Formation WordPress : Créer un thème “from Scratch” à l’aide de HTML/CSS, PHP, JS

Samir MOUSSAÏD

1. Introduction

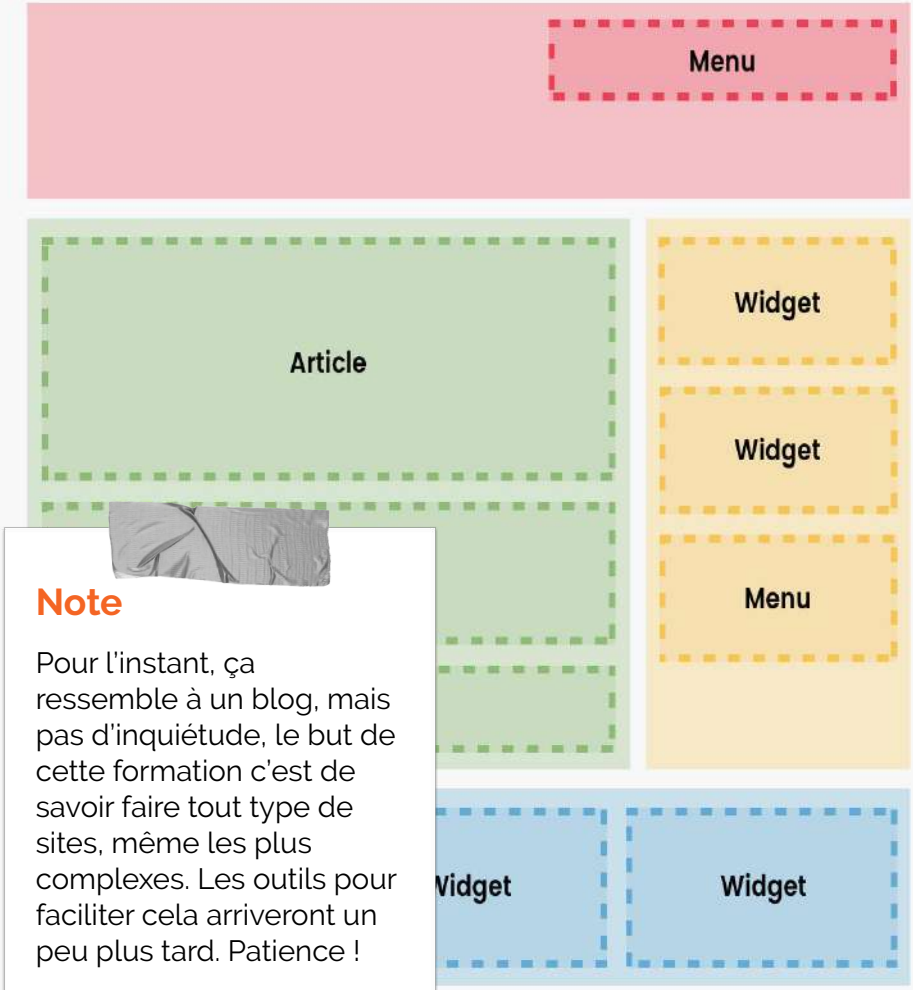
WordPress fournit toute une architecture ainsi que des fonctions pour vous aider dans la conception de votre thème. Et les maîtriser vous fera gagner beaucoup de temps !

- ➔ **Les zones**
  Mettez en avant ce qui est nouveau, inhabituel ou surprenant.
- ➔ **Les composants**
  Donnez aux gens des raisons de s'intéresser à votre propos.

Les zones

En me basant sur la structure d'un thème de type « blog » classique, je peux le découper en 4 zones :

- Le **Header** : l'en-tête, en haut, toujours la même selon la page
- Le **Footer** : le pied de page, en bas, également toujours le même sur chaque page
- Le **Content** : Le contenu principal, différent sur chaque page
- La **Sidebar** : Une éventuelle barre latérale contenant des informations transverses



Note

Pour l'instant, ça ressemble à un blog, mais pas d'inquiétude, le but de cette formation c'est de savoir faire tout type de sites, même les plus complexes. Les outils pour faciliter cela arriveront un peu plus tard. Patience !

Les composants

Menus, commentaires, navigation, widget sont des composants habituels d'un thème

Le rôle de ces composants est de rendre le thème dynamique, et le site administrable. On retrouve :

- Les **menus** : un dans le **header** en général, et certains dans la **sidebar** ou le **footer**
- Les **commentaires** : à la suite d'un article (dans les pages articles seulement)
- Les **widgets** : dans les **sidebars** ou le **footer** pouvant afficher les derniers articles, les catégories, un contenu statique…
- La **navigation** : pour aller à l'article suivant ou précédent.

2. Créer la base

On va maintenant créer la structure de code minimale qui va nous permettre de déclarer notre thème afin de l'activer dans l'interface d'administration de **WordPress.**

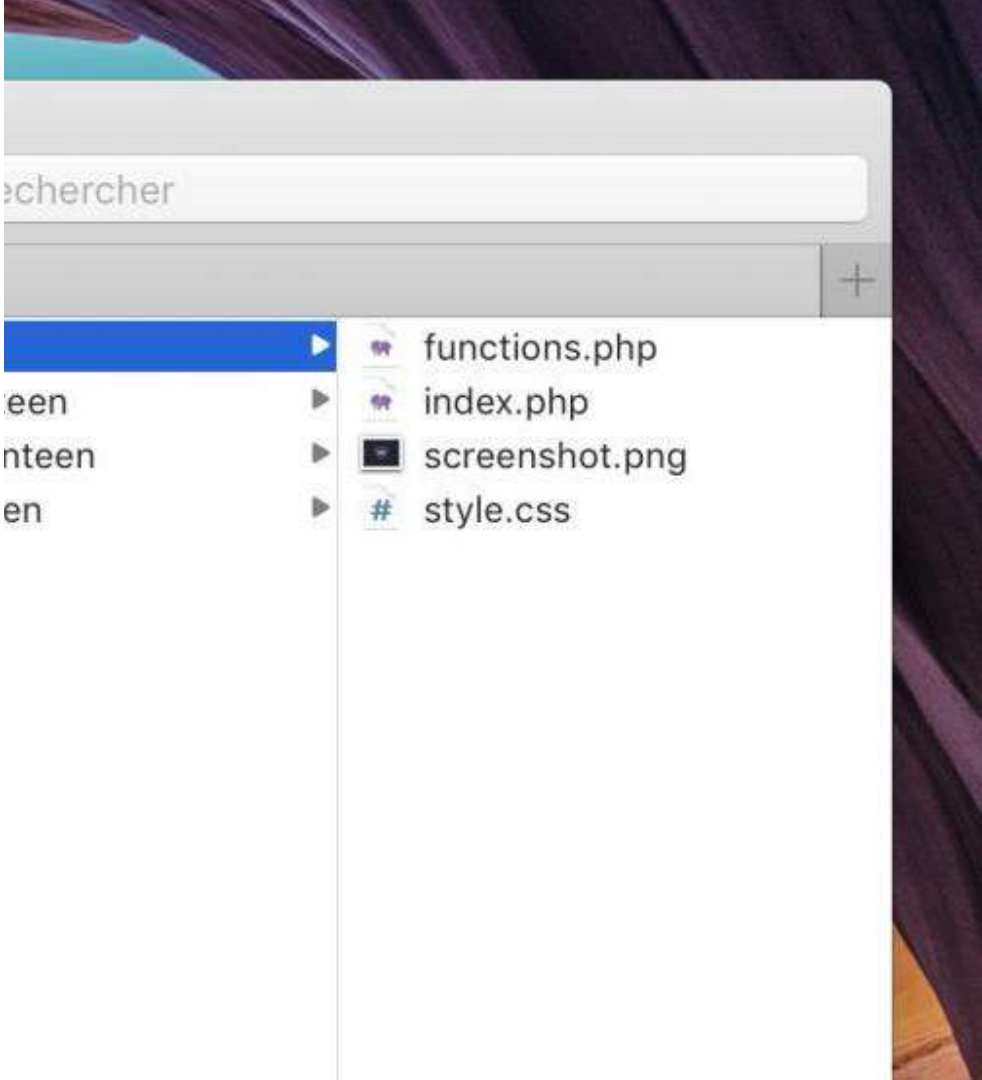
- **Déclarer son thème**
  1. Style.css
  2. Index.php
  3. Functions.php
- **Activer son thème**
  Expliquez le cas particulier d'une personne qui aurait besoin de votre solution.

Déclarer son thème

En tout premier lieu, on va se rendre dans le dossier wp-content/themes de notre site WordPress et créer un nouveau dossier qui va accueillir les fichiers de notre thème.

Je l'appelle mytheme puis je l'ouvre avec mon éditeur de code.
On va créer quelques fichiers :

- **style.css** , *requis*, qui va contenir nos styles mais aussi à déclarer notre thème à WordPress ;
- **index.php**, *requis*, qui contiendra pour l'instant l'unique page de notre thème ;
- **functions.php**, dans lequel on va pouvoir activer et ajouter des fonctionnalités ;



Style.css

Oui vous avez bien lu: C'est style.css qui permet à WordPress de savoir que c'est un thème, grâce aux quelques commentaires que l'on va insérer dedans.

Ouvrez votre fichier style.css et ajoutez-y les commentaires ci-dessous:

```
/*
Theme Name: MyTheme
Theme URI: https://wp.bzez.dev
Author: Samir MOUSSAÏD
Author URI: https://bzez.dev
Description: Mon premier thème !
Requires at least: WordPress 5.0
Version: 1.0
*/
```

Attention !

Ne mettez pas d'espace avant les deux points « : », sinon ça ne fonctionnera pas !

Index.php

Pour l'instant on va se contenter du minimum syndical, nous allons simplement insérer la structure HTML et un petit message type "Coucou !".

Ouvrez votre fichier index.php et entrez le code ci-dessous:

```
1  <!DOCTYPE html>
2  <html>
3  <head></head>
4  <body>
5      <h1>Coucou !</h1>
6  </body>
7  </html>
```

Functions.php

Ce fichier est essentiel puisque c'est ici que l'on va activer toutes les fonctionnalités nécessaires mais également ajouter nos propres fonctions.

Pour le moment, on va simplement ajouter 2 chose à notre fichier functions.php

```php
<?php

// Ajouter la prise en charge des images mises en avant
add_theme_support( 'post-thumbnails' );

// Ajouter automatiquement le titre du site dans l'en-tête du site
add_theme_support( 'title-tag' );
```

Note

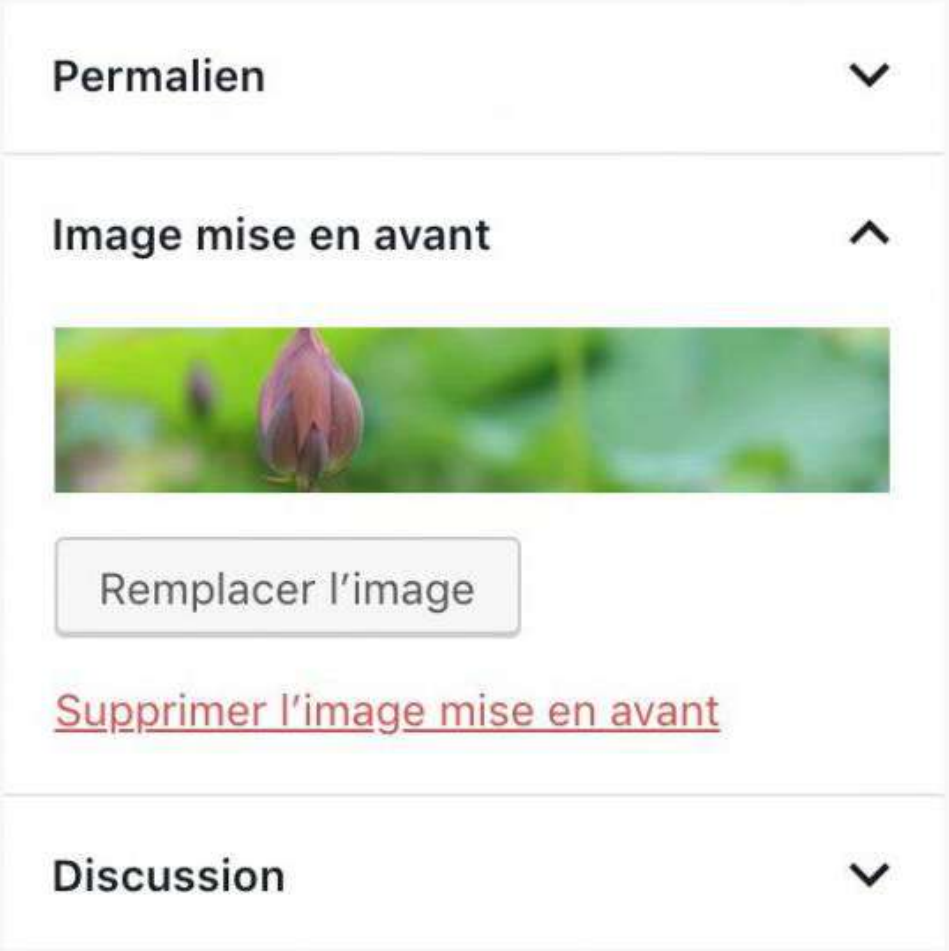
Dans les fichiers qui contiennent exclusivement du PHP, comme functions.php, on ne **referme pas** la balise PHP à la fin du fichier.

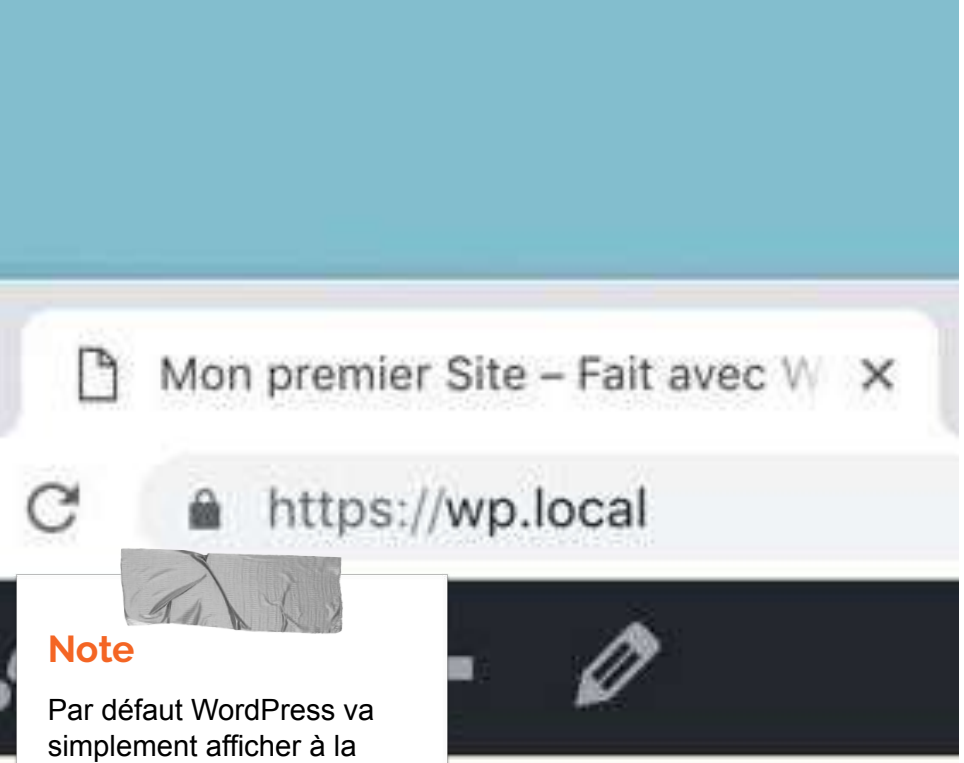
Les images misent en avant

La première chose que l'on demande, c'est d'activer la prise en charge des **images mise en avant** pour le blog, les visuels qui accompagnent l'article et son résumé, anciennement appelés image à la une et en anglais les **post thumbnails**.

Cela aura pour rôle d'ajouter la metabox ci-contre (à droite) dans votre éditeur de contenu

Vous pourrez alors choisir une image mise en avant pour chacun de vos articles





Note

Par défaut WordPress va simplement afficher à la suite les valeurs **Titre du site** et **Slogan** définies dans **Réglages > Général**.

Les images misent en avant

On lui demande également de créer automatiquement la balise <title> dans l'en-tête.

C'est le titre qui apparaît dans l'onglet du navigateur et qui est important pour le référencement naturel.

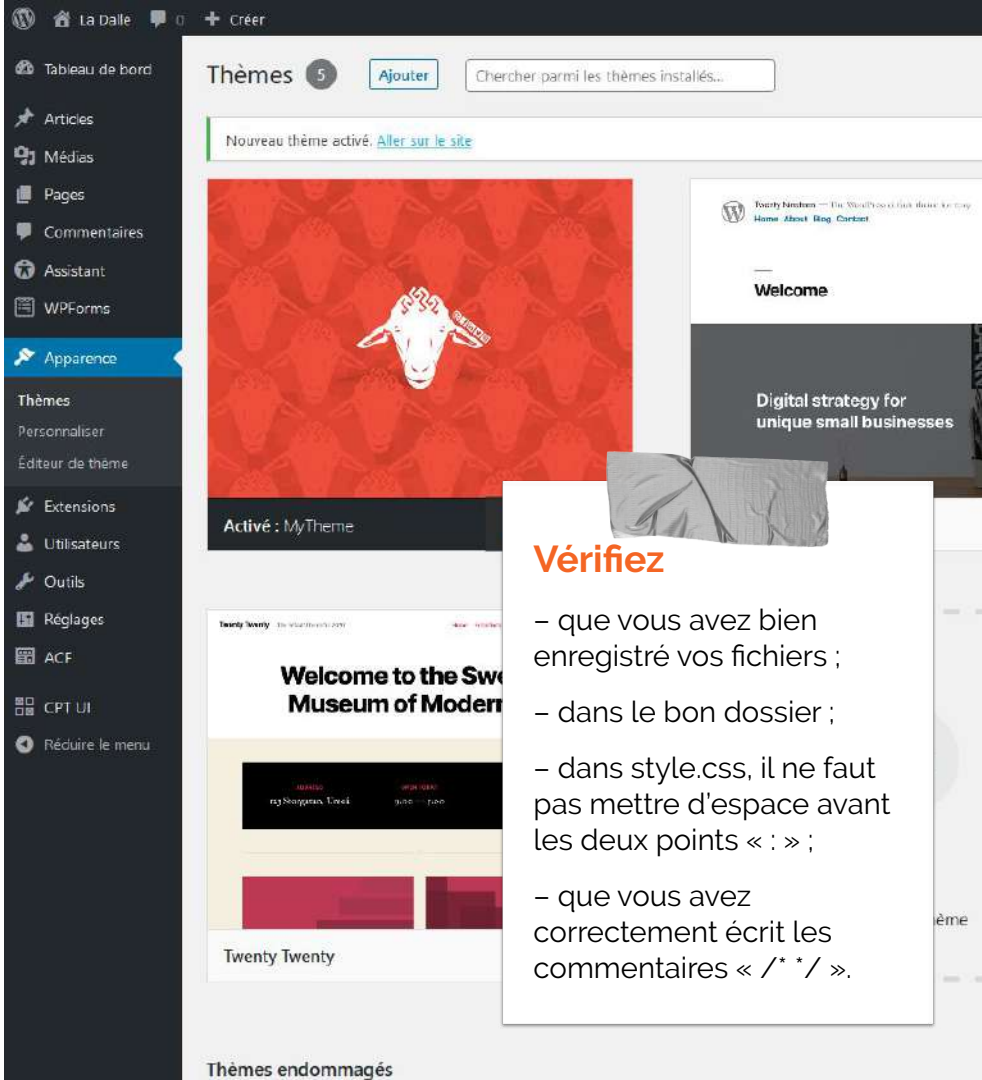
En laissant la gestion du titre à WordPress, il pourra être dynamique et différent pour chaque page.

Une extension SEO comme Yoast pourra du coup venir écrire des titres optimisés pour le référencement.

Activer son thème

Pour cela rendez-vous dans la rubrique **Apparence / Thèmes** de votre interface d'administration, survolez votre thème “MyTheme” et cliquez sur “Activer”

Si tout est bon, rendez-vous sur le site afin de voir le résultat.



Vérifiez

– que vous avez bien enregistré vos fichiers ;

– dans le bon dossier ;

– dans style.css, il ne faut pas mettre d’espace avant les deux points « : » ;

– que vous avez correctement écrit les commentaires « /* */ ».

https://wp.local

https://wp.local

Coucou !

Conseil

Afin d'éviter des aller-retours incessants entre l'admin WordPress et le site, je vous propose d'ouvrir en permanence 2 onglets dans votre navigateur pour optimiser votre travail.

3. Header/Footer

L'en-tête et le pied de page d'un site sont les éléments qui ne changent en général jamais d'une page à une autre. Afin d'éviter toute répétition de code, on va les envoyer dans leur propre fichier.

- **Séparer Header & Footer**
  Isoler le haut et le bas du site dans des fichiers à part.
- **Nouvelles fonctions WordPress**

  1. Header
  2. Footer
- **Ajouter le logo du site**

Séparer Header & Footer

On va donc isoler le haut et le bas du site dans des fichiers à part afin d'éviter toute répétition de code dans la suite de la formation.

Je vous invite donc à créer 2 nouveaux fichiers à la racine de votre thème :

- **header.php** : où l'on mettra la base du HTML et le haut du site ;
- **footer.php** : où l'on mettra le bas du site et les balises fermantes de notre page.

Dans **header.php** on vient placer ce code :

```
<!DOCTYPE html>
<html <?php language_attributes(); ?>>
<head>
    <meta charset="<?php bloginfo('charset'); ?>">
    <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no"/>

    <?php wp_head(); ?>
</head>

<body <?php body_class(); ?>>

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

Séparer Header & Footer

Dans **footer.php** on vient placer ce code :

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

C'est le strict minimum nécessaire pour le moment et on pourra agrémenter à terme en fonction des besoins.

Du coup, on va pouvoir modifier **index.php** et appeler nos nouveaux fichiers grâce à 2 fonctions fournies nativement par WordPress :

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

<h1>Coucou</h1>

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

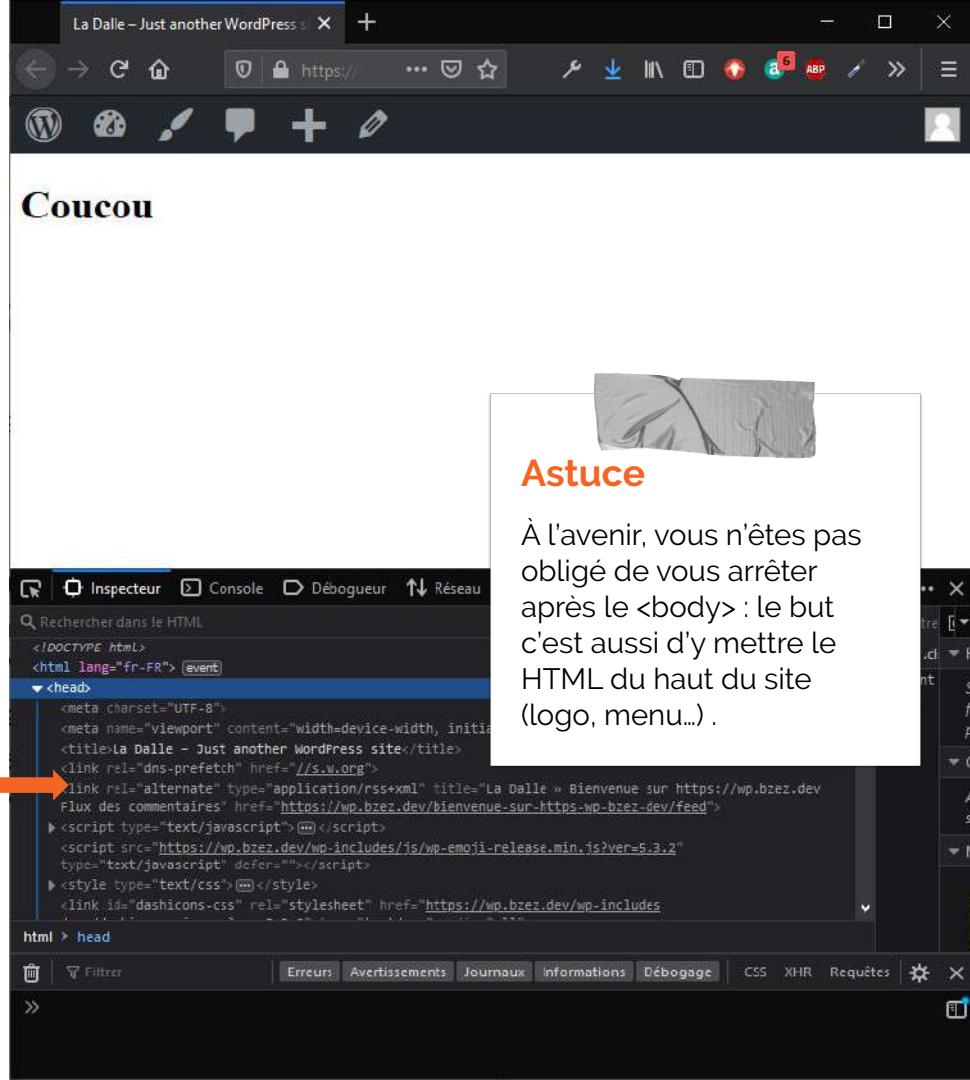
Testez, vérifiez le code source de la page afin de vérifier si vos en-têtes et pied de page sont bien correctement appelés.

Apercu

En haut, la barre d'administration fait son apparition. Dans le code, plein de nouveau HTML

Si la manipulation a bien fonctionné, vous devriez voir apparaître la barre d'administration WordPress (en noir en haut du site) et le titre dans l'onglet du navigateur.

Concernant le code de la page via l'inspecteur, on remarque qu'il y a bien plus de HTML que ce qu'on en a écrit ! C'est dû aux fonctions que l'on vient d'ajouter et que l'on va analyser juste après.



Nouvelles fonctions WordPress

HEADER

```php
<!DOCTYPE html>
<html <?php language_attributes(); ?>>
<head>
    <meta charset="<?php bloginfo('charset'); ?>">
    <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no"/>
```

language_attributes() :

- Défini automatiquement la langue du document.
- Basée sur **Réglages > Général > Langue du site**.

bloginfo('charset') :

- Défini l'encodage du site. Par défaut c'est **UTF-8**
- Cette fonction nous permettra de récupérer d'autres informations utiles pour notre site..

Infos

La fonction bloginfo est une très vieille fonction de WordPress, à l'époque où ce dernier était encore considéré comme une plateforme de blogging.

Nouvelles fonctions WordPress

HEADER

```
7       <?php wp_head(); ?>
```

wp_head() :

- Déclare le chargement des scripts et des styles.
  D'ailleurs c'est via cette fonction que WordPress va venir écrire la balise <title> que l'on a activé dans le **functions.php** dans le cours précédent.

```
10  <body <?php body_class(); ?>>
```

body_class() :

- Permet d'obtenir des noms de classe CSS en fonction de la page visitée.

```
12      <?php wp_body_open(); ?>
```

wp_body_open() :

- Permet à des extensions d'écrire du code au début du body

Warning

Cette fonction est essentielle au bon fonctionnement de votre thème alors ne l'oubliez pas

Nouvelles fonctions WordPress

FOOTER

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

wp_footer() :

- Il a exactement **le même rôle** que **wp_head()** : afficher des scripts et styles, mais cette fois en bas de page.

Astuce

En terme de performances, il est plus intéressant de charger en priorité les fichiers CSS (donc en haut de page) et reléguer les scripts en bas de page.

Le JS n'est en réalité exécuté seulement lorsque la page est « prête » pour le navigateur, donc inutile de les charger trop tôt.

Ajouter le logo

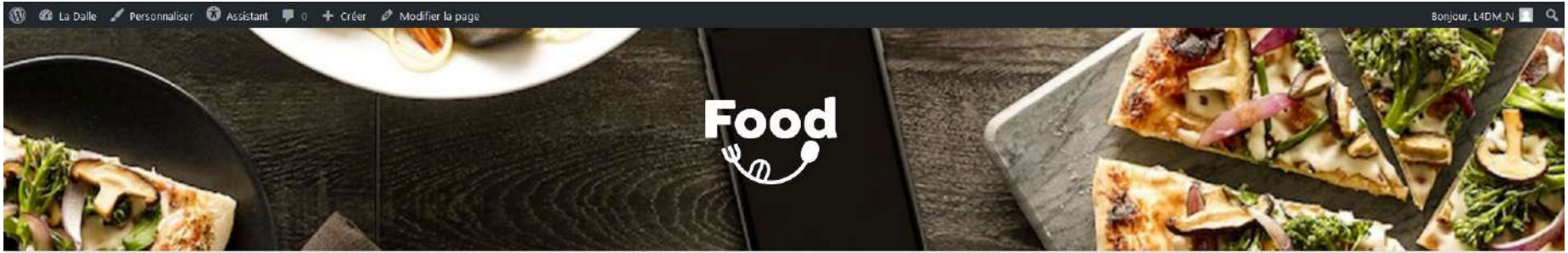
On va profiter du fichier **header.php** pour ajouter la partie haute du site, qui ne bougera jamais
Ajoutez le code suivant à votre fichier header.php

```php
<body <?php body_class(); ?>>
    <header class="header">
        <a href="<?php echo home_url( '/' ); ?>">
            <img src="<?php echo get_template_directory_uri(); ?>/img/logo.png" alt="Logo">
        </a>
    </header>
```

home_url() : Retourne l'url de la page d'accueil
get_template_directory_uri() : Permet d'obtenir l'adresse absolue (c'est-à-dire complète) du logo.
Sans ça, votre image ne s'affichera pas !

Par défaut, le fichier style.css n'est pas chargé par WordPress, si vous voulez utilisez le fichier style.css pour embellir votre header vous devrez ajouter ce code à votre fichier functions.php.

```php
// On crée la fonction qui ajoutera automatiquement le fichier "style.css" dans le <head>
function mytheme_add_style() {
    //Fonction par défaut WordPress wp_enqueue_style();
    wp_enqueue_style( 'mytheme-style', get_stylesheet_uri() );
}
add_action( 'wp_enqueue_scripts', 'mytheme_add_style' );
```



Essayez d’obtenir quelque chose comme ça :)
PS: N’oubliez, bootstrap et votre ami !

4. Templates

Le Template Hierarchy permet à WordPress de définir quel modèle de page afficher (template) en fonction de la page demandée.

- **Typologie des pages**
  Tour d'horizon
- **Le Templates Hierarchy**
  1. Les pages
  2. La page d'accueil
  3. Les archives
  4. Les singles
  5. Autres
- **Simplifications**
  Une version simplifié du TH.
- **Créer vos fichiers**

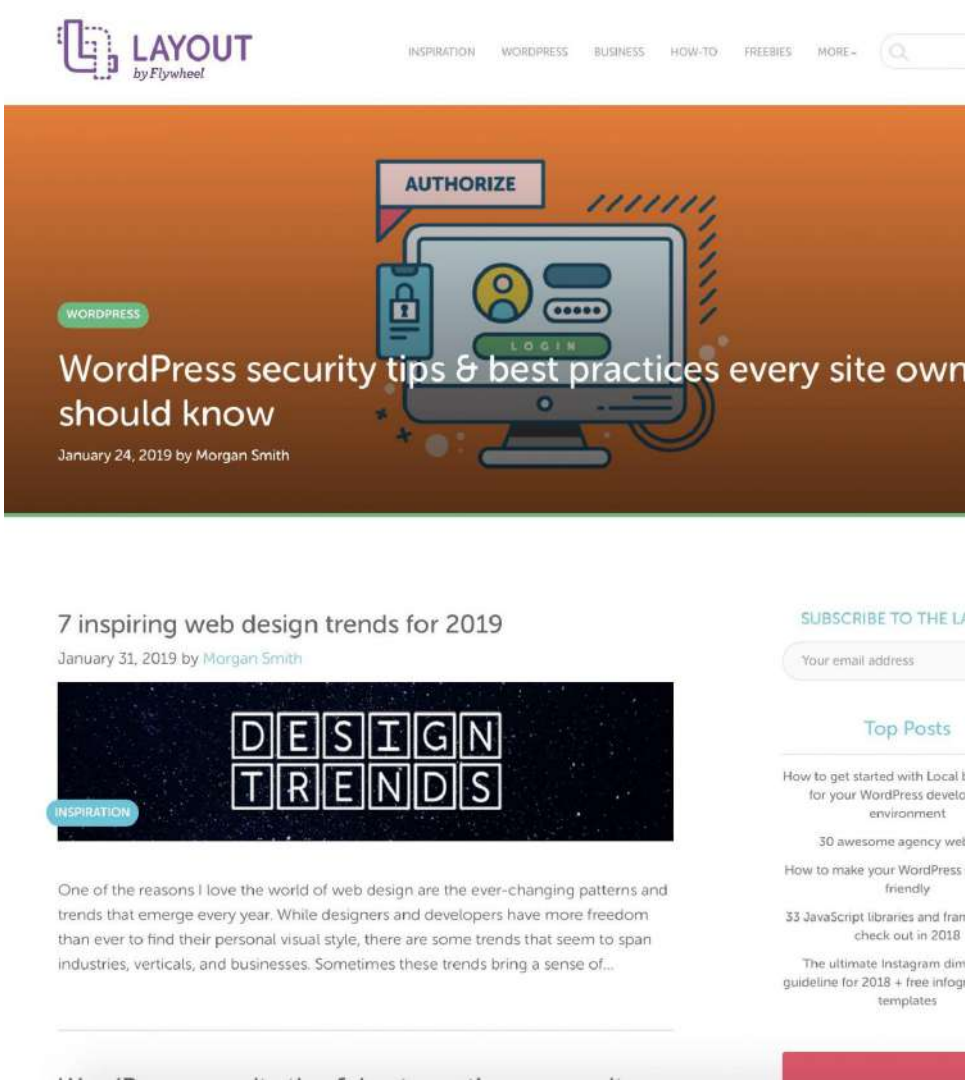
Il y a 3 grands types de page dans un site WordPress.

La page d'accueil du site, du blog

DANS WORDPRESS ON APPEL CA DES PAGES

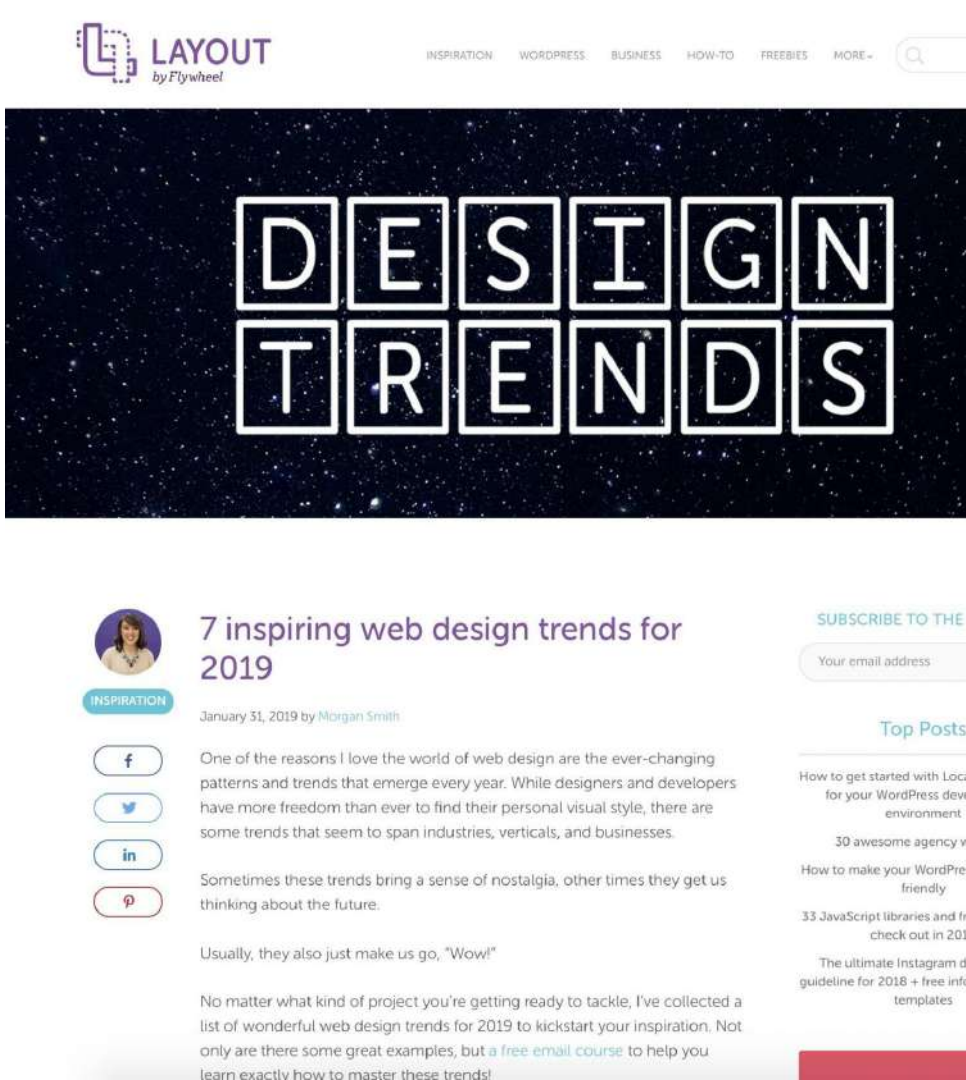
Les pages qui liste des articles

DANS WORDPRESS ON APPEL CA DES ARCHIVES



La page d'un article quand celui-ci est affiché en entier

DANS WORDPRESS ON APPEL CA DES SINGLES



A retenir

Les pages standard sont des pages

Ex: Page d'accueil

La liste des articles d'un blog sont des archives

Ex: Listing d'articles par catégorie

L'affichage d'un article est un single

Certains cas sont spécifiques tels que les pages d'accueil et accueil du blog

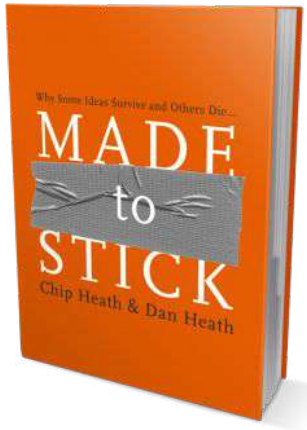
Le Templates Hierarchy

Définition:

Le Template Hierarchy est un processus du coeur de WordPress qui lui permet de **déterminer quel modèle de page utiliser en fonction de la page sélectionnée**.

Ce processus est lancé pour chaque page demandée par WordPress. **On parle de hiérarchie car WordPress va procéder par ordre pour définir le bon fichier**, en regardant d'abord dans les cas spécifiques, et en terminant sur des modèles plus génériques.

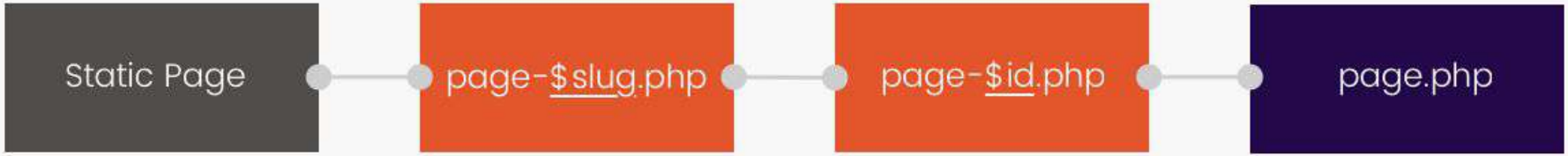
Le développeur de thèmes peut donc aller **créer des modèles de page pour chaque situation**, en suivant une nomenclature spécifique définie par le schéma du Template hierarchy.



Rappel

9 fois sur 10, on n'utilisera pas les cas ultra spécifiques du Template Hierarchy. On reste quasiment tout le temps dans les cas généraux.

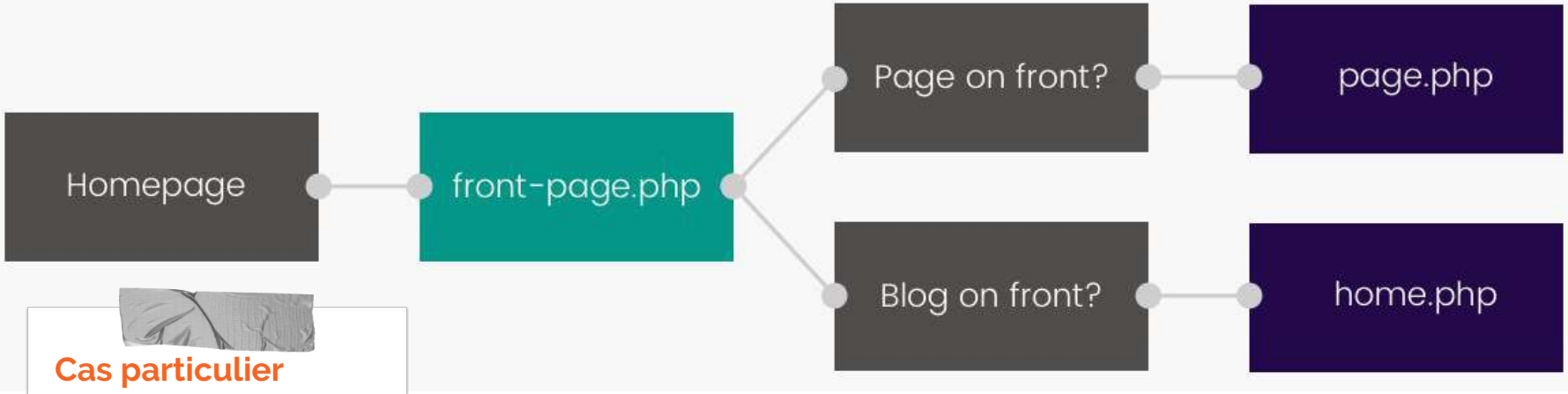
Les pages



Astuce

Je vous déconseille d'utiliser cette technique (slug ou ID) car ils peuvent changer : une personne peut supprimer la page et la recréer (l'ID change) ou changer son permalien.

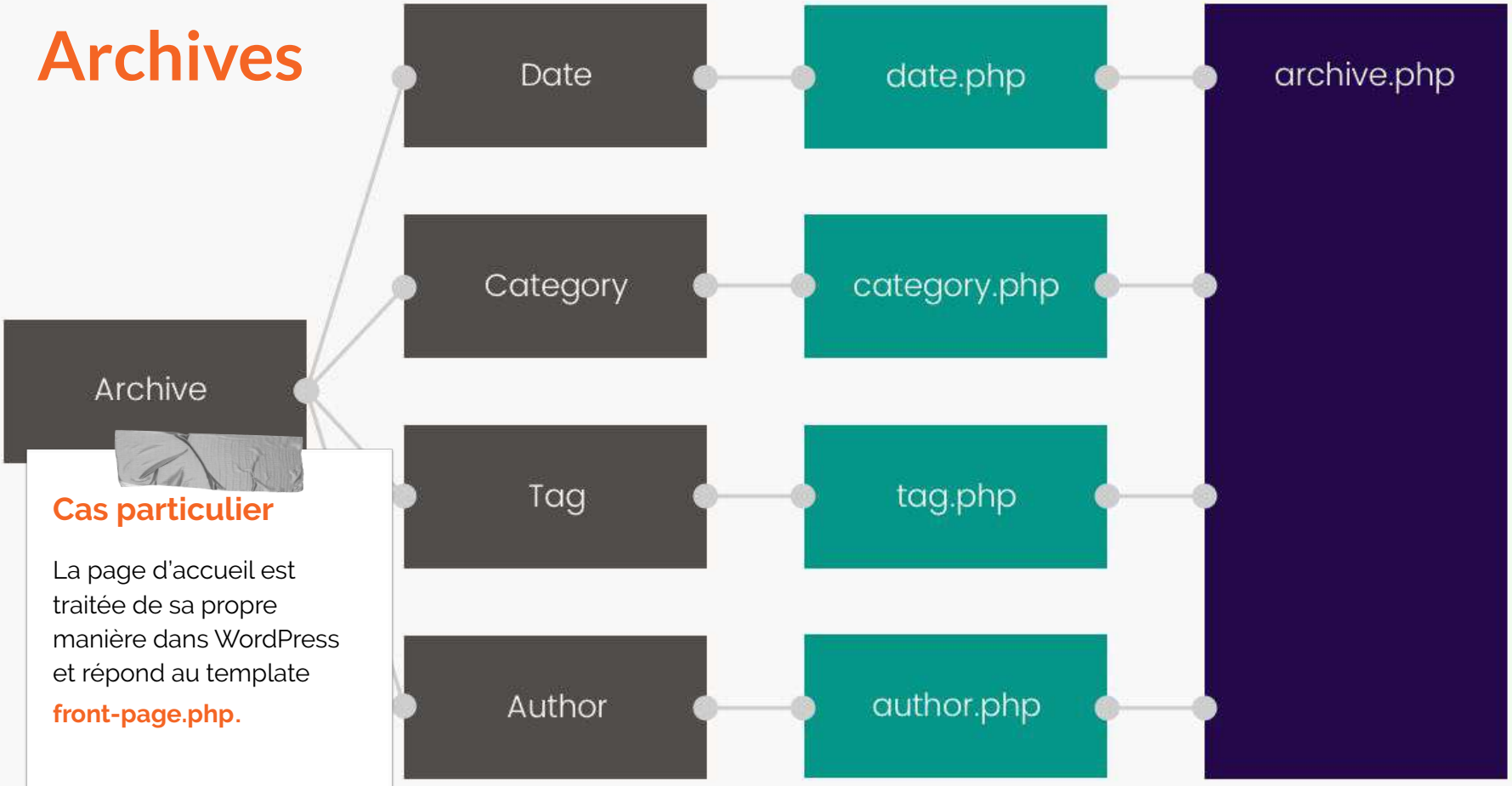
La page d’accueil



Cas particulier

La page d’accueil est traitée de sa propre manière dans WordPress et répond au template **front-page.php.**

Archives



Cas particulier

La page d'accueil est traitée de sa propre manière dans WordPress et répond au template **front-page.php.**

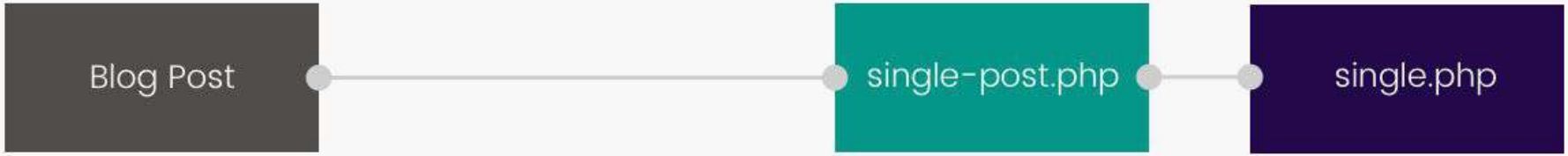
La page d’accueil

Attention

La **page d’accueil du blog**, même si c’est une archive, répond au template **home.php** même si celui-ci n’est pas défini comme page d’accueil du site.

Par exemple, vous pourriez avoir une catégorie **Promotions** qui aurait une mise en page complètement différente du reste du blog. Mais vous en conviendrez, ce genre de cas est très rare.

Les singles



Attention

La **page d'accueil du blog**, même si c'est une archive, répond au template **home.php** même si celui-ci n'est pas défini comme page d'accueil du site.

Simplifications

Si j'enlève tous les cas trop spécifiques, j'obtiens ceci.

C'est déjà plus simple.

Et pour encore plus de simplicité, contentez-vous des templates primaires **(en violet)**

Créez vos fichiers

- archive.php ;
- front-page.php ;
- home.php ;
- page.php ;
- single.php.

Place respectivement ceci dans chaque fichiers *(répétez pour chaque fichier)*

single.php

```php
<?php get_header(); ?>
    <h1>SINGLE</h1>
<?php get_footer(); ?>
```

archive.php

```php
<?php get_header(); ?>
    <h1>ARCHIVE</h1>
<?php get_footer(); ?>
```

5. “LA” boucle

Lorsque l’on développe un thème WordPress, impossible de passer à côté de la boucle WordPress. Voyons à quoi ce concept correspond.

- ➔ **La boucle WordPress c’est quoi**
  Définition & mise en pratique
- ➔ **Les Templates Tags**
  Les incontournables
- ➔ **Cas pratique: Le Blog**
  1. La liste des derniers articles
  2. Afficher un article (single)
- ➔ **Les Template Tags fréquents**
  1. Les équivalents en “get”

La boucle WordPress c'est quoi ?

Définition:

La boucle WordPress est le mécanisme matérialisé par un **petit bout de code PHP** qui permet **d'afficher les données entrées via l'interface** d'administration.

Elle permet de **préparer les données** (titre, contenu, catégories, lien, etc…) et de les appeler **via des fonctions dédiées** au nom explicite, comme the_content().

Cette boucle a la particularité **d'être exactement la même sur tous les templates** de page que l'on va créer.
Il suffira alors de la copier/coller dans tous nos modèles.

La boucle WordPress c'est quoi ?

Maintenant que l'on a vu le **template hierarchy**, on va commencer par aller modifier le fichier **front-page.php** qui s'affichera lorsqu'on demande la **page d'accueil** :

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

    <?php if( have_posts() ) : while( have_posts() ) : the_post(); ?>

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

        <?php the_content(); ?>

    <?php endwhile; endif; ?>

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

Copiez exactement le même code pour **page.php**.
À l'avenir notre page d'accueil sera différente des pages « standards » mais pour le moment ça ira.


La Dalle
Personnaliser
Assistant
0
Créer
Modifier la page
Bonjour, L4DM_N
graphicnaivety
Bienvenue sur https://wp.bzez.dev
Voici votre page d'accueil. C'est différent d'un article de blog parce qu'elle restera au même endroit et apparaîtra dans la navigation de votre site (dans la plupart des thèmes).
En tant qu'utilisateur ou utilisatrice connecté·e à WordPress, vous pouvez modifier cette page en cliquant sur le bouton Modifier la page dans la barre d'outils. Cette page peut servir à vous présenter ou à présenter votre entreprise aux visiteurs du site et ressembler à quelque chose comme cela :
Bonjour ! Je suis un mécanicien qui aspire à devenir acteur, et voici mon site. J'habite à Bordeaux, j'ai un super chien baptisé Russell, et j'aime la vodka-ananas (ainsi qu'être surpris par la pluie soudaine lors de longues balades sur la plage au coucher du soleil).
... ou quelque chose comme cela :
La société 123 Machin Truc a été créée en 1971, et n'a cessé de proposer au public des machins-trucs de qualité depuis lors. Située à Saint-Remy-en-Bouzemont-Saint-Genest-et-Isson, 123 Machin Truc emploie 2 000 personnes, et fabrique toutes sortes de bidules super pour la communauté bouzemontoise.
Si vous préférez afficher vos derniers articles sur la page d'accueil, rendez-vous sur la page de Réglages et choisissez l'option «Les derniers articles» pour la page d'accueil.
Amusez-vous bien !

Les Templates Tags

Définition:

Les Templates Tags sont des **fonctions incluses dans WordPress** permettant **d'afficher dans votre thème les contenus** que vous avez administré dans l'interface d'édition.
Ces fonctions sont **à destination des développeurs** de thèmes afin de rendre leurs pages dynamiques.
Il en **existe des dizaines**, certaines permettant d'afficher le contenu de l'article comme **the_title()** et **the_content()**, d'autres sont là pour afficher les commentaires, l'auteur… ou même encore certaines qui permettent de récupérer des informations générales du site comme **bloginfo()**.

Attention

Les template tags doivent impérativement être utilisés **à l'intérieur de la boucle** WordPress afin de fonctionner de manière optimale.

Les Templates Tags

Dans le code vu plus haut, on en a utilisé deux :

```php
<?php if( have_posts() ) : while( have_posts() ) : the_post(); ?>
    <h1><?php the_title(); ?></h1>
    <?php the_content(); ?>
<?php endwhile; endif; ?>
```

the_title() permet d'afficher le titre
the_content() va afficher le contenu de la page, c'est-à-dire ce que l'on a écrit avec l'éditeur visuel.

Ce sont les deux template tags les plus utilisés !

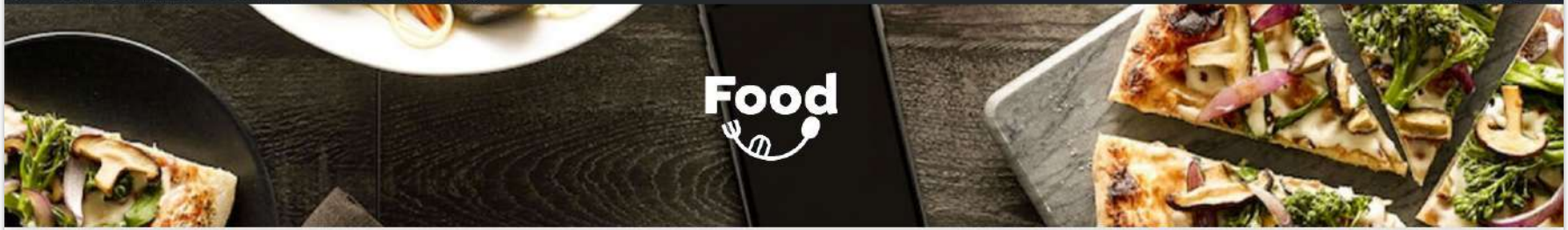
Il en existe d'autres que l'on va voir avec les pages du blog, mais pour les pages, on n'a pas besoin de grand chose de plus.

Cas pratique: Le Blog

La liste des derniers articles (archive.php et home.php)

On commence avec la page d'archives, qui liste par défaut les 10 derniers articles publiés sur le blog :

```php
<?php get_header(); ?>
    <h1>Le blog La Dalle</h1>

    <?php if( have_posts() ) : while( have_posts() ) : the_post(); ?>

        <article class="post">
            <h2><?php the_title(); ?></h2>

            <?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>

            <?php the_excerpt(); ?>

            <p>
                <a href="<?php the_permalink(); ?>" class="post__link">Lire la suite</a>
            </p>
        </article>

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



Le Blog

Fallout 76

Publié le janvier 6, 2020 par L4DM_N • Pas de commentaire

Lire la suite

Hello world!

Publié le janvier 3, 2020 par L4DM_N • Un commentaire

Welcome to WordPress. This is your first post. Edit or delete it, then start writing!

Lire la suite

Qui ducimus est nobis voluptas

Cas pratique: Le Blog

Voyons maintenant les nouveaux templates tags ajoutés :

```
9           <?php the_post_thumbnail(); ?>
10
```

the_post_thumbnail() : Permet d'afficher l'image mise en avant.
Retourne la balise <img>
Pour rappel on avait activé cette fonctionnalité grâce à la **fonction add_theme_support('post-thumbnails')** dans le fichier **functions.php**

```
12              Publié le <?php the_time( get_option( 'date_format' ) ); ?>
```

Pour afficher la date, on dispose de 2 fonctions : **the_date()**, et **the_time()**.
Dans cette dernière on doit indiquer le format d'affichage via les paramètres de Date PHP classiques.
D'ailleurs je récupère, grâce à la fonction **get_option()**, la valeur indiquée dans **Réglages > général > Format de date**.

```
13              par <?php the_author(); ?> • <?php comments_number(); ?>
```

La fonction **the_author()** permet d'afficher le nom de l'auteur, tel qu'il l'a défini dans **Utilisateurs > Votre profil > Nom à afficher publiquement**. Par mesure de sécurité, évitez d'afficher votre identifiant de connexion.
comments_number() qui affiche le nombre de commentaires publiés dans cet article.
Par défaut il va afficher : *Pas de commentaire*, *1 commentaire* ou *X commentaires*.

Cas pratique: Le Blog

Voyons maintenant les nouveaux templates tags ajoutés :

```
16          <?php the_excerpt(); ?>
```

Cette fois on utilise la fonction **the_excerpt()** à la place de **the_content()** afin de n'afficher qu'un extrait de l'article, et pas l'article en entier.

```
19              <a href="<?php the_permalink(); ?>" class="post__link">Lire la suite</a>
```

La fonction **the_permalink()** permet d'afficher le lien vers l'article.
Vous pouvez configurer la structure des permaliens dans **Réglages > Permaliens**.

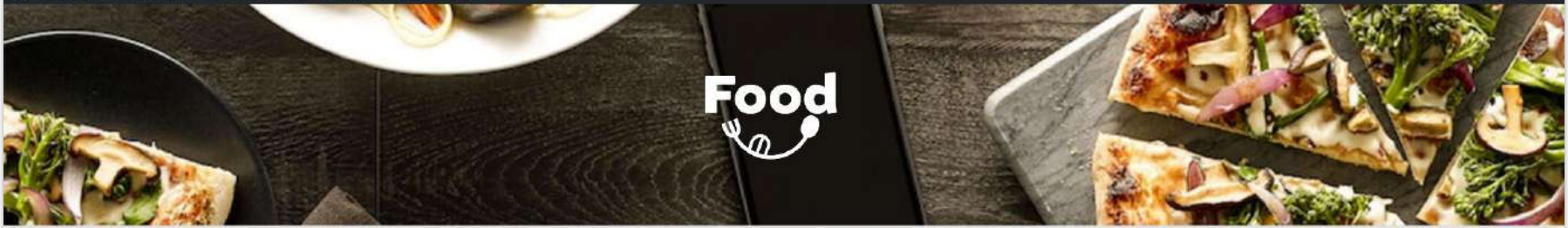
CONSEIL
Pour de belles URLs et un meilleur référencement optez pour la configuration suivante:
« Titre de publication » /%postname%/ qui est normalement le réglage par défaut.

Cas pratique: Le Blog

Afficher un article (single)

Voici le code PHP que j'ai choisi de mettre dans **single.php** :

```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(); ?>
```



Mon premier article WordPress

Sortie le

Publié le janvier 2, 2020 par L4DM_N Dans la catégorie

- Uncategorized

Avec les étiquettes

Mon premier contenu d'article.

Cas pratique: Le Blog

Voyons maintenant les nouveaux templates tags ajoutés :

```
10        <?php echo get_avatar( get_the_author_meta( 'ID' ), 40 ); ?>
```

la fonction **get_avatar()** permet de récupérer l'avatar de l'auteur, en se servant du service Gravatar.

```
14        Dans la catégorie <?php the_category(); ?>
15        Avec les étiquettes <?php the_tags(); ?>
```

Et enfin on utilise **the_category()** et **the_tags()** pour afficher la catégorie et les éventuelles étiquettes (anciennement mots-clés).
Cela aura pour effet d'afficher des listes **
**

Les Templates Tags fréquents

Pour résumer, voici la liste des templates tags que l'on va utiliser le plus régulièrement :

- **the_title()** : affiche le titre de l'article ou la page ;
- **the_content()** : affiche le contenu, écrit depuis l'éditeur visuel (Gutenberg ou TinyMCE) ;
- **the_post_thumbnail()** : affiche l'éventuelle image mise en avant ;
- **the_excerpt()** : affiche un extrait de l'article, soit le contenu du champ extrait, soit le début de l'article jusqu'à la balise Lire la suite ;
- **the_category()** : affiche une liste de la ou les catégories sélectionnées ;
- **the_tags()** : affiche une liste des éventuelles étiquettes de l'article ;
- **the_author()** : affiche le nom de l'auteur ;
- **the_author_link()** : affiche le nom de l'auteur avec un lien vers son site personnel ;
- **the_date()** : affiche une fois les dates où des articles ont été publiés ;
- **the_time()** : affiche la date de publication de chaque article ;
- **the_permalink()** : affiche l'URL vers l'article, à mettre dans une balise <a> ;
- **comment_number()** : affiche le nombre de commentaires publiés dans l'article ;
- **get_avatar()** : Récupère l'avatar d'un utilisateur depuis le service Gravatar ;

Les équivalents “get”

Parfois, vous aurez besoin de récupérer la valeur sans directement l’afficher dans votre template, afin de la traiter en PHP.
Si vous souhaitez récupérer la valeur du titre sans l’afficher par exemple, the_title() ne fonctionnera pas, il faudra alors utiliser son équivalent get_the_title() :

```php
<?php
    $title = get_the_title();

    // Si le titre contient le mot "promo", j'ajoute un emoji
    if( strpos( $title, 'Promo' ) ) {
        $title = '💰' . $title;
    }
?>
    <h1><?php echo $title; ?></h1>
```

Retenez donc que :

- Les fonctions en the_ **affichent directement la donnée** dans le template ;
- Les fonctions en get_ **la récupère sans l’afficher**, en vue d’un traitement.

Comme vous pouvez vous en douter, il existe un équivalent « get » à chacun des template tags.

6. Les templates parts

Dans le but d'éviter des répétitions de code, il est possible d'appeler des sous-templates grâce à la fonction **get_template_part()** que l'on va voir dans ce cours.

- ➔ **Appeler un sous-template**
  Cas: Newsletter
- ➔ **get_template_part vs include**
  Explications
- ➔ **Scinder le templates hierarchy**
  1. Le blog et les archives
  2. La recherche et les archives

Appeler un sous-template

Imaginons que vous ayez créé un formulaire d'abonnement à votre newsletter à la main, et que vous souhaitez l'intégrer dans votre thème à différents endroits.
Pour cela vous créez un fichier **newsletter.php**

```
<form class="newsletter-form">
    <p>Des offres exclusives, des actus WordPress et du contenu bonus en avant-première</p>
    <input type="email" name="email" placeholder="E-mail">
    <input type="submit" value="Je m'abonne">
</form>
```

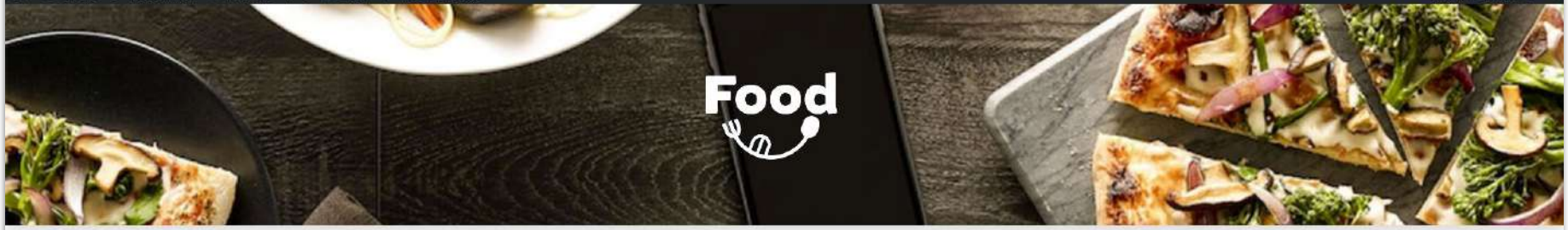
Puis dans **footer.php** ajouter:

```
    <footer>
        <?php get_template_part( 'newsletter' ); ?>
    </footer>
```

Vous remarquerez que l'**on ne met pas l'extension de fichier à la fin**.
Donc simplement **newsletter**, et pas **newsletter.php**.
Dans le but de garder votre dossier de thème organisé, vous pouvez stocker ces templates dans un sous-dossier, par exemple **parts/newsletter.php**.

Conseil

Vous pouvez très bien utiliser cette fonction pour alléger un template un peu trop lourd, en séparant certaines parties dans des sous-templates.



Bienvenue sur https://wp.bzez.dev

Voici votre page d'accueil. C'est différent d'un article de blog parce qu'elle restera au même endroit et apparaîtra dans la navigation de votre site (dans la plupart des thèmes).

En tant qu'utilisateur ou utilisatrice connecté·e à WordPress, vous pouvez modifier cette page en cliquant sur le bouton **Modifier la page** dans la barre d'outils. Cette page peut servir à vous présenter ou à présenter votre entreprise aux visiteurs du site et ressembler à quelque chose comme cela :

Bonjour ! Je suis un mécanicien qui aspire à devenir acteur, et voici mon site. J'habite à Bordeaux, j'ai un super chien baptisé Russell, et j'aime la vodka-ananas (ainsi qu'être surpris par la pluie soudaine lors de longues balades sur la plage au coucher du soleil).

… ou quelque chose comme cela :

La société 123 Machin Truc a été créée en 1971, et n'a cessé de proposer au public des machins-trucs de qualité depuis lors. Située à Saint-Remy-en-Bouzemont-Saint-Genest-et-Isson, 123 Machin Truc emploie 2 000 personnes, et fabrique toutes sortes de bidules super pour la communauté bouzemontoise.

Si vous préférez afficher vos derniers articles sur la page d'accueil, rendez-vous sur la page de Réglages et choisissez l'option «Les derniers articles» pour la page d'accueil.

Si vous préférez afficher vos derniers articles sur la page d'accueil, rendez-vous sur la page de Réglages et choisissez l'option «Les derniers articles» pour la page d'accueil.

Amusez-vous bien !

Notre newsletter

Les équivalents “get”

Parfois, vous aurez besoin de récupérer la valeur sans directement l’afficher dans votre template, afin de la traiter en PHP.
Si vous souhaitez récupérer la valeur du titre sans l’afficher par exemple, the_title() ne fonctionnera pas, il faudra alors utiliser son équivalent get_the_title() :

```php
<?php
    $title = get_the_title();

    // Si le titre contient le mot "promo", j'ajoute un emoji
    if( strpos( $title, 'Promo' ) ) {
        $title = '💰' . $title;
    }
?>
    <h1><?php echo $title; ?></h1>
```

Retenez donc que :

- Les fonctions en the_ **affichent directement la donnée** dans le template ;
- Les fonctions en get_ **la récupère sans l’afficher**, en vue d’un traitement.

Comme vous pouvez vous en douter, il existe un équivalent « get » à chacun des template tags.

get_template_part() vs include()

Au fait, pourquoi ne pas utiliser simplement les fonctions **include()** ou **require()** en PHP directement ?

En fait **get_template_part()** est adapté pour WordPress et ses différents cas de figure : `

- En premier lieu, cette **fonction ne renvoie pas d'erreur** si elle n'a pas trouvé le template correspondant (il se peut dans certains cas qu'on cherche à trouver un template dynamiquement, et il arrive qu'il n'existe pas) ;
- De plus, la fonction sait récupérer le bon template dans le bon dossier dans le cas où vous avez utilisé un **thème enfant.**

Donc préférez l'utilisation de **get_template_part()** dans toutes les situations.

Scinder le Templates Hierarchy

Pour terminer, je vais vous donner une astuce que j'utilise pour améliorer le **Template Hierarchy** car il y a quelques cheminements qui ne me plaisent pas.

1. **Le Blog et les archive**
   Comme on l'a vu précédemment, la page d'accueil du blog est toujours **home.php**, alors que les autres archives sont représentées par **archive.php**.
   Ce fonctionnement aurait un sens si votre blog avait une mise en page différente de ses archives, mais c'est rarement le cas.

2. **La recherche et les archives**
   Pour la recherche, c'est pareil : bien souvent les résultats sont affichés de la même manière que le blog.
   Donc si c'est votre cas, on va également mettre le code suivant mais cette fois dans **search.php**

Je vous propose donc de mettre ce code dans vos fichiers **home.php** et **search.php**

```
1  <?php get_template_part( 'archive' ); ?>
```

Home (Page d'accueil du blog)
Search (Page des résultats)

Redirigent désormais directement vers

Archive (Listing des articles)

7. Les conditional tags

Les Conditional Tags sont des fonctions fournies qui permettent d'éviter des répétitions de code

➔ **Définition**

➔ **Utilisation**
Comment les utiliser

➔ **Les tags utiles**

1. is_single()
2. is_page()
3. is_archive()
4. is_front_page()
5. is_home()
6. is_user_logged_in()
7. has_post_thumbnail()

Les Conditional Tags

Définition:

Les Conditional Tags sont des fonctions PHP fournies par WordPress à utiliser dans le template ou vos fonctions et qui permet de savoir dans quelle situation se trouve la page actuelle.
Par exemple, ils permettent de savoir si l'utilisateur est connecté, si la page actuelle est l'accueil, une archive, un single, une page…
Elles renvoient toujours **vrai ou faux (true/false)**.
Leur rôle est de permettre, au sein d'un même template, de distinguer des cas de figure différents et agir en conséquent.

Attention

Les template tags doivent impérativement être utilisés **à l'intérieur de la boucle** WordPress afin de fonctionner de manière optimale.

Utilisation

Les fonctions sont utilisées en général dans une condition et renvoie une valeur **booléenne** true/false.

Par exemple, il existe la fonction **is_user_logged_in()** qui permet de savoir si l'internaute est connecté.

```php
<?php
if ( is_user_logged_in() ):
    get_currentuserinfo();
    echo "Bonjour, " . $current_user->user_firstname;
else:
    echo "Bonjour, visiteur !";
endif;
```

Exemple avec la page archive.php, ajouter ce code a votre fichier.

```php
<?php get_header(); ?>
    <?php
    if ( is_category() ) {
        $title = "Catégorie : " . single_tag_title( '', false );
    }
    elseif ( is_tag() ) {
        $title = "Étiquette : " . single_tag_title( '', false );
    }
    elseif ( is_search() ) {
        $title = "Vous avez recherché : " . get_search_query();
    }
    else {
        $title = 'Le Blog';
    }
?>
<h1><?php echo $title; ?></h1>
```

Les Tags utiles

is_single():

```php
<?php
    if( is_single() ) { } // Teste si la page est de type single
    if( is_single( 12 ) ) { } // Teste si l'article porte l'ID 12
    if( is_single( 'mon-article' ) ) { } // Teste si le slug est 'mon-article'
```

is_page():

```php
<?php
    if( is_page() ) { } // Teste si la page est de type page
    if( is_page( 16 ) ) { } // Teste si la page porte l'ID 16
    if( is_page( 'contact' ) ) { } // Teste si le slug est 'contact'
```

is_archive(): Pour ce tag pas de paramètre. Il teste simplement si vous êtes dans une **archive** ou non. Attention cependant, si vous faites des Custom Post Types, il faudra utiliser **is_post_type_archive()** mais on va l'aborder plus tard dans la formation.

is_front_page(): Pour tester si vous êtes sur la page d'accueil du site.

Les Tags utiles

is_home(): A contrario, ce tag ne renvoie true seulement si vous êtes sur la page d'accueil des articles du blog.

Donc attention :

- Si le blog est en page d'accueil du site, **is_home()** et **is_front_page()** répondront tous deux true ;
- Si le blog est défini par exemple sur la page « blog », alors **is_home()** sera true seulement sur cette

is_user_logged_in(): Contrôle si l'utilisateur est connecté ou non.
Ajouter ce code dans votre fichier **header.php**

```php
<?php
if ( is_user_logged_in() ):
    get_currentuserinfo();
?>
    <p>
        <?php echo $current_user->user_firstname; ?>
        <a href="<?php echo wp_login_url(); ?>"> Déconnexion </a>
    </p>
<?php else: ?>
    <p>
        <a href="<?php echo wp_login_url(); ?>"> Connexion </a>
    </p>
<?php endif; ?>
```

Les Tags utiles

has_post_thumbnail():

Cette fonction permet de contrôler si l'article possède ou non une **image mise en avant**.
Par défaut on n'en n'a pas forcément besoin : la fonction **the_post_thumbnail()** ne génère la balise de l'image seulement si l'article en a une définie.

Mais ça peut être utile dans le cas où vous voudriez également ajouter un peu de markup HTML autour, comme une ** ** par exemple.

Ajoutez ce code à votre fichier **single.php**:

```
1  <?php if ( has_post_thumbnail() ): ?>
2      <div class="post__thumbnail">
3          <?php the_post_thumbnail(); ?>
4      </div>
5  <?php endif; ?>
```

8. Les templates perso

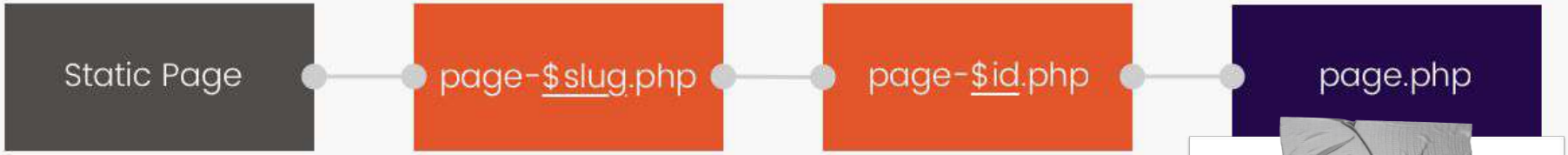
Dans ce cours nous allons aborder un moyen pour créer des templates personnalisés que le rédacteur pourra manuellement assigner à ses pages.
Cela permettra de créer des mises en page complètement différentes pour chaque page du site.

➔ **Déclarer un template perso**
Comment les utiliser

➔ **Autoriser le templates**

Rappel

Dans le cours sur le Template Hierarchy on a vu que l'on pouvait créer des templates spécialisés pour les pages en ciblant leur **ID** ou leur **Slug** :



Astuce

Je vous déconseille d'utiliser cette technique (slug ou ID) car ils peuvent changer : une personne peut supprimer la page et la recréer (l'ID change) ou changer son permalien.

Déclarer un Template perso

Dans un site moderne, les pages principales (accueil, services…) sont bien différentes, et plus complexes que des pages standard comme les mention légales ou la page contact.

C'est pour cela que l'on va créer des fichiers templates qui leur seront dédiées.

Pour cela il suffit de créer un nouveau fichier, par exemple **services.php** et d'y indiquer :

```php
<?php
/*
  Template Name: Services
*/
    get_header();
    if ( have_posts() ) : while ( have_posts() ) : the_post();
?>
    <h1><?php the_title(); ?></h1>
    <div class="content">
        <?php the_content(); ?>
    </div>
<?php
    endwhile; endif;
    get_footer();
?>
```

Attention

Vérifiez qu'il n'y a pas d'espace entre Template Name et les deux points : sinon WordPress ne détectera pas votre modèle.

Autoriser le template perso

Les thèmes premium utilisent cette technique pour proposer des articles standards, et des articles **« full width »** (pleine largeur).

Pour autoriser plusieurs types de publication, il faut ajouter une seconde ligne avec **Template Post Type :**

```php
<?php
    /*
    Template Name: Full-width
    Template Post Type: post, page, product
    */
    get_header();
?>
```

Pour cet exemple j'ai autorisé dans les articles (post), pages et même les produits (product) dans le cas où WooCommerce est installé.

Cette fonctionnalité est plus récente puisqu'elle a été introduite seulement dans WordPress 4.7.

9. Les custom fields

Les champs personnalisés permettent d'ajouter des informations dans une publication en plus du contenu principal. L'interface des custom fields est cependant très limitée, c'est pour cela qu'à terme on basculera sur un outil comme **ACF**.

➔ **Utilité des custom fields**
Découverte

➔ **Activer les custom fields**
1.Alternative

➔ **Afficher les custom fields**

Utilité des CF

Pour ce cours je vais reprendre l'exemple du blog de jeux vidéo.
Imaginons que l'on veut faire des tests de jeux, et qu'à la fin de l'article on veuille donner une note, un avis, ainsi que les plus et les moins.
Un peu comme on le voit sur tous les sites de tests en quelques sortes.

NOTRE AVIS

Jauger un remake est toujours complexe. Reliquats de codes désormais désuets, trahison d'un esprit original, intérêt d'un remaniement visuel... tout se mélange dans diverses proportions pour créer une recette jamais miracle. Certains s'en sortent mieux que d'autres et c'est le cas de celui de Resident Evil 2. Loin d'être paresseux, il donne un nouveau visage à un jeu vieux de plus de 20 ans, ancré dans un genre codifié à l'extrême.

7 / 10

LES PLUS

- Une réalisation très correcte....
- Le sound-design travaillé avec soin
- Le côté organique de la progression
- Deux scénarios, plus deux histoires bis
- Mr X, l'idée qui change tout
- Le rechargement sans passer par le menu
- Le mode Hardcore pour les nostalgiques
- Certains thèmes musicaux flamboyants
- L'utilisation de l'obscurité
- Rythme global bien géré
- Quelques moments inédits

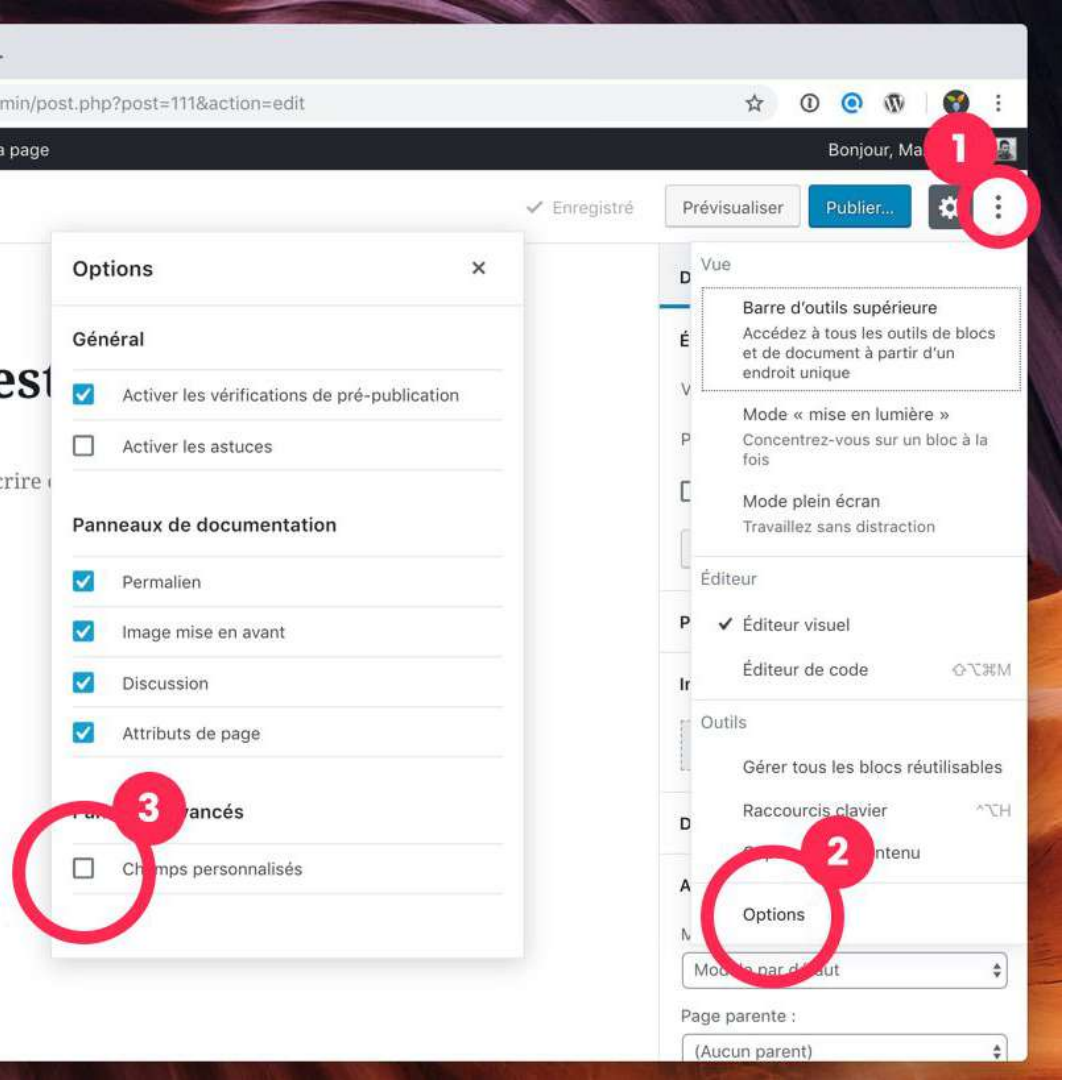
LES MOINS

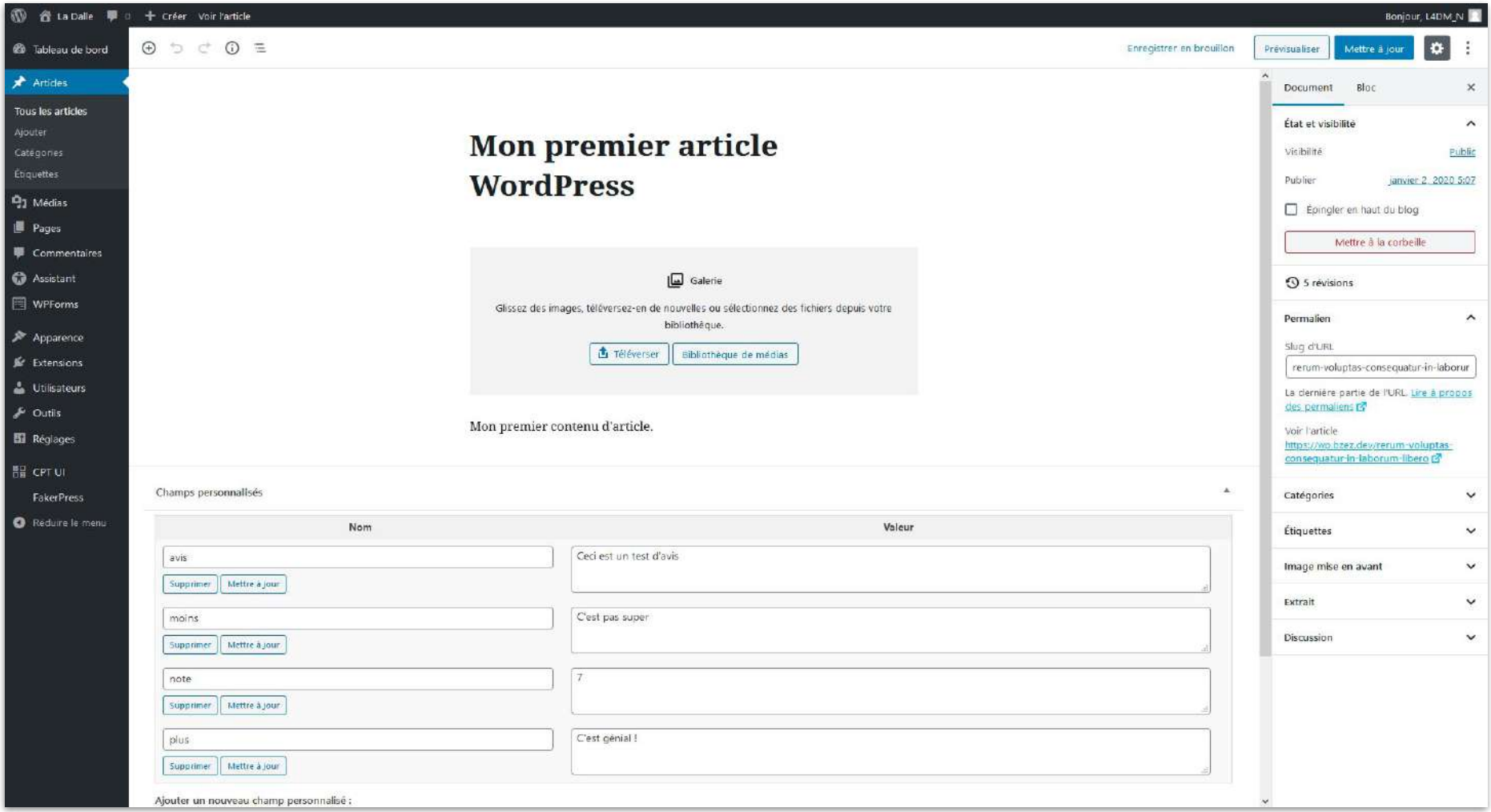
- ... malgré des problèmes sur certains reflets
- Ecriture clichée au possible
- Les boss inintéressants
- Manque de personnalité
- Des archaïsmes un peu pénibles
- Les énigmes toujours aussi banales

Activer les custom fields

Par défaut l'interface des champs additionnels est cachée.
Dans votre éditeur visuel allez dans le **menu 3 points** en haut à droite, puis **Options**, et enfin cochez **Champs personnalisés**.

L'interface des champs va alors apparaître en bas de page.




La Dalle
Créer
Voir l'article
Bonjour, L4DM_N
Tableau de bord
Articles
Tous les articles
Ajouter
Catégories
Étiquettes
Médias
Pages
Commentaires
Assistant
WPForms
Apparence
Extensions
Utilisateurs
Outils
Réglages
CPT UI
FakerPress
Réduire le menu
Enregistrer en brouillon
Prévisualiser
Mettre à jour
Document
Bloc
Mon premier article WordPress
Galerie
Glissez des images, téléversez-en de nouvelles ou sélectionnez des fichiers depuis votre bibliothèque.
Téléverser
Bibliothèque de médias
Mon premier contenu d'article.
Champs personnalisés
Nom
Valeur
avis
Ceci est un test d'avis
Supprimer
Mettre à jour
moins
C'est pas super
Supprimer
Mettre à jour
note
7
Supprimer
Mettre à jour
plus
C'est génial !
Supprimer
Mettre à jour
Ajouter un nouveau champ personnalisé :
État et visibilité
Visibilité
Public
Publier
janvier 2, 2020 5:07
Épingler en haut du blog
Mettre à la corbeille
5 révisions
Permalien
Slug d'URL
rerum-voluptas-consequatur-in-laborur
La dernière partie de l'URL. Lire à propos des permaliens
Voir l'article
https://wo.bzez.dev/rerum-voluptas-consequatur-in-laborum-libero
Catégories
Étiquettes
Image mise en avant
Extrait
Discussion

Afficher le contenu des custom fields

Pour cela, on va utiliser la fonction **get_post_meta()** dans **single.php** ou même dans **archive.php** :

```
<?php the_content(); ?>
<p>
    <strong>Avis :</strong>
    <?php echo get_post_meta( get_the_ID(), 'avis', true ); ?>
</p>

<p>
    <strong>Note :</strong>
    <?php echo get_post_meta( get_the_ID(), 'note', true ); ?> / 10
</p>

<div class="plus-moins">
    <div class="plus">
        <?php echo get_post_meta( get_the_ID(), 'plus', true ); ?>
    </div>
    <div class="plus">
        <?php echo get_post_meta( get_the_ID(), 'moins', true ); ?>
    </div>
</div>
```

10. Les HOOKS

C'est dans le fichier **functions.php** de notre thème que l'on va ajouter de nouvelles fonctionnalités et configurer certains aspects du CMS, grâce à un système très bien pensé nommé **Hooks**.

➔ **A quoi sert le functions.php**

➔ **Les hooks dans WordPress**

➔ **Hooks et functions.php**

➔ **Scinder functions.php**

A quoi sert le functions.php ?

Dans un fichier **functions.php** on peut :

- Activer des fonctionnalités comme les images mises en avant ;
- Déclarer des emplacements de menus, zones de widgets ;
- Déclarer des feuilles de styles CSS et scripts JS ;
- Déclarer de nouveaux types de publications et taxonomies ;
- Déclarer des fonctions sur mesure ;
- Gérer les requêtes Ajax ;
- Réécrire des URLs ;
- Personnaliser certaines réglages d'extensions ;
- Personnaliser l'interface d'administration ;
- Créer des routes dans l'API Rest ;
- Et bien plus encore…

Les hooks dans WordPress

Définition:

Les Hooks sont des fonctions déclarées par le développeur de thème ou d'extension qui permettent d'interagir avec le coeur de WordPress, d'autres thèmes ou extensions et lancés à des moments clés de leur exécution.
Par exemple, on peut intercepter le moment où WordPress enregistre un article dans la base, afin d'y apporter des modifications.
Il existe **2 types de Hooks** : les actions, un moment clé pour lancer ses propres fonctions, et les filtres, pour intercepter une valeur à un moment donné et la modifier.

Exemples :

- Un développeur peut insérer une fonction qui calcule le nombre de mots d'u
  où WordPress l'enregistre, via une **action** ;
- Une extension de SEO comme **Yoast** intercepte le **Title** de la page via un **filt**
  une version plus adaptée pour le référencement naturel.

Astuce

Grâce aux hooks, on va pouvoir modifier le comportement de WordPress sans pour autant modifier le code source de celui-ci.

Hooks et functions.php

En ce qui concerne la création de thèmes, notre fichier **functions.php** va lui aussi principalement s'appuyer sur des **hooks**.
Voici un exemple :

```php
//Cette fonction me permet de retirer des éléments inutiles de l'admin WordPress, afin de l'alléger
function mytheme_remove_menu_pages() {
    remove_menu_page( 'tools.php' );
    remove_menu_page( 'edit-comments.php' );
}
add_action( 'admin_menu', 'mytheme_remove_menu_pages' );
```

Le **hook** est appelé via la fonction **add_action()** et le moment clé est **admin_menu**.
Cette **action** est appelée par WordPress au moment ou il s'apprête à afficher le m
Ce n'est donc pas choisi arbitrairement.

J'indique à WordPress, qu'avant de générer le menu d'admin, je veux lancer ma pr
mytheme_remove_menu_pages (ça par contre c'est un nom arbitraire) afin d'ajou
entrées dans ce menu.

A éviter

Si vous mettez noms de fonctions trop génériques elles pourraient entrer en conflit avec des fonctions natives de WordPress, ou déclarées par une extension et qui porteraient le même nom.

Scinder functions.php

Dans certains thèmes, il se peut qu'au bout d'un moment votre fichier **functions.php** commence à être assez conséquent ! Du coup rien n'empêche de le scinder en plusieurs parties :

```php
//Séparer la config d'un thème (functions.php)
// Configuration du thème
require_once get_template_directory() . '/inc/config.php';

// Types de publication et taxonomies
require_once get_template_directory() . '/inc/post-types.php';

// Fonctionnalités
require_once get_template_directory() . '/inc/features.php';
```

On utilise la fonction **get_template_directory()** qui permet de récupérer le nom du dossier du thème actif (son Path).

Les thèmes premium ont tous tendance à utiliser cette technique, sinon leur fichier possèderait plusieurs milliers de lignes de code.

11. Scripts & Styles

Le fichier **functions.php** permet de charger les scripts JS et les feuilles de styles CSS de la bonne manière, afin de prendre en compte les éventuelles dépendances.

➔ **Pourquoi utiliser functions.php**

➔ **Bien déclarer ses scripts**

1. Le Handle
2. L'adresse du fichier
3. Éventuelles dépendances
4. Numéro de version
5. Chargement en haut ou en bas

➔ **Changer la version jQuery**

Pourquoi utiliser functions.php ?

En fait il faut voir votre site WordPress comme un écosystème dans lequel interagissent le coeur du CMS, les extensions et votre thème.

Et tout ce petit monde aura ses propres scripts et styles à charger, mais encore faut-il que ce soit fait dans le bon ordre : Imaginez que votre script utilise la librairie **select2**, qui elle même a besoin de jQuery.
Il vous faudra alors :

- Charger **jQuery** en premier ;
- Charger ensuite **Select2** ;
- Charger enfin votre **script**, qui pourra appeler le composant Select2 sans souci.

Pareil pour les styles : vous aurez peut-être envie de charger votre feuille de style en dernier afin d'écraser plus facilement des styles CSS apportés par une extension.

Le principe est donc simple : le thème et les extensions vont déclarer leurs scripts et styles à WordPress, indiquer les dépendances nécessaires à chacun d'entre eux. Et c'est le CMS qui va automatiquement charger le tout dans le bon ordre, au travers de la fonction wp_head() que l'on avait mise dans notre fichier **header.php** (d'où l'importance capitale de cette fonction).

Bien déclarer ses scripts (et styles)

Pour déclarer tout cela, on va utiliser le **hook** suivant :

```
25    add_action( 'wp_enqueue_scripts', 'votre_fonction' );
```

Pour rappel, le principe des Hooks est de dire à WordPress que lorsqu'il arrive au point clé **wp_enqueue_scripts,** il lance notre fonction nommée en deuxième paramètre (on appelle cela une **fonction de callback**).

```php
function mytheme_register_assets() {

    // Déclarer jQuery
    wp_enqueue_script(
        'jquery',
        'https://ajax.googleapis.com/ajax/libs/jquery/3.3.1/jquery.min.js',
        false,
        '3.3.1',
        true
    );

    // Déclarer le JS
    wp_enqueue_script(
        'mytheme',
        get_template_directory_uri() . '/asset/js/bootstrap.js',
        array( 'jquery' ),
        '1.0',
        true
    );

    // Déclarer style.css à la racine du thème
    wp_enqueue_style(
        'mytheme',
        get_stylesheet_uri(),
        array(),
        '1.0'
    );

    // Déclarer un autre fichier CSS
    wp_enqueue_style(
        'capitaine',
        get_template_directory_uri() . '/asset/css/bootstrap.css',
        array(),
        '1.0'
    );
}
add_action( 'wp_enqueue_scripts', 'mytheme_register_assets' );
```

Rappel

wp_enqueue_script(): permet d'inscrire un fichier JS

wp_enqueue_style(): permet d'inscrire un fichier CSS

Bien déclarer ses scripts (et styles)

Le handle : Le premier paramètre est le **handle**, c'est le nom unique que vous allez donner à votre fichier. Je met en général le nom du thème pour les deux.

L'adresse du fichier: C'est l'emplacement absolu où se trouve le fichier. Pour cela on va utiliser **get_template_directory_uri()** pour donner le chemin vers le thème, suivi du nom de fichier et éventuellement des sous-répertoires intermédiaires.
Si vous voulez charger directement le fichier style.css à la racine du thème, utilisez alors la fonction **get_stylesheet_uri()** seule.
Pour rappel ce fichier **style.css** est obligatoire car c'est lui qui contient la définition du thème.

Les éventuelles dépendances: Indique à WordPress que mon script/style devrait absolument être chargé **après** les scripts/styles indiqués en dépendances.
Pour cela on indique un tableau via **array()**, et on liste le **handle** des scripts/styles qui devront être chargés avant.

Le numéro de version: Ce numéro de version est important car il va permettre **d'invalider le cache** du navigateur lorsque vous changerez la valeur. Il permet d'ajouter un paramètre à la fin de l'URL du fichier : **script.js?ver=1.0.**

Le chargement en haut ou bas de page (scripts seulement): Et enfin, le dernier paramètre pour les scripts est un booléen : s'il est passé à true, le script sera chargé en bas de page, via le **wp_footer** et non pas **wp_header**, en haut.

Changer la version de jQuery

Par défaut WordPress va charger une version de jQuery avec le complément jQuery Migrate afin de garder la rétrocompatibilité avec les très vieux navigateurs (on parle des vieux Internet Explorer). Ce n'est donc plus très utile.

En plus, ça fait 2 requêtes HTTP pour pas grand chose…

Afin d'éviter cela on va devoir annuler l'inscription de jQuery de la liste des scripts déclarés, et refaire notre propre déclaration, avec la toute dernière version en branche 3.x :

```
// Déclarer jQuery
wp_deregister_script( 'jquery' ); // On annule l'inscription du jQuery de WP
wp_enqueue_script( // On déclare une version plus moderne
    'jquery',
    'https://ajax.googleapis.com/ajax/libs/jquery/3.3.1/jquery.min.js',
    false,
    '3.3.1',
    true
);
```

12. Les images

WordPress propose par défaut 3 tailles d'images, réglables depuis l'interface d'administration. En tant que développeur de thèmes, on va pouvoir en définir de nouvelles qui seront optimisées au design de celui-ci.

➔ **Les images mises en avant**

1. Activation
2. Affichage

➔ **Le "srcset"**

➔ **Déclarer de nouvelles tailles**

➔ **Utiliser les tailles**

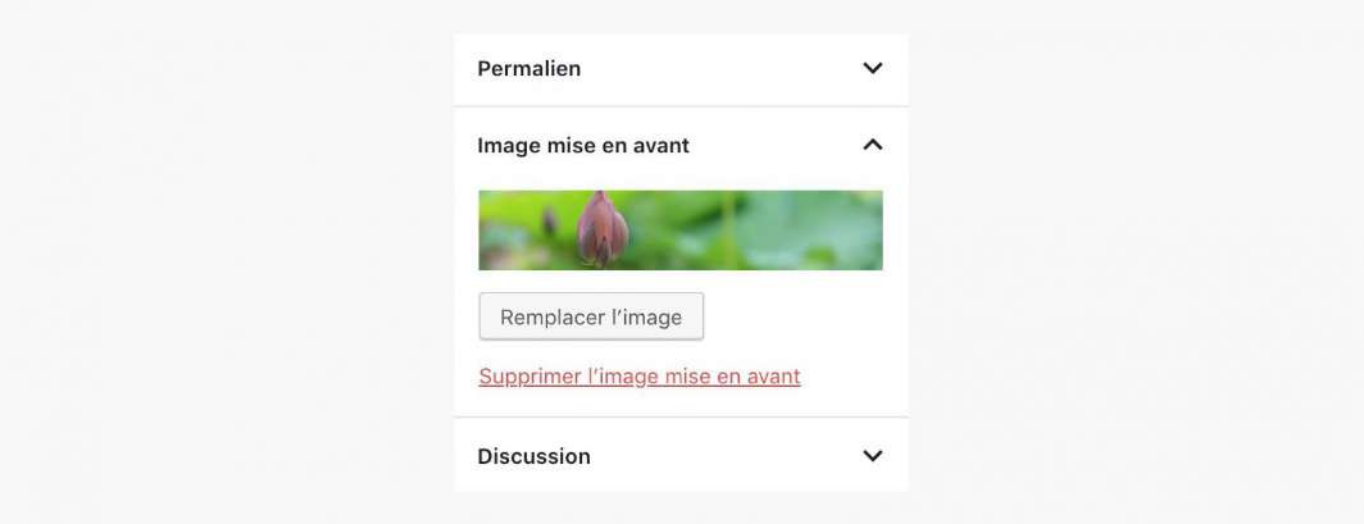
Les image mises en avant

Activation

Dans le cours sur la préparation des bases du thème, on avait déjà vu comment activer la fonctionnalité grâce à la fonction **add_theme_support()** dans **functions.php**

```
// Ajouter la prise en charge des images mises en avant
add_theme_support( 'post-thumbnails' );
```

Cela a pour but de rajouter la metabox Image mise en avant dans l'interface de l'éditeur



Les image mises en avant

Affichage

Pour afficher cette image dans votre thème, il suffit d'appeler la fonction **the_post_thumbnail()** dans **archive.php** ou **single.php** par exemple, à **l'intérieur de la boucle WordPress** :

```php
<?php
get_header();
if (have_posts()) : while (have_posts()) : the_post();
    ?>

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

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

Pas besoin de créer la balise **<img>** car la fonction la créé pour nous. Contrairement à ce que l'on pourrait croire, elle ne renvoie pas juste l'URL de l'image.

Le “srcset” des images

La balise <img> fournit les attributs srcset et sizes en plus du classique src.

C’est en fait une norme HTML5 pour améliorer l’affichage des images dans l’ère du web responsive et des périphériques aux multiples résolutions d’écran.

```
<img
        width="2507"
        height="1002"
        src="http://wp.local/wp-content/uploads/2019/01/chevaux.jpg"
        class="attachment-post-thumbnail size-post-thumbnail wp-post-image"
        alt="Mon image mise en avant"
        srcset="http://wp.local/wp-content/uploads/2019/01/chevaux.jpg 2507w,
        http://wp.local/wp-content/uploads/2019/01/chevaux-300x120.jpg 300w,
        http://wp.local/wp-content/uploads/2019/01/chevaux-768x307.jpg 768w,
        http://wp.local/wp-content/uploads/2019/01/chevaux-1024x409.jpg 1024w"
        sizes="(max-width: 2507px) 100vw, 2507px"
/>
```

- Dans l’attribut scrset, WordPress liste toutes les tailles d’images, et indique leur largeur (300w, 1024w…) ;
- Dans l’attribut sizes, WordPress indique la plus grande taille de l’image.

Déclarer de nouvelles tailles

Si les 3 tailles d'images proposées par défaut ne suffisent pas ?
Et bien vous allez pouvoir en créer d'autres grâce au **functions.php** !

```php
<?php

add_theme_support( 'post-thumbnails' );

// Définir la taille des images mises en avant
set_post_thumbnail_size( 2000, 400, true );

// Définir d'autres tailles d'images
add_image_size( 'products', 800, 600, false );
add_image_size( 'square', 256, 256, false );
```

La fonction **set_post_thumbnail_size()** permet de définir la taille des images mises en avant.
Cette taille sera donc utilisée par défaut par **the_post_thumbnail().**

Utiliser les tailles

Pour appeler une taille d'image spécifique, c'est tout simple. Il suffit d'indiquer le nom de l'image que vous souhaitez afficher en paramètre :

```php
<?php the_post_thumbnail( 'square' ); ?>
```

13. Menus & Search

WordPress nous permet d'administrer des menus, et d'afficher un moteur de recherche sur le site. On va voir les notions de menus et emplacements de menus, et voir pourquoi cette distinction est fondamentale.

- ➔ **Les menus dans WordPress**
- ➔ **Menus & emplacement de menu, quelle différence ?**
- ➔ **Déclarer un emplacement menu**
- ➔ **Créer un menu et l'assigner**
- ➔ **Ajouter le moteur de recherche**

Les menus dans WordPress

Il y a deux façons de gérer les menus dans WordPress, soit en passant par **Apparence > Menus**, soit via le customizer dans **Apparence > Personnaliser > Menus**.

Mais pour le moment, comme on n'a pas déclaré de menus dans notre thème, **vous ne trouverez pas ces entrées** dans l'administration WordPress.

Dans les deux cas, c'est exactement la même chose. A vous de choisir la technique que vous préférez.

Le **Customizer** est surtout utile pour les personnes qui utilisent un thème premium, qui propose alors pleins d'options de personnalisation à ses utilisateurs.

Vous, quand vous créez votre thème, c'est soit pour vous, soit pour le client, du coup le but c'est d'avoir un thème qui répond à un besoin précis.

Depuis chacun de ces deux interfaces, on voit que l'on peut créer plusieurs **menus**. Mais pour le voir apparaître il faut l'assigner à un **emplacement de menu**.

Menus et emplacement de menus

On peut utiliser un menu de plusieurs manières dans WordPress : l'assigner quelque part dans le thème, ou le mettre dans un Widget.
Ce qui va nous intéresser ici c'est d'assigner le menu dans notre thème. Mais pour cela on va devoir déclarer un ou plusieurs emplacements de menus.
L'emplacement de menu est une zone dans laquelle l'utilisateur va pouvoir créer et assigner un menu qu'il aura créé.

Alors pourquoi distinguer un menu et un emplacement de menus ?

Vous créez un menu et vous l'assignez à l'emplacement « **Menu Principal** ». Mais demain, vous décidez de changer de thème.
Cette fois les emplacements ne seront pas forcément les mêmes, il n'y aura plus d'emplacement « **Menu Principal** », mais à la place « **Menu primaire** ». C'est la même chose, seul le nom change.

Déclarer un emplacement de menu

On va maintenant aller déclarer 2 emplacements dans notre thème, un que l'on mettra dans l'entête, ce sera le menu principal, et un autre en pied de page (pour les mentions légales, la politique de confidentialité, le lien vers contact…).

Pour cela, rendez-vous dans **functions.php** et ajoutez :

```php
//On déclarer les emplacements de menus
register_nav_menus( array(
    'main' => 'Menu Principal',
    'footer' => 'Bas de page',
) );
```

Le premier paramètre est le **slug** du menu, le second le nom qui apparaîtra dans l'administration de WordPress.

Mais une minute ! On n'a toujours pas dit où se trouvaient ces menus dans notre template.
On va donc ouvrir **header.php** et ajouter la fonction **wp_nav_menu()** :

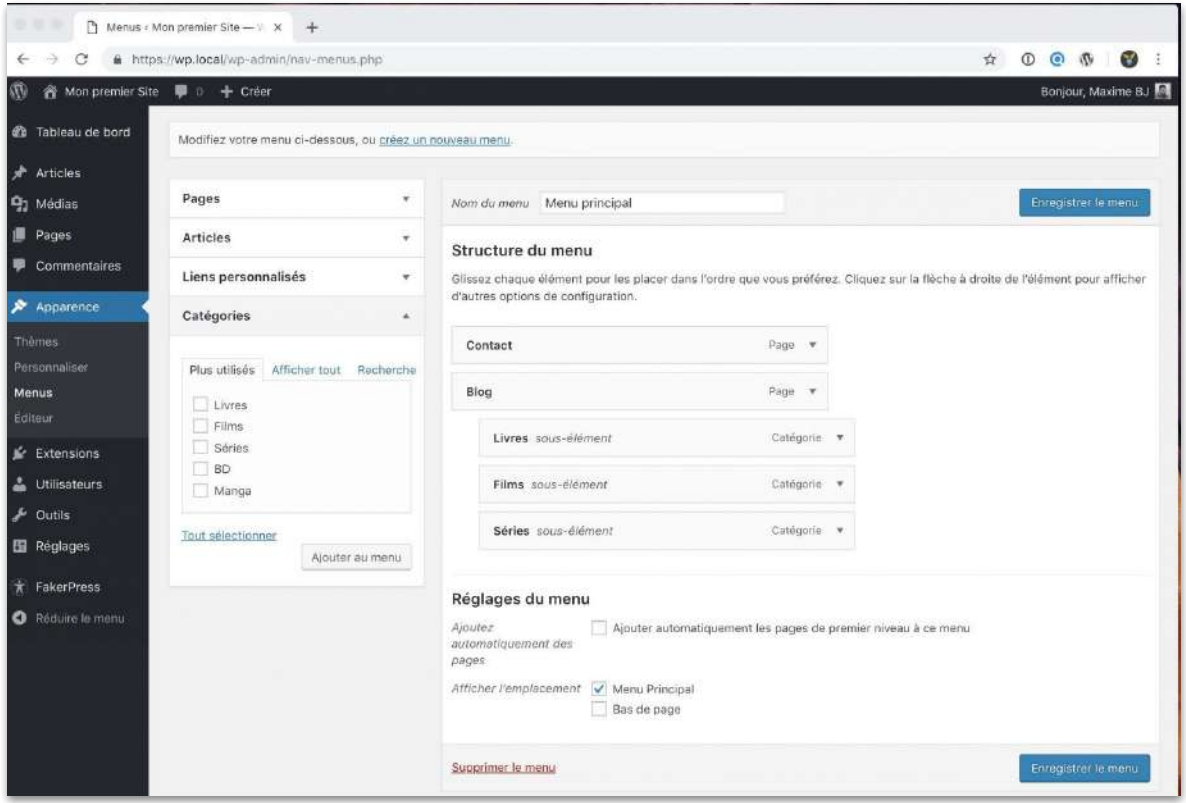
```php
<?php wp_nav_menu(
    array(
        'theme_location' => 'main',
        'container' => 'ul', // afin d'éviter d'avoir une div autour
        'menu_class' => 'site__header__menu', // ma classe personnalisée
    )
); ?>
```

Rappel
Si vous n'assignez pas de menu à un emplacement, WordPress listera automatiquement les Pages.

Le paramètre **theme_location** me permet de définir le slug de l'emplacement, parmi ceux définis juste au dessus dans le **functions.php**

Créer un menu et l'assigner

On peut maintenant aller créer notre menu, y ajouter des pages, le blog, éventuellement des catégories…



Bienvenue sur https://wp.bzez.dev

Voici votre page d'accueil. C'est différent d'un article de blog parce qu'elle restera au même endroit et apparaîtra dans la navigation de votre site (dans la plupart des thèmes).

En tant qu'utilisateur ou utilisatrice connecté-e à WordPress, vous pouvez modifier cette page en cliquant sur le bouton **Modifier la page** dans la barre d'outils. Cette page peut servir à vous présenter ou à présenter votre entreprise aux visiteurs du site et ressembler à quelque chose comme cela :

Bonjour ! Je suis un mécanicien qui aspire à devenir acteur, et voici mon site. J'habite à Bordeaux, j'ai un super chien baptisé Russell, et j'aime la vodka-ananas (ainsi qu'être surpris par la pluie soudaine lors de longues balades sur la plage au coucher du soleil).

... ou quelque chose comme cela :

La société 123 Machin Truc a été créée en 1971, et n'a cessé de proposer au public des machins-trucs de qualité depuis lors. Située à Saint-Remy-en-Bouzemont-Saint-Genest-et-Isson, 123 Machin Truc emploie 2 000 personnes, et fabrique toutes sortes de bidules super pour la communauté bouzemontoise.

Si vous préférez afficher vos derniers articles sur la page d'accueil, rendez-vous sur la page de Réglages et choisissez l'option «Les derniers articles» pour la page d'accueil.

Amusez-vous bien !

Ajouter le moteur de recherche

WordPress nous propose également son moteur de recherche, afin de trouver facilement un contenu dans une page ou les articles du blog. Et pour l'utiliser, c'est extrêmement simple.
Ajoutez simplement cette fonction dans votre **header.php** par exemple :

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

Résultat:

Il est également possible de personnaliser le HTML du moteur de recherche si celui par défaut ne vous convient pas. Pour cela créez un fichier **searchform.php** à la racine de votre thème et ajoutez ce code de base que vous pourrez personnaliser selon votre envie.

14. Sidebars & Widget

Les widgets sont des éléments interactifs que l'on retrouve général dans la barre latérale du blog, et permettant d'afficher des informations comme la liste des derniers articles, commentaires, catégories…

➔ **Déclarer & afficher une sidebar**

1. Autres paramètres

➔ **Ajouter des widgets**

Déclarer et afficher une sidebar

Pour déclarer une **sidebar**, c'est aussi simple que de déclarer un menu !
Pour cela on dispose d'une fonction **register_sidebar()** à placer dans le **functions.php** :

```
register_sidebar( array(
    'id' => 'blog-sidebar',
    'name' => 'Blog',
) );
```

Pour afficher notre sidebar dans le blog, on va ouvrir **archive.php** et ajouter la fonction **dynamic_sidebar()** :

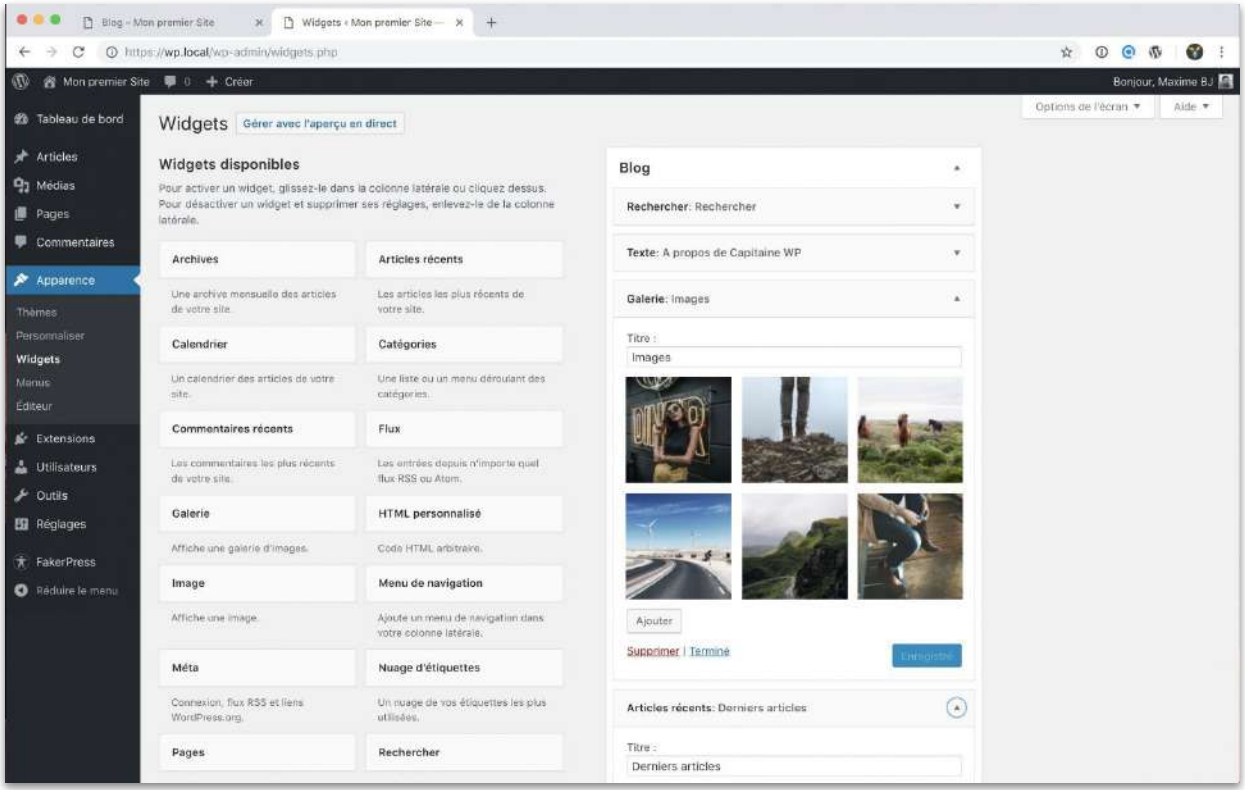
```
<aside class="site__sidebar">
    <ul>
        <?php dynamic_sidebar('blog-sidebar'); ?>
    </ul>
</aside>
```

Par défaut vous devez entourer votre sidebar d'un **<ul>** car chaque **widget** est écrit avec un **<li>**. Si vous n'êtes pas fan de **<li>**:

```
register_sidebar( array(
    'id' => 'blog-sidebar',
    'name' => 'Blog',
    'before_widget'  => '<div class="site__sidebar__widget %2$s">',
    'after_widget'  => '</div>',
    'before_title' => '<p class="site__sidebar__widget__title">',
    'after_title' => '</p>',
) );
```

Ajouter des widgets à notre sidebar

Pour cela rendez-vous dans **Apparence > Widgets** afin de faire votre sélection.



Notre sidebar

Mon premier article WordPress

Publié le janvier 2, 2020 par L4DM_N

Mon premier contenu d'article.

Lire la suite

Pas de commentaire

Calendrier

L	M	M	J	V	S	D
		1	2	3	4	5
6	7	8	9	10	11	12
13	14	15	16	17	18	19
20	21	22	23	24	25	26
27	28	29	30	31		

janvier 2020

Dernière photos

15. Pagination & Nav

WordPress propose des fonctions qui vous nous permettre de naviguer vers les entrées plus anciennes des pages archives, mais aussi d'un article à l'autre en allant facilement au suivant, ou au contraire en retournant au précédent.

➔ **Pagination pour les archives**

1. Pagination numérique
2. Alternative

➔ **Navigation entre les articles**

La pagination pour les archives

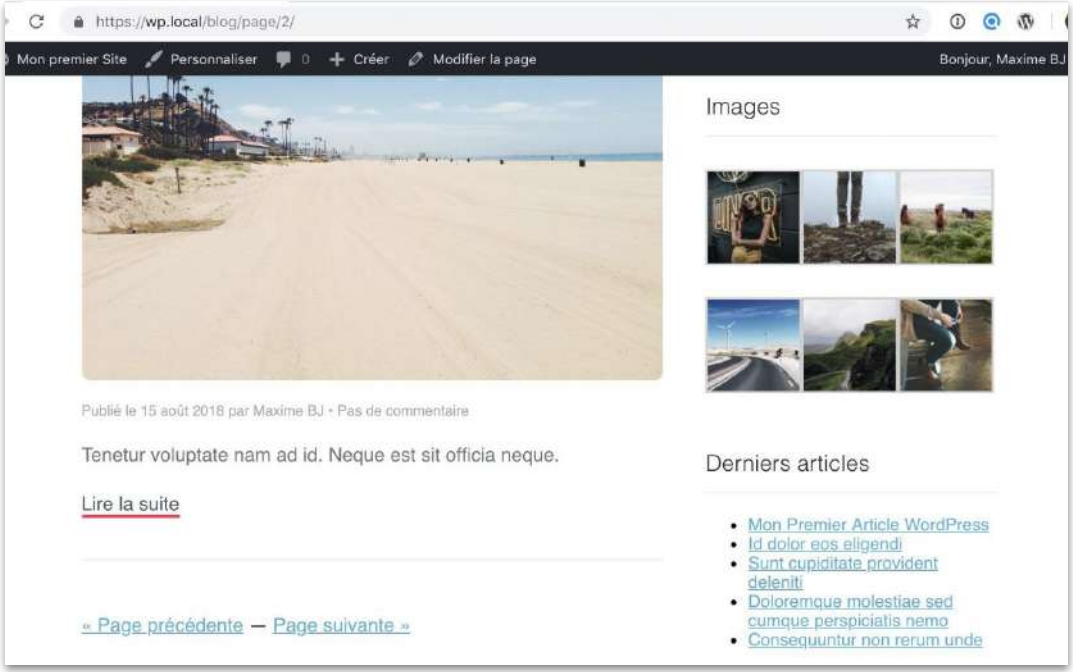
Pour ajouter une pagination simple, il vous suffit d'ajouter cette fonction dans votre fichier **archive.php**, après la fin de la boucle :

```php
<?php posts_nav_link(); // Après la boucle ?>
```

C'est un peu simpliste, mais ça marche !
Le souci avec cette fonction c'est qu'elle ne nous laisse que peu de choix de mise en forme.

Alternative:
Pour faciliter la personnalisation (CSS) je vous conseil d'utiliser ce code ou une extension:

```php
<div class="site__navigation">
    <div class="site__navigation__prev">
        <?php previous_posts_link( 'Page Précédente' ); ?>
    </div>
    <div class="site__navigation__next">
        <?php next_posts_link( 'Page Suivante' ); ?>
    </div>
</div>
```



16. Les commentaires

Les commentaires sont un bon moyen de créer de l'interaction avec vos lecteurs. De plus, ça permet de créer du contenu supplémentaire qui pourrait bien s'avérer pertinent pour les moteurs de recherche. Dans ce cours nous allons voir comment les mettre en place.

➔ **Ajouter les commentaires**

1. L'approche simple
2. L'approche sur mesure

Ajouter les commentaires

L'approche simple:
Ajoutez tout simplement cette fonction après votre contenu dans la boucle WordPress de **single.php** :

```php
<?php comments_template(); // Par ici les commentaires ;)
```

Cette ligne suffit à aller chercher le template de commentaires par défaut qui affiche d'abord la liste des commentaires publiés, puis ensuite le formulaire de saisie d'un nouveau commentaire.

Ce n'est pas très sexy, mais ça fonctionne !
Un peu de CSS pourra améliorer ça.

Ajouter les commentaires

L'approche sur mesure:
Créer un fichier comments.php à la racine de votre thème. Dès que WordPress le détecte, il utilisera ce fichier au lieu du template par défaut.

get_comments_number():
Le nombre de commentaires

wp_list_comments():
Liste des commentaires

comments_open():
Vérifier si il y en a.

comment_form():
Affiche le formulaire

```php
<div id="commentaires" class="comments">
    <?php if ( have_comments() ) : ?>
        <h2 class="comments__title">
            <?php echo get_comments_number(); // Nombre de commentaires ?> Commentaire(s)
        </h2>

        <ol class="comment__list">
            <?php
            // La fonction qui liste les commentaires
            wp_list_comments( array(
                'style'       => 'ol',
                'short_ping'  => true,
                'avatar_size' => 74,
            ) );
            ?>
        </ol>

        <?php
        // S'il n'y a pas de commentaires
        if ( ! comments_open() && get_comments_number() ) :
            ?>
            <p class="comments__none">
                Il n'y a pas de commentaires pour le moment. Soyez le premier à participer !
            </p>
        <?php endif; ?>
    <?php endif; ?>

    <?php comment_form(); // Le formulaire d'ajout de commentaire ?>
</div>
```

Conseil

Vous pouvez également vous baser sur le template des thèmes fournis par défaut avec WordPress histoire de vous en inspirer.

17. Custom Post Types

On va maintenant voir comment déclarer de nouveaux types de publication dans notre thème WordPress afin de ne pas être limité aux articles et pages. Et c'est dans le fichier **functions.php** de notre thème que ça se passe !

➔ **Déclarer un CPT**

1. Le hook
2. Version simple
3. Version complète

➔ **Afficher un CPT**

4. Rafraîchir la structure
5. Affichage

Déclarer un CPT

Le hook:
il va falloir placer le code dans un Hook WordPress. Le hook **init** fera l'affaire :

```php
<?php

function mytheme_register_post_types() {
    // La déclaration de nos Custom Post Types et Taxonomies ira ici
}
add_action( 'init', 'mytheme_register_post_types' );
```

La déclaration d'un type de publication s'effectue via la fonction **register_post_type()**.

Elle possède de nombreux paramètres.

Afin de nous simplifier la tâche, on va simplement définir le strict minimum pour l'instant, dans votre fichier **fonctions.php** ajouter le code suivant:

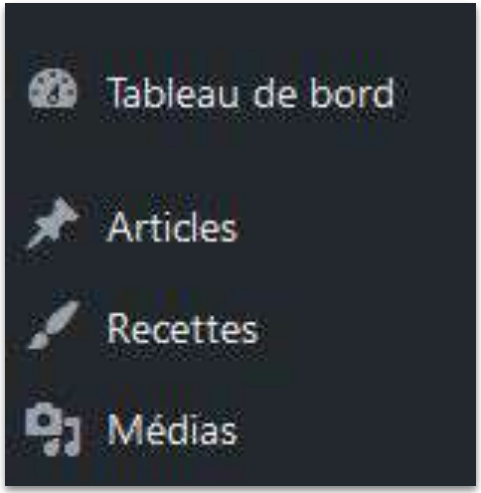
(Prochain slide)

Déclarer un CPT

```php
function mytheme_register_post_types() {

    // CPT Recettes
    $labels = array(
        'name' => 'Recettes',
        'all_items' => 'Toutes les recette',  // affiché dans le sous menu
        'singular_name' => 'Recette',
        'add_new_item' => 'Ajouter une recette',
        'edit_item' => 'Modifier la recette',
        'menu_name' => 'Recettes'
    );

    $args = array(
        'labels' => $labels,
        'public' => true,
        'show_in_rest' => true,
        'has_archive' => true,
        'supports' => array( 'title', 'editor','thumbnail','comments' ),
        'menu_position' => 5,
        'menu_icon' => 'dashicons-admin-customizer',
    );

    register_post_type( 'recettes', $args );
}
add_action( 'init', 'mytheme_register_post_types' ); // Le hook init lance la fonction
```

Déclarer un CPT

Voilà ce que vous devriez voir:



Déclarer un CPT

On va maintenant passer en revue les différents paramètres utilisés :

```php
$args = array(
    'labels' => $labels,
    'public' => true,
    'show_in_rest' => true,
    'has_archive' => true,
    'supports' => array( 'title', 'editor','thumbnail' ),
    'menu_position' => 5,
    'menu_icon' => 'dashicons-admin-customizer',
);
```

$labels - Dans le tableau vous allez pouvoir définir les phrases qui apparaissent dans l'administration de WordPress;
public - Lorsque vous créez un **Custom Post Type**, vous pouvez dire s'il est public ou non.
has_archive - C'est là où vous dites si vous voulez que le CPT se comporte comme des articles, ou comme des pages .
supports - Vous allez pouvoir choisir les champs affichés dans l'interface d'administration de vos publications.
menu_position - Vous pouvez également choisir l'endroit où apparaîtra votre CPT dans le menu WordPress.
menu_icon - Afin de distinguer visuellement notre CPT, on peut lui assigner une icône et pour cela, il y a 3 façons de procéder :

```php
$args = array(
    'menu_icon' => 'dashicons-portfolio',
    'menu_icon' => get_template_directory_uri() . '/img/cpt-icon.png',
    'menu_icon' => '' . base64_encode( data: "<svg>...</svg>" ),
);
```

Déclarer un CPT

Et pour terminer, il ne nous reste plus qu'à déclarer ce CPT.
On a **$labels** qui est passé dans **$args**, que l'on va passer à son tour dans la fonction :

```
132     register_post_type( 'recettes', $args );
```

Le premier paramètre, c'est le nom unique, le **slug** donc, de votre type de publication.

C'est également cette valeur qui va servir pour générer l'URL de l'archive qui sera donc du type :
https://wp.local/recettes.

Donc je vous déconseille de changer son nom.
À la place on va pouvoir changer plutôt son URL grâce à rewrite, que l'on va voir plus tard.

Si on veut absolument utiliser tous les paramètres, il va y avoir pas mal de lignes.
Je ne les met pas toutes là, mais à la place je vous renvoie sur la documentation officielle

Documentation:

https://developer.wordpress.org/reference/functions/register_post_type/

Attention

Si vous changez le slug de votre CPT en cours de route, les publications ne seront plus accessibles (car elles répondent toujours à l'ancien nom).

Afficher un CPT

On va maintenant voir comment rend notre nouveau CPT sur le site.
On va déjà créer des articles dans notre Portfolio, d'ailleurs, ce sont des « **recettes** » et non pas des « **articles** ».

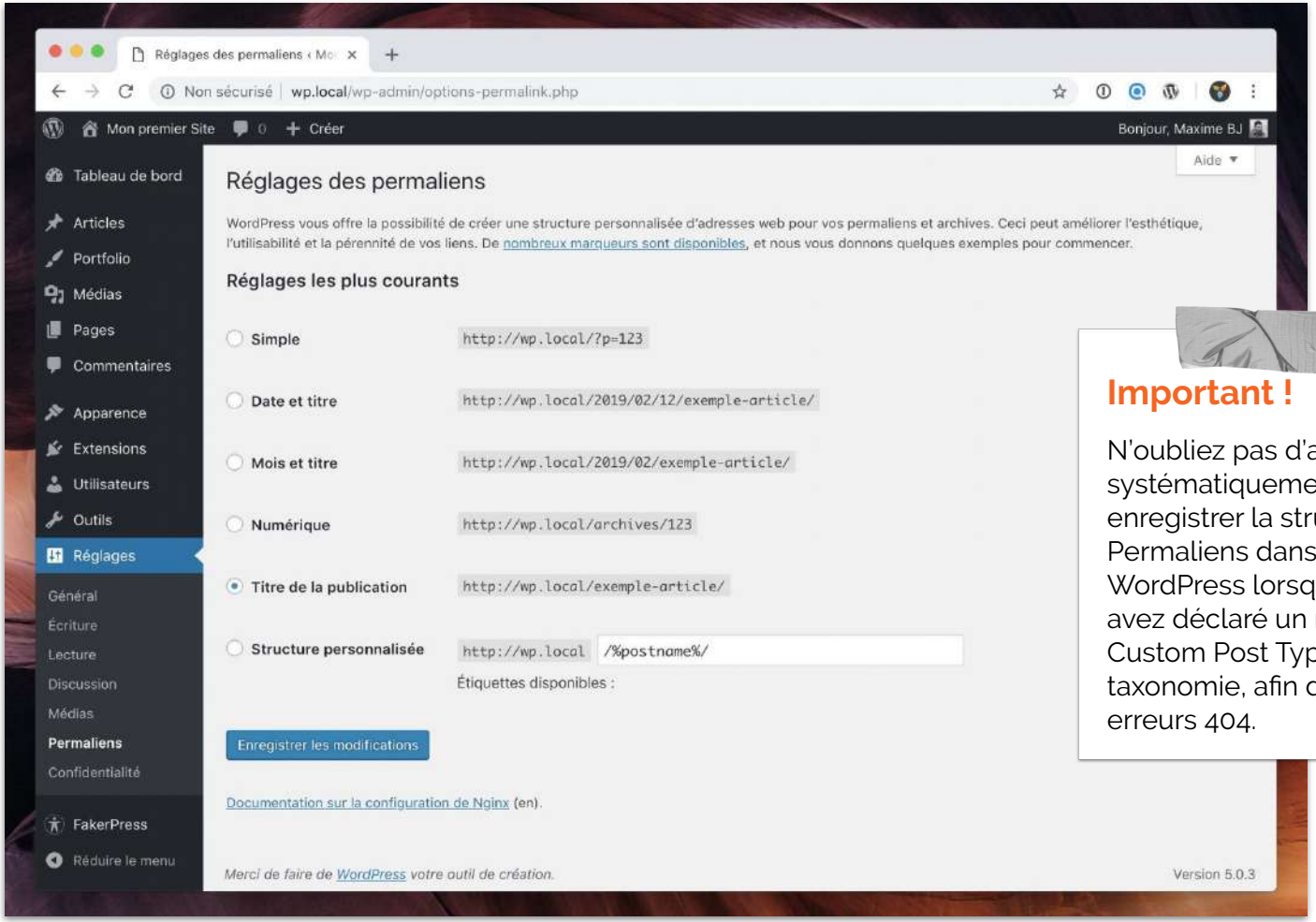
Une fois les projets créer rendez-vous sur **http://votrewp.com/recettes/**

Rafraîchir la structure des Permaliens

Vous êtes tombé sur une **erreur 404 ?** C'est normal !
C'est dû au fait que l'on utilise la réécriture des URL (afin qu'elles soient sexy) et que l'on a pas demandé à WordPress de prendre en compte le nouveau CPT et sa nouvelle URL.
Afin de corriger ce problème il va falloir réenregistrer la structure des **permaliens.**

Pour cela allez dans **Réglages > Permaliens**, et cliquez simplement sur **Enregistrer**

(voir slide suivant)



Important !

N'oubliez pas d'aller systématiquement enregistrer la structure des Permaliens dans WordPress lorsque vous avez déclaré un nouveau Custom Post Type ou une taxonomie, afin d'éviter des erreurs 404.

Important !

N'oubliez pas d'aller systématiquement enregistrer la structure des Permaliens dans WordPress lorsque vous avez déclaré un nouveau Custom Post Type ou une taxonomie, afin d'éviter des erreurs 404.

18. Templates perso

Maintenant que l'on a vu comment créer des Custom Post Types et des Taxonomies, on va voir comment leur assigner un template sur mesure grâce au Template Hierarchy ainsi que quelques **Conditional Tags** qui leur sont dédiés.

➔ **CPT & Template Hierarchy**

1. Les archives et le single
2. Taxonomies et termes

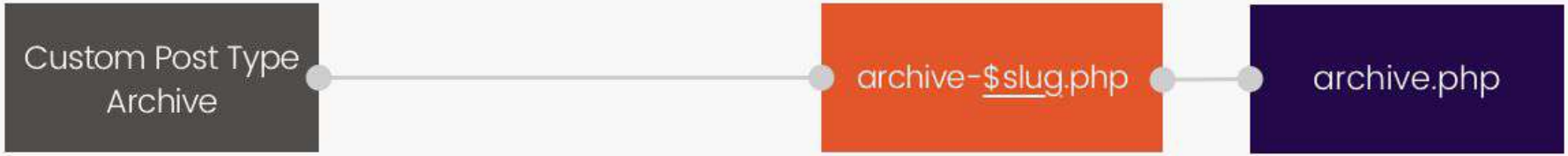
➔ **CPT & Conditional Tags**

3. Conditional Tags pour CPT
4. Conditional Tags pour Taxonomies

CPT et Template Hierarchy

Templates pour les archives et le single

Cheminement du Template Hierarchy pour les **Custom Post Types:**



Il est donc possible de créer un fichier **archive-*<type>*.php** et **single-*<type>*.php** pour gérer notre Post Type.
Dans mon cas je créé donc **archive-portfolio.php** et **single-portfolio.php** !
Au niveau du code, on va conserver la boucle WordPress (le CMS sait ce qu'il doit afficher) et on va simplement modifier le HTML ainsi que les classes CSS : *(slide suivant)*

CPT et Template Hierarchy

Templates pour les archives et le single (archive-portfolio.php)

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

    <h1 class="site__heading"><?php post_type_archive_title(); ?></h1>

    <main class="site__portfolio">
        <?php if( have_posts() ) : while( have_posts() ) : the_post(); ?>
            <div class="project">
                <h2 class="project__title">
                    <a href="<?php the_permalink(); ?>">
                        <?php the_title(); ?>
                    </a>
                </h2>
                <?php the_post_thumbnail(); ?>
            </div>
        <?php endwhile; endif; ?>
    </main>

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

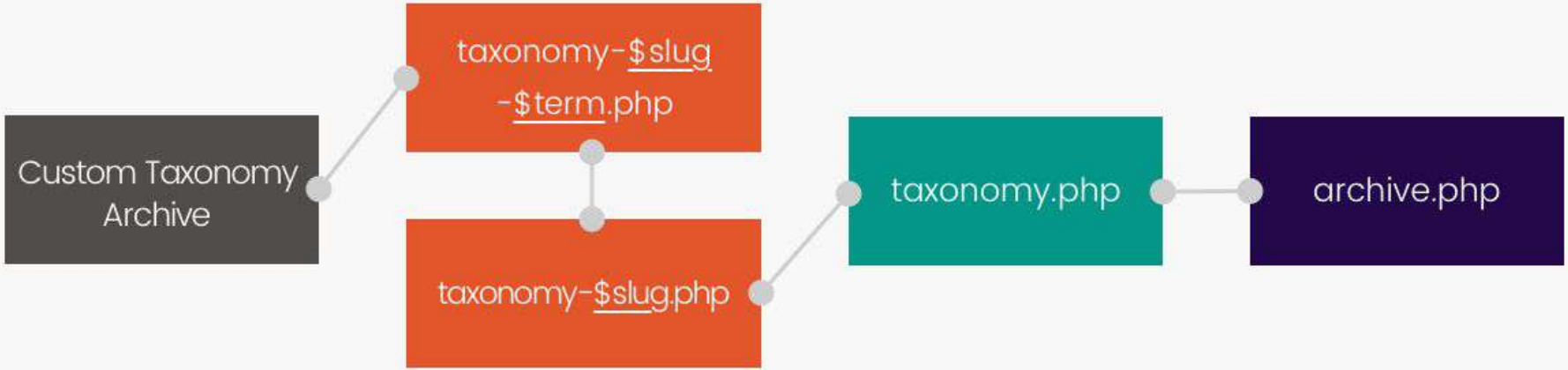
Notez la présence de la fonction **post_type_archive_title()** qui va afficher « PortFolio » en titre **<h1>.**

RECETTES

CPT et Template Hierarchy

Templates pour Taxonomies et termes

Les Taxonomies personnalisées ont aussi le droit à leur templates:



On peut donc avoir un template **taxonomy-*<taxo>*–*<term>*.php** ou plus simplement **taxonomy-*<taxo>*.php**

CPT et Conditional tags

Comme vous pouvez vous en douter, il existe aussi des **Conditionnal Tags** adaptés à notre CPT et les Taxonomies.

```php
<?php
if (is_post_type_archive('portfolio')) {
    //...
}
```

is_post_type_archive():
Permet de tester si vous êtes dans une archive de type “x”, ici “**portfolio**”

```php
<?php
if (is_singular('portfolio')) {
    //...
}
```

is_singular():
Là ici, petit piège ! Pour tester si vous êtes dans la page **single** d’un **Custom Post Type**, il faut utiliser **is_singular()** et non pas **is_single()**.

On peut également tester si on est dans une taxonomie en particulier, ou même si on a sélectionné un terme dans cette taxonomie en utilisant **is_tax()**

Pour consulter tous les Conditional Tags qui existent, rendez-vous sur la documentation !

19. Taxonomies

Pour classer les publications de nos **Custom Post Types**, on peut créer de nouvelles Taxonomies. Du coup, on n'est pas cantonné aux Catégories et aux étiquettes. Dans ce cours on va apprendre à déclarer des taxonomies et les assigner à nos types de publication.

➔ **Déclarer une taxonomie**

1. Où et comment faire ?
2. Paramétrage
3. Assignation à un ou plusieurs CPT

➔ **Afficher dans vos templates**

Déclarer une Taxonomies

Pour déclarer une nouvelle taxonomie on utilise la **fonction register_taxonomy()** que l'on place dans le **functions.php**, à la suite de la déclaration du **Custom Post Type** :

```php
function mytheme_register_post_types() {

    // CPT Portfolio
    $labels = array(...);

    $args = array(...);

    register_post_type( 'portfolio', $args );

    // Déclaration de la Taxonomie
    $labels = array(
        'name' => 'Type de projets',
        'new_item_name' => 'Nom du nouveau Projet',
        'parent_item' => 'Type de projet parent',
    );

    $args = array(
        'labels' => $labels,
        'public' => true,
        'show_in_rest' => true,
        'hierarchical' => true,
    );

    register_taxonomy( 'type-projet', 'portfolio', $args );
}
add_action( 'init', 'mytheme_register_post_types' ); // Le hook
```

$labels : Là aussi, il y a pleins de libellés disponibles mais on pourrait se cantonner seulement au nom.
Consultez la documentation pour obtenir toute la liste des **intitulés de taxonomie**.

public : Afin que notre taxonomie soit visible publiquement sur le site, on passe cette valeur à true. Il faut également penser à mettre **show_in_rest** pour la voir apparaître dans l'éditeur visuel (Gutenberg).

hierarchical : C'est le paramètre le plus important. Souhaitez-vous que votre taxonomie soit **hiérarchique**, comme les catégories, donc plus ou moins prédéfinies à l'avance, où plutôt volatile, comme les étiquettes ?

D'expérience, c'est le mode **hiérarchique** que l'on choisit dans une grande majorité des cas.

Déclarer une Taxonomies

Maintenant, il ne nous reste plus qu'à déclarer notre taxonomie grâce à la fonction **register_taxonomy()**.
Le premier paramètre est le **slug** de la taxonomie, le second celui du CPT et en troisième, les paramètres définis auparavant.

Mais il est également possible de l'assigner à plusieurs types de publication en même temps, même si on le fait rarement.

```php
<?php

// Assigner à un CPT
register_taxonomy( 'type-projet', 'portfolio', $args );

// Assigner à plusieurs CPT
register_taxonomy( 'type-projet', array( 'portfolio', 'autre' ), $args );
```

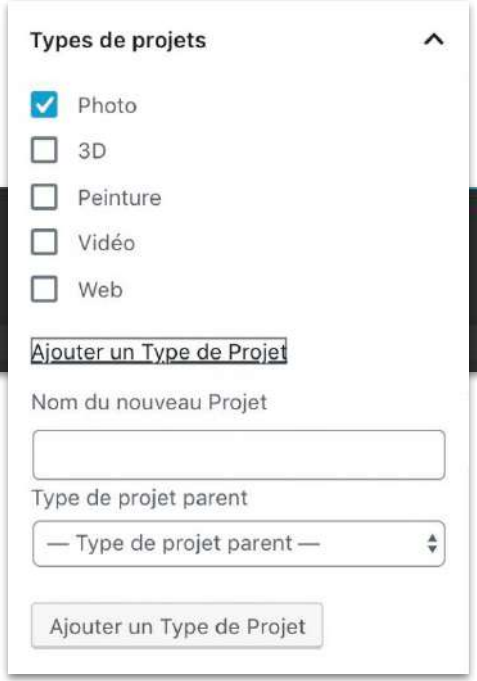
Afficher les taxonomies dans vos templates

Bien ! Il va falloir quelques **termes** dans cette **taxonomie**, et pour cela vous pouvez :

- Soit les créer via **Portfolio > Types de projets** ;
- Soit les créer directement lors de l'édition d'un projet.

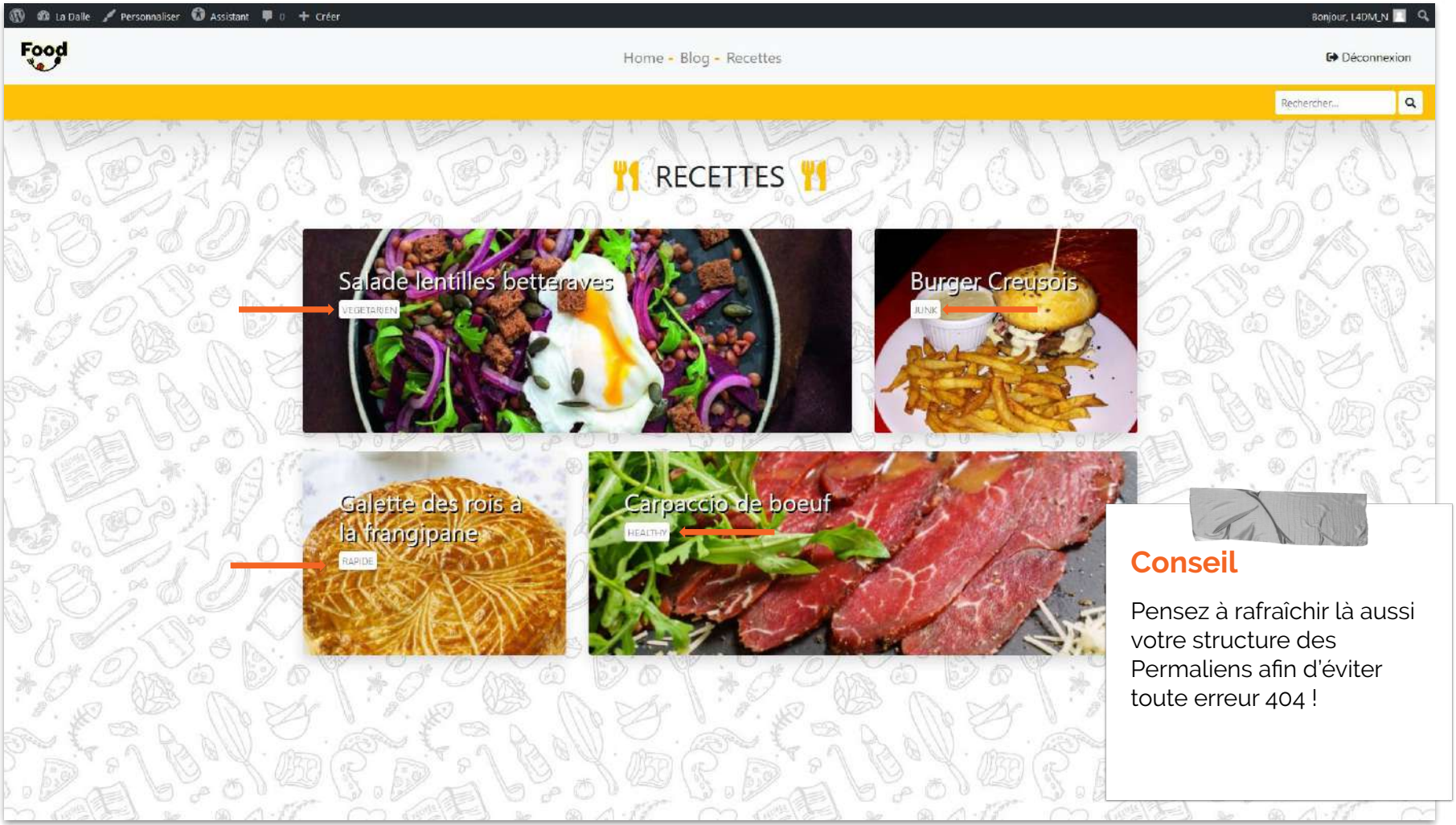
On va maintenant afficher notre taxonomie dans le template correspondant, soit dans **archive-portfolio.php** :

```php
<!-- Version simple -->
<?php the_terms( get_the_ID() , 'type-projet' ); ?>

<!-- Contrôle de l'affichafge avant / séparateur / après -->
<?php the_terms( get_the_ID() , 'type-projet', 'Type de projet :', ', ', '' ); ?>
```



Vous pouvez aussi définir ce qui va apparaître avant, après, et définir un séparateur (par défaut une virgule).
En soit, le but est de n'assigner qu'un type à chaque projet.

(résultat slide suivant)



Conseil

Pensez à rafraîchir là aussi votre structure des Permaliens afin d'éviter toute erreur 404 !

20. Advanced Custom Fields

Très vite, on se rend compte que WordPress nous limite pas mal lorsque l'on veut réaliser des mises en page modernes. Heureusement, il existe des outils et techniques qui vont nous permettre de lever tous les obstacles, comme les Page Builders ou encore ACF.

➔ La solution ACF

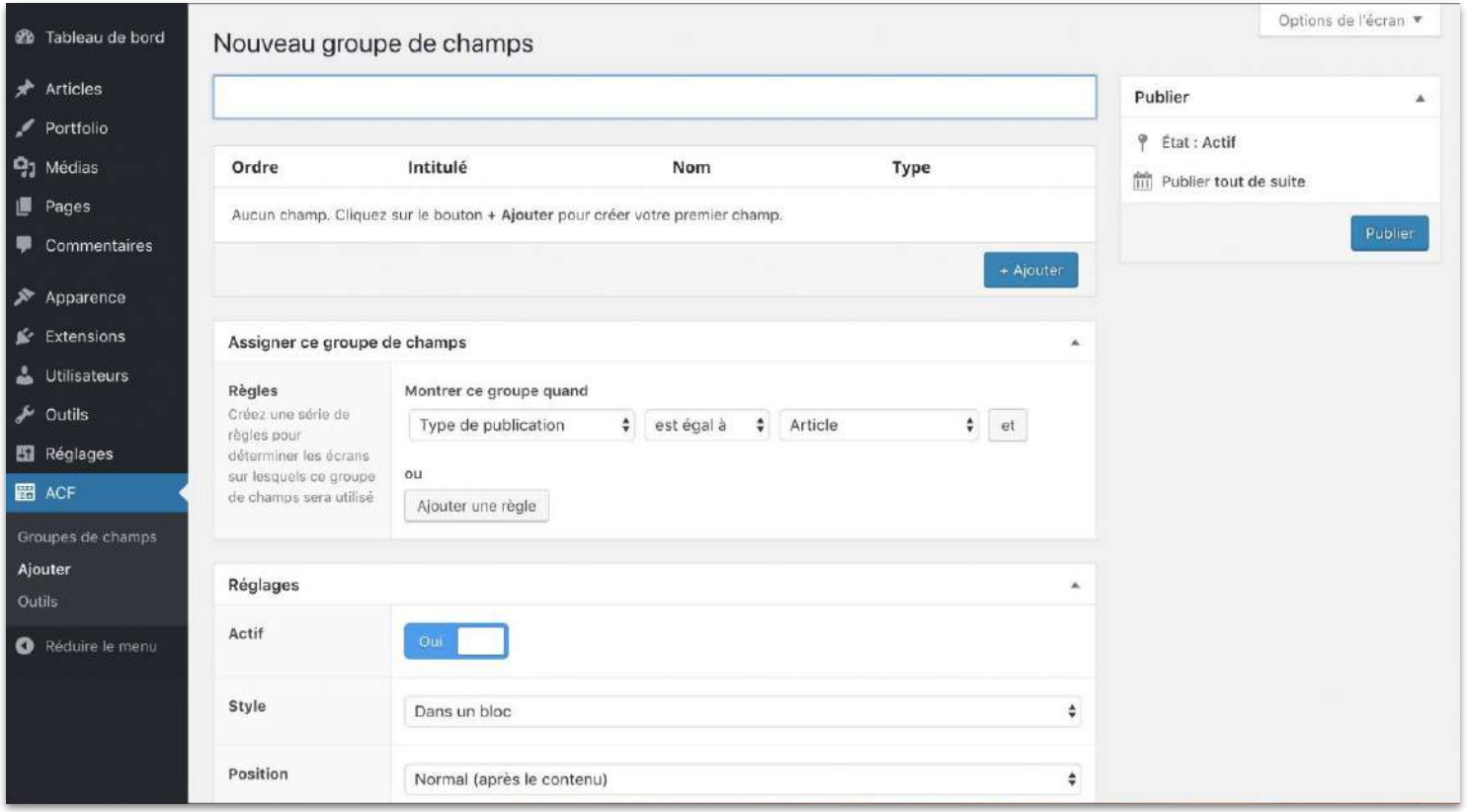
1. Déclarer un groupe de champs
2. Assigner un groupe de champs
3. Saisie du contenu
4. Affichage des champs
5. Cas particulier: les images

La solution Advanced Custom Fields

Une fois ACF activé, vous devriez voir une nouvelle entrée **ACF** en bas du menu latéral de WordPress :

La solution Advanced Custom Fields

Cliquez dessus puis sélectionnez **Ajouter**. Vous tomberez sur cette interface :



Déclarer un groupe de champs

Ensuite, on va ajouter notre premier champ. Pour cela cliquez sur le bouton bleu **+ Ajouter** nous permettant de définir un nouveau champ.

Le Temps

On va lui donner un titre, par exemple **Temps**, et définir le type de champ sur **Nombre**. Le Nom du champ est défini automatiquement à partir du titre. C'est l'identifiant qui doit être unique, et qu'on utilisera dans le template pour afficher la valeur. Autrement dit, c'est un Slug.

Ils y a plusieurs options dessous, mais pour le moment celles qui vont nous intéresser c'est :

- Suffixe : mettez **min** afin d'ajouter une indication sur le champ pour le rédacteur ;
- Valeur minimale : 0 (je ne vous fait pas un dessin) ;

Déclarer un groupe de champs

On va procéder de la même manière pour les autres champs :

Personnes

- Titre : Personnes ;
- Nom : personnes ;
- Type de champ : Nombre ;

Difficulté

- Titre : Difficulte ;
- Nom : difficulte ;
- Type de champ : Nombre.
- Valeur affichée dans le template : Les deux (array) ;

Déclarer un groupe de champs

Prix

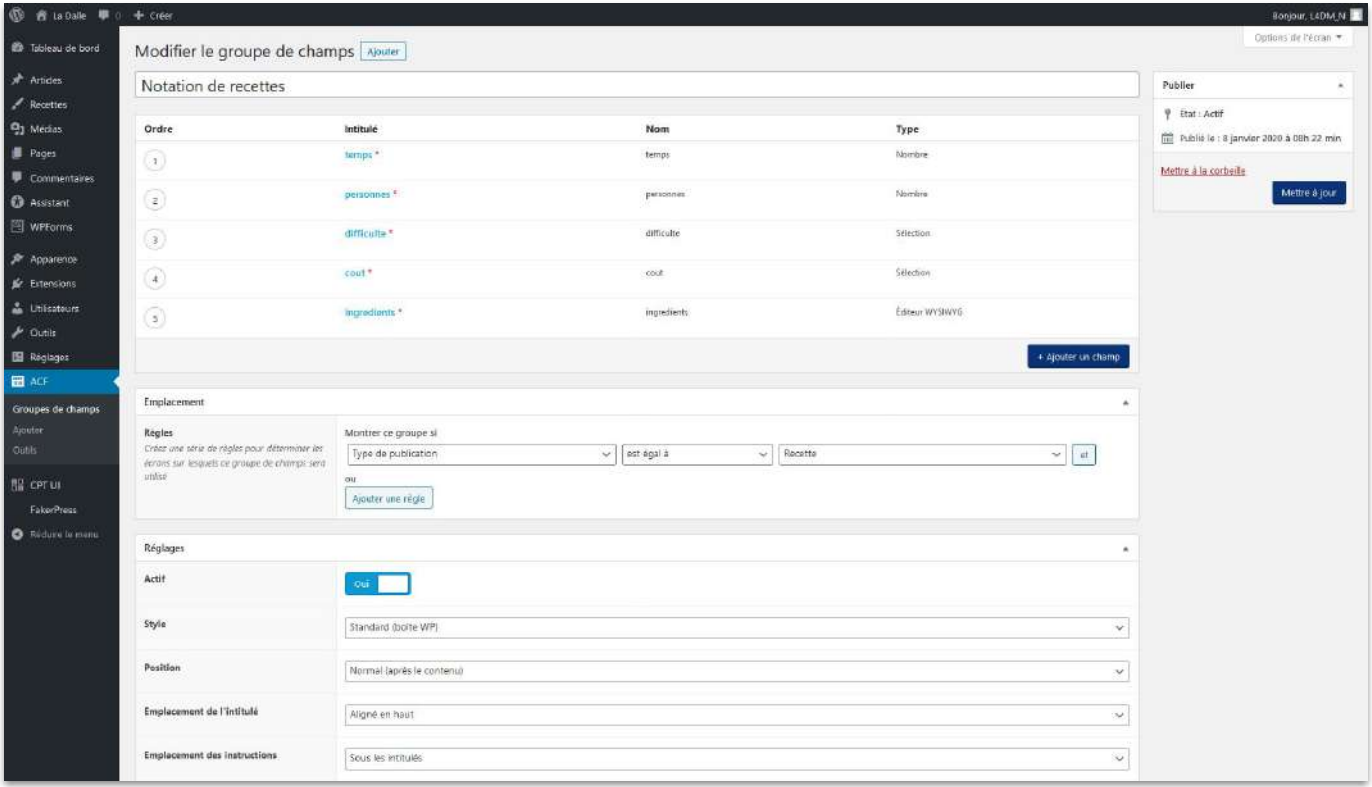
- Titre : Prix ;
- Nom : prix ;
- Type de champ : Nombre.
- Valeur minimale : 0.

Ingrédients

- Titre : Ingrédients ;
- Nom : ingredients;
- Type de champ : Editeur ;

Déclarer un groupe de champs

C’est grâce à ce genre de champs comme les images qu’ ACF va prendre tout son intérêt !
Vous devriez maintenant donc avoir tous ces champs :



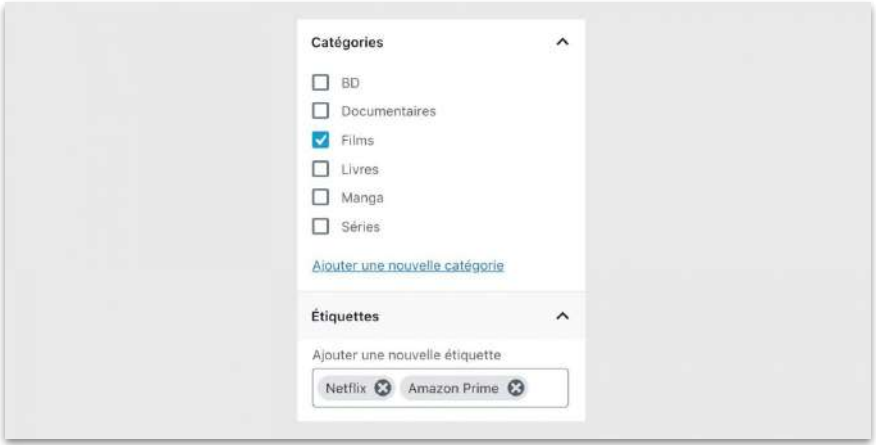
Déclarer un groupe de champs

Un champ ou une taxonomie ?

D'ailleurs, on pourrait ajouter un champ **Origine**, où l'on pourrait indiquer si la recette et d'origine *Japonaise*, *Française*…

Mais en fait là, ce serait plutôt une façon de « classer » la recette.

Du coup, dans ces deux cas, créer un champ n'est pas la meilleure approche. À la place, il faudrait plutôt créer une taxonomie ! Et ça tombe bien, car maintenant vous savez faire !

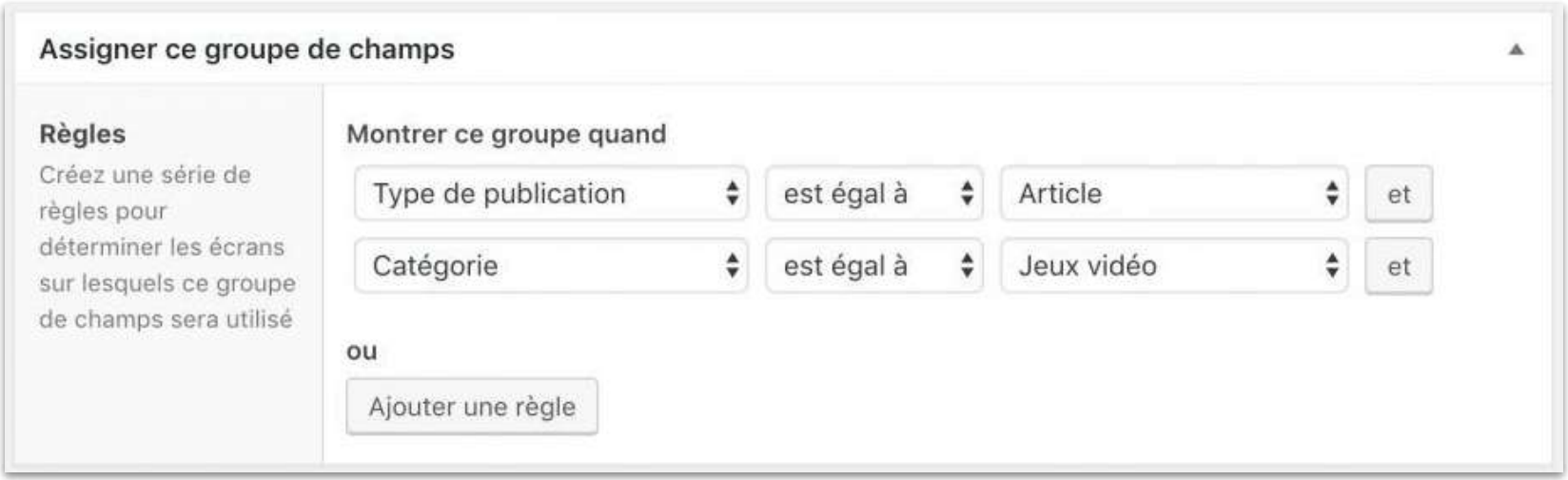


Assigner un groupe de champs à des publications

On a créé un groupe de champs, mais il faut maintenant indiquer à ACF où et quand il doit apparaître.

Nous ce que l'on veut, c'est qu'il apparaisse dans les **articles** de la **catégorie Jeux vidéo** seulement.
(Vérifiez que vous avez bien une telle catégorie créée dans votre site).

Sous l'interface d'ajout de champs que l'on vient de manipuler, vous trouverez une autre Metabox nommée **Assigner ce groupe de champs**.

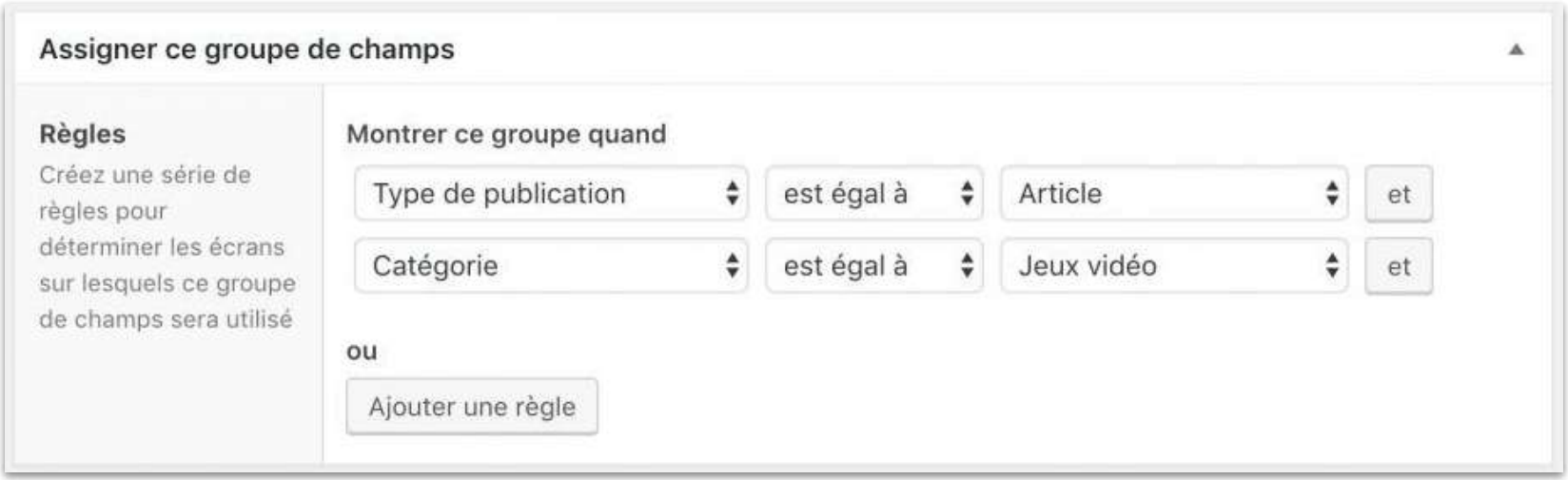


Assigner un groupe de champs à des publications

On a créé un groupe de champs, mais il faut maintenant indiquer à ACF où et quand il doit apparaître.

Nous ce que l'on veut, c'est qu'il apparaisse dans les **articles** de la **catégorie Jeux vidéo** seulement.
(Vérifiez que vous avez bien une telle catégorie créée dans votre site).

Sous l'interface d'ajout de champs que l'on vient de manipuler, vous trouverez une autre Metabox nommée **Assigner ce groupe de champs**.



Afficher les champs dans le template

Dernière étape : afficher les valeurs de nos champs dans notre article !
Pour cela ACF propose une fonction pour récupérer et afficher le contenu des champs dans votre template :

```php
<!-- Afficher une valeur -->
<?php the_field( 'note' ); ?>

<!-- Récupérer la valeur -->
<?php $note = get_field( 'note' ); ?>
```

C'est la même logique qu'avec les **Templates Tags** natifs de WordPress : une fonction **the_** permet d'afficher directement le résultat, et une fonction **get_** pour récupérer la valeur.

En fait la fonction **the_field** utilise la fonction native **get_post_meta** que l'on avait pu voir dans le cours sur les champs personnalisés.
On pourrait d'ailleurs utiliser cette dernière fonction mais celle d'ACF simplifie un peu les choses.

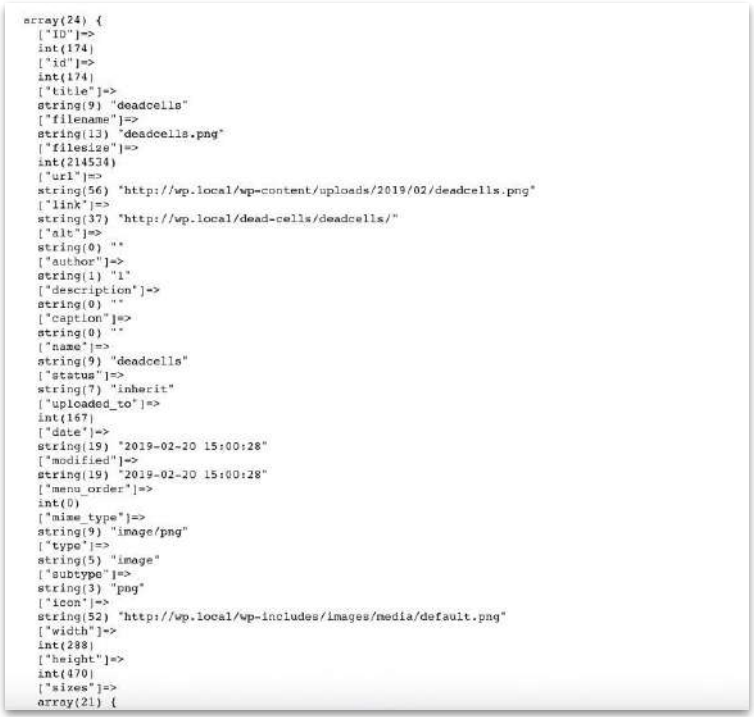
Afficher les champs dans le template

Mis en application avec mes champs dans la page, ça donne ça :

```
<div class="container p-5">
    <article class="post">
        <div class="row text-center p-5 rounded bg-light shadow-sm mb-5">
            <div class="col-3 border-right">
                <div class="row">
                    <div class="col-12">
                        <h5>Temps</h5>
                    </div>
                </div>
                <div class="row">
                    <div class="col-12 ">
                        <span class="h3 text-warning font-weight-bold"> <?php the_field('temps'); ?> min</span>
                    </div>
                </div>
            </div>
            <div class="col-3 border-right">
                <div class="row">
                    <div class="col-12">
                        <h5>Personnes</h5>
                    </div>
                </div>
                <div class="row">
                    <div class="col-12">
                        <span class="h3 text-warning font-weight-bold"><?php the_field('personnes'); ?></span>
                    </div>
                </div>
            </div>
```

Cas particulier: les images

Pour l'image, c'est un peu différent : ACF ne renvoie pas la balise image, ni même juste l'URL de celle-ci, mais un tableau contenant plusieurs données. Si je fais un **var_dump** de mon image j'obtiens :



```
array(24) {
  ["ID"]=>
  int(174)
  ["id"]=>
  int(174)
  ["title"]=>
  string(9) "deadcells"
  ["filename"]=>
  string(13) "deadcells.png"
  ["filesize"]=>
  int(214534)
  ["url"]=>
  string(56) "http://wp.local/wp-content/uploads/2019/02/deadcells.png"
  ["link"]=>
  string(37) "http://wp.local/dead-cells/deadcells/"
  ["alt"]=>
  string(0) ""
  ["author"]=>
  string(1) "1"
  ["description"]=>
  string(0) ""
  ["caption"]=>
  string(0) ""
  ["name"]=>
  string(9) "deadcells"
  ["status"]=>
  string(7) "inherit"
  ["uploaded_to"]=>
  int(167)
  ["date"]=>
  string(19) "2019-02-20 15:00:28"
  ["modified"]=>
  string(19) "2019-02-20 15:00:28"
  ["menu_order"]=>
  int(0)
  ["mime_type"]=>
  string(9) "image/png"
  ["type"]=>
  string(5) "image"
  ["subtype"]=>
  string(3) "png"
  ["icon"]=>
  string(52) "http://wp.local/wp-includes/images/media/default.png"
  ["width"]=>
  int(288)
  ["height"]=>
  int(470)
  ["sizes"]=>
  array(21) {
```

Toutes les données de notre image sont là

C'est dû au fait que, lorsque l'on a défini notre champ ACF, on a demandé un tableau de données en format de sortie :

On observe qu'ACF nous fournit pas mal d'informations sur l'image, comme son nom ou son titre. Mais ce qui va nous intéresser c'est le sous-tableau sizes qui contient toutes les tailles d'images intermédiaires que l'on a créé dans le cours sur les **tailles d'images personnalisées**.

Cas particulier: les images

Vous pouvez sinon récupérer l'ID de l'image. Dans ce cas quelques fonctions WordPress vont vous permettre de récupérer votre image :

```php
<?php
    $picture_ID = get_field( 'pochette' ); // On récupère cette fois l'ID
    $url = wp_get_attachment_image_src( $picture_ID, 'post-thumbnail' );
?>
    <img src="<?php echo $url; ?>">
```

On utilise la fonction native **wp_get_attachment_image_src** pour récupérer l'URL de l'image.
Le second paramètre permet d'indiquer quelle taille d'image on souhaite.

Les deux techniques se valent, donc faites comme vous préférez !

Si vous souhaitez en savoir plus sur la gestion des champs ACF, je vous invite à consulter la documentation officielle qui est extrêmement bien réalisée.

Documentation ACF :

https://www.advancedcustomfields.com/resources/

Voilà ce que vous devriez obtenir *(avec css)*