à Administration> Apparence> Thèmes pour accéder à l'écran Thèmes d'apparence. Cliquez sur le bouton Ajouter un nouveau. De là, vous trouverez des thèmes à utiliser sans changement. En haut de cet écran, il y a une fonction de recherche avec trois méthodes disponibles pour trouver un nouveau thème. Recherche par filtre, mot-clé et attribut.

- Utilisation de la méthode de téléchargement

  La méthode de téléchargement installe un thème via un fichier ZIP. Tous les thèmes du répertoire de thèmes WordPress peuvent être installés de cette manière. Après avoir téléchargé le fichier ZIP, accédez à Administration> Apparence> Thèmes, puis cliquez sur le bouton Ajouter. Cliquez ensuite sur le lien Télécharger le thème. Recherchez le fichier ZIP et cliquez sur Installer maintenant. Pour terminer en faisant le thème actuel, cliquez sur le lien Activer.

- Utiliser la méthode FTP

  Pour installer un thème avec la méthode FTP, vous devez d'abord télécharger les fichiers de thème sur votre ordinateur local. Extrayez le contenu du fichier ZIP en préservant la structure du fichier et ajoutez-le à un nouveau dossier. S'il y a des instructions de l'auteur du thème, assurez-vous de les suivre.

  Utilisez un client FTP pour accéder au serveur Web de votre site. Ajoutez les fichiers Theme

téléchargés dans votre répertoire wp-content / themes fourni par WordPress. Si nécessaire, créez un dossier contenant votre nouveau thème dans le répertoire wp-content / themes. Un exemple de ceci serait, si votre thème est nommé Test, il devrait vivre dans wp-content / themes / test.

Accédez à Administration> Apparence> Thèmes, puis cliquez sur le lien Activer pour sélectionner le thème comme thème actuel.

- Installation avec cPanel

  Les panneaux de contrôle cPanel offrent une autre méthode pour installer des thèmes avec un fichier ZIP ou GZ. Dans le gestionnaire cPanel, si WordPress est installé, accédez à votre dossier Thèmes. Le chemin ressemblerait à "public_html / wp-content / themes". Cliquez sur Upload file (s) et téléchargez le fichier ZIP. Sélectionnez le fichier ZIP dans cPanel et cliquez sur Extraire le contenu du fichier dans le panneau à droite pour décompresser ce fichier.

  Accédez à Administration> Apparence> Thèmes, puis cliquez sur le lien Activer pour sélectionner le thème comme thème actuel.

Toutes les informations énumérées ci-dessus sont conformes au Codex WordPress. Il a été raccourci pour être bref. Le matériel d'origine peut être trouvé ici . Ou pour plus d'informations, visitez codex.wordpress.org .

Créer un thème personnalisé

Ces instructions créent un thème WordPress très simple, conforme aux normes minimales.

La première étape consiste à créer un nouveau dossier de thème dans votre répertoire de thèmes WordPress. Le chemin correct sera:> wp-content> themes> Pour créer un thème valide, les thèmes WordPress nécessitent au moins ces deux fichiers:

- index.php
- style.css

Votre feuille de style doit contenir un commentaire qui alerte WordPress qu'un thème existe ici.

```
/*
Theme Name: <theme name>
Author: <author name>
Description: <description goes here>
Version: <theme version #>
Tags: <tag to id theme>
*/
```

Votre thème a maintenant été créé. Accédez au tableau de bord WordPress et cliquez sur Apparence> Thèmes pour l'activer.

Lire Des thèmes en ligne: https://riptutorial.com/fr/wordpress/topic/8967/des-themes

Chapitre 21: Développement de plugin

Syntaxe

- add_action (string $ tag, callable $ function_to_add, int $ priority = 10, int $ accepted_args = 1)
- add_filter (string $ tag, callable $ function_to_add, int $ priority = 10, int $ accepted_args = 1)

Paramètres

Paramètre	Détail
$ tag	(string) (Obligatoire) Nom du filtre permettant de raccorder le rappel $ function_to_add.
$ function_to_add	(callable) (obligatoire) Rappel à exécuter lors de l'application du filtre.
$ priorité	(int) (Facultatif) Utilisé pour spécifier l'ordre dans lequel les fonctions associées à une action particulière sont exécutées. Les nombres inférieurs correspondent à une exécution antérieure et les fonctions avec la même priorité sont exécutées dans l'ordre dans lequel elles ont été ajoutées à l'action. Valeur par défaut: 10
$ accepted_args	(int) (Facultatif) Le nombre d'arguments acceptés par la fonction. Valeur par défaut: 1

Remarques

La manière dont les plug-ins fonctionnent est que, à différents moments de l'exécution de WordPress, WordPress vérifie si des plug-ins ont des fonctions enregistrées à exécuter à ce moment-là, et si c'est le cas, les fonctions sont exécutées. Ces fonctions modifient le comportement par défaut de WordPress.

Il y a deux sortes de crochets:

Les filtres vous permettent de modifier la valeur d'une donnée lors de l'exécution de WordPress. Les fonctions de rappel pour les filtres seront transmises via une variable, modifiées, puis renvoyées. Ils sont censés fonctionner de manière isolée et ne devraient jamais affecter les variables globales ou autre chose en dehors de la fonction.

Les actions , en revanche, vous permettent d'ajouter ou de modifier le fonctionnement de WordPress. Votre fonction de rappel s'exécutera à un moment précis de l'exécution de WordPress

et peut effectuer une sorte de tâche, comme faire écho à la sortie de l'utilisateur ou insérer quelque chose dans la base de données.

Référence du filtre

Référence d'action

Manuel

API de plugin

Filtres vs Actions

Examples

Filtre

```
add_filter('comment_text','before_comment');
add_filter('comment_text','after_comment');
function before_comment($comment_text){
            return 'input before comment'.$comment_text;
        }
function after_comment($comment_text){
            return $comment_text.'input after comment';
        }
```

action

```
add_action('wp_head','hook_javascript');

function hook_javascript() {
    $output="<script> alert('Page is loading...'); </script>";
    echo $output;
}
```

Exemples de développement de plug-ins: Widget de chanson préférée

```
    <?php
function wpshout_register_widgets() {
    register_widget( 'Favorite_Song_Widget');
}

add_action( 'widgets_init', 'wpshout_register_widgets' );

class Favorite_Song_Widget extends WP_Widget {

function Favorite_Song_Widget() {
    // Instantiate the parent object
    parent::__construct(
            'favorite_song_widget', // Base ID
            __('Favorite Song', 'text_domain'), // Name
            array( 'description' => __( 'Widget for playable favorite song', 'text_domain' ),
) // Args
```

```
    );
}

function widget( $args, $instance ) {
    echo $args['before_widget'];
    echo '<h3>Favorite Song Lists:</h3>';
    echo $instance['songinfo'];
    echo '<a href="' . $instance['link'] . '">Download it</a><br>';
    echo $instance['description'];
            echo $args['after_widget'];
}

function update($new_abc,$old_abc) {
    $instance = $old_abc;
    // Fields
    $instance['link'] = strip_tags($new_abc['link']);
    $instance['songinfo'] = strip_tags($new_abc['songinfo']);
            $instance['description'] = strip_tags($new_abc['description']);
    return $instance;
}

// Widget form creation
function form($instance) {
     $link = '';
    $songinfo = '';
            $description = '';
    // Check values
    if( $instance) {
        $link = esc_attr($instance['link']);
        $songinfo = esc_textarea($instance['songinfo']);
                    $description = esc_textarea($instance['description']);
    } ?>


    <p>
        <label for="<?php echo $this->get_field_id('link'); ?>"><?php _e('Link',
'wp_widget_plugin'); ?></label>
        <input class="widefat" id="<?php echo $this->get_field_id('link'); ?>" name="<?php
echo $this->get_field_name('link'); ?>" type="text" value="<?php echo $link; ?>" />
    </p>

    <p>
        <label for="<?php echo $this->get_field_id('songinfo'); ?>"><?php _e('Song Info:',
'wp_widget_plugin'); ?></label>
        <input class="widefat" id="<?php echo $this->get_field_id('songinfo'); ?>" name="<?php
echo $this->get_field_name('songinfo'); ?>" type="text" value="<?php echo $songinfo; ?>" />
    </p>

            <p>
        <label for="<?php echo $this->get_field_id('description'); ?>"><?php
_e('Description:', 'wp_widget_plugin'); ?></label>
        <textarea class="widefat" id="<?php echo $this->get_field_id('description'); ?>"
name="<?php echo $this->get_field_name('description'); ?>" type="text" value="<?php echo
$description; ?>"></textarea>
    </p>

            <p><a href="#" id="add-more-tabs"><?php _e('Add More Tabs', 'wp_widget_plugin');
?></a></p>

<?php }
```

```
}
```

Lire Développement de plugin en ligne:
https://riptutorial.com/fr/wordpress/topic/6108/developpement-de-plugin

Chapitre 22: Exécutez WordPress local avec XAMPP

Introduction

Avec XAMPP, vous pouvez installer un serveur Web sur votre PC local. Il possède un serveur Web Apache, la base de données MariaDB (MySQL) et fonctionne avec Perl et PHP. Après l'installation, vous pouvez exécuter et déboguer, par exemple, des systèmes de gestion de contenu tels que WordPress sur votre PC local. Vous pouvez l'utiliser avec Windows, Linux, Mac OS X et Solaris.

Examples

1. Installation de XAMPP

1. Téléchargez le programme d'installation pour la version actuelle de XAMPP
2. Parcourez l'assistant d'installation et ne changez rien si vous n'êtes pas sûr de ce que vous modifiez.
3. Après l'installation, vous voyez le panneau de contrôle XAMPP. Cliquez simplement sur le bouton de démarrage d'Apache et de MySQL pour les exécuter avec la configuration de base.
4. Si votre ordinateur est connecté à un réseau local et que vous souhaitez également accéder au serveur Web depuis d'autres ordinateurs, assurez-vous que votre pare-feu l'autorise.

2. Installez une base de données après avoir installé XAMPP

Pour exécuter WordPress sur votre ordinateur, vous devez d'abord configurer une base de données. Assurez-vous qu'Apache et MySQL sont en cours d'exécution (voir Installation de XAMPP étape 3)

1. Lancez un navigateur Web de votre choix et entrez "localhost" dans l'adresse.
2. Choisissez votre langue préférée et sélectionnez dans la catégorie "Outils" -> `phpMyAdmin` .
3. Sélectionnez l'onglet `Databases` .
4. Entrez le nom de votre base de données (vous aurez besoin de ce nom plus tard) sous "Créer une base de données" et choisissez `utf8_general_ci` dans la liste déroulante "Assemblage".
5. Cliquez sur votre base de données créée à gauche et sélectionnez l'onglet `Users` .
6. Ajouter un nouvel utilisateur en cliquant sur `Add user` .
7. Entrez votre nom d'utilisateur, choisissez `local` dans la liste déroulante "Host" (si vous souhaitez accéder à la base de données à partir d'autres ordinateurs du réseau, choisissez `Any host`) et entrez votre mot de passe (vous en aurez besoin plus tard).
8. Vérifiez si ci-dessous "Database for user" `Grant all privileges on wildcard name` est coché.
9. Appuyez sur `Go` .

3. Installation de WordPress après avoir configuré la base de données

Après avoir installé XAMPP et configuré la base de données MySQL, vous pouvez installer WordPress.

1. Téléchargez la dernière version de WordPress.
2. Décompressez le fichier et insérez le dossier dans le répertoire racine de votre serveur Web (si vous avez utilisé le chemin suggéré pendant l'installation de XAMPP, il s'agit de "c: \ xampp \ htdocs").
3. Tous les fichiers existants deux fois peuvent être remplacés.
4. Entrez dans le champ d'adresse de votre navigateur Web "localhost / wordpress / wp-admin / install.php".
5. Après avoir choisi votre langue, cliquez sur `Continue` puis sur `Let's go` .
6. Remplacez tous les champs de texte par défaut par la configuration choisie (voir " Configurer une base de données "). Ne changez pas "Database Host" et "Table Prefix", sauf que vous souhaitez migrer votre installation WordPress locale en ligne. N'oubliez pas que la sécurité est importante. Vous trouverez plus d'informations dans Set Prefix dans New WordPress Installation .
7. Appuyez sur `Submit` et `Run the install` .
8. Vous devez maintenant entrer le nom de votre site, un nom d'utilisateur, un mot de passe et un e-mail valide.
9. Cliquez sur `Install WordPress` pour terminer la configuration et connectez-vous à votre nouveau site WordPress local.

Lire Exécutez WordPress local avec XAMPP en ligne:
https://riptutorial.com/fr/wordpress/topic/9551/executez-wordpress-local-avec-xampp

Chapitre 23: Extraits personnalisés avec excerpt_length et excerpt_more

Examples

Limite la longueur de l'extrait à 50 mots

Mettez le code suivant dans **functions.php** :

```
function themify_custom_excerpt_length( $length ) {
   return 50;
}
add_filter( 'excerpt_length', 'themify_custom_excerpt_length', 999 );
```

Utilisez 999 comme priorité pour vous assurer que la fonction s'exécute après le filtre WordPress par défaut, sinon elle remplacerait ce qui est défini ici.

Ajouter un lien Lire plus à la fin de l'extrait

Pour ce faire, placez le code suivant dans **functions.php** :

```
function custom_excerpt_more($more) {
   return '<a href="'. get_permalink($post->ID) . '">Read More</a>';
}
add_filter('excerpt_more', 'custom_excerpt_more');
```

Les résultats devraient ressembler à ceci:

Ajout de quelques points à la fin de l'extrait

Dans nos **fonctions.php**

```
function new_excerpt_more( $more ) {
    return '.....';
}
add_filter('excerpt_more', 'new_excerpt_more');
```

Nous devrions obtenir ceci:

Lire Extraits personnalisés avec excerpt_length et excerpt_more en ligne: https://riptutorial.com/fr/wordpress/topic/6104/extraits-personnalises-avec-excerpt-length-et-excerpt-more

Chapitre 24: Faire des requêtes réseau avec HTTP API

Syntaxe

- $ response = wp_remote_get ($ url, $ args);
- $ response = wp_remote_post ($ url, $ args);
- $ response = wp_safe_remote_post ($ url, $ args);

Paramètres

Paramètre	Détails
$ url	(chaîne) (obligatoire) URL du site à récupérer.
$ args	(array) (Facultatif) Arguments de demande.

Remarques

Résultats

(WP_Error | array) La réponse en tant que tableau ou WP_Error en cas d'échec.

Examples

OBTENEZ une ressource JSON distante

Cet extrait va récupérer une ressource au format JSON, le décoder et l'imprimer au format tableau PHP.

```
// Fetch
$response = wp_remote_get( 'http://www.example.com/resource.json' );

if ( ! is_wp_error( $response ) ) {
  $headers = wp_remote_retrieve_headers( $response );

  if ( isset( $headers[ 'content-type' ] ) && 'application/json' === $headers[ 'content-type'
] ) {
    print_r( json_decode( wp_remote_retrieve_body( $response ) ) );
  }
}
```

Lire Faire des requêtes réseau avec HTTP API en ligne:
https://riptutorial.com/fr/wordpress/topic/1116/faire-des-requetes-reseau-avec-http-api

Chapitre 25: Fonction: add_action ()

Syntaxe

- add_action ($ tag, $ function_to_add)
- add_action ($ tag, $ function_to_add, $ priority)
- add_action ($ tag, $ function_to_add, $ priority, $ accepted_args)

Paramètres

Paramètre	Détails
$ tag	*(string) (Obligatoire)* Nom de l'action à laquelle est attaché `$function_to_add`
$ function_to_add	*(callable) (Obligatoire)* Fonction qui doit être appelée lorsque l'action indiquée par la `$tag` est exécutée
$ priorité	*(int) (Facultatif) Valeur par défaut:* `10` Utilisé pour spécifier l'ordre dans lequel les fonctions associées à une action particulière sont exécutées. Les nombres inférieurs correspondent à une exécution antérieure et les fonctions ayant la même priorité sont exécutées dans l'ordre dans lequel elles ont été ajoutées à l'action.
$ accepted_args	*(int) (Facultatif) Valeur par défaut:* `1` Nombre d'arguments acceptés par la fonction.

Remarques

La fonction `add_action()` crée un **crochet d'action** , associant une fonction PHP à une *action* particulière "tag" ou à un nom. Lorsque l'action est "déclenchée" par un appel à `do_action()` (ou `do_action_ref_array()`) avec une balise spécifique, toutes les fonctions "accrochées" à cette balise seront exécutées.

Dans la plupart des cas, cette fonction doit être utilisée dans le fichier `functions.php` un thème ou dans un fichier de plugin - ou dans un autre fichier source chargé par l'un ou l'autre.

Cette fonction fait partie de l' **API** du **plugin**

Examples

Crochet d'action de base

L'application la plus simple de `add_action()` consiste à ajouter du code personnalisé à exécuter à

un certain emplacement dans le code source d'une installation WordPress - soit en utilisant des actions fournies par le **Core** de WordPress, soit par des codes tiers tels que des plugins et thèmes.

Pour ajouter du contenu à la section `<head></head>` du site - dites à un élément `<link>` meta add pour indiquer où les informations de copyright du site peuvent être trouvées - `add_action()` peut être utilisé pour attacher une fonction qui imprime le balisage approprié à l'action `'wp_head'` (qui déclenche lorsque WordPress construit la section `<head>`):

```
function add_copyright_meta_link() {
  echo( '<link rel="copyright" href="' . get_home_url() . '/copyright">' );
}

add_action( 'wp_head', 'add_copyright_meta_link' );
```

Priorité du crochet d'action

Un nombre quelconque de fonctions peut être "accroché" à une action donnée. Dans certains cas, il est important qu'une fonction accrochée s'exécute avant ou après les autres, ce qui correspond au troisième paramètre de `add_action()` , `$priority` .

Si l'argument `$priority` est omis, la fonction sera associée à la priorité par défaut de `10` . Lorsque l'action est "déclenchée", les fonctions "accrochées" seront appelées en commençant par celles ajoutées avec la plus petite `$priority` et progressant vers les fonctions avec la plus grande `$priority` . Toutes les fonctions accrochées partageant la même priorité seront appelées dans l'ordre où elles ont été ajoutées (l'ordre dans lequel leurs appels `add_action()` respectifs ont été exécutés).

Par exemple, disons qu'un plug-in tiers utilise une fonction liée à l'action `'template_redirect'` afin de transférer les visiteurs vers la page `daily-deal` vers un lien d'affiliation pour un site de commerce électronique externe, mais que vous souhaitez la redirection se produire uniquement pour les utilisateurs connectés. Vous devez utiliser votre propre hook `'template_redirect'` pour envoyer des visiteurs déconnectés à la page de connexion. Après avoir déterminé que le plug-in tiers joint sa fonction avec la valeur par défaut de `$piority 10` , vous pouvez `$piority` votre fonction avec une priorité de `9` pour vous assurer que votre vérification de connexion se produit en premier:

```
function redirect_deal_visitors_to_login() {
  if( is_page( 'daily-deal' ) && !user_is_logged_in() ) {
    wp_redirect( wp_login_url() );
    exit();
  }
}

add_action( 'template_redirect', 'redirect_deal_visitors_to_login', 9 );
```

Accrocher les méthodes de classe et d’objet aux actions

Les classes PHP sont un outil puissant pour améliorer l'organisation du code et minimiser les collisions de noms. À un moment ou à un autre, la question de savoir comment créer un crochet

d'action pour une méthode de classe se pose inévitablement.

L'argument `$function_to_add` est souvent affiché sous la forme d'une chaîne contenant le nom de la fonction, mais le type de données de l'argument est en réalité un " appelable ", qui peut être résumé comme "une référence à une fonction ou à une méthode".

Il existe un certain nombre de formats appelables pouvant être utilisés pour référencer des méthodes sur des classes et des objets. Dans tous les cas cependant, la méthode référencée *doit* être visible publiquement. Une méthode est publique lorsqu'elle est préfixée par le mot-clé `public` ou pas du tout par un mot-clé de visibilité (auquel cas la méthode est par défaut publique).

Méthode d'objet Crochets d'action

Les méthodes d'objet sont exécutées sur une instance particulière d'une classe.

```
class My_Class {
  // Constructor
  function My_Class() {
    // (Instantiation logic)
  }

  // Initialization function
  public function initialize() {
    // (Initialization logic)
  }
}
```

Après avoir instancié la classe ci-dessus comme suit,

```
$my_class_instance = new My_Class();
```

la méthode `initialize()` serait normalement appelée sur l'objet en appelant `$my_class_instance->initialize();` . Accrocher la méthode à l'action WordPress `'init'` se fait en passant un tableau contenant une référence à l'instance et une chaîne contenant le nom de la méthode d'objet:

```
add_action( 'init', [ $my_class_instance, 'initialize' ] );
```

Si `add_action()` est appelée dans une méthode objet, la variable `$this` peut aussi être utilisée:

```
class My_Class {
  // Constructor
  function My_Class() {
    // (Instantiation logic)
    add_action( 'init', [ $this, 'initialize' ] );
  }

  // Initialization function
  public function initialize() {
    // (Initialization logic)
  }
}
```

Méthodes de classe

Les méthodes de classe sont exécutées de manière statique sur une classe plutôt que sur une instance particulière. Compte tenu de la classe suivante,

```
class My_Class {
  // Initialization function
  public static function initialize() {
    // (Initialization logic)
  }
}
```

la méthode `initialize()` serait normalement appelée à l'aide de l'opérateur `::` scope-resolution, c'est-à-dire `My_Class::initialize();` . Accrocher une méthode de classe statique à un WordPress peut se faire de deux manières différentes:

- En utilisant un tableau composé d'une chaîne contenant le nom de la classe et une chaîne contenant le nom de la méthode:

  ```
  add_action( 'init', [ 'My_Class', 'initialize' ] );
  ```

- Passer une chaîne contenant une référence complète à la méthode, y compris l'opérateur `::`

  ```
  add_action( 'init', 'My_Class::initialize' );
  ```

- Si `add_action()` est appelée dans une méthode de classe statique, le mot-clé `self` ou la constante-constante `__CLASS__` peut être utilisé à la place du nom de la classe. Notez que cela est généralement déconseillé car les valeurs de ces éléments deviennent quelque peu contre-intuitives dans le cas d'un héritage de classe.

  ```
  class My_Class {
    // Setup function
    public static function setup_actions() {
      add_action( 'init', 'self::initialize' );
    }

    // Initialization function
    public static function initialize() {
      // (Initialization logic)
    }
  }
  ```

Lire Fonction: add_action () en ligne: https://riptutorial.com/fr/wordpress/topic/6561/fonction--add-action---

Chapitre 26: Fonction: wp_trim_words ()

Syntaxe

- `<?php $trimmed_text = wp_trim_words( $text, $num_words = 55, $more = null ); ?>`

Paramètres

Paramètre	Détails
`$text`	(Chaîne) (obligatoire) Texte qui sera raccourci ou coupé.
`$num_words`	(Entier) (obligatoire) Nombre de mots auxquels le texte sera restreint.
`$more`	(Chaîne) (Facultatif) Ce qu'il faut ajouter si $ text doit être coupé.

Examples

Ajuster le contenu des messages

Cette fonction raccourcit le texte en un nombre de mots spécifié et renvoie le texte raccourci.

```
<?php echo wp_trim_words( get_the_content(), 40, '...' ); ?>
```

Dans l'exemple ci-dessus, nous transmettons le contenu de la publication à la fonction. Cela limitera la longueur du contenu à 40 mots et réduira le reste des mots.

Lire Fonction: wp_trim_words () en ligne: https://riptutorial.com/fr/wordpress/topic/7866/fonction--wp-trim-words---

Chapitre 27: Formats de poste

Remarques

Les formats de courrier suivants sont disponibles pour les utilisateurs, si le thème leur permet de les prendre en charge.

Notez que même si l'entrée de contenu de publication ne change pas, le thème peut utiliser ce choix d'utilisateur pour afficher la publication différemment selon le format choisi. Par exemple, un thème pourrait ne pas afficher le titre d'un message "Statut". La manière dont les choses sont affichées dépend entièrement du thème, mais voici quelques lignes directrices générales.

- de côté - Typiquement sans titre. Semblable à une mise à jour de note Facebook.
- galerie - Une galerie d'images. Post contiendra probablement un shortcode de galerie et aura des pièces jointes.
- lien - Un lien vers un autre site. Les thèmes peuvent souhaiter utiliser la première balise dans le contenu de la publication comme lien externe pour cette publication. Une autre approche pourrait être que la publication ne soit composée que d'une URL, ce sera alors l'URL et le titre (post_title) sera le nom associé à l'ancre.
- image - Une image unique. La première balise dans le post pourrait être considérée comme l'image. Sinon, si la publication consiste uniquement en une URL, ce sera l'URL de l'image et le titre de la publication (post_title) sera l'attribut title de l'image.
- citation - Une citation. Contiendra probablement un blockquote contenant le contenu de la citation. Alternativement, la citation peut être juste le contenu, la source / auteur étant le titre.
- status - Une brève mise à jour de statut, similaire à une mise à jour de statut Twitter.
- video - Une seule playlist vidéo ou vidéo. Le premier tag ou objet / incorporé dans le contenu du post pourrait être considéré comme la vidéo. Sinon, si la publication consiste uniquement en une URL, ce sera l'URL de la vidéo. Peut également contenir la vidéo en pièce jointe à la publication, si le support vidéo est activé sur le blog (comme via un plugin). audio - Un fichier audio ou une liste de lecture. Peut être utilisé pour la baladodiffusion.
- chat - une transcription de chat

Examples

Ajout du type de message au thème

Ajouter des post-formats à la page post_type '

```
add_post_type_support( 'page', 'post-formats' );
```

L'exemple suivant enregistre le type de message personnalisé «my_custom_post_type» et ajoute des post-formats.

Enregistrez le type de poste personnalisé 'my_custom_post_type'

```
add_action( 'init', 'create_my_post_type' );
function create_my_post_type() {
    register_post_type( 'my_custom_post_type',
      array(
        'labels' => array( 'name' => __( 'Products' ) ),
        'public' => true
    )
  );
}
```

Ajouter des post-formats à post_type 'my_custom_post_type'

```
add_post_type_support( 'my_custom_post_type', 'post-formats' );
```

Ou dans la fonction register_post_type (), ajoutez «post-formats» dans le tableau de paramètres 'supports'. L'exemple suivant équivaut à plus d'un.

Enregistrer le type de poste personnalisé 'my_custom_post_type' avec le paramètre 'supports'

```
add_action( 'init', 'create_my_post_type' );
function create_my_post_type() {
    register_post_type( 'my_custom_post_type',
      array(
        'labels' => array( 'name' => __( 'Products' ) ),
        'public' => true,
        'supports' => array('title', 'editor', 'post-formats')
    )
  );
}
```

Ajouter un support de thème pour post

Appel de fonction

```
add_theme_support( 'post-formats' )
```

Lire Formats de poste en ligne: https://riptutorial.com/fr/wordpress/topic/6075/formats-de-poste

Chapitre 28: get_bloginfo ()

Introduction

Récupère des informations sur le site actuel.

Syntaxe

- get_bloginfo ($ show, $ filter)

Paramètres

Paramètre	Détails
$ show	*(chaîne)* Les informations de configuration du site à récupérer.
filtre $	*(string)* Spécification de l'opportunité de retourner une valeur filtrée ou non.

Remarques

$ show

Valeurs	La description	Exemple
'nom' (par défaut)	Titre du site	`'Matt Mullenweg'`
'la description'	Slogan du site	`'Just another WordPress site'`
'wpurl'	URL de l'installation WordPress. Identique à la fonction `site_url()`	`'http://example.com'`, `'http://localhost/wordpress'`
'url'	URL du site. Identique à la fonction `home_url()`	`'http://example.com'`, `'http://localhost/wordpress'`
'admin_email'	Adresse e-mail de l'administrateur principal	`'matt@mullenweg.com'`
'charset'	Encodage des caractères des pages et des flux	`'UTF-8'`

Valeurs	La description	Exemple
'version'	Version actuelle de l'installation WordPress	`'4.5'`
'html_type'	valeur de type de contenu du code HTML	`'text/html'`
'text_direction'	Direction du texte déterminée par la langue du site	`'ltr'`
'la langue'	Code de langue basé sur ISO 639-1	`'en-US'`
'stylesheet_url'	URL de la feuille de style du thème activé. Priorité valeur: **thème enfant** »Thème parent.	`'http://example.com/wp-content/themes/twentysixteen/style.css'`
'répertoire_fiche_de_fichier'	Emplacement de la ressource du thème activé. Priorité valeur: **thème enfant** »Thème parent.	`'http://example.com/wp-content/themes/twentysixteen'`
'template_url'	Répertoire d'URL du thème activé. Priorité valeur: **thème parent** »Thème enfant.	`'http://example.com/wp-content/themes/twentysixteen'`
'template_directory'	Identique à `'template_url'`	
'pingback_url'	Fichier Pingback XML-RPC	`'http://example/xmlrpc.php'`
'atom_url'	URL du flux Atom	`'http://example/feed/atom/'`
'rdf_url'	URL de flux RDF / RSS 1.0	`'http://example/feed/rdf/'`
'rss_url'	URL de flux RSS 0.92	`'http://example/feed/rss/'`

Valeurs	La description	Exemple
'rss2_url'	URL de flux RSS 2.0	`'http://example/feed/'`
'comments_atom_url'	Commentaires URL du flux Atom	`'http://example/comments/feed/atom/'`
'URL du site'	*(obsolète)* Utilisez 'url' à la place	
'maison'	*(obsolète)* Utilisez 'url' à la place	

filtre $

Valeurs	La description	Exemple
'brut' (par défaut)	Aucun filtre ne sera appliqué	*données brutes*
'afficher'	Les filtres seront appliqués à la valeur de retour si `$show` n'est ni `'url'`, `'directory'`, `'home'`	*données filtrées*

Examples

Obtenir le titre du site

```
<?php echo get_bloginfo( 'name' ); ?>
```

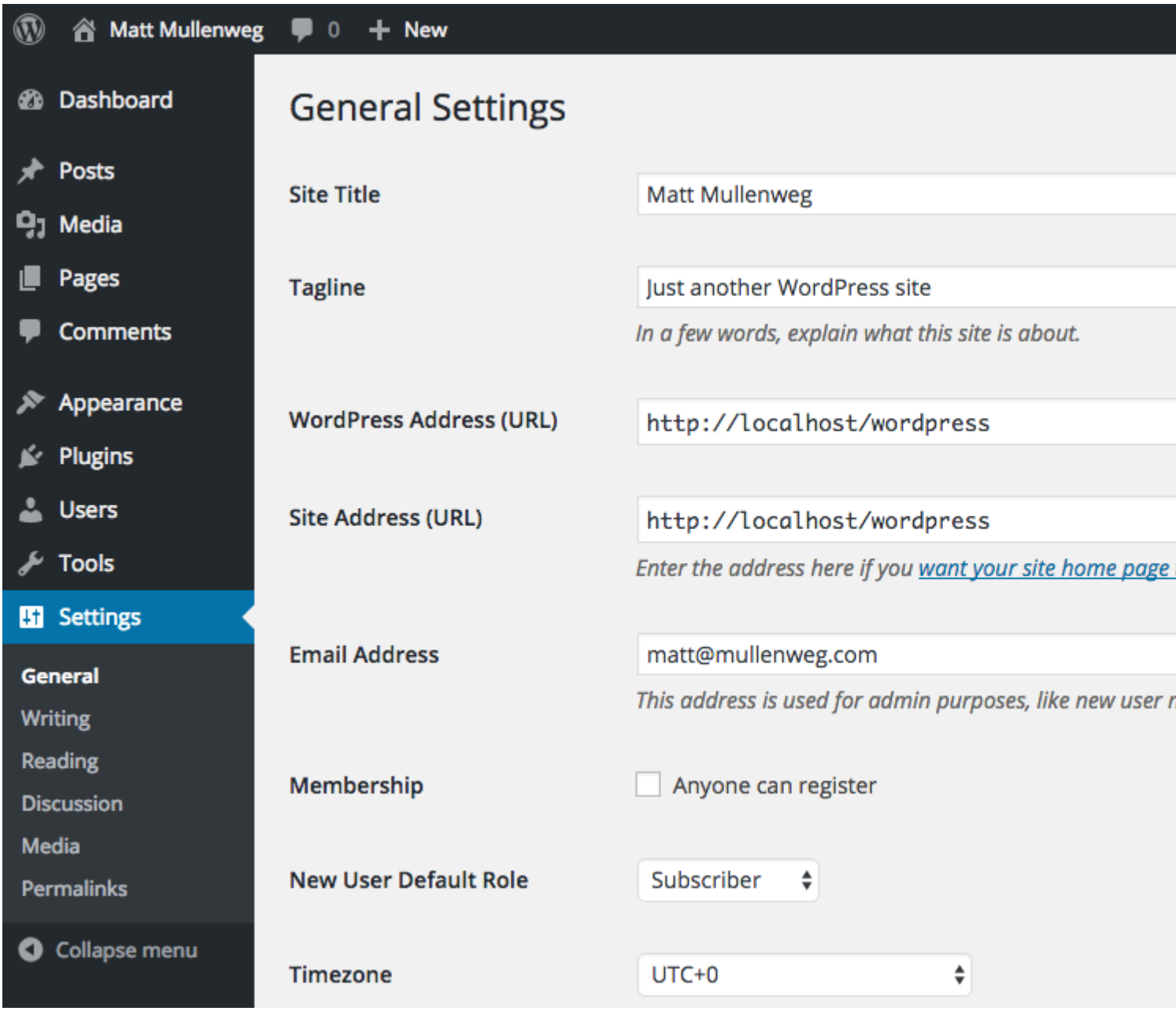
ou

```
<?php echo get_bloginfo(); ?>
```

Sortie

```
Matt Mullenweg
```

Basé sur ces exemples de paramètres



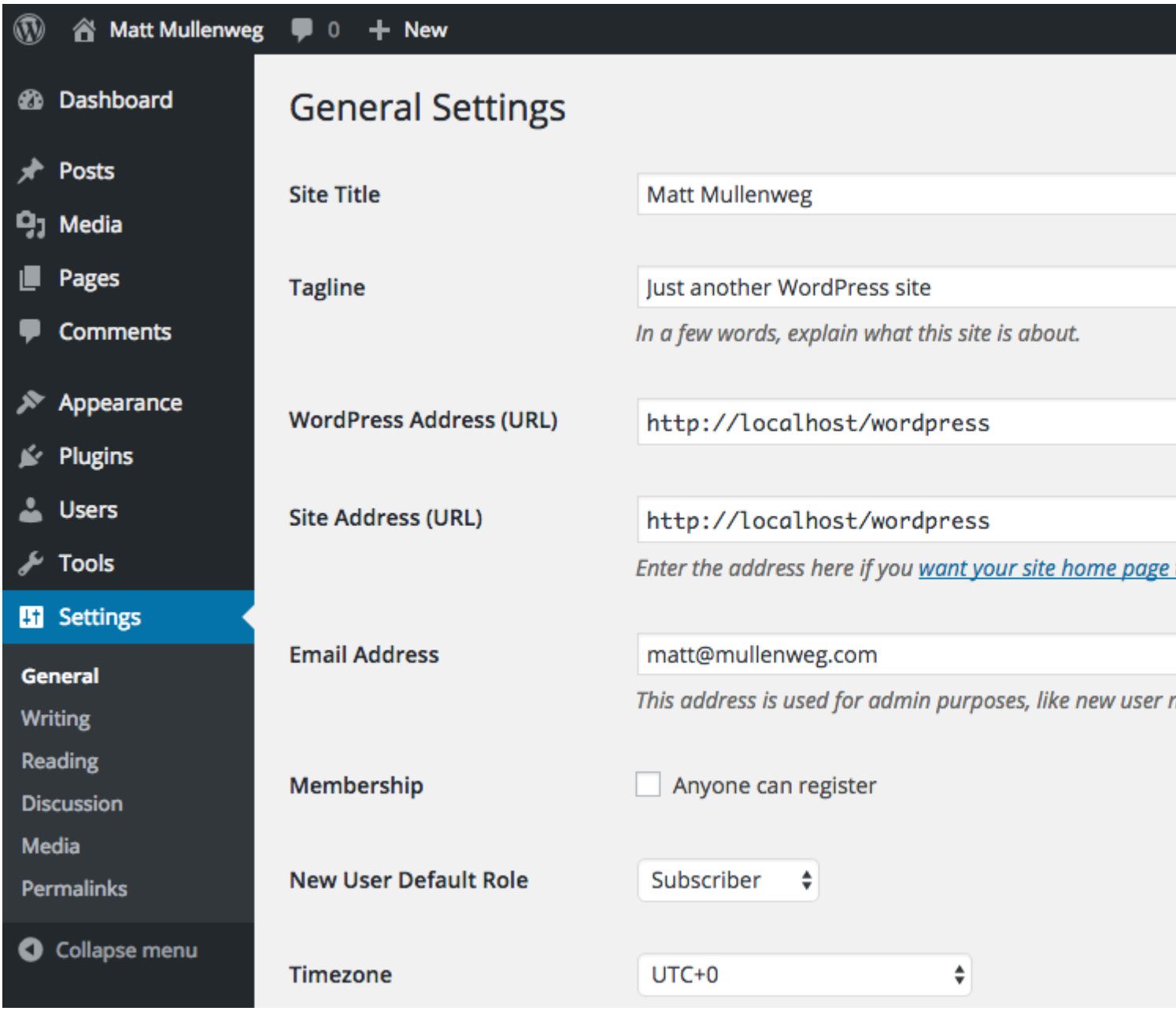
Obtenir le slogan du site

```
<?php echo get_bloginfo( 'description' ); ?>
```

Sortie

```
Just another WordPress site
```

Basé sur ces exemples de paramètres



Obtenir l'URL du thème actif

```
<?php echo esc_url( get_bloginfo( 'stylesheet_directory' ) ); ?>
```

Sortie

```
http://example.com/wp-content/themes/twentysixteen
```

Des alternatives

En interne, `get_bloginfo( 'stylesheet_directory' )` appelle `get_stylesheet_directory_uri()` , vous pouvez donc l'utiliser à la place:

```
<?php echo esc_url( get_stylesheet_directory_uri() ); ?>
```

De nombreux développeurs préfèrent utiliser ces fonctions dédiées en raison de conventions de dénomination incohérentes entre eux et `get_bloginfo()` . Par exemple, `get_stylesheet_directory()` renvoie le chemin du thème enfant. Cependant, comme l'illustre notre exemple précédent, `get_bloginfo( 'stylesheet_directory' )` renvoie l'URL du thème enfant. Si vous utilisez `get_stylesheet_directory_uri()` , il y a moins de risque de confusion si vous récupérez un chemin ou une URL.

Obtenir l'URL du site

```
<?php echo esc_url(get_bloginfo('url')); ?>
```

ou si vous deviez créer un lien vers une sous-page

```
<?php echo esc_url(get_bloginfo('url') . '/some-sub-page');  ?>
```

Obtenir l'adresse e-mail de l'administrateur du site

Nous pouvons utiliser la fonction `get_bloginfo` pour récupérer l'adresse e-mail de l'administrateur du site.

```
<?php echo get_bloginfo('admin_email'); ?>
```

Lire get_bloginfo () en ligne: https://riptutorial.com/fr/wordpress/topic/524/get-bloginfo---

Chapitre 29: get_home_path ()

Introduction

Obtenez le chemin d'accès absolu au système de fichiers à la racine de l'installation WordPress.

Paramètres

Paramètre	Détails
Aucun	Cette fonction n'accepte aucun paramètre.

Remarques

Différence importante entre `get_home_path()` et `ABSTPATH`

Gardez à l'esprit la différence entre `ABSPATH` et `get_home_path()` si WordPress est installé dans un sous-dossier.

La fonction `get_home_path()` renvoie toujours un chemin **sans** le sous-dossier:

- http://www.example.com - / var / www / htdocs / example
- http://www.example.com/wp - **/ var / www / htdocs / example**

Voici comment il diffère de `ABSPATH` , qui renverra des valeurs différentes:

- http://www.example.com - / var / www / htdocs / example
- http://www.example.com/wp - **/ var / www / htdocs / example / wp**

`ABSPATH` est d'abord défini dans `wp-load.php` qui sera situé dans `/var/www/htdocs/example/wp/wp-load.php` d'où `ABSPATH` prendra sa définition.

`get_home_path()` vérifie si les `site_url` et `home_url` diffèrent et supprime la sous-chaîne du chemin. Sinon, il renvoie la valeur `ABSPATH` :

```
function get_home_path() {
    $home    = set_url_scheme( get_option( 'home' ), 'http' );
    $siteurl = set_url_scheme( get_option( 'siteurl' ), 'http' );
    if ( ! empty( $home ) && 0 !== strcasecmp( $home, $siteurl ) ) {
        $wp_path_rel_to_home = str_ireplace( $home, '', $siteurl ); /* $siteurl - $home */
        $pos = strripos( str_replace( '\\', '/', $_SERVER['SCRIPT_FILENAME'] ),
trailingslashit( $wp_path_rel_to_home ) );
        $home_path = substr( $_SERVER['SCRIPT_FILENAME'], 0, $pos );
        $home_path = trailingslashit( $home_path );
    } else {
        $home_path = ABSPATH;
    }
```

```
    return str_replace( '\\', '/', $home_path );
}
```

En l'utilisant dans votre code

L'appel de `get_home_path()` doit être effectué dans un contexte où `wp-admin/includes/file.php` a déjà été inclus.

Par exemple, l'utilisation de `get_home_path()` dans le hook `admin_init` est `admin_init` , mais l'utiliser dans `init` n'est pas une erreur fatale:

```
Call to undefined function get_home_path()
```

Ce fichier n'est inclus qu'à partir du contexte admin (tableau de bord), si vous en avez absolument besoin en dehors de ce contexte, vous devrez inclure le fichier vous-même avant d'appeler la fonction:

```
require_once(ABSPATH . 'wp-admin/includes/file.php');
```

Examples

Usage

```
$path = get_home_path();
```

Valeur de retour:

`string`

Chemin complet du système de fichiers à la racine de l'installation WordPress, même s'il est installé dans un sous-dossier.

Exemple:

`/var/www/htdocs/example`

Lire get_home_path () en ligne: https://riptutorial.com/fr/wordpress/topic/9699/get-home-path---

Chapitre 30: get_option ()

Introduction

Récupère une valeur d'option basée sur un nom d'option.

Syntaxe

- get_option (option $, $ default)

Paramètres

Paramètre	Détails
option $	(chaîne) Nom de l'option à récupérer. Attendu pour ne pas être SQL-échappé.
$ par défaut	(mixed) (Facultatif) Valeur par défaut à renvoyer si l'option n'existe pas.

Remarques

Liste des arguments pour l'option $

- 'admin_email'
- "blogname"
- 'blogdescription'
- 'blog_charset'
- 'format de date'
- 'default_category'
- 'maison'
- 'URL du site'
- 'modèle'
- 'start_of_week'
- 'upload_path'
- 'users_can_register'
- 'posts_per_page'
- 'posts_per_rss'

Examples

Afficher le titre du blog

Code

```
<h1><?php echo get_option( 'blogname' ); ?></h1>
```

Sortie

Nom du blog dans le style H1

Afficher le jeu de caractères

Code

```
<p><?php echo esc_html( sprintf( __( 'Character set: %s', 'textdomain' ), get_option(
'blog_charset' ) ) ); ?></p>
```

Sortie

Jeu de caractères: UTF-8

Gestion des options non existantes

Code

```
<?php echo get_option( 'something_bla_bla_bla' ); ?>
```

Sortie

faux

Code

```
<?php echo get_option( 'something_bla_bla_bla', 'Oh no!' ); ?>
```

Sortie

Oh non!

Lire get_option () en ligne: https://riptutorial.com/fr/wordpress/topic/9194/get-option---

Chapitre 31: get_permalink ()

Introduction

Cette fonction renvoie le paralink complet de la publication en cours ou de la publication désignée.

Syntaxe

- get_permalink ($ post, $ leavename)

Paramètres

Paramètre	Détails
$ post	(int) (facultatif) Identifiant de poste ou objet postal. La valeur par défaut est l'ID du message actuel.
$ leavename	(bool) (facultatif) Indique s'il faut conserver le nom de la publication ou le nom de la page.

Remarques

Pour le paramètre $ leavename, il est faux par défaut.

Examples

Utilisation simple de get_parmalink

Code

```
echo get_permalink();
```

Sortie

Le lien de la page en cours, par exemple: http://website.com/category/name-of-post/

Spécifier le message pour obtenir le lien

Code

```
echo get_permalink( 44 );
```

Sortie

Le lien de l'identifiant du poste: 44, par exemple: http://website.com/category/name-of-post/

Obtenez le lien sans le nom du poste

Code

```
echo get_permalink( 44 , false );
```

Sortie

Le lien du post id: 44 sans le nom du post, par exemple:
http://website.com/category/%postname%/

Lire get_permalink () en ligne: https://riptutorial.com/fr/wordpress/topic/9209/get-permalink---

Chapitre 32: get_template_part ()

Syntaxe

- get_template_part ($ slug, $ name);
- get_template_part ($ slug);

Paramètres

Paramètre	Détails
$ slug	*(string)* Nom du slug générique du template
$ nom	*(string)* Nom du template spécialisé

Examples

Charger un composant de modèle à partir d'un sous-dossier

```
get_template_part( 'foo/bar', 'page');
```

nécessitera "bar-page.php" du répertoire "foo".

Obtenir un fichier spécifique

Vous pouvez obtenir un fichier spécifique en utilisant cette fonction.

```
get_template_part('template-parts/layout');
```

Incluez le fichier layout.php du sous-répertoire template-parts placé à la racine de votre dossier de thème.

Lire get_template_part () en ligne: https://riptutorial.com/fr/wordpress/topic/5897/get-template-part--

Chapitre 33: get_template_part ()

Introduction

Le but de cette fonction est de standardiser la manière d'importer des partiels ou des composants d'un thème dans le modèle de thème principal. Vous pouvez utiliser un SSI PHP standard (inclus côté serveur), cependant, il y a quelques avantages à utiliser get_template_part (). L'utilisation de cette fonction réduit les erreurs susceptibles d'affecter les développeurs moins expérimentés qui tentent d'identifier un chemin d'accès complet sur le serveur. En outre, elle échoue normalement lorsque des fichiers n'existent pas

Syntaxe

- get_template_part ($ slug)
- get_template_part ($ slug, $ name)

Paramètres

Paramètre	Détails
$ slug	*(chaîne)* Nom du bloc du modèle personnalisé.
$ nom	*(string)* Nom du modèle spécialisé. Optionnel

Examples

Y compris un modèle personnalisé

```
<?php get_template_part( 'foo' ); ?>
```

Comprend

```
../wp-content/themes/your-theme-slug/foo.php
```

Inclure un modèle personnalisé avec un nom de fichier séparé par des tirets

```
<?php get_template_part( 'foo','bar' ); ?>
```

Comprend

```
../wp-content/themes/your-theme-slug/foo-bar.php
```

Inclure un modèle personnalisé dans un répertoire

```
<?php get_template_part( 'dir/foo' ); ?>
```

Comprend

```
../wp-content/themes/your-theme-slug/dir/foo.php
```

Inclure un modèle personnalisé avec un nom de fichier séparé par des tirets situé dans un répertoire

```
<?php get_template_part( 'dir/foo', 'bar' ); ?>
```

Comprend

```
../wp-content/themes/your-theme-slug/dir/foo-bar.php
```

Passage de variable à l'étendue du modèle personnalisé

```
<?php
set_query_var( 'passed_var', $my_var );
get_template_part( 'foo', 'bar' );
?>
```

Accéder à `foo-bar.php`

```
<?php echo $passed_var; ?>
```

Lire get_template_part () en ligne: https://riptutorial.com/fr/wordpress/topic/5993/get-template-part--

Chapitre 34: get_template_part ()

Syntaxe

- get_template_part ('nom-fichier-non-extension');

Paramètres

Paramètre	La description
nom de fichier sans extension	Le nom du composant de modèle sans extension. Par exemple 'foo' au lieu de 'foo.php'

Examples

Chargement du modèle

Extrait le code d'un certain fichier spécifié dans un autre fichier où l'appel a été effectué.

Par exemple dans example.php

```
<h1>Hello World!</h1>
```

Inside page.php

```
// header code
get_template_part('example');
// rest of page code
```

Sortie:

```
// header code
<h1>Hello World</h1>
// rest of page code
```

Lire get_template_part () en ligne: https://riptutorial.com/fr/wordpress/topic/6267/get-template-part--

Chapitre 35: get_the_category ()

Introduction

Cette fonction renvoie toutes les catégories sous la forme d'un tableau de l'article ou de la page en cours ou du message ou de la page désigné.

Syntaxe

- get_the_category ($ id)

Paramètres

Paramètre	Détails
$ id	(int) (facultatif) par défaut à l'identifiant de publication actuel. L'identifiant de poste.

Remarques

Veuillez noter que get_the_category () renvoie un tableau, ce qui signifie que vous ne pouvez pas directement faire écho à la valeur renvoyée.

Voici une liste d'objets de chaque catégorie que vous pouvez imprimer:

- term_id
- prénom
- limace
- term_group
- terme_taxonomie_id
- taxonomie
- la description
- parent
- compter
- object_id
- filtre
- cat_ID
- category_count
- description de la catégorie
- cat_name
- category_nicename
- category_parent

Examples

Obtenez tous les noms des catégories du message

Code

```
$categories = get_the_category();
 foreach( $categories as $category ) {
    echo  $category->name . '<br />';
}
```

Sortie

Tous les noms des catégories de la page en cours, un sur chaque ligne.

Obtenez tous les identifiants des catégories du message

Code

```
$categories = get_the_category();
 foreach( $categories as $category ) {
    echo  $category->term_id . '<br />';
}
```

Sortie

Tous les identifiants des catégories de la page en cours, un sur chaque ligne.

Lire get_the_category () en ligne: https://riptutorial.com/fr/wordpress/topic/9211/get-the-category---

Chapitre 36: get_the_title ()

Introduction

Cette fonction renvoie le titre du message en cours ou du message désigné.

Syntaxe

- get_the_title ($ post)

Paramètres

Paramètre	Détails
$ post	(int) (facultatif) Identifiant de poste ou objet postal. La valeur par défaut est l'ID du message actuel.

Remarques

Si vous prévoyez d'obtenir le titre d'un article ou d'une page à l'aide d'une post-boucle, il est conseillé d'utiliser the_title () à la place.

Examples

Utilisation simple de get_the_title

Code

```
get_the_title();
```

Sortie

Le titre du poste ou de la page en cours

Obtenir le titre d'un identifiant de poste spécifié

Code

```
get_the_title( 44 );
```

Sortie

Le titre de l'identifiant du poste: 44.

Lire get_the_title () en ligne: https://riptutorial.com/fr/wordpress/topic/9214/get-the-title---

Chapitre 37: Hiérarchie de modèles

Remarques

Plugins pour le débogage dans WordPress:

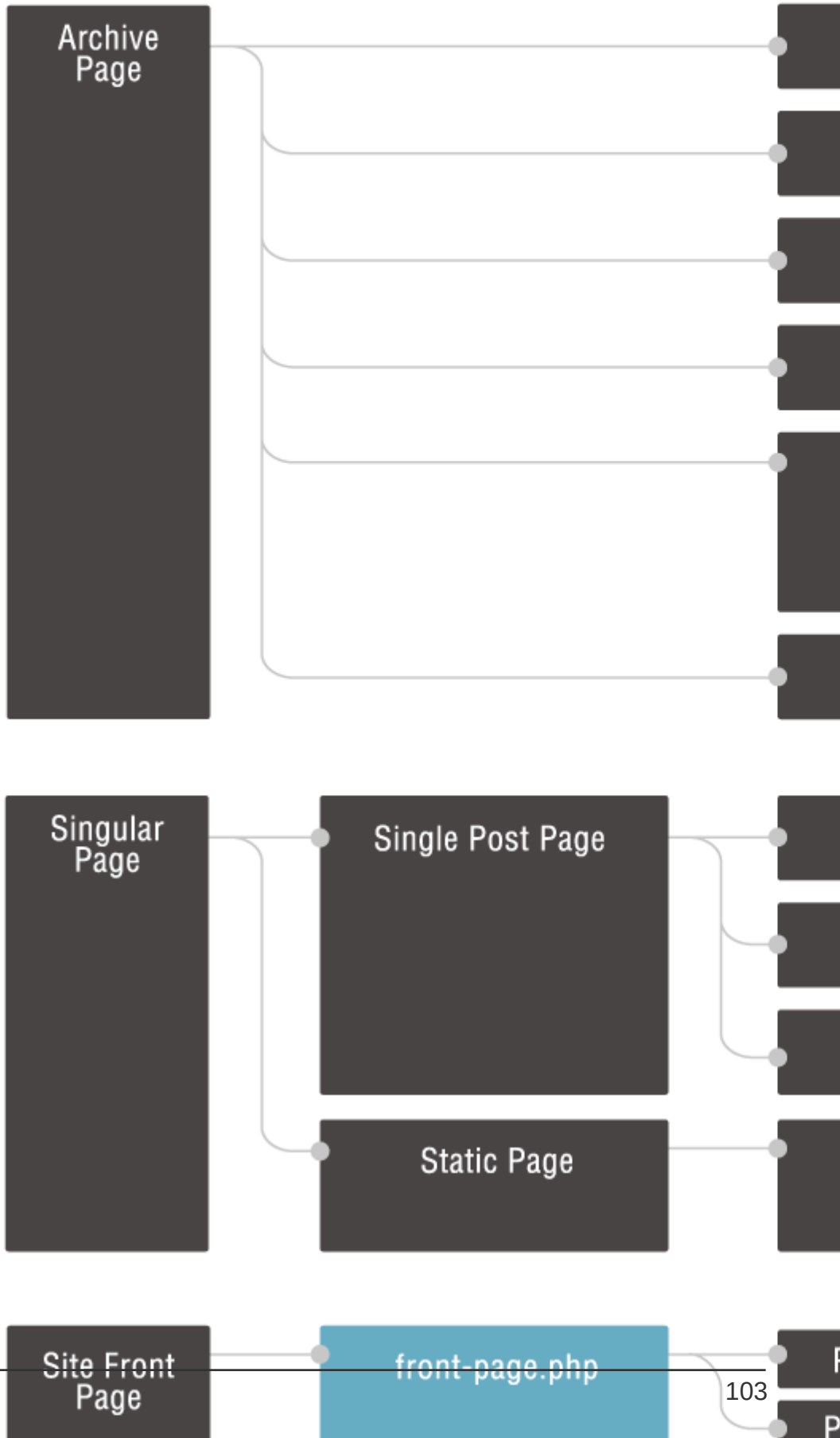
- https://wordpress.org/plugins/query-monitor/
- https://wordpress.org/plugins/debug-bar/
- https://wordpress.org/plugins/debug-bar-console/
- https://wordpress.org/plugins/kint-debugger/
- https://wordpress.org/plugins/rest-api-console/

Examples

introduction

Une des choses les plus importantes à apprendre lorsque vous créez un thème WordPress est la hiérarchie WordPress Template pour les thèmes. La hiérarchie de modèles définit quel fichier de modèle sera chargé pour chaque demande et dans quel ordre. Si le premier modèle n'existe pas dans la hiérarchie, WordPress essaiera de charger le suivant et ainsi de suite jusqu'à ce que vous vous `index.php` dans `index.php` .

Pour décrire la hiérarchie du modèle en détail, le mieux est bien sûr d’utiliser une image avec la structure complète:


Archive Page
Singular Page
Single Post Page
Static Page
Site Front Page
front-page.php

qui ne cible que la catégorie avec un slug spécifique, par exemple `category-books.php` ne serait utilisé que pour la catégorie avec le slug `book` . Un autre exemple est `page-$id.php` qui ne cible qu'une page avec un identifiant spécifique, par exemple `page-41.php` ne ciblerait que la page avec l'ID 41.

Après les modèles qui ciblent des types ou des publications spécifiques, nous arrivons aux modèles de type générique, comme archive.php pour toutes les pages d'archive ou `page.php` pour toutes les pages. Mais souvenez-vous que ceux-ci ne seront utilisés que si la page en cours ne correspond à aucun des modèles les plus élevés de la hiérarchie.

Enfin, si WordPress n'a pu trouver aucun modèle correspondant dans le répertoire du modèle, le dernier repli est toujours `index.php` qui est le seul fichier de modèle requis dans un thème WordPress.

Le débogage

Il est facile de se perdre lors du débogage de la hiérarchie. Vous pouvez utiliser la commande `debug_backtrace` .

Mettez l'extrait de code suivant dans n'importe quel modèle que vous souhaitez déboguer et affichez la page générée:

```
<!--
<?php print_r( debug_backtrace() ) ?>
-->
```

Lire Hiérarchie de modèles en ligne: https://riptutorial.com/fr/wordpress/topic/6116/hierarchie-de-modeles

Chapitre 38: home_url ()

Syntaxe

- home_url ($ path, $ scheme);

Paramètres

Paramètre	Détails
$ path	(*Chaîne* , *Facultatif*) Pour ajouter plus de segment après l'URL d'origine.
$ schéma	(*String* , *Facultatif*) Schéma pour donner le contexte de l'URL à la maison. Accepte 'http', 'https', 'relatif', 'repos' ou null.

Examples

Obtenir l'URL de la maison

`home_url()` est utilisé pour récupérer l'URL d'origine du site.

```
<?php echo esc_url( home_url( '/' ) ) ); ?>
```

Sortie

```
http://www.example.com
```

URL du site

Renvoie l'option 'site_url' avec le protocole approprié (https: //)

```
<?php echo site_url(); ?>
```

Sortie

```
http://www.example.com
```

Lire home_url () en ligne: https://riptutorial.com/fr/wordpress/topic/1252/home-url---

Chapitre 39: init

Syntaxe

1. add_action ('init', fonction callable $)

Remarques

`init` est un hook d'action qui est déclenché une fois le chargement de WordPress terminé, mais avant l'envoi des en-têtes HTTP.

Examples

Traitement des données de requête $ _POST

add_action ('init', 'process_post_data');

```
function process_post_data() {
    if( isset( $_POST ) ) {
        // process $_POST data here
    }
}
```

Traitement des données de requête $ _GET

```
add_action('init', 'process_get_data');

function process_get_data() {
    if( isset( $_GET ) ) {
        // process $_GET data here
    }
}
```

Enregistrement d'un type de poste personnalisé

```
add_action( 'init', function() {
    register_post_type( 'event', array(
            'public' => true,
            'label'  => 'Events'
        );
    );
});
```

Enregistre un nouveau type de poste personnalisé avec un libellé `Events` et un `event` slug

Lire init en ligne: https://riptutorial.com/fr/wordpress/topic/6375/init

Chapitre 40: Installation et configuration

Examples

Wordpress sur LAMP

J'ai testé les étapes suivantes sur Amazon EC2 avec Ubuntu 14.04

Voici un résumé des lignes de commande pour installer la pile technologique LAMP (avant d'installer wordpress):

1: Installer la pile technologique LAMP

Mettre à jour Linux

`#> sudo apt-get update` -> mettre à jour les informations des dépôts de paquets

`#> sudo apt-get upgrade` -> installer les mises à jour du paquet

Installer apache2

```
#> sudo apt-get install apache2
```

Installer php

```
#> sudo apt-get install php5 libapache2-mod-php5 php5-mcrypt
```

Installer mysql

```
#> sudo apt-get install mysql-server php5-mysql
```

Installer phpmyadmin

```
#> sudo apt-get install phpmyadmin
```

En cas de problème, supprimez le package installé et ses fichiers de configuration

```
#> sudo apt-get purge package_name
```

Une fois la pile installée, procédez comme suit pour terminer l'installation de wordpress:

2: Installer WordPress

1. `wget https://wordpress.org/latest.zip`
2. `sudo mkdir /var/www/wordpress`
3. Déplacer le fichier zip dans / var / www / wordpress et extraire.
4. `cd /etc/apache2/sites-available/`
5. `sudo cp 000-default.conf wordpress.conf`

6. Modifiez wordpress.conf pour que apache2 sache transférer toute requête ou votre domaine vers le dossier wordpress app.
7. `sudo a2ensite wordpress.conf`
8. `sudo service apache2 restart`
9. Accédez à PhpMyAdmin via votre navigateur par youdomain.com/phpmyadmin. Créez un nouvel utilisateur «wordpress» et cochez pour créer la base de données correspondante. `cd /var/www/wordpress`
10. `sudo cp wp-config-example.php wp-config.php`
11. Modifiez le fichier de configuration pour ajouter vos informations de base de données MySQL wordpress.
12. `sudo chown -R www-data:www-data /var/www/wordpress`
13. `sudo chmod -R 755 /var/www/wordpress/`
14. Ouvrez le navigateur et tapez dans votre domaine. Vous devriez voir la page d'installation de WordPress. Suivez les instructions et vous avez terminé!

WP d'installation en MAMP

Il est assez simple d'installer wordpress dans MAMP.

Vous devez suivre quelques étapes simples:

1 - Téléchargez MAMP d' ici , c'est gratuit et vous n'avez probablement pas besoin de la version pro.

2 - Installez sur votre PC ou Mac.

3 - Exécutez le programme -> vous verrez une petite fenêtre ouverte, à partir de là vous pouvez définir MAMP. Laissez pour le moment toutes les valeurs prédéfinies, pour la première installation, vous n'avez pas besoin de compliquer votre vie! À l'avenir, rappelez-vous qu'il est recommandé de changer votre mot de passe et votre nom d'utilisateur pour la base de données MySQL. Par défaut, c'est la racine.

4 - Cliquez sur "Ouvrir la page WebStart" - ici vous pouvez voir vos informations de données comme mot de passe, nom de l'administrateur et aussi des informations sur MAMP.

5 - Cliquez sur Outils -> phpMyAdmin et vous serez redirigé vers la page de la base de données.

6 - Créez une nouvelle base de données, cliquez sur Nouveau et donnez-lui le nom que vous voulez, vous aurez besoin de ce nom plus tard.

7 - Maintenant, recherchez un appel de dossier "htdocs", il se trouve dans le dossier MAMP de votre PC ou Mac. Si vous utilisez un Mac, vous devez en créer une dans le Finder et ouvrir l'application en utilisant "Afficher le contenu du package". À l'intérieur, vous trouverez le dossier

`htdocs` .

8 - Prenez le dossier Wordpress et copiez-le dans les htdocs, vous devez mettre dans tout le dossier, sans mettre le fichier zip. Renommez le dossier, vous pouvez appeler avec votre nom de projet.

9 - Revenez à la fenêtre de MAMP dans votre navigateur et cliquez sur "Mon site" - cela ouvrira un appel URL `http://localhost:8888` (8888 est le port par défaut, vous pouvez le changer si vous le souhaitez). Vous pouvez voir ici le dossier que vous avez placé dans le dossier `htdocs` , cliquez sur le nom que vous avez donné au dossier.

10 - Commencez maintenant une installation normale de WordPress, vous devez utiliser l'administrateur et le mot de passe. Par défaut, il s'agit de `root` et `root` , et utilisez le nom de la base de données que vous avez créée auparavant.

11 - Exécutez l'installation et terminez!

Vous verrez votre site Web sur `http://localhost:8888/namefolder` Bien sûr, vous devez continuer à utiliser MAMP.

Lire Installation et configuration en ligne: https://riptutorial.com/fr/wordpress/topic/6606/installation-et-configuration

Chapitre 41: Interroger les messages

Syntaxe

- $ the_query = new WP_Query ($ args);
- $ posts_array = get_posts ($ args);

Paramètres

Paramètre	La description
$ args	*(array)* Tableau d'arguments nécessaires à une requête - peut être personnalisé en fonction de vos besoins, par exemple en interrogeant des publications d'une seule catégorie, d'un type de publication personnalisé ou même en interrogeant certaines taxonomies

Remarques

Les arguments de requête sont nombreux. La page de codex WP_Query () contient une liste de paramètres. Certains d'entre eux sont

- Paramètres d'auteur
- Paramètres de catégorie
- Paramètres de balise
- Paramètres de taxonomie
- Paramètre de recherche
- Paramètres de poste et de page
- Paramètres de mot de passe
- Paramètres de type
- Paramètres d'état
- Paramètres de pagination
- Paramètres de commande et de commande
- Paramètres de date
- Paramètres de champs personnalisés
- Paramètres d'autorisation
- Paramètres de type MIME
- Paramètres de mise en cache
- Paramètre des champs de retour

L'une des choses les plus importantes à retenir est la suivante:

Ne jamais utiliser query_posts ()

```
query_posts()
```

remplace la requête principale et peut entraîner des problèmes dans le reste de votre thème. Chaque fois que vous devez modifier la requête principale (ou une requête quelconque), utilisez le filtre pre_get_posts . Cela vous permettra de modifier la requête avant son exécution.

De même, lorsque vous interrogez des publications, vous devez toujours les réinitialiser à l'aide de wp_reset_postdata () . Cela restaurera la variable globale `$post` de la boucle de requête principale et vous ne rencontrerez aucun problème ultérieurement (par exemple, les catégories étant exclues, car dans votre boucle secondaire, vous les avez exclues et avez oublié de réinitialiser la requête).

Examples

Utiliser l'objet WP_Query ()

La création d'une instance distincte de l'objet `WP_Query` est simple:

```
$query_args = array(
                'post_type' => 'post',
                'post_per_page' => 10
            );

$my_query = new WP_Query($query_args);

if( $my_query->have_posts() ):
    while( $my_query->have_posts() ): $my_query->the_post();
       //My custom query loop
    endwhile;
endif;

wp_reset_postdata();
```

Notez que vous devez créer le tableau des arguments de la requête en fonction de vos spécifications. Pour plus de détails, consultez la page du codex WP_Query .

Utiliser get_posts ()

`get_posts()` est un wrapper pour une instance distincte d'un objet `WP_Query` . La valeur renvoyée est un tableau de post-objet.

```
global $post;

$args = array(
    'numberposts' => 5,
    'offset'=> 1,
    'category' => 1
);

$myposts = get_posts( $args );

foreach( $myposts as $post ) :
    setup_postdata($post); ?>
    <h2><a href="<?php the_permalink(); ?>"><?php the_title(); ?></a></h2>
<?php endforeach;
```

```
wp_reset_postdata(); ?>
```

Pour plus d'informations, consultez la page de codex get_posts () .

Lire Interroger les messages en ligne: https://riptutorial.com/fr/wordpress/topic/4002/interroger-les-messages

Chapitre 42: L'objet $ wpdb

Remarques

Il existe deux manières de référencer l'objet `$wpdb` . La première consiste à utiliser le mot-clé PHP `global` pour agir sur l'instance globale de l'objet.

```
global $wpdb;
echo $wpdb->prefix;
// Outputs the prefix for the database
```

La deuxième façon d'utiliser l'objet `$wpdb` est de faire référence à la variable globale `$GLOBALS` PHP.

```
echo $GLOBALS['wpdb']->prefix;
// This will also output the prefix for the database
```

La deuxième voie est déconseillée car elle peut ne pas être considérée comme la meilleure pratique.

Examples

Sélection d'une variable

Dans la forme la plus élémentaire, il est possible de sélectionner une seule variable dans une table en appelant la méthode `get_var` l'objet en passant dans une requête SQL.

```
global $wpdb;
$user = $wpdb->get_var( "SELECT ID FROM $wpdb->users WHERE user_email='foo@bar.com'" );
```

Il est très important de noter que toute valeur non fiable utilisée dans les requêtes *doit* être échappée afin de se protéger contre les attaques. Cela peut être fait en utilisant la méthode de `prepare` l'objet.

```
global $wpdb;
$email = $_POST['email'];
$user = $wpdb->get_var(
    $wpdb->prepare( "SELECT ID FROM $wpdb->users WHERE user_email=%s", $email )
);
if( !is_null( $user ){
    echo $user;
} else {
    echo 'User not found';
}
```

Sélection de plusieurs lignes

Vous pouvez utiliser get_results pour obtenir plusieurs lignes de la base de données.

```
global $wpdb;
$userTable =$wpdb->prefix."users";

$selectUser = $wpdb->get_results("SELECT * FROM $userTable");
```

Cela affichera toutes les listes d'utilisateurs dans un tableau.

Lire L'objet $ wpdb en ligne: https://riptutorial.com/fr/wordpress/topic/2691/l-objet---wpdb

Chapitre 43: La barre d'administration (aussi appelée "la barre d'outils")

Remarques

La barre d'outils d'administration WordPress a été ajoutée dans la version 3.1 et contient des liens vers des tâches administratives courantes ainsi que des liens vers le profil de l'utilisateur et d'autres informations WordPress. Cependant, de nombreux propriétaires de sites n'aiment pas montrer la barre d'outils par défaut à tous les utilisateurs connectés et / ou souhaitent y ajouter leurs propres options.

Examples

Suppression de la barre d'outils d'administration de tous les administrateurs sauf

Ajoutez le code suivant à `functions.php` pour le supprimer de tout le monde sauf le niveau d'utilisateur administrateur:

```
add_action('after_setup_theme', 'no_admin_bar');

    function no_admin_bar() {
        if (!current_user_can('administrator') && !is_admin()) {
          show_admin_bar(false);
        }
    }
```

Suppression de la barre d'outils Admin à l'aide de filtres

Une autre façon de masquer la barre d'administration est d'ajouter

```
if ( !current_user_can( 'manage_options' ) ) {
    add_filter( 'show_admin_bar', '__return_false' , 1000 );
}
```

Les utilisateurs qui n'ont pas les privilèges pour accéder à la page Paramètres ne pourront pas voir la barre d'administration.

Comment faire pour supprimer le logo WordPress de la barre d'administration

Les développeurs peuvent utiliser l'action **admin_bar_menu** pour supprimer des éléments de la barre d'administration ou de la barre d'outils WordPress.

```
add_action('admin_bar_menu', 'remove_wp_logo_from_admin_bar', 999);
function remove_wp_logo_from_admin_bar( $wp_admin_bar ) {
```

```
    $wp_admin_bar->remove_node('wp-logo');
}
```

Le code ci-dessus supprime le logo WordPress de la barre d'administration. Tout ce que vous avez à faire est de coller le code dans votre fichier functions.php.

Le paramètre passé à la méthode remove_node est l'ID du noeud que vous souhaitez supprimer. Les identifiants peuvent être trouvés dans le code source HTML de la page WordPress avec une barre d'outils. Par exemple, l'ID de l'élément li pour le logo WordPress sur la gauche de la barre d'outils est "wp-admin-bar-wp-logo":

```
<li id="wp-admin-bar-wp-logo" class="menupop"> … </li>
```

Supprimez "wp-admin-bar-" de l'ID de li pour obtenir l'ID du noeud. Dans cet exemple, l'identifiant du noeud est "wp-logo".

Vous pouvez utiliser des outils d'inspecteur de navigateur pour identifier les identifiants de nœud de divers éléments ou nœuds de votre barre d'administration.

Ajoutez votre logo personnalisé et un lien personnalisé sur la page de connexion de l'administrateur

Vous pouvez ajouter ci-dessous des crochets pour ajouter votre propre logo et lien pour remplacer le logo wordpress par défaut.

Pour ajouter un logo personnalisé

```
function custom_login_logo() {
echo '<style type="text/css">
h1 a { background-image: url('.get_bloginfo('template_directory').'/images/custom-logo.png)
!important; background-size : 100% !important; width: 300px !important; height : 100px
!important;}
</style>';
}
add_action('login_head', 'custom_login_logo');
```

Pour ajouter un lien logo personnalisé

```
add_filter( 'login_headerurl', 'custom_loginlogo_url' );
function custom_loginlogo_url($url) {
    return home_url();
}
```

Lire La barre d'administration (aussi appelée "la barre d'outils") en ligne: https://riptutorial.com/fr/wordpress/topic/2932/la-barre-d-administration--aussi-appelee--la-barre-d-outils--

Chapitre 44: La boucle (boucle WordPress principale)

Examples

Structure de boucle WordPress de base

Chaque fois que WordPress charge la page, celle-ci exécute *la boucle principale* .

La boucle est le moyen de parcourir tous les éléments liés à la page sur laquelle vous vous trouvez actuellement.

La boucle principale fonctionnera sur un objet `WP_Query` global. La requête a une méthode globalisée `have_posts()` , qui nous permet de parcourir tous les résultats. Enfin, à l'intérieur de la boucle, vous pouvez appeler la `the_post()` (également en tant que fonction globale), qui définit l'objet post global sur la publication en cours dans la boucle, et définit la postdata sur la publication en cours. Grâce à cela, vous pouvez appeler des fonctions comme `the_title` , `the_content` , `the_author` (*balises template*) directement dans la boucle.

Par exemple, si vous êtes sur des listes de publications, la boucle principale contiendra un objet de requête avec tous les messages.

Si vous êtes sur un seul message (ou page), il contiendra une requête avec un seul message (page) sur lequel vous êtes actuellement.

```
if ( have_posts() ) :
    while ( have_posts() ) :
        the_post();
        var_dump( $post );
    endwhile;
endif;
```

Syntaxe de boucle alternative

Vous pouvez également utiliser une boucle avec des accolades comme ceci:

```
if ( have_posts() ) {
    while ( have_posts() ) {

        the_post();
        var_dump( $post );

    }
}
```

Manipulation d'aucun élément dans la boucle

Si vous voulez gérer un tel scénario, ajoutez simplement une instruction `if/else` .

```
if ( have_posts() ) : while ( have_posts() ) :

    the_post();
    var_dump( $post );

endwhile; else :

    __('This Query does not have any results');

endif;
```

Lire La boucle (boucle WordPress principale) en ligne:
https://riptutorial.com/fr/wordpress/topic/1803/la-boucle--boucle-wordpress-principale-

Chapitre 45: Le débogage

Introduction

https://codex.wordpress.org/Debugging_in_WordPress

Le débogage du code PHP fait partie de tout projet, mais WordPress est livré avec des systèmes de débogage spécifiques conçus pour simplifier le processus, ainsi que pour standardiser le code à travers le noyau, les plugins et les thèmes.

Remarques

Plugins pour le débogage dans WordPress:

- https://wordpress.org/plugins/query-monitor/
- https://wordpress.org/plugins/debug-bar/
- https://wordpress.org/plugins/debug-bar-console/
- https://wordpress.org/plugins/kint-debugger/
- https://wordpress.org/plugins/rest-api-console/

Examples

WP_DEBUG

`WP_DEBUG` est une constante PHP (une variable globale permanente) qui peut être utilisée pour déclencher le mode "debug" dans WordPress. Il est supposé être faux par défaut et est généralement défini sur true dans le fichier `wp-config.php` sur les copies de développement de WordPress.

```
define( 'WP_DEBUG', true );
define( 'WP_DEBUG', false );
```

WP_DEBUG_LOG

`WP_DEBUG_LOG` est un compagnon de WP_DEBUG qui entraîne l'enregistrement de toutes les erreurs dans un fichier journal debug.log situé dans le répertoire / wp-content /. Ceci est utile si vous souhaitez revoir toutes les notifications plus tard ou si vous avez besoin d'afficher les avis générés hors écran (par exemple, lors d'une requête AJAX ou d'une exécution wp-cron).

```
//enable
define( 'WP_DEBUG_LOG', true );

//disable
define( 'WP_DEBUG_LOG', false );
```

WP_DEBUG_DISPLAY

`WP_DEBUG_DISPLAY` est un autre compagnon de WP_DEBUG qui contrôle si les messages de débogage sont affichés dans le code HTML des pages ou non. La valeur par défaut est "true", qui affiche les erreurs et les avertissements au fur et à mesure qu'ils sont générés. Si vous définissez cette valeur sur false, toutes les erreurs seront masquées. Ceci doit être utilisé conjointement avec WP_DEBUG_LOG afin que les erreurs puissent être examinées plus tard. Remarque: pour que WP_DEBUG_DISPLAY fasse quelque chose, WP_DEBUG doit être activé (true).

```
//enable
define( 'WP_DEBUG_DISPLAY', true );

//disable
define( 'WP_DEBUG_DISPLAY', false );
```

SCRIPT_DEBUG

`SCRIPT_DEBUG` est une constante liée qui obligera WordPress à utiliser les versions "dev" des fichiers CSS et JavaScript principaux plutôt que les versions minifiées normalement chargées. Ceci est utile lorsque vous testez des modifications sur des fichiers .js ou .css intégrés. La valeur par défaut est false.

```
//enable
define( 'SCRIPT_DEBUG', true );

//disable
define( 'SCRIPT_DEBUG', false );
```

SAUVEGARDES

La définition SAVEQUERIES enregistre les requêtes de base de données dans un tableau et ce tableau peut être affiché pour faciliter l'analyse de ces requêtes. La constante définie comme true entraîne l'enregistrement de chaque requête, la durée d'exécution de la requête et la fonction appelée. REMARQUE: Cela aura un impact sur les performances de votre site. Veillez donc à désactiver cette option lorsque vous ne déboguez pas.

```
define( 'SAVEQUERIES', true );
```

Le tableau est stocké dans le

```
global $wpdb->queries;
```

Exemple wp-config.php et bonnes pratiques pour le débogage

Le code suivant, inséré dans votre fichier wp-config.php, enregistre toutes les erreurs, notifications et avertissements dans un fichier appelé debug.log dans le répertoire wp-content. Il masquera également les erreurs afin de ne pas interrompre la génération de page.

```
 // Enable WP_DEBUG mode
define( 'WP_DEBUG', true );

// Enable Debug logging to the /wp-content/debug.log file
define( 'WP_DEBUG_LOG', true );

// Disable display of errors and warnings
define( 'WP_DEBUG_DISPLAY', false );
@ini_set( 'display_errors', 0 );

// Use dev versions of core JS and CSS files (only needed if you are modifying these core
files)
define( 'SCRIPT_DEBUG', true );
```

Bonne pratique Si vous souhaitez ajouter des messages personnalisés au débogage du journal, ajoutez le code suivant dans votre plugin ou votre thème.

```
//Checking is function_exists
    if ( !function_exists( 'print_to_log' ) ) {
            //function writes a message to debug.log if debugging is turned on.
            function print_to_log( $message )
            {
                if ( true === WP_DEBUG ) {
                    if ( is_array( $message ) || is_object( $message ) ) {
                        error_log( print_r( $message, true ) );
                    } else {
                        error_log( $message );
                    }
                }
            }
        }
```

Voir les journaux dans un fichier séparé

Lorsque vous avez un appel ajax, il est extrêmement difficile d'obtenir un journal depuis l'intérieur de la fonction de rappel. Mais si vous activez le débogage

```
define('WP_DEBUG', true);
```

et après cela ajouter

```
ini_set('log_errors',TRUE);
ini_set('error_reporting', E_ALL);
ini_set('error_log', dirname(__FILE__) . '/error_log.txt');
```

vous aurez un fichier `error.log.txt` dans votre dossier racine où se trouvent tous vos journaux. vous pouvez même les connecter avec

```
error_log( print_r( 'what I want to check goes here', true) );
```

à l'intérieur de votre code. Cela vous facilitera la vie.

Lire Le débogage en ligne: https://riptutorial.com/fr/wordpress/topic/9170/le-debogage

Chapitre 46: le titre()

Introduction

Cette fonction renvoie le titre du message en cours.

Syntaxe

- the_title ($ before, $ after, $ echo);

Paramètres

Paramètre	Détails
$ avant	(chaîne) (facultatif) Texte à placer avant le titre.
$ après	(chaîne) (facultatif) Texte à placer après le titre.
$ echo	(Booléen) (facultatif) Afficher le titre ou le renvoyer pour une utilisation en PHP

Remarques

- Pour le paramètre $ echo, utilisez true pour afficher le titre et utilisez false pour le renvoyer en PHP
- Veuillez noter que the_title ne peut être utilisé que dans des boucles, si vous voulez l'utiliser en dehors des boucles, veuillez utiliser get_the_title

Examples

Utilisation simple de the_title

Code

```
the_title( );
```

Sortie

Le titre du poste ou de la page en cours

Imprimer le titre avec le code avant et après

Code

```
the_title( '<h1>', '</h1>' );
```

Sortie

Le titre du message ou de la page en cours dans les balises h1

Lire le titre() en ligne: https://riptutorial.com/fr/wordpress/topic/9213/le-titre--

Chapitre 47: Les bases du thème de l'enfant

Syntaxe

- **template** — est le nom du thème WordPress principal, le parent.
- **child-theme** — est le paquet qui remplace le **modèle** .

Remarques

J'ai annoncé que l'utilisation d'un thème enfant est toujours une bonne chose mais il y a toujours un *but ...*

Dans notre exemple de remplacement de modèle, imaginons que l'auteur d'un thème ajoute ses propres améliorations au modèle de barre latérale et qu'il y en aura une nouvelle à

`/themes/template/sidebar.php`

```
<?php
/**
 * The template for the sidebar containing the main widget area
 *
 * @link https://developer.wordpress.org/themes/basics/template-files/#template-partials
 */

if ( is_active_sidebar( 'sidebar-1' )  ) : ?>
    <aside id="secondary" class="sidebar widget-area" role="complementary">
        <?php dynamic_sidebar( 'sidebar-1' ); ?>
    </aside><!-- .sidebar .widget-area -->
<?php endif; ?>
```

Maintenant, notre site Web ne bénéficiera pas de la nouvelle spécification `role="complementary"` car notre thème enfant est toujours en train d'écraser le **modèle** avec son propre fichier sur `/themes/child-theme/sidebar.php`

Il est de notre devoir, en tant que mainteneurs de sites Web, de garder une trace des modèles que nous remplaçons et, dans le cas imminent d'une mise à jour, examinez attentivement le journal des modifications afin de mettre à jour les fichiers enfants.

Examples

2) le but

Les thèmes enfants sont un moyen sûr de conserver les personnalisations du modèle principal sans craindre de les perdre lors d'une mise à jour de thème.

Fondamentalement, chaque fois que vous voulez éditer un fichier dans le modèle actif de votre site Web, vous devez vous demander " **Qu'est-ce qui va se passer lorsque je mettrai à jour le thème?**

Et la réponse est simple: **vous les perdez car la mise à jour apportera un dossier de thème entièrement nouveau** .

Examinons donc un thème enfant en tant que dossier contenant des fichiers qui écrasent les fichiers ayant le même chemin dans le thème parent. Maintenant, apportons de vrais exemples:

Définition et exigences

Un thème enfant est identifié par WordPress lorsqu'il existe un répertoire (par exemple `child-theme`) dans `/wp-content/themes/` avec les fichiers suivants:

- `style.css`

  Ce fichier doit commencer par un modèle de commentaire comme celui-ci:

  ```
  /*
  Theme Name: Example Child
  Author: Me
  Author URI: https://example.com/
  Template: example
  Text Domain: example-child-theme
  Domain Path: /languages/
  */
  ```

  Les choses les plus importantes à considérer ici sont:

  - Le nom du `Template` doit être exactement le nom du dossier qui contient le thème parent (le slug du thème parent)
  - Nommez votre thème enfant de manière à pouvoir l’identifier facilement dans Dashboard (en général, il suffit d’ajouter `Child` au nom du parent, comme `Example Child`).

- `index.php`
- `functions.php`

3) Réécriture du modèle

L'utilisation la plus courante d'un thème enfant consiste à remplacer les parties de modèle. Par exemple, une barre latérale, si nous avons un thème avec le fichier suivant à

`/themes/template/sidebar.php`

```
<?php
/**
 * The sidebar containing the main widget area.
 *
 * @link https://developer.wordpress.org/themes/basics/template-files/#template-partials
 */

if ( ! is_active_sidebar( 'sidebar-1' ) ) {
    return;
```

```
}?>
<div id="sidebar" class="widget-area">
    <?php dynamic_sidebar( 'sidebar-1' ); ?>
</div>
```

Nous pouvons certainement ajouter notre propre fichier dans le thème enfant (avec les spécifications du premier exemple) avec le fichier suivant

`/themes/child-theme/sidebar.php`

```
<?php
/**
 * The sidebar containing the main widget area.
 */

if ( ! is_active_sidebar( 'sidebar-1' ) ) {
    return;
}?>
<div id="my-sidebar" class="my-own-awesome-class widget-area">
    <div class="my-wrapper">
        <?php dynamic_sidebar( 'sidebar-1' ); ?>
    </div>
</div>
```

Maintenant, `my-own-awesome-class` est sûr dans **le thème enfant** et il ne sera supprimé d'aucune mise à jour de thème et WordPress choisira toujours un modèle parmi les thèmes enfants lorsqu'il en trouvera un sur le même chemin.

Remplacement d'actifs

Même si ce n'est pas une bonne pratique, vous devez parfois remplacer des éléments tels que des fichiers CSS ou JS ou des bibliothèques.

Notez que le système de remplacement de modèle WordPress ne fonctionne pas avec autre chose que les fichiers `.php` , donc lorsque nous parlons d'actifs, nous nous référons à des actifs enregistrés.

Un exemple pourrait être le remplacement de la bibliothèque jQuery par votre version souhaitée. Dans notre fichier `functions.php` thème enfant, nous devons ajouter une fonction qui supprime la version actuelle de `jQuery` et ajoutons notre propre version de CDN (rappelez-vous que ce n'est qu'un exemple).

```
/**
 * Dequeue the jQuery script and add our own version.
 *
 * Hooked to the wp_print_scripts action, with a late priority (100),
 * so that it is after the script was enqueued.
 */
function my_own_theme_scripts() {
    // remove the current version
    wp_dequeue_script( 'jquery' );
    // register my desired version
    wp_register_script( 'jquery', 'https://code.jquery.com/jquery-3.1.0.min.js', false,
'3.1.0' );
```

```
    // load my version, here or somewhere else
    wp_enqueue_script( 'jquery' );
}
add_action( 'wp_print_scripts', 'my_own_theme_scripts' );
```

Lire Les bases du thème de l'enfant en ligne: https://riptutorial.com/fr/wordpress/topic/6238/les-bases-du-theme-de-l-enfant

Chapitre 48: Meta Box

Introduction

Utilisation simple d'une méta-boîte dans les éditeurs de contenu wp-admin

Syntaxe

- _x ('Text', 'Description', 'textdomain') est utilisé pour ajouter une description du service de traduction au lieu de __ ('Text', 'textdomain') qui n'est que la traduction
- _ex ('Text', 'Description', 'textdomain') est utilisé pour faire écho au texte traduit avec une description

Remarques

Le contenu de la boîte de métadonnées de rendu peut être n'importe quoi. Au lieu que les valeurs soient directement intégrées, vous pouvez également utiliser un `include` avec un modèle PHP et utiliser la méthode `set_query_var` pour lui transmettre des données. La sauvegarde fonctionnerait de la même manière.

Examples

Un exemple simple avec une entrée régulière et une entrée de sélection

```
/**
* Add meta box to post types.
*
* @since  1.0.0
*/
function myplugin_add_meta_box() {
    // Set up the default post types/
    $types = array(
        'post',
    );

    // Optional filter for adding the meta box to more types. Uncomment to use.
    // $types = apply_filters( 'myplugin_meta_box_types', $types );

    // Add the meta box to the page
    add_meta_box(
        'myplugin-meta-box', // Meta Box Id. Can be anything.
        _x( 'Custom Meta', 'Custom Meta Box', 'myplugin' ), // The title of the meta box.
Translation is optional. Can just be string.
        'myplugin_render_meta_box', // The render meta box function.
        $types, // Add the post types to which to add the meta box.
        'side', // Show on the side of edit.
        'high' // Show at top of edit.
    );
}
```

```
add_action( 'add_meta_boxes', 'myplugin_add_meta_box' );

/**
* Render the meta box.
*
* This shows examples of a basic input and a select inside a meta box. These can be anything.
*
* @since  1.0.0
*
* @param $post WP_Post The post being edited.
*/
function myplugin_render_meta_box( $post ) {
    // Get the current post meta values for our custom meta box.
    $city  = get_post_meta( $post->ID, 'city', true ); // True is for returning a single
value.
    $country = get_post_meta( $post->ID, 'country', true ); // True is for returning a single
value.

    // Add the WP Nonce field for security.
    wp_nonce_field( plugin_basename( __FILE__ ), 'myplugin_meta_nonce' );
?>

<p>
    <label for="city">
        <?php _ex( 'City', 'Custom Meta Box Template', 'myplugin' ); ?>
    </label>
    <input name="city" id="city" value="<?php echo $city; ?>" />
</p>
<p>
    <label for="country">
        <?php _ex( 'Country', 'Custom Meta Box Template', 'myplugin' ); ?>
    </label>
    <select name="country" id="country">
        <option value="United States" <?php selected( $country, 'United States' ); ?>><?php
_ex( 'United States', 'Custom Meta Box Template', 'myplugin' ); ?></option>
        <option value="Mexico" <?php selected( $country, 'Mexico' ); ?>><?php _ex( 'Mexico',
'Custom Meta Box Template', 'myplugin' ); ?></option>
        <option value="Canada" <?php selected( $country, 'Canada' ); ?>><?php _ex( 'Canada',
'Custom Meta Box Template', 'myplugin' ); ?></option>
    </select>
</p>

<?php
}

/**
* Save meta box data.
*
* @since  1.0.0
*
* @param $post_id int The Id of the Post being saved.
*/
function myplugin_save_meta_data( $post_id ) {
    // Verify this is not an auto save.
    if ( defined( 'DOING_AUTOSAVE' ) && DOING_AUTOSAVE ) {
        return;
    }

    // Validate the meta box nonce for security.
    if ( ! isset( $_POST['myplugin_meta_nonce'] ) || ! wp_verify_nonce(
```

```
$_POST['myplugin_meta_nonce'], plugin_basename( __FILE__ ) ) ) {
        return;
    }

    // Get the new values from the form.
    $city    = $_POST['city'];
    $country = $_POST['country'];

    // update_post_meta will add the value if it doesn't exist or update it if it does.
    update_post_meta( $post_id, 'city', $city );
    update_post_meta( $post_id, 'country', $country );

    /*
    * OPTIONAL ALTERNATIVE
    *
    * Instead of just using update_post_meta, you could also check the values and
    * issue create/update/delete on the post meta value.
    */
    // $current_city_value = get_post_meta( $post_id, 'city', true ); // True is for returning
a single value.
    //
    // // Add the post meta if it doesn't exist.
    // if ( $city && '' === $city ) {
    //     add_post_meta( $post_id, 'city', $city, true ); // True means the key is unique to
the post. False is default and more than one can be added.
    // }
    // // Edit the post meta if it does exist and there is a new value.
    // elseif ( $city && $city != $current_city_value ) {
    //     update_post_meta( $post_id, 'city', $city );
    // }
    // // Delete the post meta if there is no new value.
    // elseif ( '' === $city && $current_city_value ) {
    //     delete_post_meta( $post_id, 'city', $current_city_value ); // $current_city_value
is optional and is used to differentiate between other values with the same key.
    // }
}

add_action( 'save_post', 'myplugin_save_meta_data' );
```

Lire Meta Box en ligne: https://riptutorial.com/fr/wordpress/topic/9611/meta-box

Chapitre 49: Mettre à jour WordPress manuellement

Examples

VIA FTP

1. Téléchargez la version souhaitée de WordPress à partir de www.wordpress.org sur votre ordinateur local et décompressez le fichier.
    - Gardez également une sauvegarde de votre version actuelle ... au cas où.
2. Connectez-vous à votre site Web avec votre client FTP préféré (FileZilla est populaire et facile, mais tout client FTP sera correct).
    - *Les instructions pour cela sont hors du domaine d'application de WordPress, mais peuvent être trouvées à une date ultérieure dans le sujet FTP proposé* .
3. Téléchargez les dossiers (et leur contenu) intitulés "wp-admin" et "wp-includes" dans leurs répertoires correspondants sur votre serveur. Veillez à écraser les dossiers actuels.
    - Vous pouvez omettre de télécharger le dossier "wp-content", sauf si vous choisissez d'utiliser l'un des thèmes inclus. Si vous souhaitez mettre à jour / télécharger les thèmes par défaut inclus dans la version choisie, vous devez également télécharger ce dossier.
4. Téléchargez les fichiers individuels dans le dossier d'accueil (index.php, wp - *. Php, etc.).
    - Vous pouvez omettre les fichiers intitulés "liscense.txt" et "readme.html" car ils ne sont pas obligés de fonctionner et ils peuvent être utilisés comme méthodes pour déterminer votre version de WP pour les exploits de sécurité.
5. Visitez et connectez-vous à votre site Web pour effectuer les mises à jour de base de données requises.
    - Toutes les mises à jour de WP ne comportent pas de modifications de base de données, mais certaines le sont.

Remarque : cette méthode crée des fichiers orphelins qui peuvent / vont se constituer avec le temps et peuvent présenter des risques de sécurité. Assurez-vous de faire une comparaison de fichiers après la fin et de supprimer les anciens fichiers de votre serveur des versions précédentes de WP qui ne sont plus utilisés.

Lire Mettre à jour WordPress manuellement en ligne:
https://riptutorial.com/fr/wordpress/topic/8663/mettre-a-jour-wordpress-manuellement

Chapitre 50: Migration de site

Syntaxe

- OLD_SITE - L'ancien site en cours de migration (par exemple: http: // localhost / example)
- NEW_SITE - Le nouveau site vers lequel migrer (ex: https://example.com

Examples

Mise à jour des tables avec une nouvelle URL

```
UPDATE wp_options SET option_value = replace(option_value, 'OLD_SITE, 'NEW_SITE') WHERE
option_name = 'home' OR option_name = 'siteurl';
UPDATE wp_posts SET guid = replace(guid, 'OLD_SITE','NEW_SITE');
UPDATE wp_posts SET post_content = replace(post_content, 'OLD_SITE', 'NEW_SITE');
```

Lire Migration de site en ligne: https://riptutorial.com/fr/wordpress/topic/9610/migration-de-site

Chapitre 51: Notions de base sur le Customizer (Ajouter un panneau, une section, un paramètre, un contrôle)

Examples

Ajouter un panneau de personnalisation

```
<?php
/**
 * Panel: WPCustomize
 *
 * Basic Customizer panel with basic controls.
 *
 * @since    1.0.0
 * @package  WPC
 */

// Exit if accessed directly.
if ( ! defined( 'ABSPATH' ) ) {
    exit;
}

// Customize function.
if ( ! function_exists( 'wpc_panel_wpcustomize' ) ) {
    // Customize Register action.
    add_action( 'customize_register', 'wpc_panel_wpcustomize' );
    /**
     * Customize Panel.
     *
     * Adds a Panel, Section with basic controls.
     *
     * @param  object WP_Customize $wp_customize Instance of the WP_Customize_Manager class.
     * @since  1.0.0
     */
    function wpc_panel_wpcustomize( $wp_customize ) {
        // Panel: Basic.
        $wp_customize->add_panel( 'wpc_panel_wpcustomize', array(
            'priority'       => 10,
            'title'          => __( 'WPCustomize Panel Title', 'WPC' ),
            'description'    => __( 'Panel Description', 'WPC' ),
            'capability'     => 'edit_theme_options'
        ) );
    }
}
```

Ajouter une section de personnalisation avec les paramètres de base et leurs contrôles

Les panneaux peuvent avoir des sections, les sections peuvent avoir des paramètres et les paramètres peuvent avoir des contrôles. Les paramètres sont enregistrés dans la base de

données, tandis que les contrôles des paramètres particuliers ne sont utilisés que pour afficher les paramètres correspondants pour l'utilisateur.

Ce code crée une `section` base dans le `panel` ci-dessus. À l'intérieur se trouvent quelques `settings` base avec des `controls` attachés.

```
<?php
/**
 * Section: Basic
 *
 * Basic Customizer section with basic controls.
 *
 * @since     1.0.0
 * @package   WPC
 */

// Exit if accessed directly.
if ( ! defined( 'ABSPATH' ) ) {
    exit;
}

// Customize function.
if ( ! function_exists( 'wpc_customize_panel_basic' ) ) {
    // Customize Register action.
    add_action( 'customize_register', 'wpc_customize_panel_basic' );

    /**
     * Customize Panel.
     *
     * Adds a Panel, Section with basic controls.
     *
     * @param  object WP_Customize $wp_customize Instance of the WP_Customize_Manager class.
     * @since  1.0.0
     */
    function wpc_customize_panel_basic( $wp_customize ) {
        // Section: Basic.
        $wp_customize->add_section( 'wpc_section_basic', array(
            'priority'       => 10,
            'panel'          => 'wpc_panel_wpcustomize',
            'title'          => __( 'Basic Section Title', 'WPC' ),
            'description'    => __( 'Section Description.', 'WPC' ),
            'capability'     => 'edit_theme_options'
        ) );

        // Setting: Text.
        $wp_customize->add_setting( 'wpc_text', array(
            'type'                 => 'theme_mod',
            'default'              => 'Placeholder.',
            'transport'            => 'refresh', // Options: refresh or postMessage.
            'capability'           => 'edit_theme_options',
            'sanitize_callback'    => 'esc_attr'
        ) );

        // Control: Text.
        $wp_customize->add_control( 'wpc_text', array(
            'label'       => __( 'Text', 'WPC' ),
            'description' => __( 'Description', 'WPC' ),
            'section'     => 'wpc_section_basic',
            'type'        => 'text'
        ) );
```

```
        // Setting: Textarea.
        $wp_customize->add_setting( 'wpc_textarea', array(
            'type'                 => 'theme_mod',
            'default'              => 'Placeholder textarea.',
            'transport'            => 'refresh', // Options: refresh or postMessage.
            'capability'           => 'edit_theme_options',
            'sanitize_callback'    => 'exc_textarea'
        ) );

        // Control: Textarea.
        $wp_customize->add_control( 'wpc_textarea', array(
            'label'       => __( 'Textarea', 'WPC' ),
            'description' => __( 'Description', 'WPC' ),
            'section'     => 'wpc_section_basic',
            'type'        => 'textarea'
        ) );

        // Setting: Checkbox.
        $wp_customize->add_setting( 'wpc_checkbox', array(
            'type'                 => 'theme_mod',
            'default'              => 'enable',
            'transport'            => 'refresh', // Options: refresh or postMessage.
            'capability'           => 'edit_theme_options',
            'sanitize_callback'    => 'wpc_sanitize_checkbox' // Custom function in
customizer-sanitization.php file.
        ) );

        // Control: Checkbox.
        $wp_customize->add_control( 'wpc_checkbox', array(
            'label'       => __( 'Checkbox', 'WPC' ),
            'description' => __( 'Description', 'WPC' ),
            'section'     => 'wpc_section_basic',
            'type'        => 'checkbox'
        ) );

        // Setting: Radio.
        $wp_customize->add_setting( 'wpc_radio', array(
            'type'                 => 'theme_mod',
            'default'              => 'on',
            'transport'            => 'refresh', // Options: refresh or postMessage.
            'capability'           => 'edit_theme_options',
            'sanitize_callback'    => 'wpc_sanitize_select', // Custom function in customizer-
sanitization.php file.
        ) );

        // Control: Radio.
        $wp_customize->add_control( 'wpc_radio', array(
            'label'       => __( 'Radio', 'WPC' ),
            'description' => __( 'Description', 'WPC' ),
            'section'     => 'wpc_section_basic',
            'type'        => 'radio',
            'choices'  => array(
                'enable'  => 'Enable',
                'disable' => 'Disable'
            )
        ) );

        // Setting: Select.
        $wp_customize->add_setting( 'wpc_select', array(
            'type'                 => 'theme_mod',
```

```
            'default'              => 'enable',
            'transport'            => 'refresh', // Options: refresh or postMessage.
            'capability'           => 'edit_theme_options',
            'sanitize_callback'    => 'wpc_sanitize_select' // Custom function in customizer-
sanitization.php file.
        ) );

        // Control: Select.
        $wp_customize->add_control( 'wpc_select', array(
            'label'       => __( 'Select', 'WPC' ),
            'description' => __( 'Description', 'WPC' ),
            'section'     => 'wpc_section_basic',
            'type'        => 'select',
            'choices'  => array(
                'enable'  => 'Enable',
                'disable' => 'Disable'
            )
        ) );
    }
}
```

Lire Notions de base sur le Customizer (Ajouter un panneau, une section, un paramètre, un contrôle) en ligne: https://riptutorial.com/fr/wordpress/topic/2930/notions-de-base-sur-le-customizer--ajouter-un-panneau--une-section--un-parametre--un-controle-

Chapitre 52: Options API

Introduction

Les options sont des éléments de données que WordPress utilise pour stocker diverses préférences et paramètres de configuration. L'API Options est un moyen simple et standardisé de stocker des données dans la base de données. L'API facilite la création, l'accès, la mise à jour et la suppression des options.

Syntaxe

- // Créer une nouvelle option dans WordPress
  add_option ($ option, $ value =, $ deprecated =, $ autoload = 'yes');
- // Supprime une option de la base de données.
  delete_option (option $)
- // Récupère une option enregistrée
  get_option ($ option, $ default = false);
- // Met à jour la valeur d'une option déjà ajoutée.
  update_option ($ option, $ newvalue);
- // Il existe également des versions * _site_option () de ces fonctions,
  // manipuler les options du réseau dans WordPress Multisite
- // Créer une nouvelle option de réseau
  add_site_option (option $, $ value =, $ deprecated =, $ autoload = 'yes');
- // Supprime une option réseau
  delete_site_option (option $);
- // Récupère une option de réseau enregistrée
  get_site_option ($ option, $ default = false);
- // Met à jour la valeur d'une option déjà ajoutée.
  update_site_option (option $, $ newvalue);

Remarques

L'API Options est un moyen simple et standardisé de travailler avec des données stockées dans la table d'options de la base de données MySQL. L'API facilite la création, la lecture, la mise à jour et la suppression des options.

Examples

get_option

La fonction get_option permet de récupérer une valeur depuis la table d'options en fonction du nom de l'option.

Vous pouvez utiliser le code suivant pour obtenir l'adresse électronique d'un administrateur de site WordPress.

```
<?php echo get_option('admin_email'); ?>
```

`get_option()` a un deuxième argument facultatif, qui vous permet de définir une valeur par défaut à renvoyer dans le cas où l'option demandée n'est pas définie. Par défaut, cet argument est `false` .

Pour récupérer une chaîne de texte et utiliser une chaîne standard si le texte n'est pas défini dans la table d'options, vous pouvez le faire:

```
<?php get_option( 'my_text', "I don't have anything written. Yet." ); ?>
```

add_option

La fonction add_option ins permet d'insérer une nouvelle ligne dans la table d'options.

Cela insérera une nouvelle ligne dans la table d'options avec le nom de l'option *nom_option_prix* et la valeur *some_option_value*

```
<?php add_option( 'some_option_name', 'some_option_value' ); ?>
```

delete_option

La fonction delete_option est utilisée pour supprimer une option du tableau d'options.

Cela supprimera *my_custom_option* de la table d'options.

```
<?php delete_option( 'my_custom_option' ); ?>
```

update_option

La fonction update_option est utilisée pour mettre à jour une valeur qui existe déjà dans la table d'options. Si l'option n'existe pas, alors l'option sera ajoutée avec la valeur de l'option.

Cela définira le statut de commentaire par défaut sur «fermé»:

```
update_option( 'default_comment_status', 'closed' );
```

Lire Options API en ligne: https://riptutorial.com/fr/wordpress/topic/7854/options-api

Chapitre 53: Petit code

Examples

Enregistrement de shortcode

Shortcode est un petit morceau de code qui peut être ajouté dans l'éditeur WordPress et qui affichera quelque chose de différent une fois la page publiée ou prévisualisée.

Souvent, les shortcodes sont ajoutés au fichier `functions.php` du thème, mais ce n'est pas une bonne pratique car les raccourcis sont censés continuer à fonctionner après avoir changé de thème. Au lieu de cela, écrivez un plugin pour ajouter cette fonctionnalité.

La structure d'enregistrement du shortcode est la suivante:

```
function new_shortcode($atts, $content = null){
    // if parameters are needed in the shortcode
    // parameters can be set to default to something
    extract( shortcode_atts( array(
        'param_one' => 'h1'
    ), $atts ) );
    $shortcode = '<'.$param_one'>'.$content.'</'.$param_one.'>';
    return $shortcode;
}
// this is what registers the shortcode with wordpress
add_shortcode('demo-shortcode','new_shortcode');
```

Dans l'éditeur WordPress, vous pouvez taper:

```
[demo-shortcode param_one="h2"]Demo[/demo-shortcode]
// you don't need to insert param_one into the editor if it has a default value.
// having it in the editor will override the default
```

Une fois la page publiée, cela se transformera en

```
<h2>Demo</h2>
```

Utiliser des raccourcis dans WordPress Backend

```
[footag foo="value of 1" attribute-2="value of 2"]
```

Dans wordpress admin, nous utilisons des shortcodes prédéfinis en écrivant le nom du shortcode dans des crochets et en y ajoutant éventuellement des attributs en les séparant par des espaces.

Ajout de nouveaux raccourcis

```
function footag_func( $atts ) {
    return "foo = {$atts['foo']}";
```

```
}
add_shortcode( 'footag', 'footag_func' );
```

Dans les plugins, nous pouvons ajouter des shortcodes en utilisant la fonction add_shortcode.

Le shortcode peut être utilisé dans n'importe quelle page Wordpress ou poste simplement en le plaçant entre crochets.

```
[footag]
```

Utiliser des shortcodes à l'intérieur du code PHP (thèmes et plugins)

```
<?php echo do_shortcode("[footag foo='Hi! I am a foo output']"); ?>
```

Pour imprimer un shortcode à l'aide de php, utilisez la fonction `do_shortcode` et `do_shortcode` écho à la valeur renvoyée.

Utilisation de raccourcis dans les widgets

```
add_filter( 'widget_text', 'shortcode_unautop' );
add_filter( 'widget_text', 'do_shortcode' );enter code here
```

Ajoutez ceci à un plugin ou au fichier `functions.php` pour activer les shortcodes dans les widgets. Le code arrête d'abord WordPress en tournant les sauts de ligne en balises de paragraphe et laisse ensuite les codes abrégés analyser les widgets. L'ordre des deux lignes est important.

Lire Petit code en ligne: https://riptutorial.com/fr/wordpress/topic/4952/petit-code

Chapitre 54: Scripts de mise en file d'attente

Syntaxe

- wp_enqueue_script ($ handle, $ src, $ deps, $ ver, $ in_footer)

Paramètres

Paramètre	Détails
$ handle	*(chaîne)* (obligatoire) Nom du script. Devrait être unique.
$ src	*(string)* (Facultatif) URL complète du script ou chemin du script relatif au répertoire racine de WordPress. *Valeur par défaut: false*
$ deps	*(array)* (Facultatif) Un tableau de descripteurs de script enregistrés dépend de ce script. *Valeur par défaut: array ()*
$ ver	*(string | bool | null)* (Facultatif) Chaîne spécifiant le numéro de version du script, s'il en a un, qui est ajouté à l'URL en tant que chaîne de requête à des fins de contournement du cache. Si la version est définie sur false, un numéro de version est automatiquement ajouté égal à la version WordPress installée actuelle. Si défini sur null, aucune version n'est ajoutée. *Valeur par défaut: false*
$ in_footer	*(bool)* (Facultatif) Indique si le script doit être mis en file d'attente avant `</body>` plutôt que dans `<head>` . *Valeur par défaut: false*

Examples

Saisie de scripts dans functions.php

Si vous souhaitez ajouter `custom.js` script `custom.js` situé dans le dossier `js/` de votre thème, vous devez le mettre en file d'attente. Dans `functions.php` add

```
<?php

add_action( 'after_setup_theme', 'yourtheme_theme_setup' );

if ( ! function_exists( 'yourtheme_theme_setup' ) ) {
    function yourtheme_theme_setup() {

        add_action( 'wp_enqueue_scripts', 'yourtheme_scripts' );
        add_action( 'admin_enqueue_scripts', 'yourtheme_admin_scripts' );

    }
}
```

```
if ( ! function_exists( 'yourtheme_scripts' ) ) {
    function yourtheme_scripts() {

        wp_enqueue_script( 'yourtheme_custom', get_template_directory_uri().'/js/custom.js',
array( 'jquery' ), '1.0.0', true );

    }
}

if ( ! function_exists( 'yourtheme_admin_scripts' ) ) {
    function yourtheme_admin_scripts() {

        wp_enqueue_script( 'yourtheme_custom', get_template_directory_uri().'/js/custom.js',
array( 'jquery-ui-autocomplete', 'jquery' ), '1.0.0', true );

    }
}
```

Mettre en file d'attente les scripts pour IE uniquement

```
add_action( 'wp_enqueue_scripts', 'enqueue_my_styles_and_scripts' );

/**
 * Enqueue scripts (or styles) conditionally.
 *
 * Load scripts (or stylesheets) specifically for IE. IE10 and above does
 * not support conditional comments in standards mode.
 *
 * @link https://gist.github.com/wpscholar/4947518
 * @link https://msdn.microsoft.com/en-us/library/ms537512(v=vs.85).aspx
 */
function enqueue_my_styles_and_scripts() {

     // Internet Explorer HTML5 support
    wp_enqueue_script( 'html5shiv',get_template_directory_uri().'/js/html5shiv.js', array(),
'3.7.3', false);
    wp_script_add_data( 'html5shiv', 'conditional', 'lt IE 9' );

    // Internet Explorer 8 media query support
    wp_enqueue_script( 'respond', get_template_directory_uri().'/js/respond.js', array(),
'1.4.2', false);
    wp_script_add_data( 'respond', 'conditional', 'lt IE 9' );

}
```

Mise en place des scripts sous certaines conditions pour des pages spécifiques

Vous pouvez utiliser des opérateurs conditionnels dans WordPress pour mettre en file d'attente des scripts sur des pages spécifiques de votre site Web.

```
function load_script_for_single_post(){
    if(is_single()){
        wp_enqueue_script(
                'some',
                get_template_directory_uri().'/js/some.js',
```

```
                array('jquery),
                '1.0.0',
                false
        );

    }
}
add_action('wp_enqueue_scripts','load_script_for_single_post');
```

Dans l'exemple ci-dessus, si la page Web en cours est à publication unique, le script sera mis en file d'attente. Sinon, la fonction *wp_enqueue_script* ne sera pas exécutée.

Lire Scripts de mise en file d'attente en ligne:
https://riptutorial.com/fr/wordpress/topic/1103/scripts-de-mise-en-file-d-attente

Chapitre 55: Sécurisez votre installation

Remarques

Les sites Web WordPress sont fréquemment piratés. Cette rubrique concerne les techniques et les pratiques qui augmentent la sécurité de votre installation WordPress au-delà de ce qui est réalisé dans une installation de base.

En dehors de ce sujet, un autre bon endroit pour lire sur la sécurisation d'une installation WordPress est la page Hardening WordPress Codex .

Examples

Désactiver l'éditeur de fichier

L'éditeur de fichiers fourni avec WordPress est un risque de sécurité. Si un attaquant obtient un accès administrateur à votre site WordPress, il pourra facilement insérer du code malveillant dans les fichiers de thème et de plug-in. C'est également un risque pour les clients qui ne savent pas ce qu'ils font. Une fois le côlon mal placé dans l'éditeur de fichiers peut casser un site et le rendre inaccessible depuis le navigateur.

Dans votre fichier WordPress `wp-config.php` , désactivez l'éditeur de fichiers en ajoutant la ligne de code suivante.

```
define( 'DISALLOW_FILE_EDIT', true );
```

Cette ligne aura l'effet désiré lorsqu'elle sera ajoutée au fichier `functions.php` votre thème, mais il est préférable d'ajouter à `wp-config.php` .

Si vous utilisez WordPress CLI pour installer WordPress, vous pouvez utiliser la commande suivante pour créer un fichier `wp-config.php` avec une modification de fichier désactivée.

```
/* declare variables beforehand or substitute strings in */
wp core config --dbname="$MYSQL_DBNAME" --dbuser="$MYSQL_USERNAME" --dbpass="$MYSQL_PASS" --
dbprefix="$WP_DBPREFIX"_ --locale=en_AU --extra-php <<PHP
define( 'DISALLOW_FILE_EDIT', true );
PHP
```

Cette méthode est utile si vous installez WordPress avec un script.

Déplacer wp-config.php

Les informations les plus sensibles d'une installation WordPress sont stockées dans le fichier `wp-config.php` . Si un pirate a accès à ce fichier, il contrôle totalement votre site Web.

Par défaut, `wp-config.php` est stocké dans le dossier d'installation de WordPress. Pour rendre ce

fichier plus difficile à voler, vous pouvez le sortir du dossier accessible par le Web. Si vous ne le déplacez que d'un seul dossier, WordPress le trouvera automatiquement. Si vous déplacez `wp-config.php` vers un autre emplacement, créez un fichier vide appelé `wp-config.php` dans le dossier d'installation de WordPress. Puis ajoutez les éléments suivants:

```
define('ABSPATH', dirname(__FILE__) . '/');
// '../../wp-config.php' defines location two folders above installation folder.
// Substitute with actual location of wp-config.php file as necessary.
require_once(ABSPATH . '../../wp-config.php');
```

Vous devrez peut-être faire en sorte que `php` exécutable dans le dossier dans lequel vous placez wp-config.php. Vous devriez créer un exécutable php dans le moins de dossiers possible. Un bon système place l'installation de WordPress dans `/path/to/wordpress/install/` et la configuration dans `/path/to/wordpress/config` . Assurez-vous que le dossier de configuration n'est pas accessible sur le Web et ne placez aucune autre information sensible placée dans `/path/to/` ou plus haut dans la hiérarchie des dossiers. Dans ce cas, vous écrivez une ligne similaire à la suivante dans votre `php.ini` :

```
open_basedir = "/path/to/wordpress/install/;/path/to/wordpress/config"
```

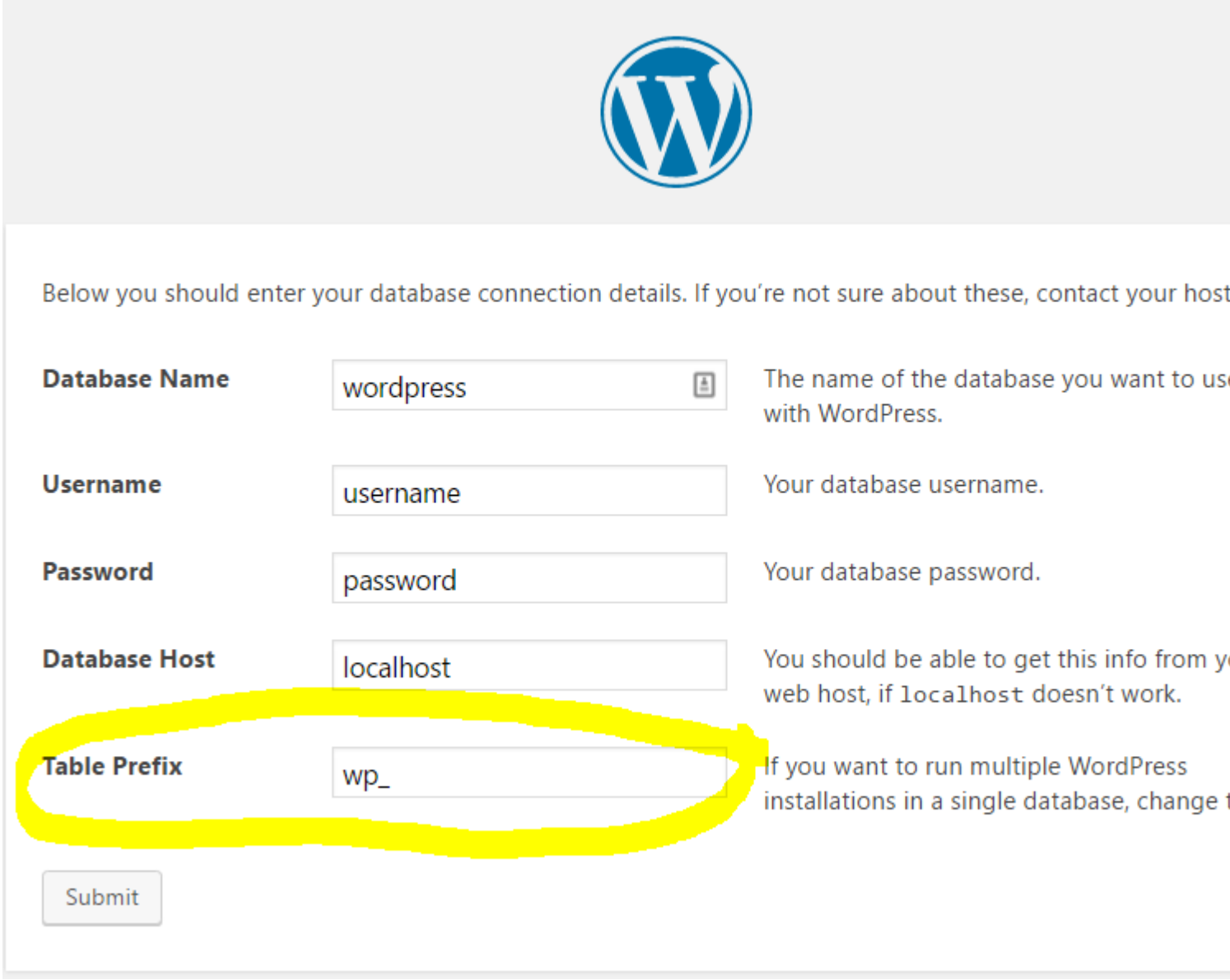
Cette technique est controversée et certaines personnes ne pensent pas que cela améliore la sécurité. Une discussion approfondie sur le sujet peut être lue à cette question WordPress StackExchange .

Définir un préfixe personnalisé pour les tables WordPress

Lorsque vous installez WordPress sur votre serveur, le script d'installation place un préfixe devant tous les noms de tables MySQL WordPress. Ce préfixe est défini sur 'wp_' par défaut. La table des publications WordPress s'appellera par exemple `wp_posts` . En modifiant le préfixe de la table, vous pouvez créer une certaine sécurité par obscurité. De cette façon, lorsqu'un pirate tente d'attaquer des attaques par injection SQL, il devra deviner le préfixe de votre table plutôt que d'utiliser simplement «wp_». Vous pouvez définir ce préfixe comme vous le souhaitez.

Définir le préfixe dans une nouvelle installation WordPress

Si vous utilisez une installation de 5 minutes, changez le préfixe dans le champ pendant l'installation.



Si vous installez via WordPress CLI, utilisez la commande suivante:

```
// set other variables above, or substitute your strings in.
WP_DBPREFIX=foo
wp core config --dbname="$MYSQL_DBNAME" --dbuser="$MYSQL_USERNAME" --dbpass="$MYSQL_PASS" --
dbprefix="$WP_DBPREFIX"_ --locale=en_AU
```

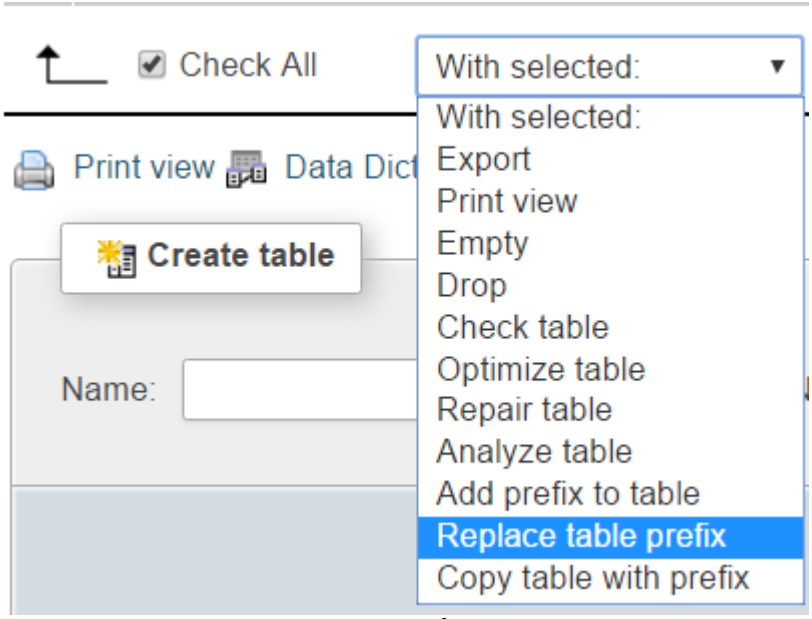
Modifier le préfixe dans une installation existante

Changer le préfixe est un peu plus difficile. Tout d'abord, utilisez un programme FTP tel que FileZilla pour éditer le fichier `wp-config.php` . Changez l'entrée `$table_prefix = 'wp_';` à `$table_prefix = 'foo_';` substituer «foo» à votre préfixe désiré.

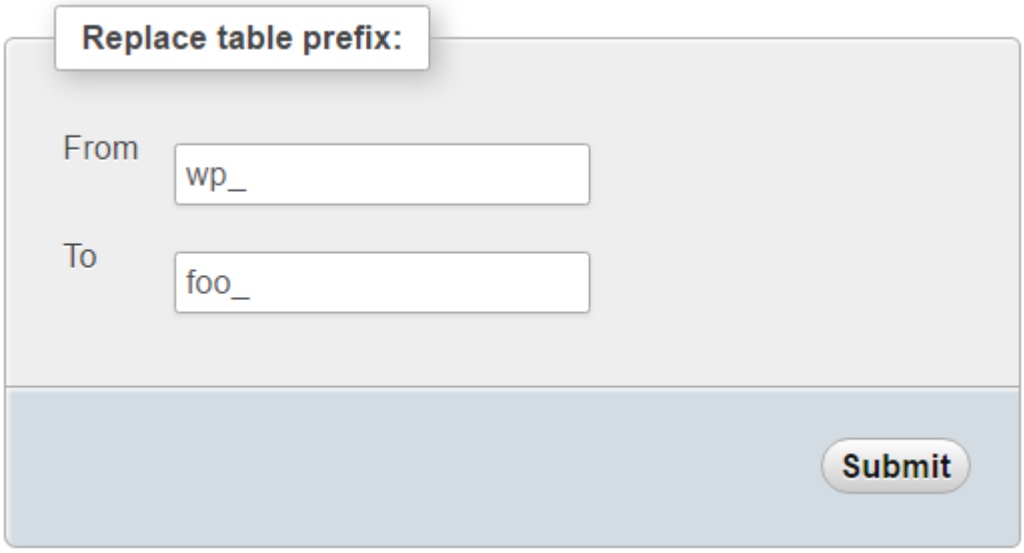
Ensuite, nous devrons modifier la base de données. Si vous avez accès à phpMyAdmin, connectez-vous et procédez comme suit:

- Sélectionnez la base de données WordPress

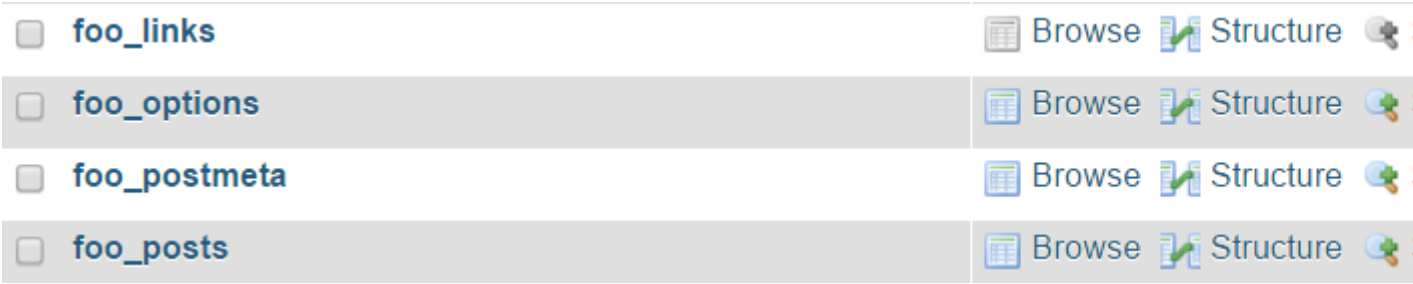
- Sélectionnez toutes les tables et, dans la liste déroulante, sélectionnez le préfixe de la table.



- Dans "De" tapez "wp_". Dans "À" tapez votre préfixe, "foo_" dans cet exemple et appuyez sur "Soumettre".



- Les tableaux doivent maintenant ressembler à ceci:



Si vous ne pouvez pas utiliser phpMyAdmin, utilisez la commande MySQL suivante:

```
RENAME table `wp_comments` TO `foo_comments`
```

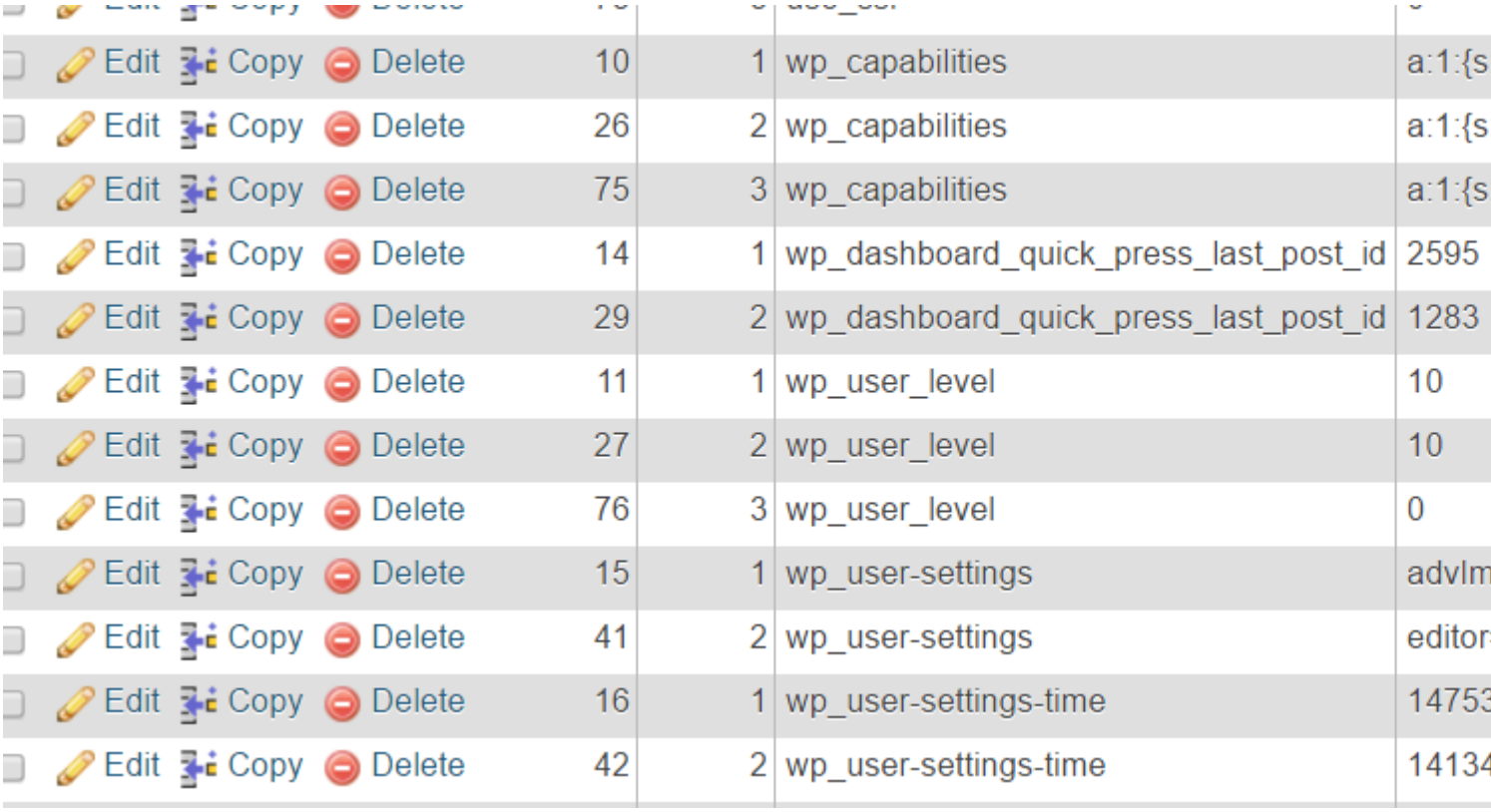
Vous devrez exécuter cette commande pour chaque table, en remplaçant les autres noms de table par des commentaires.

Ensuite, nous devons modifier quelques entrées dans certaines tables. Exécutez cette requête sur la table 'foo_options'

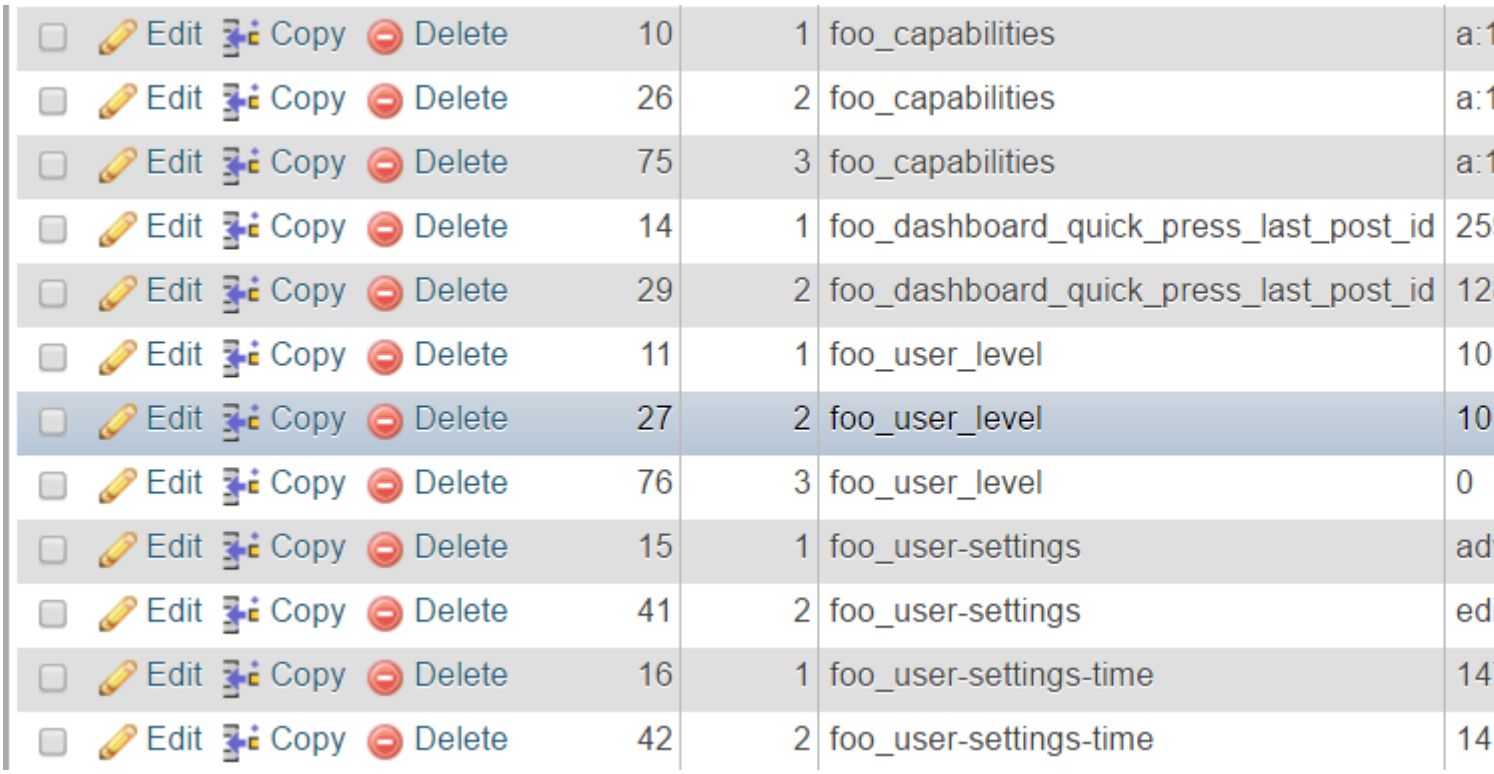
```
SELECT * FROM  foo_options WHERE option_name LIKE '%user_roles%'
```

Une entrée avec option_name de 'wp_user_roles' devrait apparaître. Dans cette entrée, remplacez l'entrée 'nom_option' de `wp_user_roles` par `foo_user_roles` .

Ouvrez ensuite le tableau 'foo_usermeta' et trouvez chaque entrée avec 'wp_' au début.



et changez-le en 'foo_'. Le nombre d'entrées à modifier dépend du nombre d'utilisateurs que vous avez.



Edit Copy Delete	10	1	foo_capabilities	a:1
Edit Copy Delete	26	2	foo_capabilities	a:1
Edit Copy Delete	75	3	foo_capabilities	a:1
Edit Copy Delete	14	1	foo_dashboard_quick_press_last_post_id	25
Edit Copy Delete	29	2	foo_dashboard_quick_press_last_post_id	12
Edit Copy Delete	11	1	foo_user_level	10
Edit Copy Delete	27	2	foo_user_level	10
Edit Copy Delete	76	3	foo_user_level	0
Edit Copy Delete	15	1	foo_user-settings	ad
Edit Copy Delete	41	2	foo_user-settings	ed
Edit Copy Delete	16	1	foo_user-settings-time	14
Edit Copy Delete	42	2	foo_user-settings-time	14

Cela devrait être tout ce dont vous avez besoin pour changer le préfixe dans une installation existante

Lire Sécurisez votre installation en ligne: https://riptutorial.com/fr/wordpress/topic/7594/securisez-votre-installation

Chapitre 56: Sécurité dans WordPress - S'échapper

Syntaxe

- esc_html (chaîne $ text)
- esc_url (chaîne $ url, tableau $ protocoles, chaîne $ _context)
- esc_js (chaîne $ text)
- wp_json_encode (mixte $ data, int $ options, int $ depth = 512)
- esc_attr (chaîne $ text)
- esc_textarea (chaîne $ text)

Remarques

La sécurité doit toujours être à l'esprit lors du développement. Sans sécurité, une application est ouverte à diverses attaques telles que des injections SQL, XSS, CSRF, RFI, etc. qui peuvent entraîner de graves problèmes.

Les données non fiables proviennent de nombreuses sources (utilisateurs, sites tiers, votre propre base de données! ...) et tout cela doit être validé à la fois en entrée et en sortie. (Source: WordPress Codex)

Les données doivent être validées, désinfectées ou échappées en fonction de l'utilisation et du but.

Pour valider, vous devez vous assurer que les données que vous avez demandées à l'utilisateur correspondent à celles qu'elles ont soumises. (Source: WordPress Codex)

La désinfection est une approche un peu plus libérale de l'acceptation des données utilisateur. Nous pouvons recourir à ces méthodes lorsqu'il existe une gamme de données acceptables. (Source: WordPress Codex)

Echapper, c'est prendre les données que vous possédez déjà et aider à les sécuriser avant de les rendre à l'utilisateur final. (Source: WordPress Codex)

Examples

données d'échappement dans le code HTML

esc_html devrait être utilisé chaque fois que nous produisons des données à l'intérieur du code HTML.

```
<h4><?php echo esc_html( $title ); ?></h4>
```

échapper à une URL

```
<a href="<?php echo esc_url( home_url( '/' ) ); ?>">Home</a>

<img src="<?php echo esc_url( $user_picture_url ); ?>" />
```

données d'échappement en code js

`esc_js()` est destiné à être utilisé pour JS en ligne, à l'intérieur d'un attribut de balise.

Pour les données `wp_json_encode()` dans une `<script>` utilisez `wp_json_encode()` .

```
<input type="text" onfocus="if( this.value == '<?php echo esc_js( $fields['input_text'] ); ?>'
) { this.value = ''; }" name="name">
```

`wp_json_encode()` code une variable dans JSON, avec quelques vérifications de cohérence.

Notez que `wp_json_encode()` inclut automatiquement les guillemets de délimitation de chaîne.

```
<?php
$book = array(
    "title" => "JavaScript: The Definitive Guide",
    "author" => "Stack Overflow",
);
?>
<script type="text/javascript">
var book = <?php echo wp_json_encode($book) ?>;
/* var book = {
    "title": "Security in WordPress",
    "author" => "Stack Overflow",
}; */
</script>
```

ou

```
<script type="text/javascript">
    var title = <?php echo wp_json_encode( $title ); ?>;
    var content = <?php echo wp_json_encode( $content ); ?>;
    var comment_count = <?php echo wp_json_encode( $comment_count ); ?>;
</script>
```

attributs d'échappement

```
<input type="text" value="<?php echo esc_attr($_POST['username']); ?>" />
```

données d'échappement dans textarea

```
<textarea><?php echo esc_textarea( $text ); ?></textarea>
```

Lire Sécurité dans WordPress - S'échapper en ligne:

https://riptutorial.com/fr/wordpress/topic/6115/securite-dans-wordpress---s-echapper

Chapitre 57: Sécurité dans WordPress - Sanitization

Syntaxe

- sanitize_text_field (string $ str)
- sanitize_title (chaîne $ title, chaîne $ fallback_title, chaîne $ context)
- sanitize_email (chaîne $ email)
- sanitize_html_class (string $ class, string $ fallback)
- sanitize_file_name (string $ name)
- sanitize_user (chaîne $ nom_utilisateur, booléen $ strict)

Remarques

La sécurité doit toujours être à l'esprit lors du développement. Sans sécurité, une application est ouverte à diverses attaques telles que des injections SQL, XSS, CSRF, RFI, etc. qui peuvent entraîner de graves problèmes.

Les données non fiables proviennent de nombreuses sources (utilisateurs, sites tiers, votre propre base de données! ...) et tout cela doit être validé à la fois en entrée et en sortie. (Cours: WordPress Codex)

Les données doivent être validées, désinfectées ou échappées en fonction de l'utilisation et du but.

Pour valider, vous devez vous assurer que les données que vous avez demandées à l'utilisateur correspondent à celles qu'elles ont soumises. (Cours: WordPress Codex)

La désinfection est une approche un peu plus libérale de l'acceptation des données utilisateur. Nous pouvons recourir à ces méthodes lorsqu'il existe une gamme de données acceptables. (Cours: Wordpress Codex)

Echapper, c'est prendre les données que vous possédez déjà et aider à les sécuriser avant de les rendre à l'utilisateur final. (Cours: WordPress Codex)

Examples

Assainir le champ de texte

```
$title = sanitize_text_field( $_POST['title'] );
```

Désinfecter le titre

La valeur renvoyée est destinée à être utilisée dans une URL, et non en tant que titre lisible par

l'homme. Utilisez plutôt sanitize_text_field.

```
$new_url = sanitize_title($title);
```

Désinfecter l'email

```
$sanitized_email = sanitize_email('     admin@example.com!     ');
```

Sanitize html class

```
$post_class = sanitize_html_class( $post->post_title );
echo '<div class="' . $post_class . '">';
```

Désinfecter le nom du fichier

```
$incfile = sanitize_file_name($_REQUEST["file"]);
include($incfile . ".php");
```

Sans désinfecter le nom du fichier, un attaquant pourrait simplement transmettre http: // site_attaquant / malicous_page en entrée et exécuter le code de votre serveur.

Désinfecter le nom d'utilisateur

```
$user = sanitize_user("attacker username<script>console.log(document.cookie)</script>");
```

$ user value après sanitize est "nom d'utilisateur attaquant"

Lire Sécurité dans WordPress - Sanitization en ligne:
https://riptutorial.com/fr/wordpress/topic/6348/securite-dans-wordpress---sanitization

Chapitre 58: Shortcode avec attribut

Syntaxe

- add_shortcode ('your_short_code', 'your_function_name');

Paramètres

Paramètres	Discription et utilisation
$ tag	(chaîne) (obligatoire) Balise de code court à rechercher dans le contenu de la publication Valeur par défaut: Aucune
$ func	(appelable) (obligatoire) Crochet pour s'exécuter lorsque le shortcode est trouvé Par défaut: Aucun

Remarques

IMPORTANT - N'utilisez pas camelCase ou UPPER-CASE pour vos attributs

Vous pouvez générer un shortcode avec l'attribut Here

Examples

Exemples de shortcodes

Les codes courts WordPress ont été introduits dans 2.5

Voici un exemple de shortcode

```
[button]
```

pour utiliser le shortcode directement dans le thème, vous devez utiliser `do_shortcode()`

```
<?php echo do_shortcode('[button]'); ?>
```

Pour personnaliser le bouton, nous pourrions simplement ajouter quelque chose comme:

```
[button type="twitter"]
```

Ou pour le rendre encore meilleur, nous pourrions utiliser un shortcode englobant:

```
[button type="twitter"]Follow me on Twitter![/button]
```

Créer un shortcode à fermeture automatique

Le shortcode le plus simple est celui à fermeture automatique. Nous allons créer un simple lien vers notre compte Twitter, puis l'ajouter à un article de blog. Tout le code va dans `functions.php` , qui se trouve dans `/wp-content/themes/your-theme/` . Si vous n'en avez pas, créez-le et insérez-le.

```
<?php
function button_shortcode() {
return '<a href="http://twitter.com/rupomkhondaker" class="twitter-button">Follow me on
Twitter!</a>"';
}
add_shortcode('button', 'button_shortcode');
?>
```

Utilisation: `[button]`

Créer un shortcode à fermeture automatique avec des paramètres

Créer un shortcode à fermeture automatique avec des paramètres

```
<?php
function button_shortcode( $type ) {

    extract( shortcode_atts(
        array(
            'type' => 'value'
         ), $type ) );

    // check what type user entered
    switch ($type) {

        case 'twitter':
            return '<a href="http://twitter.com/rupomkhondaker" class="twitter-button">Follw
me on Twitter!</a>';
            break;

        case 'rss':
            return '<a href="http://example.com/rss" class="rss-button">Subscribe to the
feed!</a>'
            break;
    }
}
add_shortcode( 'button', 'button_shortcode' );
?>
```

Vous pouvez maintenant choisir quel bouton afficher en définissant le type dans votre shortcode.

```
[button type="twitter"]
[button type="rss"]
```

Créer un shortcode englobant

code court

Le shortcode englobant vous permet d'intégrer du contenu dans votre shortcode, tout comme le BBCode si vous l'avez déjà utilisé.

```
<?php
function button_shortcode( $attr, $content = null ) {
return '<a href="http://twitter.com/filipstefansson" class="twitter-button">' . $content .
'</a>';
}
add_shortcode('button', 'button_shortcode');
?>
```

Pour utiliser ce shortcode, vous devez incorporer le texte que vous souhaitez utiliser comme suit:

```
[button]Follow me on Twitter![/button]
```

Pour rendre ce bouton encore meilleur, nous pourrions ajouter des paramètres comme nous l'avons fait dans l'exemple précédent.

```
<?php
function button_shortcode( $atts, $content = null ) {
extract( shortcode_atts( array(
'account' => 'account',
 'style' => 'style'
 ), $atts ) );
return '<a href="http://twitter.com/' . esc_attr($account) . '" class="twitter-button ' .
esc_attr($style) . '">' . $content . '</a>';
}
add_shortcode('button', 'button_shortcode');
?>
```

Usage:

```
[button account="rupomkhondaker" style="simple"]Follow me on Twitter![/button]
```

Shortcodes dans les widgets

Par défaut, WordPress ne prend pas en charge les codes abrégés dans les widgets de la barre latérale. Il ne fait que développer les codes courts dans le contenu d'un type de publication, de page ou personnalisé. Pour ajouter un support de shortcode aux widgets de la barre latérale, vous pouvez installer un plug-in ou utiliser le code ci-dessous:

```
add_filter( 'widget_text', 'shortcode_unautop' );
add_filter( 'widget_text', 'do_shortcode' );
```

Il est important que ces lignes soient ajoutées dans cet ordre. La première ligne empêche WordPress de transformer les sauts de ligne en balises de paragraphe, car cela empêche les shortcodes de fonctionner. La deuxième ligne est celle qui fait fonctionner les shortcodes.

Lire Shortcode avec attribut en ligne: https://riptutorial.com/fr/wordpress/topic/6291/shortcode-avec-attribut

Chapitre 59: Shortcodes

Examples

Introduction au shortcode

Les raccourcis sont utiles lorsque vous souhaitez pouvoir ajouter des éléments plus complexes en ligne dans l'éditeur de contenu normal.

Un shortcode dans sa forme la plus simple ressemble à ceci:

```
function my_shortcode( ){
    return "This is a shortcode";
}
add_shortcode( 'my_shortcode', 'my_shortcode' );
```

Il afficherait le texte `"This is a shortcode"` et vous l'utiliseriez en écrivant [my_shortcode] dans l'éditeur de contenu.

Bouton shortcode

Voici un exemple de code court de bouton simple:

```
<?php
function my_button_shortcode( $atts ) {

    // Parse the input attributes and assgn default values for the
    // attributes that are not specified on the shortcode
    $a = shortcode_atts( array(
        'id' => '',
        'url' => '#',
        'class' => '',
        'text' => ''
    ), $atts );

    // Open the anchor tag and add role=button for better accessibility
    $btn_html = '<a role="button"';

    // Add the href(link) attribute
    $btn_html .= ' href="' . $a['url'] . '"';

    // Add id attribute to output if specified
    if ( !empty( $a['id'] ) ) {
        $btn_html .= ' id="' . $a['id'] . '"';
    }

    $btn_classes = 'button';


    // Add class attribute to output
    $btn_html .= ' class="button ' . ( !empty(a['class']) ? $a['class'] : '' ) . '"';

    // Close opening anchor tag
```

```
    $btn_html .= '>'.
        // Add button text
        $a['text'].
        // Add closing anchor tag
        '</a>'."\n";

    return $btn_html;
}
add_shortcode( 'button', 'my_button_shortcode' );
```

Ce shortcode peut être utilisé en tapant `[button url="/my-other-page" id="my-other-page-button" class="dark" text="Click me!"]` Dans l'éditeur et affichera le HTML suivant:

```
<a role="button" href="/my-other-page" id="my-other-page-button"
class="button dark">Click me!</a>
```

Lire Shortcodes en ligne: https://riptutorial.com/fr/wordpress/topic/6070/shortcodes

Chapitre 60: Styles d'enveloppement

Syntaxe

1. wp_enqueue_style ($ handle, $ src, $ dependency, $ version, $ media);

Paramètres

Paramètre	Détails
`$handle`	(String) (Obligatoire) Nom unique de la feuille de style.
`$src`	(String) (Facultatif) URL de la feuille de style qui sera utilisée dans l'attribut src de la balise de **lien** .
`$deps`	(Chaîne) (Facultatif) Un tableau de styles prend en charge cette feuille de style.
`$ver`	(String) (Facultatif) Chaîne spécifiant la version de la feuille de style de la feuille de style.
`$media`	(Chaîne) (Facultatif) Le média pour lequel cette feuille de style est créée. par exemple "tout", "imprimer", "écran", etc.

Examples

Inclure un fichier css interne avec un autre fichier CSS en tant que dépendance

```
function themeSlug_enqueue_scripts() {
    wp_enqueue_style( 'themeSlug-reset', get_template_directory_uri() .'/css/reset.css',
'1.0.0' );
    wp_enqueue_style( 'themeSlug-style', get_template_directory_uri() .'/style.css',
'themeSlug-reset', '1.0.0');
}
add_action('wp_enqueue_scripts', 'themeSlug_enqueue_scripts');
```

Y compris le fichier css interne

Dans ce cas, `style.css` est situé à la racine du dossier du thème.

```
function themeSlug_enqueue_scripts() {
    wp_enqueue_style( 'themeSlug-style', get_template_directory_uri() .'/style.css', '1.0.0');
}
add_action('wp_enqueue_scripts', 'themeSlug_enqueue_scripts');
```

Y compris le fichier css externe

Dans cet exemple, nous voulons inclure la police police icon awesome

```
function themeSlug_enqueue_scripts() {
    wp_enqueue_style( 'font-awesome', '//cdnjs.cloudflare.com/ajax/libs/font-
awesome/4.6.3/css/font-awesome.css');
}
add_action('wp_enqueue_scripts', 'themeSlug_enqueue_scripts');
```

Enqueue feuilles de style pour IE uniquement

```
add_action( 'wp_enqueue_scripts', 'enqueue_my_styles_and_scripts' );

/**
 * Enqueue styles (or scripts) conditionally.
 *
 * Load stylesheets (or scripts) specifically for IE. IE10 and above does
 * not support conditional comments in standards mode.
 *
 * @link https://gist.github.com/wpscholar/4947518
 * @link https://msdn.microsoft.com/en-us/library/ms537512(v=vs.85).aspx
 */
function enqueue_my_styles_and_scripts() {

    // Internet Explorer specific stylesheet.
    wp_enqueue_style( 'themename-ie', get_stylesheet_directory_uri() . '/css/ie.css', array(
'twentyfifteen-style' ), '20141010' );
    wp_style_add_data( 'themename-ie', 'conditional', 'lte IE 9' );

    // Internet Explorer 7 specific stylesheet.
    wp_enqueue_style( 'themename-ie7', get_stylesheet_directory_uri() . '/css/ie7.css', array(
'twentyfifteen-style' ), '20141010' );
    wp_style_add_data( 'themename-ie7', 'conditional', 'lt IE 8' );

}
```

Y compris le fichier css interne pour votre classe de plugin

```
class My_Plugin() {
  function __construct() {
    add_action( 'wp_enqueue_scripts', array( $this, 'init_fe_assets' ) );
  }

  public function init_fe_assests() {
    wp_enqueue_style( 'my-plugin', plugin_dir_url( __FILE__ ) .
'assets/css/frontend/plugin.css', array(), '0.0.1', true );
  }
}

new My_Plugin();
```

Ajouter des feuilles de style alternatives

```
    <?php wp_enqueue_style('theme-five', get_template_directory_uri() .
'/path/to/additional/css');
wp_style_add_data('theme-five', 'alt', true);
wp_style_add_data('theme-five', 'title', __('theme-five.css', 'your-theme-name')); ?>
```

wp_style_add_data

Lire Styles d'enveloppement en ligne: https://riptutorial.com/fr/wordpress/topic/1247/styles-d-enveloppement

Chapitre 61: Supprimer la version de Wordpress et Stylesheets

Introduction

Pour rendre plus difficile pour les autres de pirater votre site Web, vous pouvez supprimer le numéro de version WordPress de votre site, vos css et js. Sans ce numéro, il n'est pas possible de voir si vous ne lancez pas la version actuelle pour exploiter les bogues des anciennes versions.

De plus, cela peut améliorer la vitesse de chargement de votre site, car sans les chaînes de requête dans l'URL, les fichiers css et js peuvent être mis en cache.

Syntaxe

- add_filter ($ tag, $ function_to_add, $ priority, $ accepted_args)

Paramètres

Paramètre	Détails
$ tag	*(chaîne requise)* Nom du filtre où $ function_to_add est connecté à
$ function_to_add	*(callable required)* Nom de la fonction qui s'exécute lorsque le filtre est appliqué
$ priorité	*(int optionnel)* place de $ function_to_add entre les autres fonctions en une seule action (par défaut = 10)
$ accepted_args	*(int optionnel)* Nombre de paramètres acceptés par $ function_to_add (par défaut = 1)

Remarques

Destiné à améliorer la vitesse et la sécurité du site.

Examples

Supprimer les numéros de version de css / js

Ajoutez simplement cette fonction à vos functions.php. Il supprimera la version de tous les fichiers js et css en file d'attente.

```
function remove_cssjs_ver( $src ) {
   if( strpos( $src, '?ver=' ) )
   $src = remove_query_arg( 'ver', $src );
   return $src;
}

add_filter( 'style_loader_src', 'remove_cssjs_ver', 999 );
add_filter( 'script_loader_src', 'remove_cssjs_ver', 999 );
```

Supprimer les numéros de version de WordPress

Si vous ajoutez ceci à vos functions.php, il supprime le numéro de version WordPress du flux RSS et de l'en-tête.

```
function remove_wordpress_ver() {
     return '';
}
add_filter('the_generator', 'remove_wordpress_ver', 999);
```

Lire Supprimer la version de Wordpress et Stylesheets en ligne:
https://riptutorial.com/fr/wordpress/topic/6218/supprimer-la-version-de-wordpress-et-stylesheets

Chapitre 62: Supprimer les sauts de ligne automatiques du contenu et de l'extrait

Introduction

Pour les sites qui s'appuient sur HTML à la main dans l'éditeur ou des extraits, ceux que vous souhaitez coder vous-même, les sauts de ligne automatiques peuvent être gênants. Vous pouvez les désactiver en supprimant ces filtres.

Remarques

Celles-ci doivent être exécutées directement dans un fichier include. Que ce soit dans functions.php ou dans un autre fichier include, ceux-ci ne peuvent pas être intégrés dans un hook. Ils ne fonctionneront pas sur init ou autre que j'ai trouvé jusqu'à présent.

Ils peuvent également être inclus directement dans un modèle tel que page.php à exécuter uniquement pour ce modèle.

REMARQUE: NE PAS INCLURE CELA DANS UN THEME OU PLUGIN DISTRIBUÉ (sauf s'il est désactivé par défaut, comme si vous n'incluiez pas le fichier include à moins que l'utilisateur ne le spécifie).

C'est une mauvaise pratique à inclure dans un site que vous ne contrôlez pas car il peut et va casser la sortie de tout autre thème ou plugin.

Examples

Supprimer les filtres

```
// Remove the auto-paragraph and auto-line-break from the content
remove_filter( 'the_content', 'wpautop' );

// Remove the auto-paragraph and auto-line-break from the excerpt
remove_filter( 'the_excerpt', 'wpautop' );
```

Fonction pour supprimer les filtres

```
/**
 * Remove the automatic line breaks from content and excerpts.
 *
 * @since 1.0.0
 */
function remove_content_auto_line_breaks() {
    // Remove the auto-paragraph and auto-line-break from the content
    remove_filter( 'the_content', 'wpautop' );
```

```
    // Remove the auto-paragraph and auto-line-break from the excerpt
    remove_filter( 'the_excerpt', 'wpautop' );
}

// Execute the function
remove_content_auto_line_breaks();
```

Lire Supprimer les sauts de ligne automatiques du contenu et de l'extrait en ligne: https://riptutorial.com/fr/wordpress/topic/9614/supprimer-les-sauts-de-ligne-automatiques-du-contenu-et-de-l-extrait

Chapitre 63: Taxonomies

Syntaxe

- register_taxonomy ($ taxonomy, $ object_type, $ args);

Paramètres

Paramètre	Détails
$ taxonomie	*(chaîne) (obligatoire)* Nom de la taxonomie. Le nom ne doit contenir que des lettres minuscules et le caractère de soulignement, et ne pas comporter plus de 32 caractères (restriction de la structure de la base de données).
$ object_type	*(array / string) (requis)* Nom du type d'objet pour l'objet taxonomie. Les types d'objet peuvent être de type post intégré ou de tout type de poste personnalisé pouvant être enregistré.
$ args	*(array / string) (facultatif)* Un tableau d'arguments.

Examples

Exemple d'enregistrement d'une taxonomie pour les genres

```
<?php
// hook into the init action and call create_book_taxonomies when it fires
add_action( 'init', 'create_book_taxonomies', 0 );

// create taxonomy genres for the post type "book"
function create_book_taxonomies() {
    // Add new taxonomy, make it hierarchical (like categories)
    $labels = array(
        'name'              => _x( 'Genres', 'taxonomy general name' ),
        'singular_name'     => _x( 'Genre', 'taxonomy singular name' ),
        'search_items'      => __( 'Search Genres' ),
        'all_items'         => __( 'All Genres' ),
        'parent_item'       => __( 'Parent Genre' ),
        'parent_item_colon' => __( 'Parent Genre:' ),
        'edit_item'         => __( 'Edit Genre' ),
        'update_item'       => __( 'Update Genre' ),
        'add_new_item'      => __( 'Add New Genre' ),
        'new_item_name'     => __( 'New Genre Name' ),
        'menu_name'         => __( 'Genre' ),
    );

    $args = array(
        'hierarchical'      => true,
        'labels'            => $labels,
        'show_ui'           => true,
```

```
        'show_admin_column' => true,
        'query_var'         => true,
        'rewrite'           => array( 'slug' => 'genre' ),
    );

    register_taxonomy( 'genre', array( 'book' ), $args );

}
?>
```

Vous pouvez définir des taxonomies personnalisées dans le fichier de modèle `functions.php` un thème:

Ajouter une catégorie dans la page

Vous pouvez également ajouter la même taxonomie créée dans la page de type de poste en utilisant le code ci-dessous.

```
function add_taxonomies_to_pages() {
     register_taxonomy_for_object_type( 'genre', 'page' );
 }
add_action( 'init', 'add_taxonomies_to_pages' );
```

Ajoutez le code ci-dessus dans le fichier functions.php de votre thème. De la même manière, vous pouvez ajouter la valeur personnalisée ou la valeur par défaut `post_tag` dans la page de type post.

Pour obtenir des pages en utilisant une requête de taxonomie personnalisée, vous devez ajouter le code ci-dessous dans le même fichier.

```
if ( ! is_admin() ) {
     add_action( 'pre_get_posts', 'category_and_tag_archives' );
 }

function category_and_tag_archives( $wp_query ) {
    $my_post_array = array('page');
    if ( $wp_query->get( 'category_name' ) || $wp_query->get( 'cat' ) )
    $wp_query->set( 'post_type', $my_post_array );
}
```

Ajouter des catégories et des balises aux pages et les insérer en tant que classe dans

```
// add tags and categories to pages

function add_taxonomies_to_pages() {
 register_taxonomy_for_object_type( 'post_tag', 'page' );
 register_taxonomy_for_object_type( 'category', 'page' );
 }
add_action( 'init', 'add_taxonomies_to_pages' );
 if ( ! is_admin() ) {
 add_action( 'pre_get_posts', 'category_and_tag_archives' );

 }
```

```
function category_and_tag_archives( $wp_query ) {
$my_post_array = array('post','page');

 if ( $wp_query->get( 'category_name' ) || $wp_query->get( 'cat' ) )
 $wp_query->set( 'post_type', $my_post_array );

 if ( $wp_query->get( 'tag' ) )
 $wp_query->set( 'post_type', $my_post_array );
}

// add tags and categorys as class to <body>

function add_categories_and_tags( $classes = '' ) {
    if( is_page() ) {
        $categories = get_the_category();
        foreach( $categories as $category ) {
            $classes[] = 'category-'.$category->slug;
        }
        $tags = get_the_tags();
        foreach( $tags as $tag ) {
            $classes[] = 'tag-'.$tag->slug;
        }
    }
return $classes;
}
add_filter( 'body_class', 'add_categories_and_tags' );
```

Ajouter des catégories et des balises aux pages et insérer en tant que classe dans

Vous pouvez ajouter ce code à votre fichier functions.php personnalisé:

```
// add tags and categories to pages

function add_taxonomies_to_pages() {
    register_taxonomy_for_object_type( 'post_tag', 'page' );
    register_taxonomy_for_object_type( 'category', 'page' );
}
add_action( 'init', 'add_taxonomies_to_pages' );

if ( ! is_admin() ) {
    add_action( 'pre_get_posts', 'category_and_tag_archives' );
}

function category_and_tag_archives( $wp_query ) {
$my_post_array = array('post','page');

    if ( $wp_query->get( 'category_name' ) || $wp_query->get( 'cat' ) )
    $wp_query->set( 'post_type', $my_post_array );

    if ( $wp_query->get( 'tag' ) )
    $wp_query->set( 'post_type', $my_post_array );

}

// add tags and categorys as class to <body>

function add_categories_and_tags( $classes = '' ) {
```

```
    if( is_page() ) {
        $categories = get_the_category();
        foreach( $categories as $category ) {
            $classes[] = 'category-'.$category->slug;
        }
        $tags = get_the_tags();
        foreach( $tags as $tag ) {
            $classes[] = 'tag-'.$tag->slug;
        }
    }
return $classes;
}
add_filter( 'body_class', 'add_categories_and_tags' );
```

Fonctionne parfaitement, testé dans **WordPress 4.8**

Lire Taxonomies en ligne: https://riptutorial.com/fr/wordpress/topic/5943/taxonomies

Chapitre 64: template_include

Paramètres

Paramètre	Explication
`$template`	Transmet un paramètre au filtre, `$template` est le chemin d'accès actuel au fichier approprié pour le type de publication tel que trouvé dans le thème enfant ou le thème parent actif (si aucun thème enfant en place ou thème enfant n'a de modèle inférieur). pour plus de détails).

Remarques

Vous devez retourner `$template` même si vous ne modifiez pas. Si cela vous `apply_filter()` , regardez des exemples où `apply_filter()` a été utilisé dans le code

Vous ne devez pas configurer de variables ici pour les utiliser plus tard, il existe de meilleurs crochets pour cela.

Un flux de programme utile pour ce filtre est:

1. Check `$template` inclut notre nom de poste personnalisé -> hiérarchie de modèles!
2. Si ce n'est pas le cas, recherchez dans notre plugin les fichiers appropriés -> Il est préférable de pointer vers des fichiers spécifiques plutôt que de chercher dans les dossiers des fichiers. Plus efficace. Mais complètement au développeur.
3. renvoyer le modèle

Examples

Exemple simple

Ce filtre est très utile. L'un des problèmes les plus courants pour les développeurs est d'inclure les modèles dans les plugins qu'ils développent.

Le filtre est appliqué immédiatement après que wordpress a localisé le modèle approprié dans le thème enfant / parent actif à l'aide de la hiérarchie wp.

Veillez à définir quand vous souhaitez modifier le chemin du modèle. Dans l'exemple ci-dessous, le code vérifie si la page actuelle est la vue unique de notre type de publication personnalisé `cpt` .

Exemple assez simple pour commencer!

```
add_filter('template_include', 'custom_function');
```

```
function custom_function($template){

    //change a single post template...

    if( is_singular('cpt')  ){
         $template= 'path/to/another/template_file';
    }


    return $template;

}
```

Plus exemple Adv

```
add_filter('template_include', 'custom_function');


function custom_function($template){

    /*
    *     This example is a little more advanced.
    *    It will check to see if $template contains our post-type in the path.
    *    If it does, the theme contains a high level template e.g. single-cpt.php
    *    If not we look in the plugin parent folder for the file. e.g. single-cpt.php
    */


    //check to see if the post type is in the filename (high priority file)
    //return template if it is!

    global $post;

    if( strpos($template, 'single-'.$post->post_type.'php' ) !== false && strpos($template,
'archive-'.$post->post_type.'php' ) !== false ){
        return $template;
    }

    $plugin_path = 'var/etc/wp-content/plugins/plugin'; //include own logic here...

    if( is_singular($post->post_type) && file_exists($plugin_path.'/single-'.$post-
>post_type.'.php')   ){
         $template= $plugin_path.'/single-'.$post->post_type.'.php';
    } elseif ( is_archive($post->post_type) && file_exists($plugin_path.'/archive-'.$post-
>post_type.'.php') ) {
         $template= $plugin_path.'/archive-'.$post->post_type.'.php';
    }

    return $template;
}
```

Lire template_include en ligne: https://riptutorial.com/fr/wordpress/topic/1439/template-include

Chapitre 65: Thème Wordpress et développement du thème enfant

Introduction

Wordpress est un système de gestion de contenu largement utilisé pour créer des sites Web d'informations simples, mais aussi pour créer des sites Web plus sophistiqués et même de petites boutiques en ligne.

Wordpress utilise des thèmes. Ces thèmes sont utilisés pour créer les fonctionnalités de mise en page et de contenu d'un site Web Wordpress. Les thèmes peuvent être trouvés sur Internet.

Chaque site a ses propres fonctionnalités et sa mise en page, mais il est parfois difficile de trouver le bon thème pour un site Web. Heureusement, nous sommes également en mesure de créer notre propre thème.

Examples

Développer votre propre thème

Un thème wordpress comprend deux types de fichiers. Les fichiers de base de chaque thème et les fichiers définissant la disposition et les fonctionnalités du thème. Ce second groupe je vais appeler les fichiers spécifiques au thème.

Les fichiers de thème de base
Les fichiers de thème de base sont les fichiers utilisés pour configurer et enregistrer un thème. Dans la liste ci-dessous, je vais décrire brièvement chaque fichier et son utilisation. Plus tard, je vais ajouter les exemples de fichiers les plus élémentaires nécessaires pour configurer votre propre thème wordpress.

- `functions.php` : Le fichier functions.php est utilisé pour enregistrer toutes les fonctions, barres latérales, scripts et inclut le thème. Dans ce fichier, vous pouvez par exemple inclure des fichiers CSS, des fichiers JS, etc.
- `Header and footer` : Les fichiers d'en-tête et de pied de page (header.php et footer.php) sont les fichiers utilisés pour appeler l'en-tête et le pied de page. Le fichier d'en-tête et de pied de page, par exemple, contient le lien vers le système dorsal wordpress.
- `index.php` : Le fichier index.php est le fichier qui crée le modèle de page par défaut. Dans ce fichier, vous pouvez voir, éditer et supprimer des éléments de cette présentation par défaut.
- `single.php` : Le fichier single.php est le fichier qui crée la page de modèle de publication unique. Tout comme le modèle par défaut pour les pages, mais maintenant pour les pages à publication unique.
- `format.php` Le fichier format.php est le fichier qui construit le modèle content-text à partir d'une page. Donc, si vous avez une page d'accueil et que vous la modifiez à partir du back-end en ajoutant un texte. Ce fichier crée le balisage standard de ce texte.

- `404.php` Le fichier 404.php crée le modèle 404. Ce fichier consiste en la présentation de base de cette page.
- `archive.php` Le fichier archive.php crée la mise en page de la page d'archive.
- `style.css` Le fichier de feuille de style de base.

Donc, dans cette liste, vous pouvez voir tous les fichiers **requis** pour la configuration de votre propre thème Wordpress. Jetons maintenant un coup d'œil à certains fichiers que vous pouvez créer si vous voulez, mais **ne** sont **pas des** fichiers **requis** pour un thème wordpress. Ces fichiers sont principalement des fichiers de modèles et d'autres extensions fonctionnelles.

Modèles de page personnalisés

`page-<your own name>.php` : dans un thème Wordpress, vous pouvez créer plusieurs modèles de page. en créant de nouveaux fichiers de modèle de page. Un fichier de modèle de page standard est constitué des attributs de nom suivants. `name of the template` `page name of the template` et `.php` Si, par exemple, vous souhaitez créer un nouveau modèle de page pour votre page de blog, vous pouvez l'appeler `page-blog.php` Wordpress lit automatiquement le fichier et ajoute le fichier au menu de sélection de modèle. Assurez-vous que vous avez au moins inclus les fonctions `get_header()` et `get_footer()` . Assurez-vous également de nommer votre modèle dans un commentaire en haut du fichier en ajoutant l'exemple suivant.

```
<?php
    /*
     * Template Name: Homepage Template
     */
    get_header();
?>
```

Modèles de pages individuelles personnalisées

`single-<your own name>.php` : Dans un thème Wordpress tout comme le modèle de page décrit ci-dessus, vous pouvez également créer vos propres modèles de page à un seul message. Tout comme le modèle de page, le fichier se compose de trois parties `single` pour déclarer qu'il s'agit d'une seule page de post `<your name of the template>` et l'extension de fichier `.php` . Tout comme la configuration minimale requise par le modèle de page pour s'assurer que Wordpress lit le nouveau modèle, il ajoute les fonctions `get_header()` et `get_footer()` . Et bien sûr aussi ajouter votre nom de modèle comme dans l'exemple ci-dessous

```
<?php

/*
* Template Name: Post Portfolio
* Template Post Type: post, page
*/

?>
```

Nous indiquons également le `Template post type:` qui correspond au type de template, dans ce cas post et page.

Modèles de texte de message personnalisés

`format -<your own name>.php` : Dans un thème Wordpress, vous pouvez également créer des

modèles de post-publication. Ces modèles de format sont la mise en page et le contenu d'un message. Par exemple, si dans certains cas, vous souhaitez que le message affiche uniquement le contenu ou le titre du message, vous pouvez utiliser ces modèles pour créer ce type d'ajustement. Étant donné que ces types de modèles ne mettent en forme que le contenu des post- `get_header()` créé par un utilisateur, nous n'avons pas besoin d'inclure `get_header()` et `get_footer()` car ceux-ci sont déjà définis dans les modèles de pages. Assurez-vous que votre modèle est capable de reconnaître un message en utilisant l'exemple de base suivant.

```
<div>
    <article id="post-<?php the_ID(); ?>" <?php post_class(); ?>>
    </article>
</div>
```

Donc, maintenant que nous savons quelque chose sur les fichiers de base et certains des nombreux fichiers spécifiques au modèle, il est temps de parler des barres latérales et des widgets. Dans le futur, cela sera ajouté avec un tutoriel sur la création d'un thème Wordpress.

Lire Thème Wordpress et développement du thème enfant en ligne: https://riptutorial.com/fr/wordpress/topic/9940/theme-wordpress-et-developpement-du-theme-enfant

Chapitre 66: Types de messages personnalisés

Syntaxe

- register_post_type ($ post_type, $ args);

Paramètres

Paramètre	Détails
$ post_type	(chaîne) (obligatoire)
$ args	(tableau / chaîne) (facultatif)

Examples

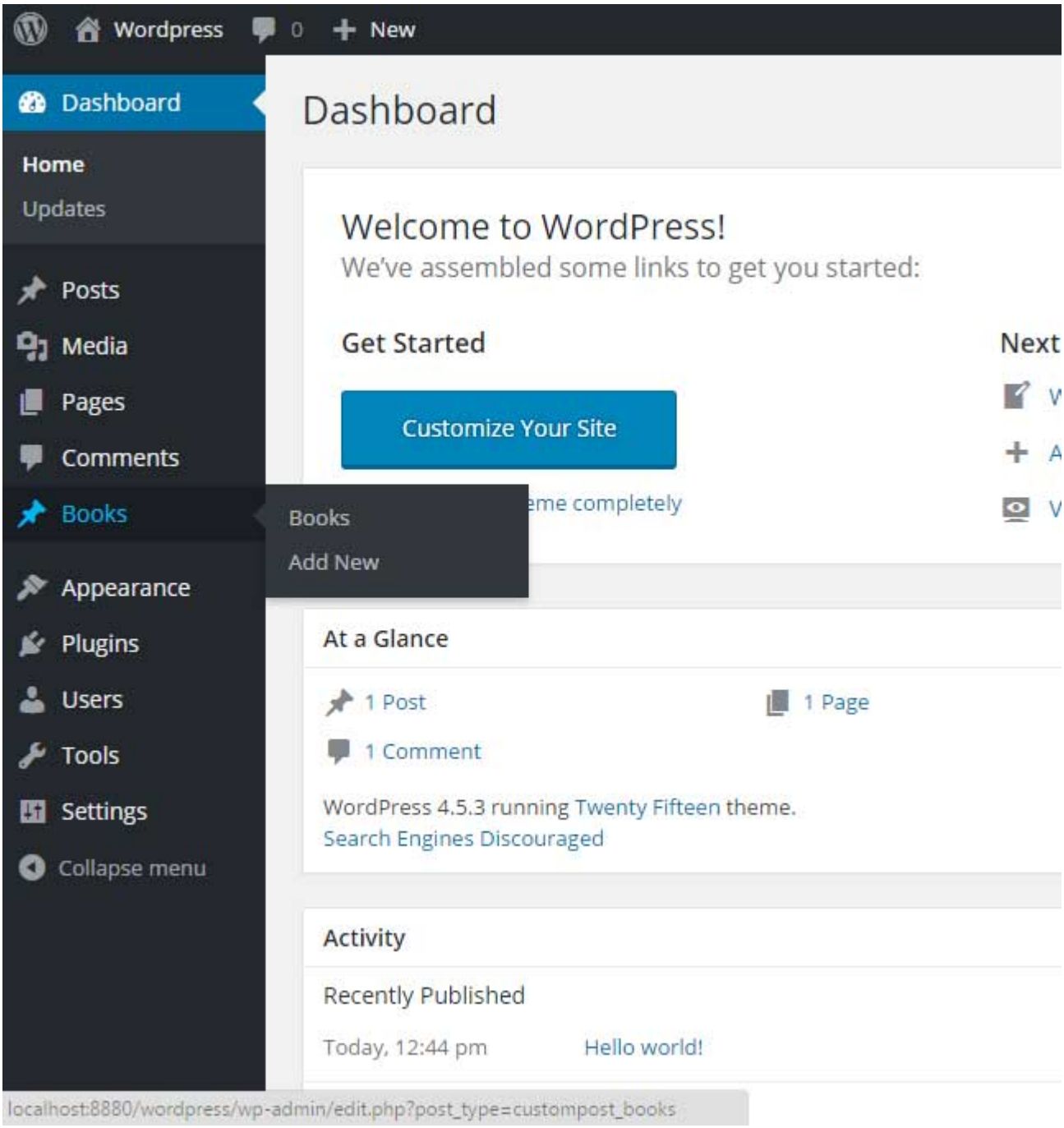
Enregistrement d'un type de message personnalisé

Supposons que vous ayez un site Web de bibliothèque et que vous souhaitiez un type de publication personnalisé nommé *Livres* . Il peut être enregistré comme

```
function create_bookposttype() {
    $args = array(
        'public' => true,
        'labels' => array(
            'name' => __( 'Books' ),
            'singular_name' => __( 'Book' )
        ),
    );
    register_post_type( 'custompost_books', $args );
}

add_action( 'init', 'create_bookposttype' );
```

et, aussi simple soit-il, vous avez maintenant un type de poste personnalisé enregistré.



Cet extrait peut être placé dans votre fichier `functions.php` ou dans une structure de plugin.

Ajouter des types de publication personnalisés à la requête principale

L'enregistrement d'un type de publication personnalisé ne signifie pas qu'il est ajouté automatiquement à la requête principale. Vous devez utiliser le filtre `pre_get_posts` pour ajouter des types de publication personnalisés à la requête principale.

```
// Show posts of 'post' and 'book' custom post types on home page
add_action( 'pre_get_posts', 'add_my_post_types_to_query' );

function add_my_post_types_to_query( $query ) {
  if ( is_home() && $query->is_main_query() )
    $query->set( 'post_type', array( 'post', 'book' ) );
  return $query;
```

```
}
```

Ajout de types de publication personnalisés au flux RSS principal

Enregistrer un type de publication personnalisé ne signifie pas qu'il est ajouté automatiquement au flux RSS principal. Vous devez utiliser le filtre de `request` pour ajouter des types de publication personnalisés au flux RSS principal.

```
// Add 'books' custom post types on main RSS feed
function add_book_post_types_to_rss($qv) {
    if (isset($qv['feed']) && !isset($qv['post_type']))
        $qv['post_type'] = array('post', 'books', );
    return $qv;
}
add_filter('request', 'add_book_post_types_to_rss');
```

Enregistrer le type de poste personnalisé

```
if ( ! function_exists('products_post_type') ) {

function products_post_type() {

    $labels = array(
        'name'                  => _x( 'Products', 'Post Type General Name', 'text_domain' ),
        'singular_name'         => _x( 'Product', 'Post Type Singular Name', 'text_domain' ),
        'menu_name'             => __( 'Products', 'text_domain' ),
        'name_admin_bar'        => __( 'Product', 'text_domain' ),
        'archives'              => __( 'Item Archives', 'text_domain' ),
        'attributes'            => __( 'Item Attributes', 'text_domain' ),
        'parent_item_colon'     => __( 'Parent Product:', 'text_domain' ),
        'all_items'             => __( 'All Products', 'text_domain' ),
        'add_new_item'          => __( 'Add New Product', 'text_domain' ),
        'add_new'               => __( 'New Product', 'text_domain' ),
        'new_item'              => __( 'New Item', 'text_domain' ),
        'edit_item'             => __( 'Edit Product', 'text_domain' ),
        'update_item'           => __( 'Update Product', 'text_domain' ),
        'view_item'             => __( 'View Product', 'text_domain' ),
        'view_items'            => __( 'View Items', 'text_domain' ),
        'search_items'          => __( 'Search products', 'text_domain' ),
        'not_found'             => __( 'No products found', 'text_domain' ),
        'not_found_in_trash'    => __( 'No products found in Trash', 'text_domain' ),
        'featured_image'        => __( 'Featured Image', 'text_domain' ),
        'set_featured_image'    => __( 'Set featured image', 'text_domain' ),
        'remove_featured_image' => __( 'Remove featured image', 'text_domain' ),
        'use_featured_image'    => __( 'Use as featured image', 'text_domain' ),
        'insert_into_item'      => __( 'Insert into item', 'text_domain' ),
        'uploaded_to_this_item' => __( 'Uploaded to this item', 'text_domain' ),
        'items_list'            => __( 'Items list', 'text_domain' ),
        'items_list_navigation' => __( 'Items list navigation', 'text_domain' ),
        'filter_items_list'     => __( 'Filter items list', 'text_domain' ),
    );
    $args = array(
        'label'                 => __( 'Product', 'text_domain' ),
        'description'           => __( 'Product information pages.', 'text_domain' ),
        'labels'                => $labels,
        'supports'              => array( 'title', 'editor', 'excerpt', 'author', 'thumbnail',
```

```
'comments', 'custom-fields', ),
        'taxonomies'            => array( 'category', 'post_tag' ),
        'hierarchical'          => false,
        'public'                => true,
        'show_ui'               => true,
        'show_in_menu'          => true,
        'menu_position'         => 5,
        'menu_icon'             => 'dashicons-products',
        'show_in_admin_bar'     => true,
        'show_in_nav_menus'     => true,
        'can_export'            => true,
        'has_archive'           => true,
        'exclude_from_search'   => false,
        'publicly_queryable'    => true,
        'capability_type'       => 'page',
        'show_in_rest'          => true,
    );
    register_post_type( 'product', $args );

}
add_action( 'init', 'products_post_type', 0 );

}
```

Type de message personnalisé à l'aide du thème WordPress Twenty Fifteen

Vous pouvez utiliser n'importe quel nom pour la fonction.

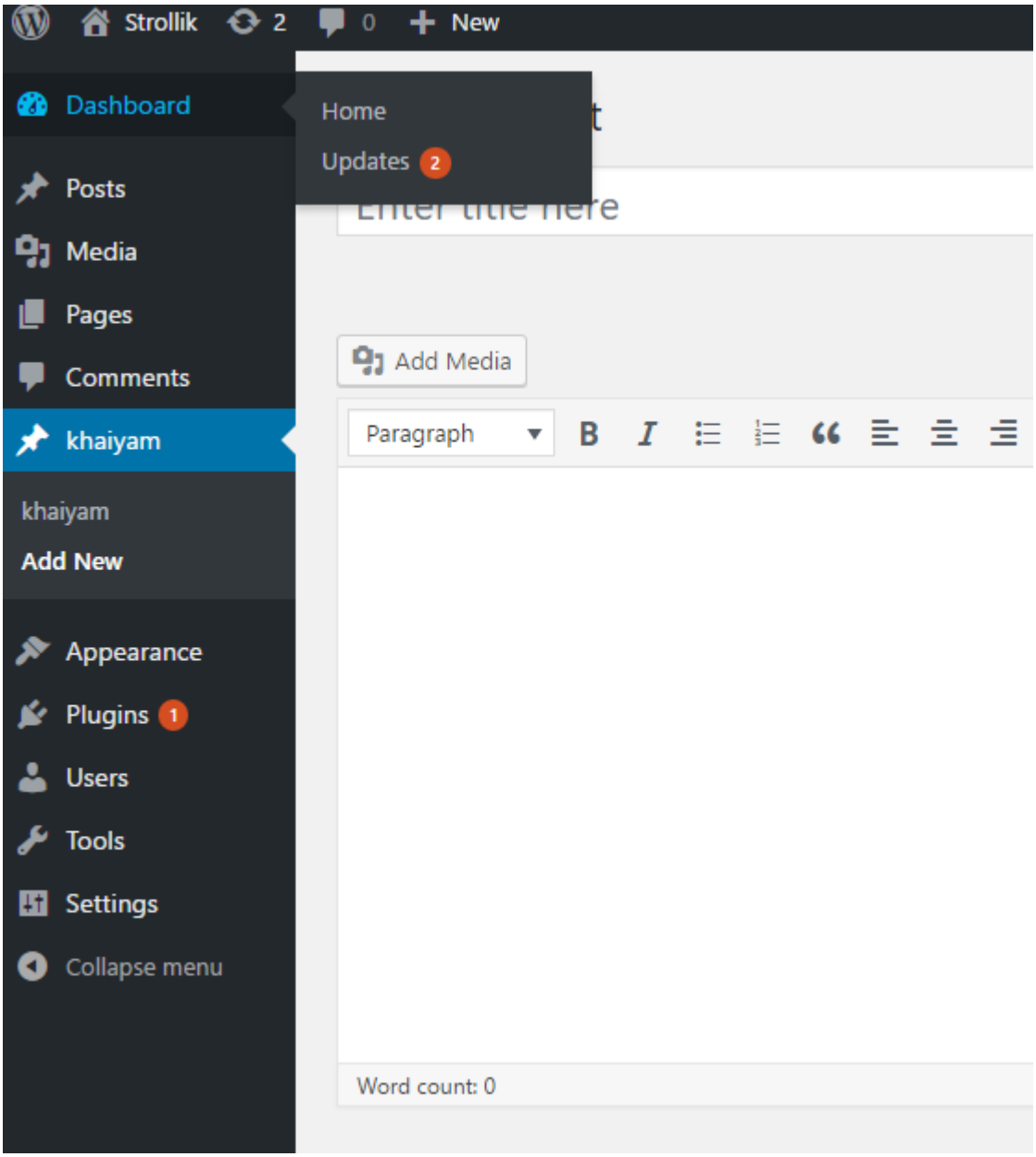
```
function custom_postype(){
    register_post_type('cus_post',array(

        'labels'=>array(
            'name'=>'khaiyam'// Use any name you want to show in menu for your users
            ),
        'public'=>true,// **Must required
        'supports'=>array('title','editor','thumbnail')// Features you want to provide on your
posts
        ));
}
add_action('after_setup_theme','custom_postytpe');
```

ou

```
add_action('init','custom_postytpe');
```

Vous pouvez utiliser n'importe lequel des crochets que vous voulez, mais ils ont bien sûr une signification et des utilisations différentes.



Type de poste personnalisé dans la recherche par défaut

Vous pouvez ajouter des posts de type post personnalisés sur la recherche wordpress par défaut, Ajouter le code ci-dessous dans le theme functions.php

```
function my_search_filter($query) {
  if ( !is_admin() && $query->is_main_query() ) {
    if ($query->is_search) {
      $query->set('post_type', array( 'news','post','article' ) );
    }
  }
}
add_action('pre_get_posts','my_search_filter');
```

Lire Types de messages personnalisés en ligne:
https://riptutorial.com/fr/wordpress/topic/1374/types-de-messages-personnalises

Chapitre 67: Widgets Admin Dashboard

Introduction

Avec un widget de tableau de bord admin, vous pouvez afficher tout type d'informations sur le tableau de bord de l'administrateur. Vous pouvez créer plusieurs widgets si vous le souhaitez. Vous pouvez ajouter le code aux fonctions.php de votre thème ou à votre plugin.

Syntaxe

- add_action ($ tag, $ function_to_add, $ priority, $ accepted_args);
- wp_add_dashboard_widget ($ widget_id, $ widget_name, $ callback, $ control_callback, $ callback_args);

Paramètres

Paramètre	Détails
$ tag	(*chaîne requise*) Nom de l'action où $ function_to_add est accroché
$ function_to_add	(*callable required*) Nom de la fonction que vous souhaitez appeler.
$ priorité	(*int optionnel*) Place de l'appel de fonction dans toutes les fonctions d'action (par défaut = 10)
$ accepted_args	(*int facultatif*) Nombre de paramètres acceptés par la fonction (par défaut = 1)
$ widget_id	(*chaîne requise*) Un slug unique pour votre widget
$ widget_name	(*chaîne requise*) Nom de votre widget (affiché dans la tête)
$ callback	(*callable required*) Nom de la fonction qui affiche le contenu de votre widget
$ control_callback	(*callable optional*) Nom de la fonction qui gère les formulaires d'options du widget
$ callback_args	(*array optional*) Paramètres de la fonction $ control_callback

Examples

Widget simple (affiche le texte)

Cela va ajouter un widget simple qui affiche juste un petit message.

```
 add_action('wp_dashboard_setup', 'register_my_dashboard_widgets');

 function register_my_dashboard_widgets() {
     wp_add_dashboard_widget('myInfo_widget', 'Important Information', 'display_infoWidget');
 }

function display_infoWidget() {
     echo '<p>At the first of february this site gets a new design.
     Therefore is wont be available this day. To see the current progress you can visit
     <a href="http://www.justanexample.com" >this site</a></p>';
 }
```

Lire Widgets Admin Dashboard en ligne: https://riptutorial.com/fr/wordpress/topic/9571/widgets-admin-dashboard

Chapitre 68: wp_get_current_user ()

Syntaxe

- wp_get_current_user ()

Examples

Obtenir l'utilisateur actuel

Récupérer toutes les informations de l'utilisateur actuel dans wordpress en utilisant la fonction prédéfinie `wp_get_current_user();`

```
<?php

        $current_user = wp_get_current_user();

        echo "Username :".$current_user->user_login;
        echo "Username :".$current_user->ID;
        echo "Username :".$current_user->user_pass;
        echo "Username :".$current_user->user_nicename;
        echo "Username :".$current_user->user_email;
        echo "Username :".$current_user->user_url;
        echo "Username :".$current_user->user_registered;
        echo "Username :".$current_user->user_activation_key;
        echo "Username :".$current_user->user_status;
        echo "Username :".$current_user->display_name;

    ?>
```

Utilisez la boucle foreach pour obtenir les informations utilisateur de wp_get_current_user ()

```
$user = wp_get_current_user();

foreach($user->data as $key=>$user_data){
    if($key == 'user_pass' || $key == 'user_activation_key' || $key=='user_status'){}
    else{
        $nice_key = ucfirst(str_replace('_', ' ', $key));

        if($key == 'user_registered'){
            $user_data = date_i18n(get_option('date_format'), strtotime($user_data));
        }

    echo $nice_key . ' : ' . $user_data . '<br />';
   }
}
```

Lire wp_get_current_user () en ligne: https://riptutorial.com/fr/wordpress/topic/2693/wp-get-current-user---

Chapitre 69: wp_get_current_user ()

Examples

Obtenir les informations actuelles sur l'utilisateur connecté

Récupère les informations relatives à l'utilisateur actuellement connecté et les place dans la variable globale $ current_user

Cette fonction n'accepte aucun paramètre.

Usage:

```
<?php wp_get_current_user(); ?>
```

Exemple:

```
  <?php
  $current_user = wp_get_current_user();

  echo 'Username: ' . $current_user->user_login . "\n";
  echo 'User email: ' . $current_user->user_email . "\n";
  echo 'User level: ' . $current_user->user_level . "\n";
  echo 'User first name: ' . $current_user->user_firstname . "\n";
  echo 'User last name: ' . $current_user->user_lastname . "\n";
  echo 'User display name: ' . $current_user->display_name . "\n";
  echo 'User ID: ' . $current_user->ID . "\n";
  ?>
```

Pour déterminer si un visiteur est connecté ou non, vous pouvez utiliser is_user_logged_in () avant, puis obtenir les informations utilisateur actuelles si le visiteur est connecté:

```
<?php
if ( is_user_logged_in() ) {
    $current_user = wp_get_current_user();

    echo 'Welcome, ' . $current_user->user_login . '!';
} else {
    echo 'Welcome, visitor!';
}
?>
```

Lire wp_get_current_user () en ligne: https://riptutorial.com/fr/wordpress/topic/6649/wp-get-current-user---

Chapitre 70: WP_Query () Boucle

Introduction

WP_Query pour interroger les messages, les pages et les types de publication personnalisés. Vous obtiendrez la liste des messages et des pages spécifiques ou des types de publication personnalisés. WP_Query vous permet d'extraire des publications de la base de données en fonction de vos critères.

Examples

Récupérer les 10 derniers articles

```
$args = array(
    'post_type'=>'post',
    'posts_per_page' =>'10'
);
$latest_posts_query = new WP_Query( $args );
if($latest_posts_query->have_posts()) :
while ( $latest_posts_query-> have_posts()) : $latest_posts_query->the_post();
//Get post details here
endwhile;
endif;
```

Lire WP_Query () Boucle en ligne: https://riptutorial.com/fr/wordpress/topic/8301/wp-query----boucle

Chapitre 71: WP-CLI

Introduction

WP-CLI est un ensemble d'outils de ligne de commande pour gérer les installations WordPress. Vous pouvez mettre à jour des plug-ins, configurer des installations multisites et bien plus, sans utiliser de navigateur Web.

Examples

Gérer des thèmes

Obtenez une liste de thèmes.

```
$ wp theme list
```

Installer la dernière version de wordpress.org et activer

```
$ wp theme install twentysixteen --activate
```

Installer à partir d'un fichier zip local

```
$ wp theme install ../my-theme.zip
```

Installer à partir d'un fichier zip distant

```
$ wp theme install http://s3.amazonaws.com/bucketname/my-
theme.zip?AWSAccessKeyId=123&amp;Expires=456&amp;Signature=abcdef
```

Obtenir des détails sur un thème installé

```
$ wp theme get twentysixteen --fields=name,title,version
```

Obtenir le statut du thème

```
$ wp theme status twentysixteen
```

Gérer les plugins

Obtenir une liste de plugins

```
$ wp plugin list
```

Liste des plugins actifs sur le site.

```
$ wp plugin list --status=active --format=json
```

Liste des plugins sur chaque site d'un réseau.

```
$ wp site list --field=url | xargs -I % wp plugin list --url=%
```

Activer le plugin

```
$ wp plugin activate hello-dolly
```

Désactiver le plugin

```
$ wp plugin deactivate hello-dolly
```

Supprimer le plugin

```
$ wp plugin delete hello-dolly
```

Installer la dernière version de wordpress.org et activer

```
$ wp plugin install bbpress --activate
```

Gérer WP-CLI lui-même

Affiche la version actuellement installée.

```
$ wp cli version
```

Vérifiez les mises à jour de WP-CLI.

```
$ wp cli check-update
```

Mettez à jour WP-CLI vers la dernière version stable.

```
$ wp cli update
```

Répertorie tous les alias disponibles.

```
$ wp cli alias
```

Imprimez divers détails sur l'environnement WP-CLI.

```
$ wp cli info
```

Videz la liste des commandes installées, en tant que JSON.

```
$ wp cli cmd-dump
```

Téléchargez, installez, mettez à jour et gérez une installation WordPress.

Télécharger le noyau WordPress

```
$ wp core download --locale=nl_NL
```

Installer WordPress

```
$ wp core install --url=example.com --title=Example --admin_user=supervisor --
admin_password=strongpassword --admin_email=info@example.com
```

Afficher la version WordPress

```
$ wp core version
```

Transformez une installation sur un seul site en une installation multisite WordPress.

```
$ wp core multisite-convert
```

Installez WordPress multisite à partir de zéro.

```
$ wp core multisite-install
```

Gérer les utilisateurs

Liste des ID utilisateur

```
$ wp user list --field=ID
```

Créer un nouvel utilisateur

```
$ wp user create bob bob@example.com --role=author
```

Mettre à jour un utilisateur existant.

```
$ wp user update 123 --display_name=Mary --user_pass=marypass
```

Supprimer l'utilisateur 123 et réaffecter les messages à l'utilisateur 567

```
$ wp user delete 123 --reassign=567
```

Effectuer des opérations de base de données de base en utilisant les informations d'identification stockées dans wp-config.php

Créez une nouvelle base de données.

```
$ wp db create
```

Déposer une base de données existante.

```
$ wp db drop --yes
```

Réinitialiser la base de données actuelle.

```
$ wp db reset --yes
```

Exécutez une requête SQL stockée dans un fichier.

```
$ wp db query < debug.sql
```

Lire WP-CLI en ligne: https://riptutorial.com/fr/wordpress/topic/9169/wp-cli

Chapitre 72: WP-Cron

Examples

wp_schedule_event () exemple

```
// register activation hook
register_activation_hook( __FILE__, 'example_activation' );

// function for activation hook
function example_activation() {
    // check if scheduled hook exists
    if ( !wp_next_scheduled( 'my_event' )) {
         // Schedules a hook
         // time() - the first time of an event to run ( UNIX timestamp format )
         // 'hourly' - recurrence ('hourly', 'twicedaily', 'daily' )
         // 'my_event' - the name of an action hook to execute.
         wp_schedule_event( time(), 'hourly', 'my_event' );
    }
}

add_action( 'my_event', 'do_this_hourly' );

// the code of your hourly event
function do_this_hourly() {
   // put your code here
}

// register deactivation hook
register_deactivation_hook(__FILE__, 'example_deactivation');

// function for deactivation hook
function example_deactivation() {
    // clear scheduled hook
    wp_clear_scheduled_hook( 'my_event' );
}
```

Important: le cron WordPress s'exécute uniquement lorsque certaines pages de votre site Web sont touchées. Donc, pour les sites à faible trafic, vous devez configurer le cron sur votre hébergement pour accéder aux pages.

intervalle de récurrence personnalisé dans wp_schedule_event ()

```
// this function add custom interval (5 minutes) to the $schedules
function five_minutes_interval( $schedules ) {
      $schedules['five_minutes'] = array(
            'interval'  => 60 * 5,
            'display'   =>  '5 minutes';
    );
    return $schedules;
}

// add a custom interval filter
add_filter( 'cron_schedules', 'five_minutes_interval' );
```

```
// Schedules a hook
wp_schedule_event( time(), 'five_minutes', 'my_event' );
```

Lire WP-Cron en ligne: https://riptutorial.com/fr/wordpress/topic/6783/wp-cron

Crédits

S. No	Chapitres	Contributeurs
1	Démarrer avec WordPress	4444, A. Raza, Andrew, animuson, Anupam, Chris Fletcher, Ciprian, Community, Florida, James Jones, JonasCz, Leo F, Marc St Raymond, Mayank Gupta, Milap, nus, Panda, rap-2-h, Seth C., Shubham, Trevor Clarke, vajrasar
2	Actions et filtres	David, Ihor Vorotnov, Mrinal Haque, Trying Tobemyself
3	add_action ()	Abel Melquiades Callejo, Waqas Bukhary
4	add_editor_style ()	Gabriel Chi Hong Lee
5	add_menu_page ()	brasofilo, Gabriel Chi Hong Lee
6	add_submenu_page ()	Gabriel Chi Hong Lee, theoretisch
7	add_theme_support ()	Gabriel Chi Hong Lee
8	AJAX	Andy, Digvijayad, Gaurav Srivastava, GreatBlakes, Nisarg Patel , Ruslan Murarov, stweb
9	Ajouter / supprimer des informations de contact pour les utilisateurs avec le hook de filtre user_contactmethods	Petar Popovic, RamenChef
10	Ajouter un code court	purvik7373, RamenChef
11	API REST	Picard
12	Barres latérales	dingo_d, Kushal Shah, theoretisch
13	Boucle principale en alternance (filtre pre_get_posts)	Dawid Urbanski, Petar Popovic
14	Comment puis-je intégrer l'éditeur Markdown à l'add-on de répéteur Advance	Fatbit

	Custom Field?	
15	Création de plugins WordPress	Seth C.
16	Créer un message par programme	RamenChef, Roel Magdaleno, RRikesh
17	Créer un modèle personnalisé	Petar Popovic
18	Créer un modèle pour un type de message personnalisé	Ashok G, Egnaro, Joe Dooley, mnoronha
19	Customizer Bonjour tout le monde	Dan Green-Leipciger
20	Des thèmes	Jef
21	Développement de plugin	Angle.R, Ping.Chen
22	Exécutez WordPress local avec XAMPP	Pierre.Vriens, theoretisch
23	Extraits personnalisés avec excerpt_length et excerpt_more	inkista, Petar Popovic, RamenChef
24	Faire des requêtes réseau avec HTTP API	Jordan, mjangda, Rarst
25	Fonction: add_action ()	bosco, RamenChef
26	Fonction: wp_trim_words ()	Harshal Limaye
27	Formats de poste	Shashank Agarwal
28	get_bloginfo ()	Abel Melquiades Callejo, Harshal Limaye, HeyCameron, KenB, Nate Beers, RamenChef, Tom J Nowell, virtualLast, Wes Moberly
29	get_home_path ()	Ihor Vorotnov

30	get_option ()	Gabriel Chi Hong Lee
31	get_permalink ()	Gabriel Chi Hong Lee
32	get_template_part ()	Dan Devine, Kushal Shah
33	get_the_category ()	Gabriel Chi Hong Lee
34	get_the_title ()	Gabriel Chi Hong Lee
35	Hiérarchie de modèles	jgraup, MarZab, Pelmered, theoretisch
36	home_url ()	dingo_d, Kushal Shah, matthew, Mr. Developer
37	init	Abel Melquiades Callejo, barbocc
38	Installation et configuration	Kenyon, Marco Romano, Ping.Chen, S.L. Barth, stig-js, theoretisch, Yuan Lung Luo
39	Interroger les messages	dingo_d
40	L'objet $ wpdb	Kushal Shah, mcon, stweb
41	La barre d'administration (aussi appelée "la barre d'outils")	dingo_d, Harshal Limaye, JCL1178, Kushal Shah
42	La boucle (boucle WordPress principale)	anik4e, Dawid Urbanski
43	Le débogage	barbocc, dingo_d, jgraup
44	le titre()	Gabriel Chi Hong Lee
45	Les bases du thème de l'enfant	Andrei, Razvan Onofrei, Vlad Olaru
46	Meta Box	Austin Winstanley
47	Mettre à jour WordPress manuellement	KnightHawk
48	Migration de site	Austin Winstanley
49	Notions de base sur le Customizer	4444, Ahmad Awais, RamenChef

	(Ajouter un panneau, une section, un paramètre, un contrôle)	
50	Options API	Harshal Limaye, Pat J, RamenChef
51	Petit code	Ad Wicks, Adam Genshaft, brasofilo, John Slegers, Kylar, Shashank Agarwal
52	Scripts de mise en file d'attente	dingo_d, Harshal Limaye, J.D., mbacon40, montrealist, Pelmered, Petar Popovic
53	Sécurisez votre installation	James Jones
54	Sécurité dans WordPress - S'échapper	Laxmana, the4kman
55	Sécurité dans WordPress - Sanitization	Laxmana
56	Shortcode avec attribut	Digvijayad, Firefog, RamenChef
57	Shortcodes	Pelmered
58	Styles d'enveloppement	dingo_d, Harshal Limaye, Laxmana, mnoronha, montrealist, Petar Popovic, RamenChef, Ruslan Murarov, virtualLast
59	Supprimer la version de Wordpress et Stylesheets	jay.jivani, mnoronha, theoretisch
60	Supprimer les sauts de ligne automatiques du contenu et de l'extrait	Austin Winstanley
61	Taxonomies	adifatz, Kushal Shah, purvik7373
62	template_include	Abel Melquiades Callejo, David, RamenChef
63	Thème Wordpress et développement du thème enfant	Deathstorm
64	Types de messages	Caio Felipe Pereira, Dan Devine, J.D., janw, jgraup, Kushal

	personnalisés	Shah, Omar Khaiyam, Ranuka, theoretisch
65	Widgets Admin Dashboard	theoretisch
66	wp_get_current_user ()	Abel Melquiades Callejo, Benoti, Muhammad Farrukh Faizy
67	WP_Query () Boucle	vrajesh
68	WP-CLI	jgraup
69	WP-Cron	stweb