Mise en place d'un thème Wordpress avec Bootstrap[1]

Préambule

Le thème que nous allons mettre en place s'inspire de celui présenté ici :
Blog Template · Bootstrap (getbootstrap.com)

Création du thème

Un thème = un dossier !

1. **Allez** dans le dossier **wp-content/themes** de WordPress
2. Créez le dossier de notre thème, appelons le **wpbootstrap**
3. Créez le fichier **style.css**
4. **Ouvrez le fichier style.css et ajoutez** les lignes suivantes :

```
/* Theme Name: wpbootstrap
Description: Thème WordPress basé sur Bootstrap Version: 1.0
*/
```

5. **Créez** une image de votre choix en **png ou jpg** et de définition **1200x900**. Le fichier doit être nommé **screenshot.png** (ou .jpg).
6. **Ajoutez** cette image dans le dossier de votre thème. Cette image sera utilisée par WP pour l'affichage de votre thème dans l'interface d'administration.

Création des fichiers nécessaires

Les fichiers bootstrap

Pour ajouter les fichiers du framework Bootstrap, nous allons créer un fichier « functions.php » à la racine de notre thème.
NB : le fichier **functions.php est obligatoire** pour les thèmes des versions actuelles de WP (02/23).

1. **Créez** le fichier **functions.php** dans votre dossier de thème.

Comme vu précédemment, Bootstrap est constitué d'un ensemble de fichiers :

- Le fichier « bootstrap.css » qui regroupe l'ensemble des styles du framework ;
- La librairie javascript jQuery ;
- La libraire Popper.js qui permet de gérer des popins sur son site web ;
- Et enfin, le fichier bootstrap.js qui contient tous les composants javascript de Bootstrap.

2. **Ouvrez** le fichier **functions.php** et insérez les lignes de code suivantes :

```php
<?php
    // Chargement des styles et des scripts Bootstrap sur WordPress
    function wpbootstrap_styles_scripts(){
        wp_enqueue_style('style', get_stylesheet_uri());
        wp_enqueue_style('bootstrap', 'https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/css/bootstrap.min.css');
        wp_enqueue_script('jquery');
        wp_enqueue_script('popper', 'https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.14.7/umd/popper.min.js', array('jquery'), 1, true);
        wp_enqueue_script('boostrap', 'https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/js/bootstrap.min.js', array('jquery', 'popper'), 1, true);
    }
    add_action('wp_enqueue_scripts', 'wpbootstrap_styles_scripts');
?>
```

[1] Merci à Quentin Bodet qui nous a autorisés à utiliser son travail !
Le blog pour apprendre WordPress de A à Z : TutoWP.fr

Leur fonctionnement a été expliqué dans le premier CM.

Les fichiers de template WP

Pour construire ce thème, nous allons créer plusieurs **fichiers de template** :

- Les templates de base **index.php**, **header.php** et **footer.php** ;
- Un template pour la **page d'accueil**.
-

1. **Créez** le fichier **index.php** dans votre dossier de thème.
2. **Insérez** les lignes de code suivantes :

```php
<?php get_header(); ?>
<?php get_template_part('loop'); ?>
<?php get_footer(); ?>
```

Les fonctions **get_header()** et **get_footer()** insèreront les instructions des templates **header.php** et **footer.php**.
Comme get_sidebar(), ces deux fonctions permettent l'inclusion spécifiques des templates correspondants.
Si l'on souhaite insérer du code provenant d'un template quelconque, on utilise alors la fonction **get_template_part()** , qui admet comme **paramètre le nom du fichier** template (sans l'extension) :

```
<?php get_template_part('loop') ; ?> //insère le code du template loop.php
```

Nous allons maintenant créer les templates « header.php », « loop.php » et « footer.php » pour que tout fonctionne.

3. **Créez** les fichiers **header.php** , **footer.php** et **loop.php** dans votre dossier de thème.
4. Dans le fichier **header.php**, insérez les lignes de code suivantes :

```php
<!DOCTYPE html>
<html lang="fr">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta http-equiv="X-UA-Compatible" content="ie=edge">
    <title><?php wp_title(); ?></title>
    <?php wp_head(); ?>
</head>
<body>
```

La fonction **wp_title()** affiche ici le titre de la page.
Rem. : wp_title(**false**) renvoie le titre au lieu de l'afficher.

La fonction **wp_head()** effectue un nombre important de tâches.
Elle permet notamment à WordPress, aux thèmes et aux plugins d'ajouter du code HTML dans un template.

WordPress utilise cette fonction pour **insérer les styles et scripts des thèmes et plugins** dans l'élément **head** du thème. Cette fonctionnalité n'est possible que si les styles/scripts ont été ajoutés à la liste grâce aux fonctions **wp_enqueue_xxxx()** dans **functions.php**.
Le fichier loop.php va contenir le code de base pour afficher les articles (*posts*).
Comme son nom l'indique, c'est une boucle dans laquelle on récupère et affiche les articles.

5. Dans le fichier **loop.php**, insérez les lignes de code suivantes :

```php
<?php if(have_posts()) : ?>
    <?php while(have_posts()) : the_post(); ?>
        <div class="blog-post">
            <h2 class="blog-post-title"><?php the_title(); ?></h2>
            <p class="blog-post-meta"><?php the_time('d/m/Y'); ?> par <?php the_author(); ?></p>
            <?php (is_singular()) ? the_content() : the_excerpt(); ?>
            <?php if(!is_singular()) : ?>
                <p><a href="<?php the_permalink(); ?>" class="btn btn-primary">Lire la suite</a></p>
            <?php endif; ?>
        </div>
    <?php endwhile; ?>
    <div id="pagination">
       <?php echo paginate_links(); ?>
   </div>
<?php endif; ?>
```

On commence par tester la présence de posts : **if(have_posts())**
Si oui, on récupère par une boucle **while/endwhile** les posts **successivement**. En effet, **the_post()** récupère le post à l'**index courant** (en partant du premier) puis **incrémente cet index** pour le prochain the_post(). Lorsque cet index dépasse le nombre de posts disponibles, **have_post()** renvoie **faux** et **la boucle s'arrête**.
Quelques fonctions de post utilisées ici :

- **the_title()** : renvoie le titre
- **the_time()** : renvoie la date de création
- **the_author()** : renvoie le nom de l'auteur
- **the_content()** : renvoie le contenu complet
- **the_excerpt()** : renvoie le contenu abrégé
- **the_permalink()** : renvoie le lien vers le post complet

On utilise ici la fonction **is_singular()** pour déterminer le contexte. S'il s'agit d'une page ou d'un article, on affiche le contenu complet : the_content() ; sinon, on affiche la version abrégée : the_excerpt(), puis on insère le lien vers l'article complet avec le texte *Lire la suite*.
Le code de l'explication ci-dessus :

```
<?php (is_singular()) ? the_content() : the_excerpt(); ?>
<?php if(!is_singular()) : ?>
     <p><a href="<?php the_permalink(); ?>" class="btn btn-primary">Lire la suite</a></p>
<?php endif; ?>
```

Enfin, la fonction **paginate_links()** met en place une pagination des différents articles.
NB : les différentes classes blog-postxxxx seront définis dans style.css. Les classes **bnt** et **btn-primary** font partie de **Bootstrap**.

6. Dans le fichier **footer.php**, insérez les lignes de code suivantes :

```php
<?php wp_footer(); ?>
</body>
</html>
```

On appelle la fonction **wp_footer()** – importante car c'est par elle que s'installent les scripts des plug-ins entre autres, puis on ferme les éléments body et html du document.
Nous allons maintenant ajouter un menu à notre thème.
Ouvrez le fichier **function.php** et insérez-y le code suivant :

```php
function wpbootstrap_after_setup_theme() {
    // On ajoute un menu
    register_nav_menu('header_menu', "Menu du header");
    // On ajoute une classe php permettant de gérer les menus Bootstrap
    require_once get_template_directory() . '/class-wp-bootstrap-navwalker.php';
}
add_action('after_setup_theme', 'wpbootstrap_after_setup_theme');
```

La fonction **register_nav_menu()** enregistre notre menu (nommé **header_menu**) et le rend disponible pour l'administration. Notre fonction **wpbootstrap_after_setup_theme()** est ensuite « accrochée » au hook d'actions **after_setup_theme**.
NB : l'affichage des menus sur WordPress n'étant pas compatible avec Bootstrap par défaut, nous ajoutons une **classe php** qui permet de gérer cela : **class-wp-bootstrap-navwalker.php**
Cette classe est disponible sur le site et doit être installée dans le dossier du thème.
Attention : notre menu est seulement déclaré, il nous faut maintenant l'installer dans notre page header.php.

7. **Ouvrez** le fichier **header.php** et insérez le code suivant :

```php
<div class="container">
    <!-- Header -->
    <header class="blog-header py-3">
        <div class="row flex-nowrap justify-content-between align-items-center">
            <div class="col-12 text-center">
                <a class="blog-header-logo text-dark" href="<?php bloginfo('url'); ?>"><?php bloginfo('name') ?></a>
            </div>
        </div>
    </header>
    <!-- Fin du header -->

    <!-- Menu header -->
    <nav class="navbar navbar-expand-md navbar-light" role="navigation">
        <div class="container">
            <button class="navbar-toggler" type="button" data-toggle="collapse" data-target="#header-menu" aria-controls="#header-menu" aria-expanded="false" aria-label="Toggle navigation">
                <span class="navbar-toggler-icon"></span>
            </button>
            <?php
                wp_nav_menu(array(
                    'theme_location'    => 'header_menu',
                    'depth'             => 2,
                    'container'         => 'div',
                    'container_class'   => 'collapse navbar-collapse',
                    'container_id'      => 'header-menu',
                    'menu_class'        => 'nav navbar-nav',
                    'fallback_cb'       => 'WP_Bootstrap_Navwalker::fallback',
                    'walker'            => new WP_Bootstrap_Navwalker(),
                ));
            ?>
        </div>
    </nav>
    <!-- Fin du menu header -->
```

Les classes blog-xxx seront définies dans **style.css**. Les autres classes sont majoritairement des classes **bootstrap**.

L'élément **header** (classe **blog-header**) contient les informations générales du site récupérées par la fonction WP **bloginfo()**. Cette fonction est utilisée ici pour afficher l'**url** du site ainsi que le **titre**, informations définies dans Settings >General de l'interface d'administration.
L'élément **nav** (classe BS **navbar**) contient le code permettant d'ajouter notre menu grâce à la fonction **wp_nav_menu()**.
Cette fonction admet comme paramètre un tableau associatif permettant de configurer le menu. Nous ajoutons donc la structure de menu de Bootstrap et nous indiquons le menu WordPress que nous avons enregistré auparavant dans le fichier **functions.php**. L'identifiant de notre menu est repris dans le paramètre **theme_location**. Pour le reste, on applique les classes Bootstrap et on ajoute le paramètre **walker** avec la classe PHP que nous avons téléchargée et qui permet de mettre en forme les menus WordPress pour Bootstrap.
Notez toutefois que ce menu devra être complété (liens) dans l'interface d'administration.

Le template de notre page d'accueil

Nous pouvons maintenant créer notre page d'accueil en partant du fichier **index.php**.

1. **Ouvrez** le fichier **index.php** et enregistrez-le avec le nom **front-page.php**.

Il s'agit du nom réservé WP pour le template de la page d'accueil.
Nous allons **modifier front-page.php** pour mettre en avant le premier post.

2. Insérez le code suivant :

```php
<?php get_header(); ?>
<?php
    $query = new WP_Query(['posts_per_page' => 1]);
    if($query->have_posts()) : while($query->have_posts()) : $query->the_post();
?>
    <div class="jumbotron p-4 p-md-5 text-white rounded bg-dark">
        <div class="col-md-6 px-0">
            <h1 class="display-4 font-italic"><?php the_title(); ?></h1>
            <p class="lead my-3"><?php the_excerpt(); ?></p>
            <p class="lead mb-0"><a href="<?php the_permalink(); ?>" class="text-white font-weight-bold">Lire la suite</a></p>
        </div>
    </div>
<?php endwhile; endif; ?>

<?php
    query_posts('offset=1');
    get_template_part('loop');
?>

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

La solution consiste à :

A. Récupérer le premier article et lui affecter des classes le distinguant des autres posts ;
B. Récupérer ensuite les autres articles grâce au template loop.php mais en commençant par le deuxième.

A. Nous créons une nouvelle requête WordPress en demandant un seul article (via le paramètre **posts_per_page**) puis nous affichons le résultat de la requête avec les classes de Bootstrap. C'est en particulier ici la classe BS **jumbotron**, qui permet de créer un bloc de mise en avant.

B. Pour que la requête de base du template **loop** n'affiche pas une deuxième fois le premier article, on décale « manuellement » l'index grâce à la fonction **query_posts()**.

La fonction **query_posts()** permet de modifier la requête principale. Nous l'utilisons ici avec le paramètre **offset** :

```
query_posts('offset=1');//décale l’index de 1 (premier article non traité dans notre cas)
get_template_part('loop');// affiche les articles suivants
```

Ajout d'une barre latérale (*sidebar*)

Dans le fichier **functions.php**, **insérez** le code suivant :

```php
// On ajoute une sidebar
function wpbootstrap_sidebar() {
    register_sidebar([
        'name'          => "Sidebar principale",
        'id'            => 'main-sidebar',
        'description'   => "La sidebar principale",
        'before_widget' => '<div id="%1$s" class="widget %2$s p-4">',
        'after_widget'  => '</div>',
        'before_title'  => '<h4 class="widget-title font-italic">',
        'after_title'   => '</h4>',
    ]);
}
add_action('widgets_init', 'wpbootstrap_sidebar');
```

Nous utilisons la fonction **register_sidebar()** pour créer la sidebar et nous l'encapsulons dans une fonction **wpbootstrap_sidebar()** que l'on accroche au hook **widgets_init**.

Nous pouvons maintenant ajouter notre sidebar dans notre template front-page.php.

1. Dans **front-page.php**, **insérez** le code suivant :

```php
<div class="container">
    <div class="row">
        <div class="col-md-8">
            <?php
                query_posts('offset=1');
                get_template_part('loop');
            ?>
        </div>
        <div class="col-md-4">
            <?php dynamic_sidebar('main-sidebar'); ?>
        </div>
    </div>
</div>
```

Il s'agit d'une organisation en colonne utilisant les classes BS. La première colonne occupe les deux premiers tiers (col-md-8) pour afficher le premier article et la deuxième occupe le dernier tiers (col-md-4) pour afficher notre sidebar.

Nous devons maintenant revenir dans footer.php pour entre autres fermer la div principale ouverte dans header.php.

2. Dans **footer.php**, **insérez** le code suivant :

```
    </div>
    <footer class="blog-footer">
    <p>Thème <a href="https://getbootstrap.com/">Bootstrap</a> pour WordPress par <a href="https://www.tutowp.fr/tutowp">TutoWP.fr</a>.</p>
    <p>
        <a href="#">Retour en haut</a>
    </p>
    </footer>
<?php wp_footer(); ?>
</body>
</html>
```

Retour vers le CSS

Nous allons insérer les éléments CSS nécessaires dans style.css.

1. Téléchargez le fichier **wp_bs_style.txt** et coller le contenu dans votre fichier **style.css**.

Commentaires pour les pages et articles

Nous allons nous commencer par modifier le template loop.php.
Ouvrez le fichier loop.php et ajoutez la ligne encadrée en rouge :

```
<?php if(have_posts()) : ?>
    <?php while(have_posts()) : the_post(); ?>
        <div class="blog-post">
            <h2 class="blog-post-title"><?php the_title(); ?></h2>
            <p class="blog-post-meta"><?php the_time('d/m/Y'); ?> par <?php the_author(); ?></p>
            <?php (is_singular()) ? the_content() : the_excerpt(); ?>
            <?php if(!is_singular()) : ?>
                <p><a href="<?php the_permalink(); ?>" class="btn btn-primary">Lire la suite</a></p>
            <?php endif; ?>
            <?php if(is_singular()) : if(comments_open()) : comments_template(); endif; endif; ?>
        </div>
    <?php endwhile; ?>
    <div id="pagination">
       <?php echo paginate_links(); ?>
   </div>
<?php endif; ?>
```

Celle-ci spécifie que si on est sur une page ou un article - **is_singular()**, ET si les commentaires sont activés – **comments_open()** , on affiche les commentaires grâce à la fonction **comment_template()**.
Toutefois, et comme pour les menus, nous devons utiliser une classe PHP pour gérer correctement l'affichage via Bootstrap. Il s'agit de la classe **wp-bootstrap-comment-walker.php** téléchargeable sur le site et à placer dans le dossier du thème.

Avant de créer notre template comments.php, nous allons activer le support html5 dans les commentaires.
Modifiez la déclaration de menu dans functions.php comme encadré en rouge :

```php
// On ajoute un menu
function wpbootstrap_header_menu() {
    register_nav_menu('header_menu', "Menu du header");
    // On ajoute une classe php permettant de gérer les menus Bootstrap
    require_once get_template_directory() . '/class-wp-bootstrap-navwalker.php';
    // On ajoute le support du html5 pour les listes de commentaires
    add_theme_support('html5', array('comment-list'));
}
add_action('after_setup_theme', 'wpbootstrap_header_menu');
```

Le template comments.php

Créez le fichier comments.php dans votre dossier de thème et insérez-y pour commencer le code suivant :

```php
<?php if(have_comments()) : ?>
    <ul class="list-unstyled">
        <?php
            // Register Custom Comment Walker
            require_once('class-wp-bootstrap-comment-walker.php');
            wp_list_comments([
                'style'         => 'ul',
                'short_ping'    => true,
                'avatar_size'   => '64',
                'walker'        => new Bootstrap_Comment_Walker(),
            ]);
        ?>
    </ul><!-- .comment-list -->
<?php endif; ?>
```

Ce code permet d'afficher la liste des commentaires.
Il reste ensuite à afficher le formulaire de commentaire. Le code est disponible dans le fichier **comments.txt**.

2. Téléchargez le fichier **comments.txt** et coller le contenu à la suite dans comments.php.