Widget de WordPress

Widget WP: définition et structure

Rappel: Un widget WP est un zone autonome d'une page web qui remplit une fonction spécifique.
C'est aussi le code qui génère cette zone.

Structure générale du widget : Un widget hérite de la classe WP_Widget
déclarée dans wp-includes/ class-wp-widget.php

```php
<?php
class Mon_Widget extends WP_Widget {

    public function __construct() {
        // initialiser le widget en appelant le constructeur de la classe parent
    }

    public function widget( $args, $instance ) {
        // traiter la fonctionnalité du widget et afficher le résultat
    }

    public function form( $instance ) {
        // coder le formulaire de saisie des réglages de configuration
        // (s'affiche dans la page d'administration des widgets)
    }

    public function update( $new_instance, $old_instance ) {
        // modifier les réglages de configuration
        // en retour du formulaire
    }
}
```

Créer la classe Mon_Widget en tant que classe enfant de la classe WP_Widget, en codant ces 4 méthodes.

```php
function register_widgets() {
    register_widget('Mon_Widget');
}

add_action('widgets_init', 'register_widgets');
```

Coder la séquence d'initialisation du widget Mon_Widget : Il s'initialise par la fonction WP register_widget() qui doit s'exécuter au moment de l'exécution du crochet d'actions widgets_init.

Widget WP: pratique, création d'un widget dans l'extension n41-recipes

Il s'agit de créer un widget qui affiche un lien vers la page de la dernière recette enregistrée dans la table recipes.

Pour cela, créer le fichier spécifique class-n41-recipes-widget-news.php qui contiendra le code de la classe N41_Recipes_Widget_News, puis y ajouter successivement les méthodes :

1. Méthode __construct()

```
class N41_Recipes_Widget_News extends WP_Widget {


// initialiser le widget en appelant le constructeur de la classe parent
public function __construct() {
        $widget_ops = array(
                            'classname'   => 'n41_recipes_widget_news',
                            'description' => 'Affiche le nom de la dernière recette enregistrée.'
                            );

        parent::__construct(
                            // identifiant unique du widget
                            'n41_recipes_widget_news',
                            // nom du widget qui apparait dans la page d'administration des widgets
                            'N41 Recipes - Dernière recette',
                            $widget_ops
                            );
}
```

Codes des couleurs: structure standard, personnalisation

Widget WP: pratique, création d'un widget dans l'extension n41-recipes

2. Méthode form($instance)

```
// coder le formulaire de saisie des réglages de configuration
// (s'affiche dans la page d'administration des widgets)
public function form( $instance ) {

    // ici un seul paramètre de configuration: le titre du widget
    // qui est affiché dans la zone du widget sur les pages du site
    $title = isset( $instance['title'] ) ? esc_attr( $instance['title'] ) : '';
    ?>
    <p>
      <label for="<?php echo $this->get_field_id('title'); ?>">
        <?php _e('Title:'); // 'echo' de la traduction du texte Title ?>
        <input class="widefat"
               id="<?php echo $this->get_field_id('title'); ?>"
               name="<?php echo $this->get_field_name('title'); ?>"
               type="text"
               value="<?php echo $title; ?>" />
      </label>
    </p>
    <?php
}
```

Codes des couleurs: structure standard, personnalisation (aucune ici)

Widget WP: pratique, création d'un widget dans l'extension n41-recipes

3. Méthode update($new_instance, $old_instance)

```
// modifier les réglages de configuration en retour du formulaire
public function update( $new_instance, $old_instance ) {

    $instance = $old_instance;

    $instance['title'] = sanitize_text_field( $new_instance['title'] );

    return $instance;
}
```

Codes des couleurs: structure standard, personnalisation (aucune ici)

Widget WP: pratique, création d'un widget dans l'extension n41-recipes

4. Méthode widget($args, $instance)

```
// traiter la fonctionnalité du widget et afficher le résultat
// ici, rechercher la dernière recette enregistrée dans la table recipes
// et afficher un lien vers la page de cette recette
public function widget( $args, $instance ) {

    $title = !empty( $instance['title'] ) ? $instance['title'] : __('Last recipe');
    /** Ce crochet de filtres est documenté
        dans wp-includes/widgets/class-wp-widget-pages.php */
    $title = apply_filters( 'widget_title', $title, $instance, $this->id_base );

    // le tableau args contient les codes html de mise en forme enregistrés
    //  par la fonction WP register_sidebar dans le thème courant
    echo $args['before_widget'];

    if ( $title ) {
        echo $args['before_title'] . $title . $args['after_title'];
    }

    // Affichage du lien vers la page de la dernière recette
    $this->get_last_recipe();

    echo $args['after_widget'];
}
```

Codes des couleurs: structure standard, personnalisation

https://codex.wordpress.org/Function_Reference/register_sidebar

Widget WP: pratique, création d'un widget dans l'extension n41-recipes

5. Méthode get_last_recipe() , spécifique pour traiter la fonctionnalité du widget

```
public function get_last_recipe() {
    global $wpdb;
    // récupération de la dernière recette dans la table recipes
    $sql = "SELECT * FROM $wpdb->prefix"."recipes ORDER BY id DESC LIMIT 1";
    $recipe = $wpdb->get_row($sql);
    if ($recipe  !== null) :
       // récupération du lien vers la page générique d'affichage d'une recette
       $postmeta = $wpdb->get_row("SELECT * FROM $wpdb->postmeta
                                   WHERE meta_key = 'n41_recipes' AND meta_value = 'single'");
       $single_permalink = get_permalink($postmeta->post_id);
    ?>
    <ul>
       <li>
         <a href="<?php echo $single_permalink.'?page='.
                             stripslashes($recipe->title).
                             '&id='.$recipe->id?>">
            <?php echo stripslashes($recipe->title) ?>
         </a>
       </li>
    </ul>
    <?php
    else :
    ?>
    <p>Aucune recette enregistrée.</p>
    <?php
    endif;
}
```

Widget WP: pratique, création d'un widget dans l'extension n41-recipes

Pour finaliser le code de ce widget, insérer dans le fichier principal de l'extension, n41-recipes.php :

```
// chargement de la classe du widget
include ("class-n41-recipes-widget-news.php");


// Initialisation de tous les widgets de l'extension (ici un seul, N41_Recipes_Widget_News)
function n41_recipes_register_widgets() {
     register_widget('N41_Recipes_Widget_News'); // classe du widget en paramètre
}


// déclenchement de la fonction d'initialisation des widgets de l'extension
// dans les actions du crochet widgets_init
add_action('widgets_init', 'n41_recipes_register_widgets');
```

Debug WP: pratique, ajout d'une trace dans un fichier à la racine du site

Plus généralement, pour faciliter la mise au point des développements sous WP, vous pouvez ajouter des séquences de code qui enregistrent des traces dans un fichier à la racine du site.
Ces traces de passage à des points précis des programmes, peuvent contenir le vidage de variables contextuelles.

Par exemple pour analyser le contenu des paramètres de la méthode "update" vue précédemment, vous pouvez insérer la séquence délimitée par les commentaires "DEBUG N41" et "FIN DEBUG N41" :

```
public function update( $new_instance, $old_instance ) {

    // DEBUG N41
    $nfile = fopen(ABSPATH."n41-debug.log", "a");

    $value = date("Y-m-d H:i:s ").__METHOD__." : old_instance"
             .print_r($old_instance, true). "\n";

    fwrite($nfile, $value);

    $value = date("Y-m-d H:i:s "). __METHOD__." : new_instance"
             .print_r($new_instance, true). "\n";

    fwrite($nfile, $value);

    fclose($nfile);
    // FIN DEBUG N41
```

La constante ABSPATH définie dans wp-config.php, permet de créer le fichier trace à la racine du site.