WordPress : la boucle

Boucle WordPress : définition

La « boucle » (ou « **loop** » en anglais) est l'un des concepts fondamentaux de WordPress qui **régit l'affichage des articles, des pages et d'autres types de contenu sur un site WordPress**. **La boucle est une structure de code qui récupère les données de contenu stockées dans la base de données et les affiche selon un modèle prédéfini**.

Elle prépare les données à afficher (titre, contenu, catégories, image à la une …) et permet de les appeler via des fonctions dédiées (ex. **the_content()** pour afficher le contenu…).

La boucle est généralement utilisée dans les **fichiers de modèle** tels que le fichier « **index.php**« , « **single.php**« , « **archive.php** » et d'autres fichiers similaires qui contrôlent l'apparence du contenu sur le site.

La logique de boucle sera la même sur l'ensemble des templates, vous pourrez d'ailleurs procéder à des copier/coller.

Voici comment fonctionne la boucle WordPress :

1. **Récupération des données :** la boucle commence par récupérer les données de contenu appropriées de la base de données. Cela peut inclure des articles, des pages, des types de publication personnalisés ou d'autres éléments, selon le contexte.
2. **Configuration des paramètres :** la boucle configure différents paramètres tels que le nombre de publications à afficher, les critères de filtrage, l'ordre de tri, etc. Ces paramètres sont souvent définis dans le fichier de modèle ou peuvent être modifiés en utilisant des fonctions spécifiques.
3. **Parcours des données :** la boucle parcourt les données récupérées une par une et exécute le code à l'intérieur de la boucle pour chaque élément. Cela permet de formater et d'afficher le contenu selon le modèle défini dans le fichier de modèle.
4. **Affichage du contenu :** à l'intérieur de la boucle, vous pouvez accéder aux différentes parties du contenu, telles que le titre, le contenu, les métadonnées, les images, etc. Vous pouvez également ajouter des conditions, des boucles imbriquées et d'autres fonctionnalités pour personnaliser l'affichage du contenu.

Structure de base de la boucle WordPress

```
<?php if( have_posts() ) : while( have_posts() ) : the_post(); ?>
        …
<?php endwhile; endif; ?>
```

La fonction **have_posts()** permet de vérifier s'il y a quelque chose à afficher.

Affichage des articles grâce à un **while** (« **boucle tant que** »).

La boucle WordPress est prévue pour fonctionner dans tous les cas de figure, qu'il y ait plusieurs

contenus à afficher (archives, résultats de recherche) ou un seul contenu (article ou page).

La fonction **`the_post()`** permet de précharger les données du contenu afin de les afficher dans la boucle.

La boucle WordPress par l'exemple

Un exemple plus complet :

header.php index.php

mon-theme > index.php

```php
<?php get_header(); ?> <!-- appel à l'en-tête -->

<main id="articles">
    <!-- Affichage de tous les articles de la page d'accueil -->
    <?php
        // Vérifier s'il y a des articles
        if(have_posts()) :
    ?>
        <?php
            // Tant qu'il y a des articles
            while(have_posts()) :
                // Récupérer son identifiant, titre, permalien, et son contenu
                the_post();
        ?>
            <article>
                <header class="titre_article">
                    <h2><a href="<?php
                            // Récupérer le permalien de l'article
                            the_permalink();
                        ?>">
                        <?php
                            // Récupérer / afficher le titre de l'article
                            the_title();
                        ?></a></h2>
                </header>
                <?php
                    // Récupérer / afficher le contenu de l'article
                    the_content();
                ?>
            </article>
        <?php endwhile; ?>
    <?php endif; ?>
</main>

<?php get_sidebar(); ?> <!-- appel à la colonne latérale -->
<?php get_footer(); ?> <!-- appel au pied de page -->
```

Des **templates tags** permettent d'afficher les données (ex. **`the_title()`** pour l'affichage du titre, **`the_content()`** pour l'affichage du contenu principal…).

Attention ! Les templates tags ne fonctionnent qu'à l'intérieur de la boucle WordPress.

Attention ! Ne placez pas de balise HTML <p> autour de the_content(), le contenu principal est déjà censé en contenir une.

Testez en intégrant une boucle dans le fichier **index.php** pour voir le rendu obtenu.

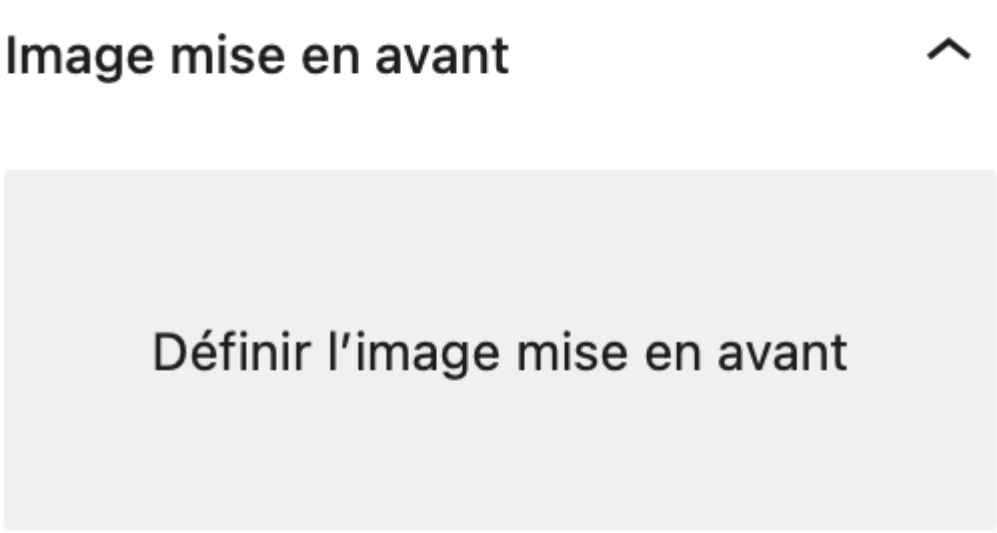
Complément : si vous souhaitez ajouter du **style CSS** pour améliorer le design, vous pouvez apprendre les bases du CSS depuis la formation gratuite proposée par MDN.

La liste des derniers articles : archive.php / home.php

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

        <h1>BLOG</h1>

        <?php if( have_posts() ) : while( have_posts() ) : the_post(); ?>
          <article class="post">
            // Affichage du titre de l'article
            <h2><?php the_title(); ?></h2>
            // Affichage de l'image à la une si renseignée
            <?php the_post_thumbnail(); ?>
            <p class="post_meta">
                Publié le <?php the_time( get_option( 'date_format' ) ); ?>
                par <?php the_author(); ?> • <?php comments_number(); ?>
            </p>
            // Affichage de l'extrait (résumé)
            <?php the_excerpt(); ?>
            <p>
                <a href="<?php the_permalink(); ?>" class="post_link">Lire la
suite</a>
            </p>
          </article>

        <?php endwhile; endif; ?>

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

Pour rappel, il ne doit y avoir qu'un seul **titre <h1> (titre principal)** par page, d'où l'utilisation du **niveau de titre <h2>** pour les titres des articles affichés dans la boucle WordPress.

Fonction the_post_thumbnail()



La gestion de l'image à la une (aujourd'hui appelée **image mise en avant**) doit au préalable avoir été activée depuis le fichier **functions.php** du thème via **`add_theme_support('post-thumbnails')`** et l'image renseignée dans le champ Image mise en avant de la publication.

La fonction **`the_post_thumbnail()`** retourne l'ensemble de la **balise HTML <img>** et non pas uniquement l'URL de l'image.

Fonctions the_time() et the_date()

La fonction **`get_option()`** avec '**date_format**' passé en paramètre va récupérer la valeur renseignée dans la configuration WordPress effectuée depuis **Réglages > Général > Format de date**.

Remarque : `the_time()` va permettre d'afficher la date même si celle-ci s'avère identique d'un article à l'autre, contrairement à **`the_date()`** qui afficherait une seule fois la date si des articles avaient la même date de publication.

Fonction the_author()

La fonction **`the_author()`** permet d'afficher le nom de l'auteur de la publication tel que renseigné depuis **Comptes > Profil > Nom à afficher publiquement**.

Fonction comment_number()

Pour afficher le nombre de commentaires dans une boucle WordPress, vous pouvez utiliser la fonction **`comments_number()`**.

```
// Afficher le nombre de commentaires
comments_number('Aucun commentaire', '1 commentaire', '% commentaires');
```

La fonction `comments_number()` accepte trois paramètres : le texte à afficher lorsque le nombre de commentaires est égal à zéro, le texte à afficher lorsque le nombre de commentaires est

égal à un, et le texte à afficher lorsque le nombre de commentaires est supérieur à un.

Dans l'exemple ci-dessous, si l'article a plus d'un commentaire, le texte « **% commentaires** » sera affiché, où `%` est remplacé par le nombre réel de commentaires.

Fonction the_excerpt()

La fonction **`the_excerpt()`** est utilisée dans WordPress pour afficher un extrait du contenu d'un article.

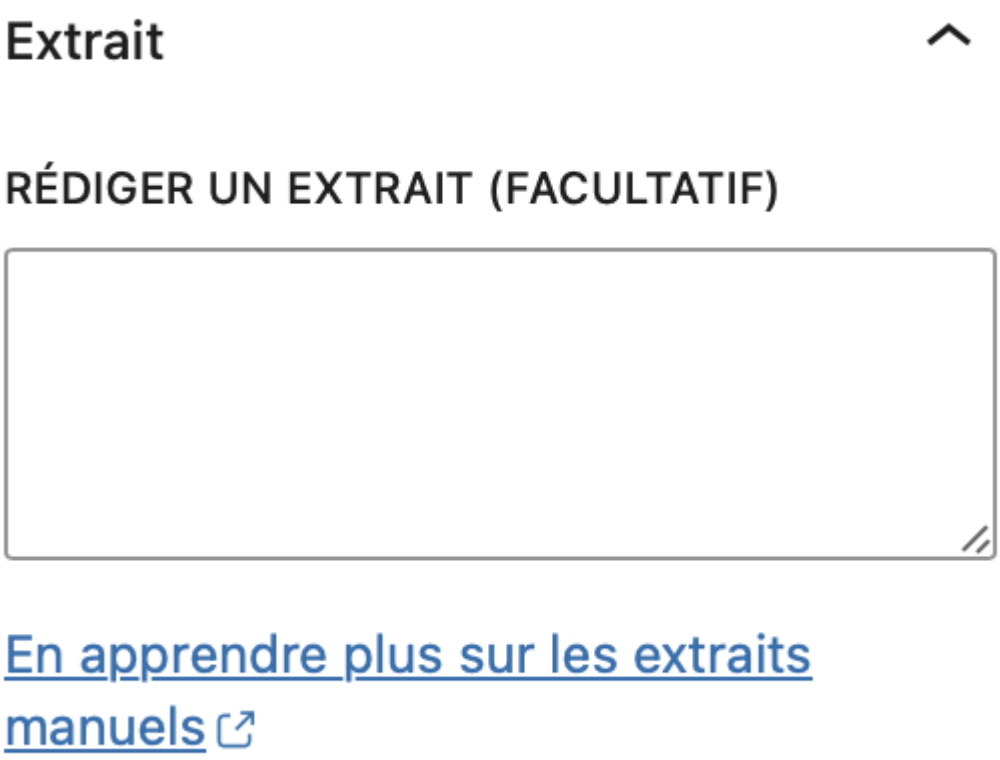
Attention ! Ne placez pas de balise HTML <p> autour de the_excerpt(), l'extrait est déjà censé en contenir une.

Par défaut, l'extrait est généré en prenant les premiers 55 mots du contenu de l'article.

WordPress ajoute également des points de suspension (…) à la fin de l'extrait par défaut.

Remarque : cette longueur par défaut peut être modifiée en utilisant le filtre **`excerpt_length`** dans votre fichier **`functions.php`** du thème WordPress.

Conseil : renseignez systématiquement un extrait, vous éviterez ainsi des découpes disgracieuses assurées par le système (ex. en plein milieu d'un mot, d'une phrase…)



Vous pouvez personnaliser l'extrait pour chaque article en utilisant la boîte d'édition « **Extrait** » dans l'éditeur WordPress, le contenu que vous ajoutez sera utilisé comme extrait pour l'article.

Si aucun extrait personnalisé n'est défini, WordPress générera automatiquement un extrait basé sur le contenu de l'article.

Vous pouvez par ailleurs utiliser le bloc « **Lire la suite** » dans le contenu l'article, si le champ « **Extrait** » est vide, c'est le contenu présent avant le bloc « Lire la suite » qui sera utilisé comme extrait.

Fonction the_permalink()

La fonction **`the_permalink()`** est utilisée dans WordPress pour afficher l'URL (lien permanent) d'un article ou d'une page.

La fonction **`the_permalink()`** génère et affiche automatiquement l'URL de l'article.

Vous pouvez l'utiliser pour envelopper du texte avec la **balise <a>** et créer un lien cliquable vers l'article.

L'URL générée par **`the_permalink()`** est basée sur les réglages des permaliens définis dans votre installation WordPress (**Réglages > Permaliens**).

Conseil : pour un meilleur référencement naturel, vérifiez bien que l'option par défaut de WordPress est bien configurée pour afficher des permaliens sous la forme « **Titre de la publication** » (**/%postname%/**).

Affichage d'un article seul : single.php

```
<?php get_header(); ?>
  <?php if( have_posts() ) : while( have_posts() ) : the_post(); ?>
    <article class="post">
      <?php the_post_thumbnail(); ?>

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

      <div class="post_meta">
        <?php echo get_avatar( get_the_author_meta( 'ID' ), 40 ); ?>
        <p>
          Publié le <?php the_date(); ?>
          par <?php the_author(); ?>
          Dans la catégorie <?php the_category(); ?>
```

```
          Avec les étiquettes <?php the_tags(); ?>
        </p>
      </div>

      <div class="post_content">
        <?php the_content(); ?>
      </div>
    </article>

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

Fonction get_avatar()

La fonction **`get_avatar()`** de WordPress est utilisée pour récupérer l'**avatar** (une représentation visuelle) d'un utilisateur.

La fonction **`get_avatar()`** prend un paramètre obligatoire, qui est l'adresse email de l'utilisateur ou l'ID pour lequel vous souhaitez obtenir l'avatar.

Par défaut, WordPress utilise les avatars fournis par **Gravatar (Globally Recognized Avatars)**, un service externe qui permet aux utilisateurs d'associer une image à leur compte utilisateur.

Si l'utilisateur a configuré un avatar Gravatar pour l'adresse email / ID fourni, **`get_avatar()`** récupérera cette image.

Si aucun avatar Gravatar n'est disponible pour l'adresse email, WordPress utilise par défaut un avatar générique fourni par le thème en cours ou par le fichier **`avatar-default.png`** du répertoire des images du thème.

Vous pouvez personnaliser la taille de l'avatar en utilisant un deuxième paramètre facultatif dans la fonction **`get_avatar()`**.

Par exemple, **`get_avatar($user_id, 80)`** affichera un avatar de 80 pixels de large.

Dans notre exemple, **`get_the_author_meta('ID')`** permet de récupérer l'identifiant du compte utilisateur ayant rédigé le contenu (auteur).

Fonctions the_category() et the_tags()

La fonction **`the_category()`** de WordPress est utilisée pour afficher les catégories auxquelles un article appartient.

Par défaut, **`the_category()`** affiche les catégories dans une liste HTML **`<ul>`**.

Vous pouvez personnaliser l'apparence des catégories en utilisant des arguments facultatifs dans la fonction **`the_category()`**.

Par exemple, **`the_category(', ')`** affichera les catégories séparées par des virgules et un espace, si un article appartient aux catégories « Actualités » et « Tutoriels », la fonction affichera « Actualités, Tutoriels ».

Chaque catégorie est un lien cliquable vers la page d'archive de cette catégorie.

Vous pouvez également utiliser la fonction **`get_the_category_list()`** pour récupérer une chaîne contenant les catégories de l'article sans l'afficher directement. Cela peut être utile si vous souhaitez personnaliser davantage la façon dont les catégories sont affichées.

La fonction **`the_tags()`** de WordPress est utilisée pour afficher les étiquettes (tags) associées à un article.

Par défaut, **`the_tags()`** affiche les étiquettes dans une liste HTML **`<ul>`**.

Vous pouvez personnaliser l'apparence des étiquettes en utilisant des arguments facultatifs dans la fonction **`the_tags()`**.

Par exemple, **`the_tags('', ', ')`** affichera les étiquettes sans aucun préfixe, séparées par des virgules et un espace, si un article a les étiquettes « WordPress » et « Développement Web », la fonction affichera « WordPress, Développement Web ».

Chaque étiquette est un lien cliquable vers la page d'archive de cette balise.

Vous pouvez également utiliser la fonction **`get_the_tag_list()`** pour récupérer une chaîne contenant les balises de l'article sans les afficher directement. Cela peut être utile si vous souhaitez personnaliser davantage la façon dont les balises sont affichées.

Les fonctions get_

Les fonctions commençant par **`get_`** dans WordPress sont utilisées pour **récupérer des informations spécifiques sans les afficher directement**.

Ces fonctions renvoient la valeur de l'information demandée plutôt que de l'afficher immédiatement.

Cela permet de manipuler les données avant de décider comment les afficher ou de les utiliser dans le code.

Voici quelques exemples courants :

- **`get_the_title()` :** récupère le titre de l'article en cours.
- **`get_the_permalink()` :** récupère l'URL (lien permanent) de l'article en cours.
- **`get_the_author()` :** récupère l'auteur de l'article en cours.
- **`get_the_category()` :** récupère les catégories auxquelles l'article en cours appartient.
- **`get_the_tags()` :** récupère les étiquettes associées à l'article en cours.
- **`get_the_excerpt()` :** récupère l'extrait de l'article en cours.

Il est important de noter que les fonctions `get_` ne doivent pas être utilisées en dehors de la boucle WordPress, car elles dépendent de l'article en cours pour récupérer les informations correctes.

```
<?php
  $title = get_the_title();
  // Ajouter un émoji devant le titre principal de l'article s'il contient le
```

```
terme "promotion"
  if( strpos( $title, 'promotion' ) !== false ) {
        $title = ' ' . $title;
   }
?>

  <h1><?php echo $title; ?></h1>
```

L’objet $post

L’objet **`$post`** en PHP WordPress est une **variable globale utilisée pour stocker les informations sur le contenu actuellement affiché ou traité dans le contexte de WordPress.** Cet objet contient toutes les données associées à une publication spécifique, qu’il s’agisse d’un article, d’une page ou d’un type de publication personnalisé.

Voici quelques utilisations courantes de l’objet `$post` :

1. **Accès aux données du contenu :** l’objet `$post` permet d’accéder aux différentes **propriétés du contenu**, telles que le titre, le contenu, les métadonnées, l’auteur, la date de publication, les catégories, les étiquettes, etc. Vous pouvez utiliser ces données pour personnaliser l’affichage du contenu dans vos thèmes ou plugins.
2. **Modification des données du contenu :** en modifiant les propriétés de l’objet `$post`, vous pouvez effectuer des modifications sur le contenu actuel. Par exemple, vous pouvez mettre à jour le titre, le contenu ou les métadonnées d’un article en utilisant les méthodes appropriées de l’objet `$post`.
3. **Utilisation des fonctions de traitement du contenu :** l’objet `$post` permet également d’utiliser des fonctions de traitement du contenu fournies par WordPress. Par exemple, vous pouvez utiliser la fonction **`get_permalink($post)`** pour obtenir l’URL de la publication actuelle, ou **`get_the_excerpt($post)`** pour obtenir un extrait du contenu.
4. **Navigation entre les publications :** en utilisant les méthodes et propriétés de l’objet `$post`, vous pouvez naviguer entre les publications liées. Par exemple, vous pouvez utiliser la méthode **`get_previous_post()`** pour obtenir la publication précédente, ou **`get_next_post()`** pour obtenir la publication suivante.

Les attributs de l’objet :

- **$post->ID :** identifiant
- **$post->post_title :** titre
- **$post->post_content :** contenu
- **$post->comment_count :** nombre de commentaires
- **$post->post_modified :** date de derrière modification
- **$post->post_type :** type de publication
- **$post->post_category :** catégorie(s) d’article

Il est important de noter que **l’objet `$post` est disponible dans le contexte approprié**, par exemple à l’intérieur de la boucle WordPress, où il est automatiquement peuplé avec les données de la publication en cours de traitement.

Hors de la boucle, vous devez utiliser des fonctions spécifiques, telles que `get_post()` ou `get_queried_object()`, pour obtenir l’objet `$post` correspondant à une publication spécifique.