

CMS – WORDPRESS

Module 18 : La création de sa propre extension



Objectifs

- Créer sa propre extension



La création de sa propre extension

Créer un fil d'ariane dans functions.php

- Utiliser la globale `$post`, des marqueurs conditionnels, des fonctions PHP et des fonctions propres à WordPress.
- Utilisez le marqueur conditionnel `is_home()` pour tester si vous êtes sur la page d'accueil
- `get_bloginfo('wpurl')` permet de récupérer l'URL de la page d'accueil.
- `get_bloginfo('name')` permet de récupérer le titre du site.
- `get_ancestors()` : affichez les liens parents, s'il y en a.
- `get_permalink()` et `get_title()`.
- `function_exists()`



La création de son propre thème

Démonstration

Créer un fil d'Ariane



La création de sa propre extension

Créer une extension Widget en PHP

- Les widgets sont des extensions. Ils se glissent dans une sidebar et vous pouvez les organiser comme vous le souhaitez, grâce à un système Ajax de drag and drop.
- Différents widgets par défaut sont à votre disposition, par exemple une liste d'articles les plus récents, un nuage d'étiquettes, un champ de recherche, un menu personnalisé...
- Vous pouvez également télécharger des widgets et ainsi ajouter des fonctionnalités à votre site.
- Onglet de l'administration **Apparence - Widgets**.
- Utiliser l'objet WP_widget dans votre propre classe et l'objet WP_Query pour faire une requête à la base de données et récupérer des informations sur les articles.



La création de sa propre extension

Créer une extension Widget en PHP

- **Configurer l'extension**

- Pour créer un widget, créer un sous-dossier au dossier wp-content/plugins et lui donner un nom.
- À l'intérieur du dossier mon-widget, créez un fichier mon-widget.php.
- Insérez dans le dossier les fichiers readme.txt, licence.txt et index.php vide, si vous le souhaitez.
- En haut du fichier mon-widget.php, ajoutez en en-tête un commentaire avec les informations sur l'extension, et activez l'extension.



La création de sa propre extension

Créer une extension Widget en PHP

```
<?php
/*
Plugin Name: Widget post
Description: widget permettant l'affichage d'une liste d'articles
selon la catégorie.
Version: 1.0
License: GPLv2
Copyright {année} {mon_nom} (email : {mon_email})
This program is free software; you can redistribute it and/or
modify it under the terms of the GNU General Public License,
version 2, as published by the Free Software Foundation.

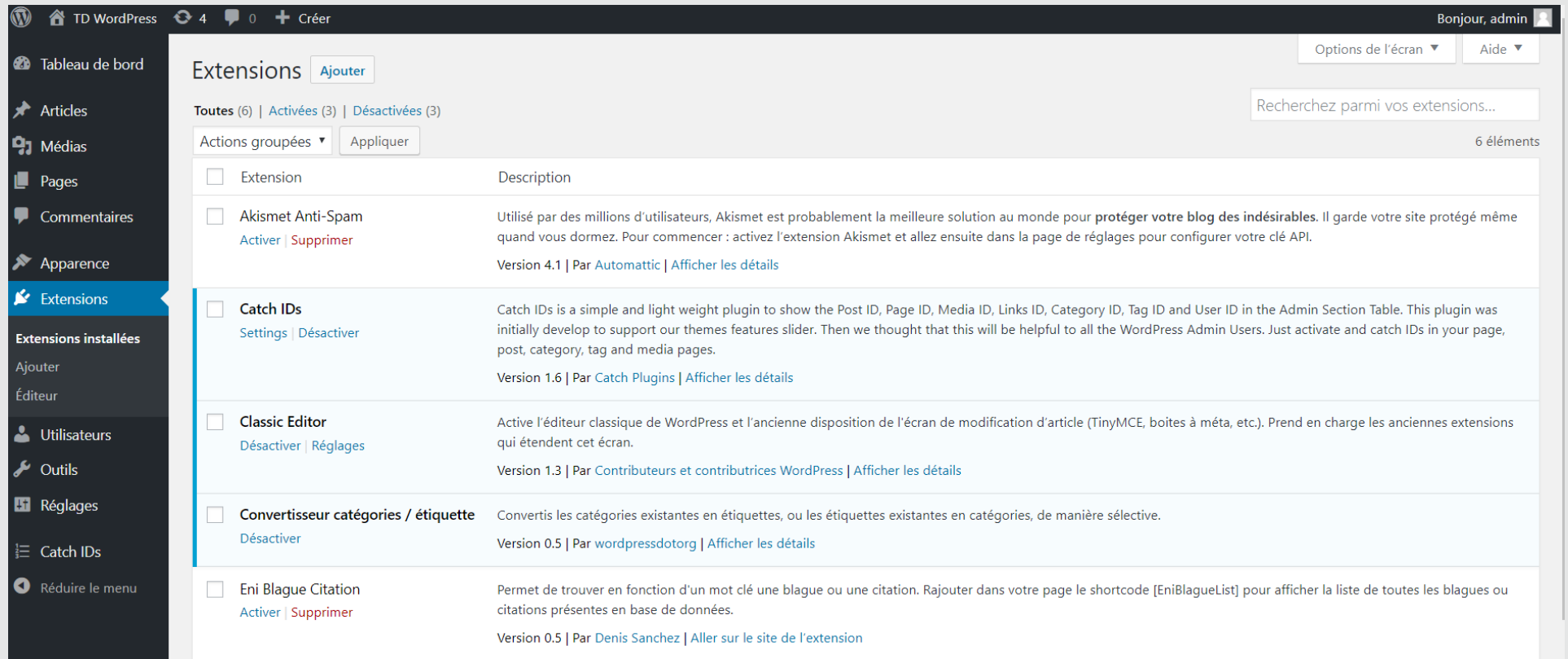
This program is distributed in the hope that it will be useful,
but WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
GNU General Public License for more details.

You should have received a copy of the GNU General Public License
along with this program; if not, write to the Free Software
Foundation, Inc., 51 Franklin St, Fifth Floor, Boston,
MA 02110-1301 USA
*/
?>
```

La création de sa propre extension

TD : Créer une extension Widget en PHP

- **Configurer l'extension**
- Vous devez voir sur la page de l'administration, dans l'onglet **Extensions** le nom de votre plugin:



The screenshot shows the WordPress Admin Dashboard with the 'Extensions' (Plugins) page selected. The left sidebar contains the standard WordPress menu items, with 'Extensions' highlighted. The main content area displays a list of installed and available plugins. The 'Ajouter' (Add) button is visible at the top right of the Extensions section. A search bar is located on the right side of the page.

Extension	Description
<input type="checkbox"/> Akismet Anti-Spam Activer Supprimer	Utilisé par des millions d'utilisateurs, Akismet est probablement la meilleure solution au monde pour protéger votre blog des indésirables . Il garde votre site protégé même quand vous dormez. Pour commencer : activez l'extension Akismet et allez ensuite dans la page de réglages pour configurer votre clé API. Version 4.1 Par Automattic Afficher les détails
<input type="checkbox"/> Catch IDs Settings Désactiver	Catch IDs is a simple and light weight plugin to show the Post ID, Page ID, Media ID, Links ID, Category ID, Tag ID and User ID in the Admin Section Table. This plugin was initially develop to support our themes features slider. Then we thought that this will be helpful to all the WordPress Admin Users. Just activate and catch IDs in your page, post, category, tag and media pages. Version 1.6 Par Catch Plugins Afficher les détails
<input type="checkbox"/> Classic Editor Désactiver Réglages	Active l'éditeur classique de WordPress et l'ancienne disposition de l'écran de modification d'article (TinyMCE, boîtes à méta, etc.). Prend en charge les anciennes extensions qui étendent cet écran. Version 1.3 Par Contributeurs et contributrices WordPress Afficher les détails
<input type="checkbox"/> Convertisseur catégories / étiquette Désactiver	Convertit les catégories existantes en étiquettes, ou les étiquettes existantes en catégories, de manière sélective. Version 0.5 Par wordpressdotorg Afficher les détails
<input type="checkbox"/> Eni Blague Citation Activer Supprimer	Permet de trouver en fonction d'un mot clé une blague ou une citation. Rajouter dans votre page le shortcode [EniBlagueList] pour afficher la liste de toutes les blagues ou citations présentes en base de données. Version 0.5 Par Denis Sanchez Aller sur le site de l'extension

La création de sa propre extension

Créer une extension Widget en PHP

- L'API des widgets de WordPress permet de gérer, créer, modifier des widgets grâce à plusieurs méthodes, et d'étendre la classe **WP_Widget**.
- http://codex.wordpress.org/Widgets_API
- La classe **WP_Widget** a été conçue spécialement pour faciliter le développement de widgets, elle autorise la création de classe étendue, ce qui va vous permettre d'utiliser ses méthodes.
- Pour créer la classe de votre propre widget, vous allez vous servir de la classe WP_Widget de cette manière :
 - `class Ma_class extends WP_Widget {}`
- Ce dispositif permet de simplifier la création de widget.



La création de sa propre extension

Créer une extension Widget en PHP

- Pour développer un widget, votre classe devra contenir quatre méthodes spécifiques :
 - La méthode de construction du même nom que la classe, qui sert à configurer le widget.
 - La méthode widget(), qui sert à afficher le widget côté internaute.
 - La méthode update(), qui sert à mettre à jour les options du widget.
 - La méthode form(), qui sert à afficher les formulaires de configuration du widget dans l'administration.

La création de sa propre extension

Créer une extension Widget en PHP

```
class Ma_class extends WP_Widget {  
  
    function Ma_class() {  
        //Configuration du widget  
    }  
  
    function widget($args,$instance) {  
        //Affichage du widget  
    }  
  
    function update($new_instance,$old_instance ){  
        //Mise à jour des options  
    }  
  
    function form($instance) {  
        //Formulaire des réglages  
    }  
}
```

- **La classe WP_Widget**
- Les arguments \$args, \$instance, \$new_instance et \$old_instance sont automatiquement pris en compte par les méthodes utilisées.
- WordPress les gère nativement, pour simplifier le développement de widgets.
 - **\$args** : tableau contenant des informations sur la sidebar.
 - **\$instance** : tableau contenant les informations du widget.
 - **\$new_instance** : nouveau tableau contenant les informations du widget après la mise à jour.
 - **\$old_instance** : ancien tableau contenant les informations du widget avant la mise à jour.

La création de sa propre extension

Créer une extension Widget en PHP

- Pour configurer le widget, ajoutez la méthode `wp_widget()` de l'objet `WP_Widget`, à la méthode du même nom que le Widget.
- Cela permet de paramétrer le widget et d'indiquer à WordPress qu'il y a un nouveau widget.
- Voici le détail de la méthode **`WP_Widget()`** :
 - **`WP_Widget($slug, $name, $widget_ops, $control_ops);`**
 - **`$slug`** : accepte le nom clé du widget.
 - **`$name`** : accepte le nom du widget.
 - **`$widget_ops`** : accepte un tableau avec un nom de classe HTML et une description.
 - Exemple
 - `$widget_ops = array('classname' => 'your_class', 'description' => 'your_description');`
 - **`$control_ops`** : accepte un tableau additionnel de paramètres (largeur, hauteur, id).
 - Exemple
 - `$control_ops = array('width' => 'your_width', 'height' => 'your_height', 'id_base' => 'your_id');`

La création de sa propre extension

Créer une extension Widget en PHP

- Pour utiliser la méthode WP_Widget() à l'intérieur d'une autre méthode, appelez la méthode en utilisant \$this, comme en PHP.
 - `$this->WP_Widget('widget-post', 'Widget post', $widget_ops, $control_ops);`

La création de sa propre extension

Créer une extension Widget en PHP

- **Enregistrer le widget avec la fonction `register_widget()`**
- Avant d'utiliser le widget, enregistrez-le à l'extérieur de la classe, grâce à la fonction WordPress `register_widget()`.
 - `<?php register_widget($widget_class); ?>`
 - Référence au codex : http://codex.wordpress.org/Function_Reference/register_widget
- Il existe une fonction inverse permettant de supprimer l'enregistrement d'un widget :
 - `<?php unregister_widget($widget_class); ?>`

La création de sa propre extension

Créer une extension Widget en PHP

- Voici les noms de classes des widgets natifs de WordPress :
 - **WP_Widget_Pages** : nom de classe pour le widget Pages.
 - **WP_Widget_Calendar** : nom de classe pour le widget Calendrier.
 - **WP_Widget_Archives** : nom de classe pour le widget Archives.
 - **WP_Widget_Meta** : nom de classe pour le widget Méta.
 - **WP_Widget_Search** : nom de classe pour le widget Chercher.
 - **WP_Widget_Text** : nom de classe pour le widget Texte.

La création de sa propre extension

Créer une extension Widget en PHP

- Voici les noms de classes des widgets natifs de WordPress :
 - **WP_Widget_Categories** : nom de classe pour le widget Catégories.
 - **WP_Widget_Recent_Posts** : nom de classe pour le widget Articles récents.
 - **WP_Widget_Recent_Comments** : nom de classe pour le widget Commentaires récents.
 - **WP_Widget_RSS** : nom de classe pour le widget Flux.
 - **WP_Widget_Tag_Cloud** : nom de classe pour le widget Nuage d'étiquettes.
 - **WP_Nav_Menu_Widget** : nom de classe pour le widget Menu personnalisé.

La création de sa propre extension

Créer une extension Widget en PHP

- **Afficher le widget grâce à un hook**
- Pour afficher le widget, greffez le module au core de WordPress à l'aide du hook d'action **widgets_init**, qui s'exécute lors de l'initialisation des widgets.
 - `add_action('widgets_init', 'register_my_widget');`
- Désormais, le widget apparaît dans l'administration **Apparence - Widgets**.



La création de sa propre extension

Créer une extension Widget en PHP

- **Créer le formulaire du widget**
- Ajouter les champs HTML
- Pour créer le formulaire du widget dans l'administration, ajoutez du code HTML dans la méthode **form()**.
- La fonction WordPress **get_categories()** retourne un tableau avec toutes les catégories de WordPress. Des arguments existent pour filtrer le tableau.
- https://developer.wordpress.org/reference/functions/get_categories

La création de sa propre extension

Créer une extension Widget en PHP

- Ajouter les méthodes `get_field_id()` et `get_field_name()`
- Pour identifier les champs du formulaire et pour en récupérer la valeur, WordPress vous simplifie la tâche grâce à deux autres méthodes qui appartiennent à l'objet `WP_Widget` :
 - `<?php get_field_id($name); ?>`
 - `<?php get_field_name($name); ?>`
 - `$name` : accepte le nom du champ.
- Lors de la validation du formulaire, les champs sont enregistrés automatiquement en base de données grâce à la méthode `update()` et viennent s'ajouter au tableau `$instance`.
- Les champs du tableau ont pour clé `$name` avec pour valeur celle du champ associé.
- Ajoutez la méthode `get_field_id()` à la propriété `label` et la méthode `get_field_name()` à la propriété `name` des champs `input` et `select` du formulaire.

La création de sa propre extension

Créer une extension Widget en PHP

- Ajouter des paramètres par défaut
- Pour ajouter des paramètres par défaut, modifiez le tableau **\$instance** passé comme argument, dans la méthode forms().
- Pour modifier les champs du tableau de **\$instance**, utilisez la fonction WordPress suivante :
 - `<?php wp_parse_args($instance, $defaults); ?>`
 - **\$instance** : accepte le tableau contenant les informations du widget.
 - **\$defaults** : accepte un tableau de champs par défaut qui s'ajoute à \$instance.
 - http://codex.wordpress.org/Function_Reference/wp_parse_args

La création de sa propre extension

Créer une extension Widget en PHP

- Dans le widget, ajoutez pour l'exemple un titre par défaut : Articles.
 - **\$defaults** = array('title' => 'Articles');
 - **\$instance** = wp_parse_args(\$instance, \$defaults);
 - Maintenant, ajoutez la valeur par défaut au champ HTML title correspondant au titre, en récupérant la valeur par défaut, contenue dans le tableau \$instance.
 - Voici le code qui permet de récupérer la valeur du champ :
 - `<?php echo $instance['title']; ?>`
 - L'input pour le titre devient alors :

```
<input type="text" id="<?php echo $this->get_field_id( 'title' ); ?>"
name="<?php echo $this->get_field_name( 'title' ); ?>" value="<?php
echo $instance['title']; ?>" style="width:100%;" />
```

- Cela permet également, lors de l'enregistrement du widget, d'actualiser la valeur du champ input et de faire apparaître la nouvelle valeur du tableau **\$instance**.
- Il faut modifier la boucle foreach() afin de sélectionner le bon champ option, dans la balise select grâce au code HTML selected.

La création de sa propre extension

Créer une extension Widget en PHP

- Il faut modifier la boucle `foreach()` de la façon suivante afin de sélectionner le bon champ option, dans la balise select grâce au code HTML `selected` :

```
foreach((get_categories()) as $cat) {  
    if($instance['category']==$cat->cat_ID){  
        $selected='selected="selected"';  
    }else{  
        $selected='';  
    }  
  
    echo '<option '.$selected.' value="'.$cat->cat_ID.'">'.$cat->  
cat_name.'</option>';  
}
```

La création de sa propre extension

Créer une extension Widget en PHP

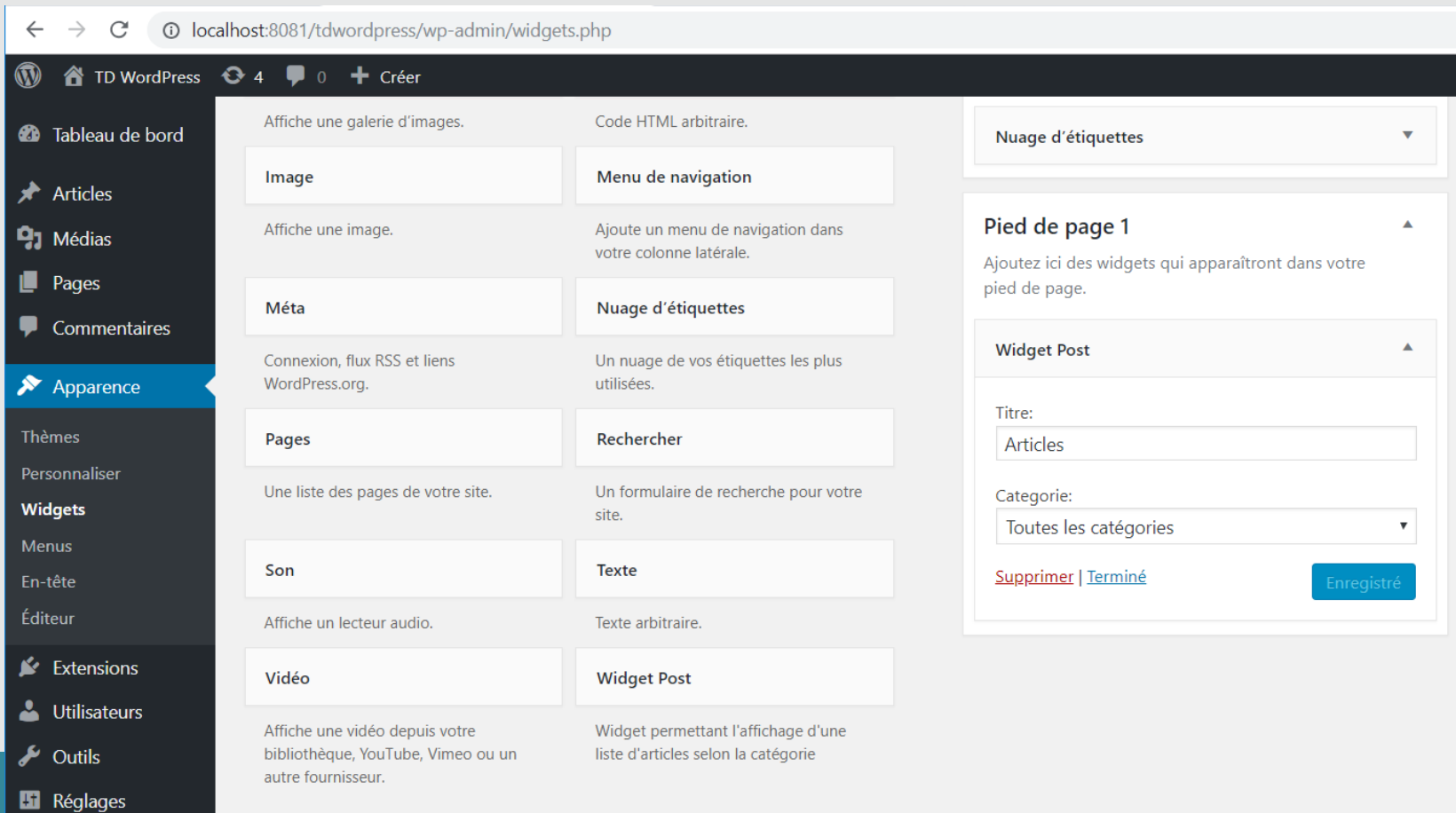
- Voici le code final de la méthode **forms()** :

```
function form($instance) {
    $defaults = array( 'title' => 'Articles' );
    $instance = wp_parse_args($instance, $defaults );
    ?>
    <p>
        <label for="<?php echo $this->get_field_id( 'title' ); ?>">
            Titre:
        </label>
        <input type="text" id="<?php echo $this->get_field_id( 'title' ); ?>"
            name="<?php echo $this->get_field_name( 'title' ); ?>" value="<?php
            echo $instance['title']; ?>" style="width:100%;" />
        </p>
    <p>
        <label for="<?php echo $this->get_field_id( 'category' ); ?>">
            Catégorie:
        </label>
        <select id="<?php echo $this->get_field_id( 'category' ); ?>" name=
            "<?php echo $this->get_field_name( 'category' ); ?>" value=
            "<?php echo $instance['category']; ?>" style="width:100%;" >
            <option>toutes les catégories</option>
        <?php
            foreach((get_categories()) as $cat) {
                if($instance['category']==$cat->cat_ID){
                    $selected='selected="selected"';
                }else{
                    $selected='';
                }
                echo ' <option ' . $selected . ' value="' . $cat->cat_ID . '">' . $cat->
                    cat_name . '</option>';
            }
        ?>
        </select>
    </p>
    <?php
```

La création de sa propre extension

Créer une extension Widget en PHP

- Dans l'administration, vous pouvez désormais configurer le widget, puis enregistrer les champs en base de données.



La création de sa propre extension

Créer une extension Widget en PHP

- **Enregistrer et mettre à jour les options du widget**
- Pour mettre à jour ou enregistrer les informations dans la base de données, modifiez la méthode `update()`.
- Retourner un tableau `$instance` avec les nouvelles valeurs, pour qu'il enregistre les modifications.
- Les deux arguments passés à la méthode sont gérés automatiquement par WordPress. Ils permettent de récupérer facilement l'ancien tableau et le nouveau tableau.
- Utilisez la fonction PHP `strip_tags()` lors de l'enregistrement du widget, pour sécuriser les champs du tableau et éviter l'injection de code.
- Voici le code pour la méthode `update()` :

```
function update($new_instance,$old_instance ){
    $instance = $old_instance;

    $instance['title'] = strip_tags( $new_instance['title'] );
    $instance['category'] = strip_tags( $new_instance['category'] );

    return $instance;
}
```

La création de sa propre extension

Créer une extension Widget en PHP

- **Afficher le widget dans la sidebar**
- Pour afficher le widget sur le site Internet, il ne reste plus qu'à alimenter la méthode d'affichage widget().
- Utilisez la fonction PHP extract() sur le tableau \$args pour extraire les clés du tableau sous forme de variables et de leur attribuer automatiquement la valeur associée.
- Exemple

```
$args=array('before_title'=>'<h1>','after_title'=>'</h1>');  
extract($args);  
echo $before_title.'mon titre'.$after_title;
```

La création de sa propre extension

Créer une extension Widget en PHP

- Le titre sera entouré des balises HTML `<h1></h1>`, les clés du tableau s'utilisent directement en tant que variables, ainsi la clé **before_title** est utilisable sous la forme `$before_title`, grâce à la fonction PHP `extract()`.
- fonction PHP **extract()** : <http://php.net/manual/fr/function.extract.php>

```
$args=array('before_title'=>'<h1>','after_title'=>'</h1>');  
extract($args);  
echo $before_title.'mon titre'.$after_title;
```

La création de sa propre extension

Créer une extension Widget en PHP

- **Afficher le widget dans la sidebar**
- Voici le code de la méthode widget :
- Dans votre cas, le tableau \$args a pour variables \$before_widget, \$after_widget, \$before_title et \$after_title.
- Récupérez également les valeurs pour le titre et pour la catégorie, contenues dans le tableau \$instance.

```
function widget($args,$instance) {  
    extract($args);  
  
}
```

```
function widget($args,$instance) {  
    extract($args);  
    echo $before_widget;  
    if($instance['title']!=''){  
        echo $before_title.$instance['title'].$after_title;  
    }  
    echo $after_widget;  
}
```

La création de sa propre extension

Créer une extension Widget en PHP

- **Afficher le widget dans la sidebar**
- Pour la catégorie, faites une requête simple sur les articles.
- Utilisez une boucle secondaire, grâce à la fonction `get_posts()`.

```
global $post;

if($instance['category']!=''){
    $args = array('category' => $instance['category'] );
}else{
    $args='';
}

$myposts = get_posts($args);
```

La création de sa propre extension

Créer une extension Widget en PHP

- Puis faites une boucle afin de récupérer les permaliens et le titre des articles. La méthode widget() doit ressembler à cela :

```
function widget($args,$instance) {
    extract($args);
    global $post;

    echo $before_widget;

    if($instance['title']!=''){
        echo $before_title.$instance['title'].$after_title;
    }

    if($instance['category']!=''){
        $args = array('category' => $instance['category'] );
    }else{
        $args='';
    }

    $myposts = get_posts($args);

    echo '<ul>';
    foreach ( $myposts as $post ){
        setup_postdata($post);

        echo '<li>';
        echo '<a href="'.get_the_permalink().'">'.get_the_title().</a>';
        echo '</li>';

    }
    echo '</ul>';
    wp_reset_postdata();

    echo $after_widget;
}
```

La création de sa propre extension

Créer une extension Widget en PHP

- Votre widget doit s'afficher côté internaute, et vous pouvez voir les liens des articles associés à la catégorie, en fonction des paramètres de votre widget dans l'administration.
- Pensez à modifier le fichier readme.txt, et d'indiquer l'utilisation du widget.

ARTICLES

Dakhla, la petite Essaouira aux portes du désert

Venise

Rome

Le château des ducs de Bretagne

Bonjour tout le monde !



La création de son propre thème

Démonstration

Créer une extension Widget en PHP



La création de sa propre extension

Créer une extension avec PHP/MySQL

- Utilisation de l'objet **wpdb**, afin de créer une table dans une base de données et d'y faire des requêtes SQL.

La création de sa propre extension

Créer une extension avec PHP/MySQL

- **Construire la classe :**
- Pour construire la classe, plusieurs méthodes sont nécessaires, dont des méthodes pour enregistrer les informations en base de données grâce à des requêtes SQL.
- Dans un premier temps, réfléchissez à la construction de la classe au préalable :
 - Méthode de création de la table, lors de l'installation de l'extension.
 - Méthode de suppression de la table, lors de la désactivation de l'extension, ou fichier de suppression de la table, lors de la suppression de l'extension.
 - Méthode d'ajout du bouton dans le menu de l'administration.
 - Méthode pour la page d'administration, où les cartes sont créées.
 - Méthode d'ajout des fichiers JavaScript et CSS pour l'administration (backoffice).
 - Méthode d'ajout des fichiers JavaScript et CSS pour les internautes (frontoffice).
 - Méthode de création d'un shortcode.
 - Méthodes de requête SQL : INSERT, UPDATE, SELECT, DELETE.
 - Hooks d'action en dehors de la classe.
- Contrairement au widget, le nom des méthodes importe peu, nommez-les de façon à vous repérer facilement.

La création de sa propre extension

Créer une extension avec PHP/MySQL

- **Créer la table :**
- Pour créer la table, l'objet **wpdb** est indispensable pour exécuter la requête SQL CREATE.
- Utilisez pour cela la globale **\$wpdb**.
- Vérifiez qu'il n'existe pas une table du même nom, afin d'éviter un éventuel conflit, grâce à la méthode **get_var()** de la classe wpdb.
- Servez-vous également du fichier upgrade.php et de la fonction **dbdelta**.
- Voici un exemple avec le code de la méthode gmap_install(), qui sert à créer la table

```
function gmap_install(){
    global $wpdb;
    $table_site = $wpdb->prefix.'mygmap';
    if($wpdb->get_var("SHOW TABLES LIKE '$table_site'")!=
    $table_site){
        $sql="CREATE TABLE `$table_site` (
            `id` BIGINT UNSIGNED NOT NULL AUTO_INCREMENT PRIMARY KEY ,
            `titre` TEXT NOT NULL,
            `longitude` TEXT NOT NULL,
            `latitude` TEXT NOT NULL
        )ENGINE=MyISAM DEFAULT CHARSET=utf8 COLLATE=utf8_unicode_ci
        ";
        require_once(ABSPATH.'wp-admin/includes/upgrade.php');
        dbDelta($sql);
    }
}
```

La création de sa propre extension

Créer une extension avec PHP/MySQL

- Pour que la méthode s'exécute lors de l'activation de l'extension, ajoutez un hook en dehors de la classe avec la fonction WordPress **register_activation_hook()**.
- Ce hook est équivalent au hook d'action 'activate_(nom_du_fichier_du_plugin)', où nom_du_fichier_du_plugin doit être remplacé par le nom de l'extension, incluant un sous-dossier si le fichier se trouve dans un sous-dossier.

```
<?php register_activation_hook( $file, $function ); ?>
```

- Cf. constantes magiques :
 - <http://www.php.net/manual/fr/language.constants.predefined.php>

La création de sa propre extension

Créer une extension avec PHP/MySQL

- Dans votre base de données, la table est bien présente :

The screenshot shows the phpMyAdmin interface for a database named 'tdwordpress'. The left sidebar lists the database structure, including tables like 'wp_commentmeta', 'wp_comments', 'wp_links', 'wp_mygmap', 'wp_options', 'wp_postmeta', 'wp_posts', 'wp_termmeta', 'wp_terms', 'wp_term_relationships', 'wp_term_taxonomy', 'wp_usermeta', and 'wp_users'. The main panel displays the 'Structure' tab for the 'tdwordpress' database. It shows a list of 13 tables with their respective actions (Parcourir, Structure, Rechercher, Insérer, Vider, Supprimer). The table 'wp_users' is highlighted, showing its structure: 1 line, InnoDB engine, utf8mb4_unicode_ci collation, 64 kio size, and 0 o. The bottom section shows the 'Nouvelle table' form with 'Nom' and 'Nombre de colonnes' fields.

Table	Action	Lignes	Type	Interclassement	Taille	Perte
wp_commentmeta	Parcourir Structure Rechercher Insérer Vider Supprimer	0	InnoDB	utf8mb4_unicode_ci	48 kio	-
wp_comments	Parcourir Structure Rechercher Insérer Vider Supprimer	3	InnoDB	utf8mb4_unicode_ci	96 kio	-
wp_links	Parcourir Structure Rechercher Insérer Vider Supprimer	0	InnoDB	utf8mb4_unicode_ci	32 kio	-
wp_mygmap	Parcourir Structure Rechercher Insérer Vider Supprimer	0	MyISAM	utf8_unicode_ci	1 kio	-
wp_options	Parcourir Structure Rechercher Insérer Vider Supprimer	139	InnoDB	utf8mb4_unicode_ci	80 kio	-
wp_postmeta	Parcourir Structure Rechercher Insérer Vider Supprimer	168	InnoDB	utf8mb4_unicode_ci	112 kio	-
wp_posts	Parcourir Structure Rechercher Insérer Vider Supprimer	69	InnoDB	utf8mb4_unicode_ci	144 kio	-
wp_termmeta	Parcourir Structure Rechercher Insérer Vider Supprimer	0	InnoDB	utf8mb4_unicode_ci	48 kio	-
wp_terms	Parcourir Structure Rechercher Insérer Vider Supprimer	15	InnoDB	utf8mb4_unicode_ci	48 kio	-
wp_term_relationships	Parcourir Structure Rechercher Insérer Vider Supprimer	20	InnoDB	utf8mb4_unicode_ci	32 kio	-
wp_term_taxonomy	Parcourir Structure Rechercher Insérer Vider Supprimer	15	InnoDB	utf8mb4_unicode_ci	48 kio	-
wp_usermeta	Parcourir Structure Rechercher Insérer Vider Supprimer	27	InnoDB	utf8mb4_unicode_ci	48 kio	-
wp_users	Parcourir Structure Rechercher Insérer Vider Supprimer	1	InnoDB	utf8mb4_unicode_ci	64 kio	-
13 tables	Somme	457	InnoDB	utf8_general_ci	801 kio	0 o

La création de sa propre extension

Créer une extension avec PHP/MySQL

- Pour supprimer la table lors de la désactivation de l'extension, utilisez la requête SQL DROP et la méthode
 - `$wpdb->query` de l'objet `wpdb`.
- La méthode `gmap_uninstall()` a le code suivant :

```
function gmap_uninstall(){ global $wpdb;

    $table_site = $wpdb->prefix.'mygmap';

    if($wpdb->get_var("SHOW TABLES LIKE '$table_site'") == $table_site){
        $sql = "DROP TABLE `$table_site`";
        $wpdb->query($sql);
    }
}
```

- Pour que la méthode s'exécute lors de la désactivation de l'extension, utilisez la fonction `hook_register_deactivation_hook()`, toujours en dehors de la classe :
- `<?php register_deactivation_hook($file, $function); ?>`

La création de sa propre extension

Créer une extension avec PHP/MySQL

- Supprimer la table lors de la suppression de l'extension
- Si vous optez pour ce choix supprimez la méthode `gmap_uninstall()` ainsi que le `hook register_desactivation_hook()`.
- Pour supprimer la table lors de la suppression de l'extension, il faut créer un fichier `uninstall.php` à la racine de votre extension.
- Ce fichier est détecté automatiquement par WordPress et s'exécutera lors de la suppression de l'extension.
- Dans le fichier `uninstall.php`, ajoutez une condition, qui dit à WordPress que si vous ne désinstallez pas l'extension, il ne faut pas exécuter le code à suivre et il faut sortir du fichier.
- Voici le code :

```
<?php
if ( ! defined( 'WP_UNINSTALL_PLUGIN' ) )
    exit();
?>
```

La création de sa propre extension

Créer une extension avec PHP/MySQL

- Puis créez à la suite une fonction avec le code de suppression de table (le même que celui de la méthode `gmap_uninstall()` de la section précédente) :
- Puis appelez la fonction à la fin du fichier :
 - `gmap_uninstall();`
- Voici le code complet :

```
<?php
if ( ! defined( 'WP_UNINSTALL_PLUGIN' ) )
    exit();

function gmap_uninstall(){

    global $wpdb;

    $table_site = $wpdb->prefix.'mygmap';

    if($wpdb->get_var("SHOW TABLES LIKE '$table_site'") == $table_site){
        $sql = "DROP TABLE `$table_site`";
        $wpdb->query($sql);
    }
}
gmap_uninstall();
?>
```


La création de sa propre extension

Créer une extension avec PHP/MySQL

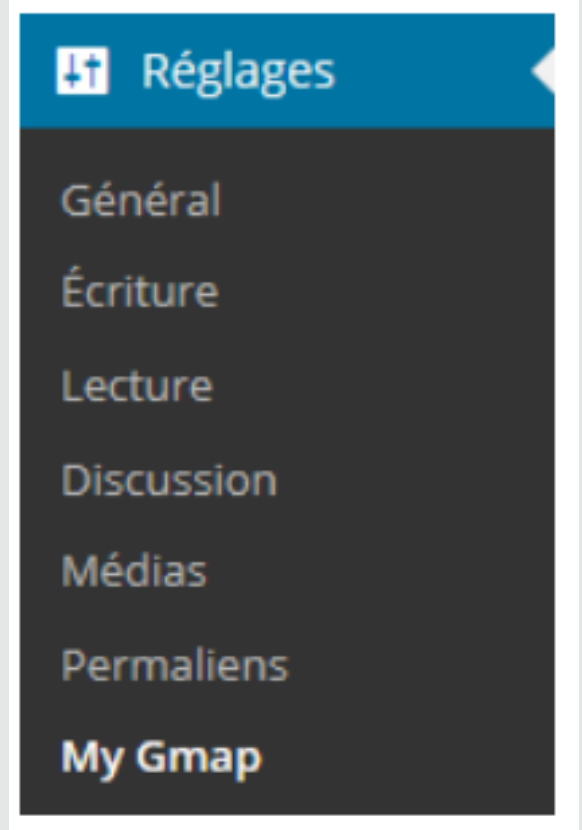
- **Ajouter le sous-menu de l'extension et la page de configuration**
- Ajouter le sous-menu
 - Pour ajouter le sous-menu à l'onglet **Réglages**, ajoutez la fonction `add_options_pages()` à la méthode `init()`,
- `__FILE__` est une constante magique en PHP renvoyant le chemin complet et le nom du fichier courant. Vous pouvez aussi écrire le nom du fichier directement :

```
function init(){  
    if (function_exists('add_options_page')){  
        add_options_page("My Gmap", 'My Gmap', 'administrator', __FILE__,  
            array($this, 'gmap_admin_page'));  
    }  
}
```

La création de sa propre extension

Créer une extension avec PHP/MySQL

- Ajoutez le hook d'action **admin_menu** à l'extérieur de la classe.
 - `add_action('admin_menu', array($inst_map, 'init'));`
- Le sous-menu apparaît désormais dans le menu **Réglages**.
 - `add_options_page()` fait appel à la méthode `gmap_admin_page()` qui contient le code HTML à afficher sur la page.
 - Pour alléger le code et éviter d'avoir l'ensemble du code HTML dans le fichier PHP,
 - Créez un fichier PHP dans le dossier de votre plugin, appelez-le **template.php**.



La création de sa propre extension

Créer une extension avec PHP/MySQL

- **Afficher la page d'administration**
- Pour afficher le fichier **template.php**, appelez le fichier à l'aide de la fonction d'inclusion PHP **require_once()** dans la méthode **gmap_admin_page()** :
 - `function gmap_admin_page() { require_once('template.php'); }`
- **Créer un formulaire pour enregistrer vos données**
- À l'intérieur du fichier `template.php`, créez le code HTML permettant de récupérer les informations

```
<div class="wrap">
  <h2>My Gmap</h2>
</div>
<div id="contentmap">
  <h3 class="title" >Créez une carte :</h3>
  <form action="?page=my-gmap/my-gmap.php&action=createmap"
method="post"></p>
  <p>Titre* :<br /><input type="text" id="Mg-title" name=
"Mg-title" /></p>
  <p>Latitude* :<br /><input type="text" id="Mg-latitude" name=
"Mg-latitude" /></p>
  <p>Longitude* :<br /><input type="text" id="Mg-longitude" name=
"Mg-longitude" /></p>
  <p><input type="button" class="button button-primary"
value= "Enregistrer" /></p>
  <small>* champs obligatoires</small>
</form>
</div>
```

Le formulaire redirigera après sa soumission à l'adresse
?page=my-gmap/my-gmap.php&action=createmap.

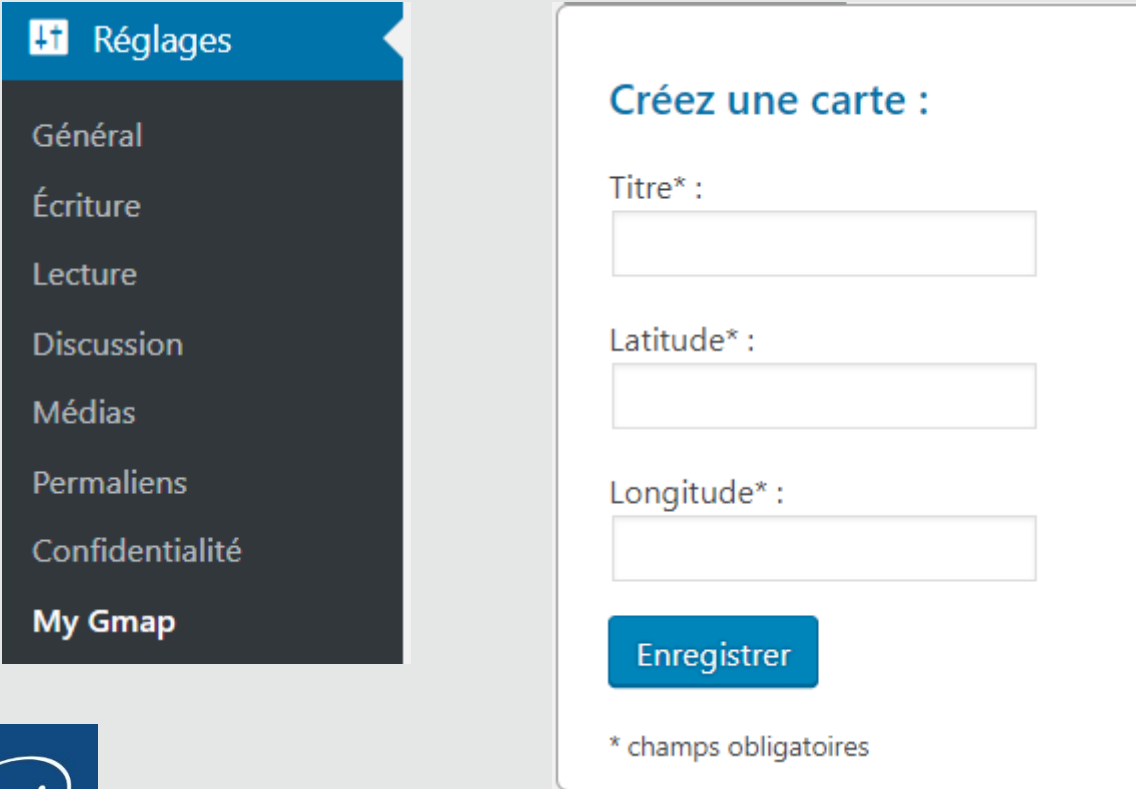
La redirection renvoie sur la page de l'extension et vers le
fichier **my-gmap.php**.

Faites passer un paramètre **action** dans l'URL avec pour
valeur **createmap**, cela permettra de récupérer cette valeur
avec **\$_GET** et d'exécuter du code PHP pour créer la carte, en
fonction de la valeur du paramètre **action**.

La création de sa propre extension

Créer une extension avec PHP/MySQL

- **Créer le formulaire pour enregistrer les cartes**
- Voici comment se présente la page **Réglages - My Gmap** :

- 

La création de sa propre extension

Créer une extension avec PHP/MySQL

- **Ajouter les fichiers JavaScript et CSS**

- Créez un dossier css et un dossier js dans le dossier de votre plugin.
- Dans le dossier css, créez un fichier admin-gmap.css.
- Dans le dossier js, créez un fichier admin-gmap.js.
- Appelez les fichiers avec les fonctions WordPress `wp_register_style()`, `wp_enqueue_style()` et `wp_enqueue_script()` dans la méthode `gmap_admin_header()`.

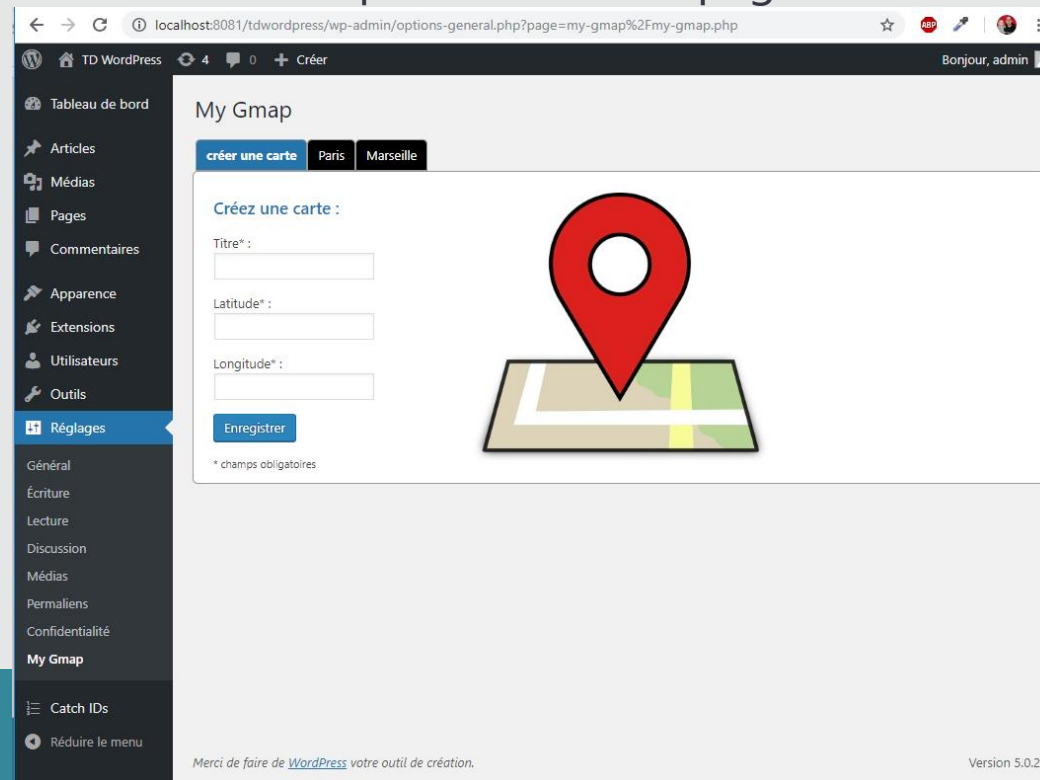
```
function gmap_admin_header(){  
    wp_register_style('my_gmap_css', plugins_url('css/  
admin-gmap.css', __FILE__));  
    wp_enqueue_style('my_gmap_css');  
    wp_enqueue_script('my_gmap_js', plugins_url('js/admin-gmap.js',  
__FILE__), array('jquery'));  
}
```

La création de sa propre extension

Créer une extension avec PHP/MySQL

- **Ajouter les fichiers JavaScript et CSS**

- Pour trouver le chemin du plugin, utilisez la fonction WordPress `plugins_url()`.
- À l'intérieur d'une extension, les fonctions ont pour arguments le nom du fichier et le chemin du fichier.
- Vous pouvez également créer un dossier images, si vous voulez appeler des images dans vos fichiers.
- Voici ce que cela rend avec le code CSS pour la mise en page :



La création de sa propre extension

Créer une extension avec PHP/MySQL

- Faire appel au hook d'action load-(page)
 - Pour que la méthode qui ajoute les fichiers CSS et JavaScript soit prise en compte, il faut greffer la méthode au core de WordPress avec un hook d'action, en dehors de la classe à la suite des autres hooks d'action.
 - Le hook peut avoir comme action **admin_init** et une condition pour que la méthode ne s'exécute que sur la page de l'extension.
 - Mais une autre méthode existe. Pour cela, il faut utiliser le hook d'action **load-(page)**.
 - L'utilisation de ce hook permet également de n'exécuter la méthode que sur la page de l'extension.
 - Si vous n'utilisez aucune de ces deux méthodes, les fichiers CSS et JavaScript seront appelés sur toutes les pages de l'administration, ce qui peut créer des conflits, soit avec les pages des autres extensions, soit avec WordPress directement.
 - [http://codex.wordpress.org/Plugin_API/Action_Reference/load-\(page\)](http://codex.wordpress.org/Plugin_API/Action_Reference/load-(page))
 - Modifiez pour cela la méthode **init()** de cette façon :

```
function init(){  
    if(function_exists('add_options_page')){  
        $mapage=add_options_page("My Gmap", 'My Gmap', 'administrator',  
        __FILE__, array($this,'gmap_admin_page'));  
        add_action('load-'.$mapage, array($this,'gmap_admin_header'));  
    }  
}
```

La création de sa propre extension

Créer une extension avec PHP/MySQL

- **Créer des shortcodes pour chaque carte**
- Créez des shortcodes en leur faisant passer des paramètres.
- Passez l'id dans la fonction du shortcode. Le résultat du shortcode doit être de cette forme :
- Ajoutez la fonction créant le shortcode en dehors de la classe, avec les hooks d'action :
- Puis, créez la méthode `gmap_shortcode()`, utilisez `$att` comme argument, puis récupérez l'id de cette façon : `$att['id']`.
- À l'intérieur de la méthode `gmap_shortcode()`, faites appel à la méthode `getmap($id)`, permettant de récupérer les informations d'une carte selon son id. Puis utilisez le code JavaScript d'affichage de la carte Google Maps, en remplaçant les valeurs pour la longitude et la latitude :

```
[mygmap id="id_de_la_carte" ]
```

```
if(function_exists('add_shortcode')){  
    add_shortcode('mygmap',array($inst_map, 'gmap_shortcode'));  
}
```

```
function gmap_shortcode($att){  
  
    $maplist=$this->getmap($att['id']);  
    ?>  
    <div id="map" style="width:400px;height:400px"></div>  
  
    <script type="text/javascript">  
        var coord=new google.maps.LatLng('php echo $maplist[0]-&gt;latitude ?&gt;','<br/                                           'php echo $maplist[0]-&gt;longitude ?&gt;'<br/                                           );  
  
        var options = {  
            center: coord,  
            zoom: 10,  
            mapTypeId: google.maps.MapTypeId.ROADMAP  
        };  
        var map=new google.maps.Map(document.getElementById("map"),options);  
    </script>  
    <?php  
}
```


La création de sa propre extension

Créer une extension avec PHP/MySQL

insérez le shortcode suivant :
`[mygmap id=";<?php echo $maplist[0]->id ?>";]`

```
<div id="placecode">
  Copiez (ctrl+c) le code et collez-le (ctrl+v) dans la
  page ou l'article où vous voulez voir apparaître votre carte :
  <input id="codemap" type="texte" value="[mygmap id=&quot;;
  <?php echo $maplist[0]->id ?>&quot;; ]" readonly="readonly" />
</div>
```

```
jQuery("#codemap").click(function(){
    this.select();
});
```



- **Afficher le code sur la page de chaque carte**
- Pour afficher la carte sur les pages du site, utilisez le shortcode `[mygmap id="id_de_la_carte"]`. Cela permet de faciliter l'utilisation de l'extension. Affichez le shortcode sur les pages de chaque carte, ainsi l'utilisateur aura le shortcode correspondant à la carte sous les yeux.
- Dans le fichier `template-map.php`, ajoutez :
- Pour une saisie plus facile, créez un input avec pour valeur le shortcode correspondant. Utilisez `readonly` pour empêcher l'écriture à l'intérieur de `input`.
- Utilisez du code jQuery, pour que l'utilisateur sélectionne automatiquement le texte, quand il clique dans le champ `input`.
- Pour cela, ajoutez à la balise `input` l'id `codemap`.
- Modifiez le code HTML précédent comme cela :
- Dans le fichier `admin-gmap.js`, ajoutez le code jQuery permettant de sélectionner automatiquement le texte de la balise `input` ayant l'id `codemap`, grâce à la fonction jQuery `select()` :
- N'oubliez pas de modifier le fichier `readme.txt`, de lui adjoindre les instructions d'utilisation, et de faire des impressions écran de l'extension.

La création de sa propre extension

Créer une extension avec PHP/MySQL

- Voici ce que vous devez voir dans l'administration :

The screenshot displays the WordPress admin dashboard for a user named 'admin'. The left sidebar contains the standard WordPress menu, with 'Réglages' (Settings) highlighted. The main content area is titled 'My Gmap' and features a tabbed interface with 'créer une carte' (create a map) selected. Below the tabs, the 'Carte : Paris' (Map: Paris) section is active. It provides instructions to copy a code snippet and shows the current map parameters: Title (Paris), Latitude (48.856713), and Longitude (2.348777). There are buttons to 'mettre à jour' (update) and 'supprimer la carte' (delete the map). An 'Aperçu' (Preview) section shows a map of Paris with a small inset map. The bottom of the map area includes links for 'Données cartographiques', 'Conditions d'utilisation', and 'Signaler une erreur cartographique'.

WordPress admin interface showing the 'My Gmap' extension settings.

Navigation menu (left sidebar):

- Tableau de bord
- Articles
- Médias
- Pages
- Commentaires
- Apparence
- Extensions
- Utilisateurs
- Outils
- Réglages
- Général
- Écriture
- Lecture
- Discussion
- Médias
- Permalien
- Confidentialité
- My Gmap
- Catch IDs
- Réduire le menu

Main content area (My Gmap):

créer une carte **Paris** Marseille

Carte : Paris

Copiez (ctrl+c) le code et collez (ctrl+v) dans la page ou l'article où vous voulez voir apparaître votre carte : `[mygmap id="1"]`

paramètres :

Titre* : Paris

Latitude* : 48.856713

Longitude* : 2.348777

mettre à jour

* champs obligatoires

supprimer la carte

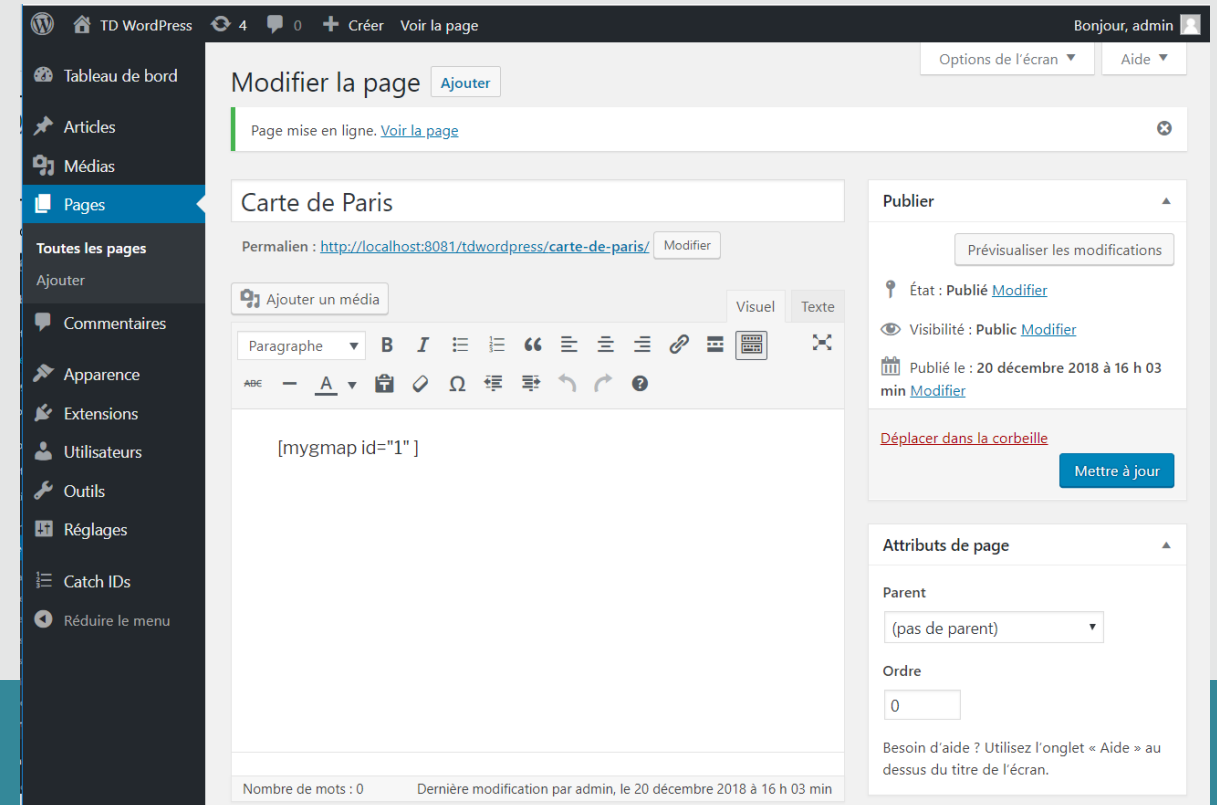
Aperçu :

Map preview showing Paris and surrounding areas.

La création de sa propre extension

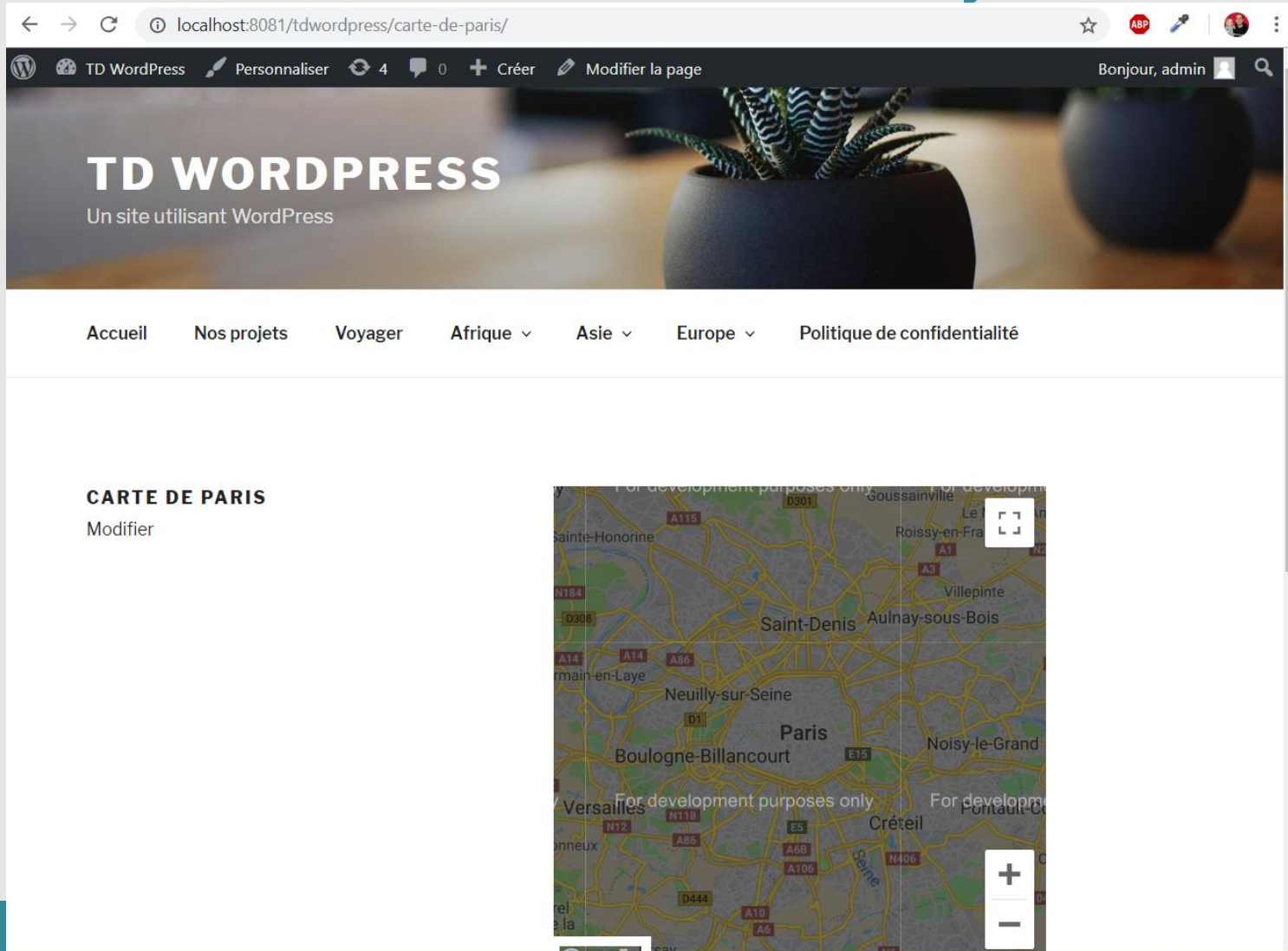
Créer une extension avec PHP/MySQL

- Quand vous ajoutez le code du shortcode sur une page, vous obtenez :
*Page **Pages**, avec l'insertion du shortcode d'une carte.* Sur le site, vous devez voir la carte apparaître, quel que soit le thème :
 - Si une erreur PHP de type « header already sent ... » survient, pensez à modifier dans le fichier php.ini de votre hébergeur ou de votre serveur local la ligne output_buffering et mettez-lui la valeur on.
- Exemple
 - Remplacez : output_buffering = 4096
par : output_buffering = on



La création de sa propre extension

Créer une extension avec PHP/MySQL



La création de son propre thème

Démonstration

Créer une extension avec PHP & MySQL

