

CMS – WORDPRESS

Module 17 : La création de son propre thème



Objectifs

- Créer son propre thème



La création de son propre thème

functions.php

- functions.php :
 - ajouter des fonctions PHP au thème.
 - Tous les thèmes disposent de ce fichier.
 - Possibilité d'ajouter des fonctions en pur PHP et des fonctions natives de WordPress.
- Exemple ajout :
 - fichiers JavaScript, des fichiers CSS, des langues, d'ajouter des menus, des sidebars.
 - réécrire des fonctions natives de WordPress,
 - créer de nouvelles fonctions. Appel à d'autres fichiers PHP souvent dans un dossier annexe (inc ou includes)
- Lors de l'installation d'un thème, vous verrez que ce fichier est plus ou moins rempli selon le thème.
- Le fichier peut impacter aussi bien le côté internaute (front office) que l'administration (back office).

La création de son propre thème

Principe Zones d'affichages

- Pour un thème séparer les différentes zones d'affichage dans plusieurs fichiers PHP séparés.
- Permet de mieux gérer son code, de ne pas avoir des centaines de lignes de code dans un même fichier.
- Permet de séparer les zones dont l'affichage est identique sur toutes les pages (en-tête et pied de page) des zones qui varient selon la page affichée (la zone centrale de contenu).
- Pour faire cela, il va falloir déterminer un fichier « maître » qui contiendra dans son code l'inclusion des autres fichiers.
- Les différentes parties d'une page
 - Dans cet exemple, nous avons :
 - Un fichier pour l'en-tête, nommé **en-tete.php**.
 - Un fichier pour la partie centrale, nommé **corps.php**.
 - Un fichier pour le pied de page, nommé **pied-page.php**.
 - Un fichier « maître » qui assemble tous ces fichiers, nommé **index.php**.




La création de son propre thème

Principe Zones d'affichages

- Le fichier d'en-tête

- Le fichier d'en-tête, **en-tete.php**, contient :
- Le début de la déclaration HTML de la page.
- Un titre de niveau **<h1>**.
- Un petit code PHP construisant une chaîne de caractères par concaténation.

A screenshot of a code editor window titled 'entete.php'. The code is written in PHP and HTML. It starts with a DOCTYPE declaration, followed by the opening tags for the HTML document, head, and body. Inside the head, there is a meta tag for content type and charset, and a title tag. Inside the body, there is an h1 tag and a PHP block that defines two variables, \$valeur and \$nom, and then echoes a string that concatenates these variables to form a personalized greeting.

```
1 <!DOCTYPE html>
2 <html>
3 <head>
4 <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
5 <title>Mon site en PHP</title>
6 </head>
7 <body>
8 <h1>Titre de mon site dans l'en-tête</h1>
9 <?php
10     $valeur='WordPress';
11     $nom='Denis';
12     echo '<p>Bonjour '.$nom.', êtes vous administrateur de '.$valeur.' ?</p>';
```

Principe Zones d'affichages

- Le fichier de la zone centrale
 - Le fichier de la zone centrale, **corps.php**, contient :
 - Un titre de niveau <h2>,
 - Un paragraphe de texte statique <p>,
 - Un deuxième titre de niveau <h2>,
 - Un petit code PHP sur un test de variable.

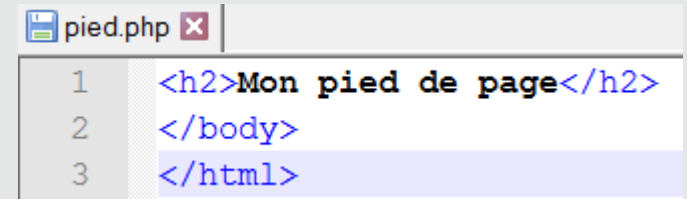
```
corps.php x
1  <h2>C'est le corps de ma page</h2>
2  <p>Cum sociis natoque penatibus et magnis dis parturient
3  montes, nascetur ridiculus mus. Curabitur blandit tempus porttitor.
4  Vestibulum id ligula porta felis euismod semper. Maecenas sed diam
5  eget risus varius blandit sit amet non magna. Donec sed odio dui. Sed
6  posuere consectetur est at lobortis.</p>
7  <h2>Un petit test avec if()</h2>
8  <?php $val=2; ?>
9  <?php if ($val>10): ?>
10     <p>La valeur est plus grande que 10.</p>
11 <?php else : ?>
12     <p>La valeur est plus petite que 10.</p>
13 <?php endif; ?>
```

La création de son propre thème

Principe Zones d'affichages

- Le fichier du pied de page

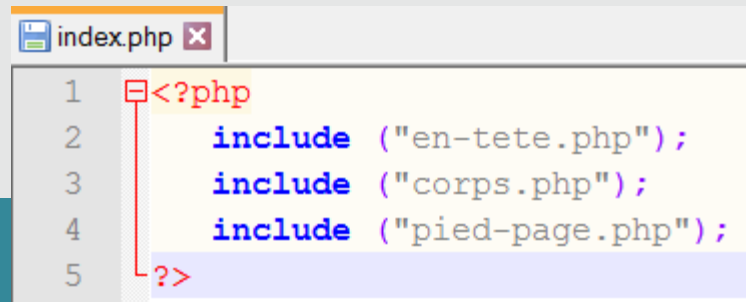
- Le fichier du pied de page, **pied-page.php**, contient :
- Un titre de niveau `<h2>`,
- Les balises de fermeture des éléments `<body>` et `<html>`.



```
pied.php
1  <h2>Mon pied de page</h2>
2  </body>
3  </html>
```

- Le fichier maître

- Le fichier « maître », **index.php**, est celui qui va assembler tous ces fichiers.
- Ce fichier utilise la fonction `include()` avec comme argument le nom du fichier à inclure.



```
index.php
1  <?php
2      include ("en-tete.php");
3      include ("corps.php");
4      include ("pied-page.php");
5  ?>
```

La création de son propre thème

Démonstration

Créer son propre thème



La création de son propre thème

La structure des thèmes

- **L'objectif**

- Comment sont structurés les thèmes WordPress et quels sont les fichiers qui peuvent intervenir pour créer un thème.
- Pour ce faire, nous allons nous baser sur le thème **Twenty Seventeen**.
- Ce qu'il faut bien comprendre c'est que chaque concepteur de thème peut structurer comme il veut ses thèmes, avec les fichiers qu'il souhaite.
- Donc chaque structure de thème pourra être unique, même s'il existe des éléments constants et obligatoires.

La création de son propre thème

Les fichiers des thèmes

- **Les fichiers obligatoires**

- Les fichiers strictement obligatoires sont au nombre de deux :

- **index.php** : c'est le fichier qui est utilisé quand aucun autre fichier de mise en page (template) n'est présent dans le dossier de votre thème. Il est aussi utilisé pour afficher la page d'accueil de votre site.
- **style.css** : c'est le fichier des styles CSS qui sont utilisés pour la mise en forme et la mise en page du thème. Ce fichier contient aussi des paramètres indispensables pour la gestion des thèmes dans l'interface d'administration de WordPress.

- Le fichier de personnalisation

- Le fichier **functions.php** va permettre, entre autre et si besoin est, d'ajouter des fonctionnalités de paramétrage, des options spécifiques de personnalisation du thème.



La création de son propre thème

Les fichiers des thèmes

- Les fichiers de structure

- Pour créer un thème, vous pouvez utiliser plusieurs fichiers.
- Chaque fichier va s'occuper de l'affichage d'une partie des pages présentes à l'écran. Cela va permettre d'éviter de répéter plusieurs fois la même structure.
- Avec une structure "classique", vous avez :
 - Le fichier **index.php** qui fait office de "chef d'orchestre". C'est dans ce fichier que sont appelés les autres fichiers spécialisés dans tel ou tel type d'affichage.
 - le fichier **header.php** s'occupe exclusivement de l'affichage de l'en-tête.
 - le fichier **footer.php** s'occupe exclusivement de l'affichage du pied de page.
 - le fichier **sidebar.php** s'occupe exclusivement de l'affichage de la colonne latérale.
 - Ces trois derniers fichiers sont appelés par le fichier **index.php** pour afficher toutes les pages du site.

La création de son propre thème

Les fichiers des thèmes

- **Les autres fichiers**
- Voici le descriptif de ces fichiers, mais cela peut varier selon la construction du thème
- **page.php** : affiche la page unique d'une page.
- **single.php** : affiche la page unique d'un article.
- **content.php** : affiche la liste des articles ou le contenu des pages ou des articles, selon le fichier qui fait appel à lui par une fonction d'inclusion. Il existe plusieurs types de fichiers "content" selon le type de contenu à afficher, comme : content-none.php, content-page.php, content-search.php, content-single.php. Dans les autres thèmes, ces fichiers peuvent se trouver directement à la racine du thème.
- **search.php** : affiche la liste des articles retournés par une recherche.



La création de son propre thème

Les fichiers des thèmes

- **archive.php** : affiche la liste des articles par mois.
- **biography.php** : permet d'afficher les informations sur un auteur.
- **404.php** : affiche la page d'erreur 404.
- **searchform.php** : affiche le formulaire de recherche. Ce fichier est appelé par d'autres fichiers à l'aide de fonctions d'inclusion.
- **comments.php** : affiche les commentaires et le formulaire des commentaires. Ce fichier est donc appelé par d'autres fichiers à l'aide de fonctions d'inclusion. Par exemple, le fichier single.php fait appel à ce fichier pour afficher les commentaires et le formulaire.
- **image.php** : affiche la page unique de l'image.



La création de son propre thème

Les fichiers des thèmes

- D'autres dossiers peuvent être présents à la racine d'un thème, en voici quelques exemples :
 - **template-parts** : contient tous les fichiers permettant d'afficher le contenu d'une page.
 - **Genericons** : c'est une bibliothèque gratuite d'icônes vectorielles basées sur des polices de caractères, qui permet d'afficher facilement des icônes sur un site Internet. Vous pouvez les mettre en forme grâce à du code CSS. L'utilisation est simple : il suffit d'inclure un code HTML dans les fichiers ou dans l'éditeur WYSIWYG en utilisant l'onglet **texte**.
- Le code suivant permet d'afficher une enveloppe devant le message "Contactez-nous !" :
 - ` ` Contactez-nous !
 - Ici, c'est l'icône genericon-mail qui est affichée, il suffit de remplacer la classe CSS par celle trouvée sur le site de Genericons : <https://genericons.com/>
- **languages** : contient les langues du thème.
- **css** ou **style** : contient les CSS du thème.



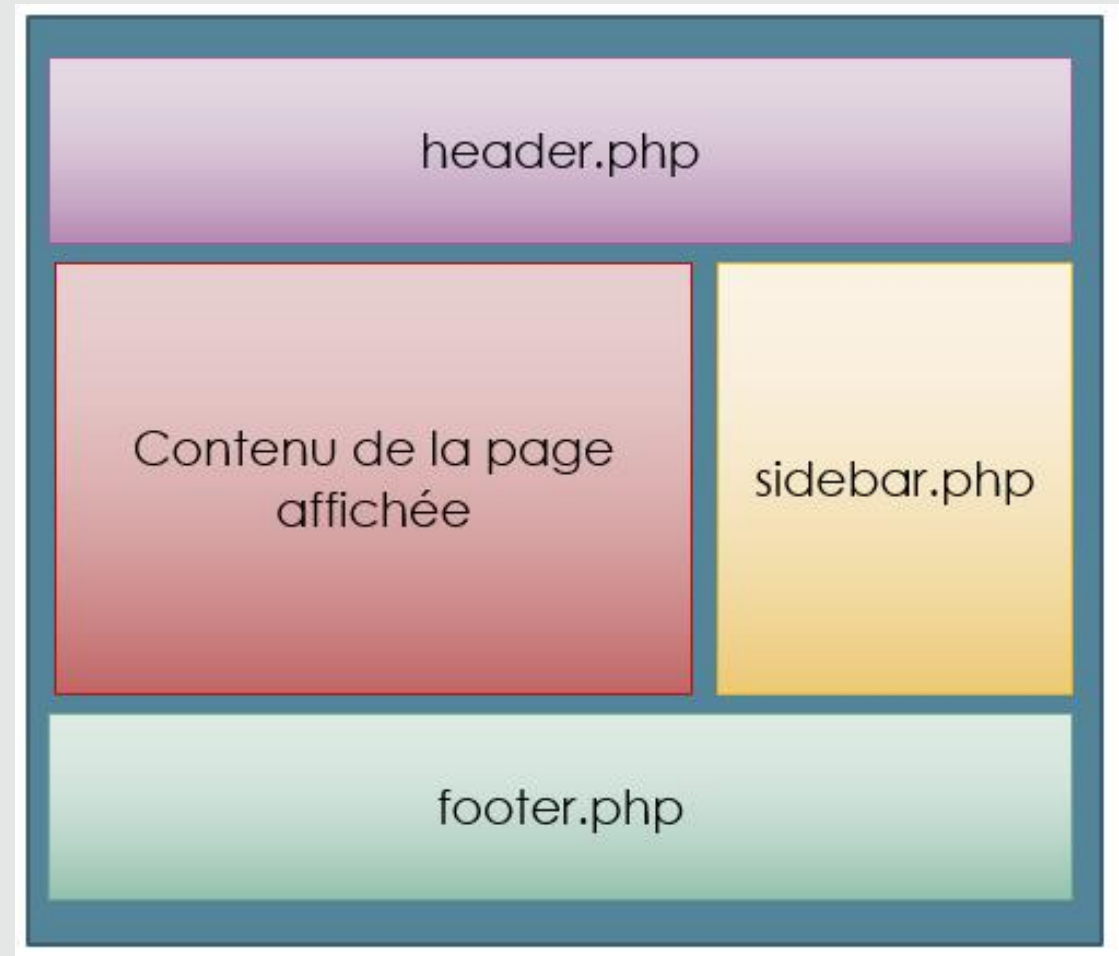
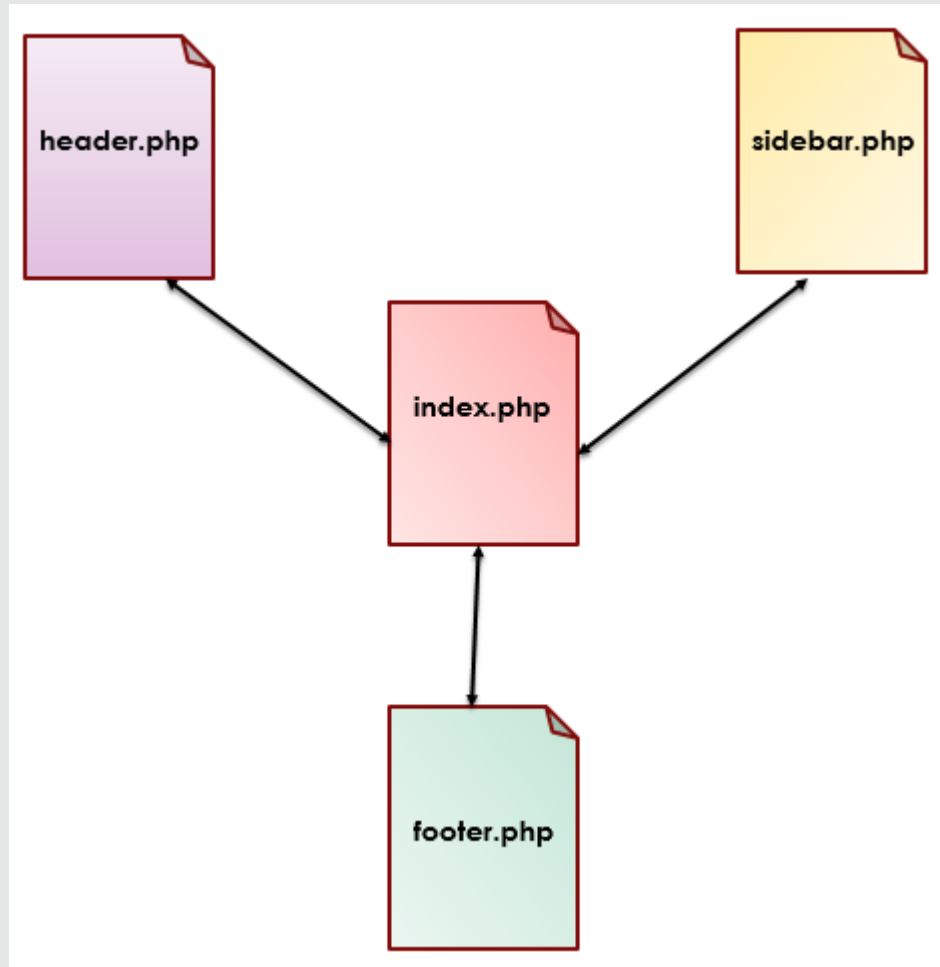
La création de son propre thème

Les fichiers des thèmes

- **js** ou **javascript** : contient les fichiers JavaScript du thème.
- **fonts** : contient des fonts (polices). Depuis CSS3, il n'est pas rare de trouver ce dossier.
- **images** ou **img** : contient les images propres au thème.
- **pages-templates** : contient des templates de page.
- **inc** ou **includes** : contient des fichiers PHP propres au thème.
- D'autres dossiers suivant le thème.

La création de son propre thème

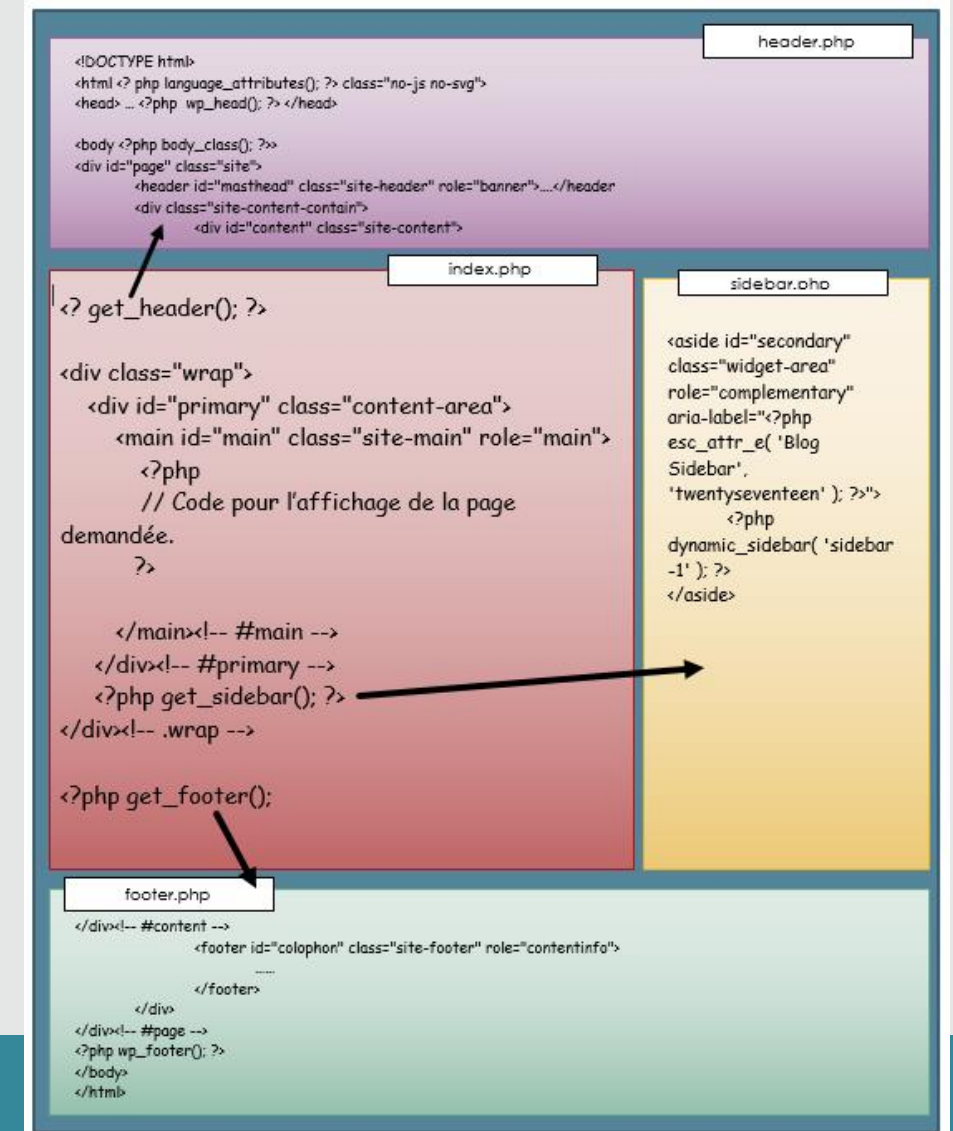
Les fichiers des thèmes



La création de son propre thème

Créer un thème WordPress : L'appel des fichiers

- Comment les fichiers constitutifs sont-ils appelés et inclus dans la page index.php ? Grâce à des fonctions propres à WordPress :
- `<? get_header(); ?>` : permet l'insertion du fichier **header.php** dans la page qui utilise cette fonction.
 - http://codex.wordpress.org/Function_Reference/get_header
- `<? get_sidebar(); ?>` : permet l'insertion du fichier **sidebar.php** dans la page qui utilise cette fonction.
 - http://codex.wordpress.org/Function_Reference/get_sidebar
- `<? get_footer(); ?>` : permet l'insertion du fichier **footer.php** dans la page qui utilise cette fonction.
 - http://codex.wordpress.org/Function_Reference/get_footer
- Structure des fichiers avec le thème **Twenty Seventeen**.



La création de son propre thème

Le code et l'affichage généré

```
index.php x codeAffichageTwentySeventeen.php x
1 <html>
2 <head>
3     <meta charset="UTF-8">
4     <meta name="viewport" content="width=device-width, initial-scale=1">
5     <title>Mon site - Un site utilisant WordPress</title>
6     <link rel="stylesheet" id="twentyseventeen-style-css"
7     href="http://localhost:8888/wordpress-theme/wp-content/
8     themes/twentyseventeen/style.css?ver=4.7.1" type="text/css" media="all">
9 </head>
10 <body class="...">
11 <div id="page" class="site">
12     <header id="masthead" class="site-header" role="banner">
13         ... Affichage de l'en-tête ...
14     </header>
15     <div class="site-content-contains">
16         <div id="content" class="site-content">
17             <div class="wrap">
18                 ... Affichage de l'en-tête de la page ...
19                 ... Affichage des articles dans la page d'accueil...
20                 ... Affichage de la sidebar ...
21             </div> // Fin de la boîte .wrap
22         </div> // Fin de la boîte #content
23         <footer id="colophon" class="site-footer"
24         role="contentinfo">
25             ...
26         </footer>
27     </div> // Fin de la boîte .site-content-contains
28 </div> // Fin de la boîte #page
29 </body>
30 </html>
```

La création de son propre thème

Le code et l'affichage généré

```
<div id="page" class="site">  
  
  <header id="masthead" class="site-header" role="banner">  
  
    <div class="custom-header">  
  
      Affichage de l'en-tête  
  
    <div class="navigation-top">  
  
      Affichage de la barre de navigation  
  
    <div class="site-content-contains">  
  
      <div id="content" class="site-content">  
  
        <div class="wrap">  
  
          <header class="entry-header">  
  
            Affichage de l'en-tête de la page  
  
          <div id="primary" class="content-area">  
  
            Affichage du contenu de la page demandée  
  
          <aside class="widget-area" role="complementary">  
  
            Affichage de la sidebar  
  
          <div id="secondary">  
  
            Affichage de la sidebar  
  
        <div id="colophon" class="site-footer" role="contentinfo">  
  
          Affichage du pied de page  
  
      </div>  
    </div>  
  </div>  
</div>
```

La création de son propre thème

La place des éléments HTML

- La place des éléments HTML

- Les balises d'ouverture des éléments de structure de page `<html>`, `<head>` et `<body>` se trouvent dans le fichier d'en-tête **header.php**.
- Vous retrouvez le même principe et la même logique avec les balises de fermeture des éléments `</div>`

(`#content`, `.site-content-contain` et `#page`) `</body>` et `</html>` qui sont placées dans le fichier **footer.php**. Ce fichier étant appelé par toutes les pages affichées dans le site.

- les pages de contenu ne se préoccupent que du contenu.
- les fichiers **header.php** et **footer.php** s'occupent de tous les éléments HTML communs à toutes les pages du site.



La création de son propre thème

L'appel aux templates

- Pour afficher les contenus des pages demandées, les thèmes peuvent utiliser des templates.
- Les templates sont des blocs de code spécialisés dans l'affichage des contenus.
 - contenu des articles,
 - le contenu des pages,
 - les articles des archives...
- Pour cela, les thèmes utilisent la fonction **get_template_part()**.
 - https://developer.wordpress.org/reference/functions/get_template_part/
 - Cette fonction, **get_template_part(string \$slug, string \$name = null)**, possède deux paramètres :
 - **\$slug** indique le chemin d'accès au dossier qui contient les templates concernés et le préfixe du nom du template. Ce chemin est indiqué depuis la racine du dossier du thème.
 - **\$name** indique le suffixe du nom du template à utiliser, sans l'extension .php.



La création de son propre thème

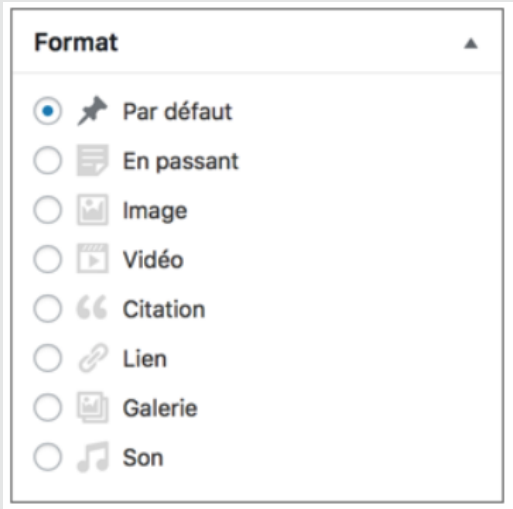
L'appel aux templates

- Voici un exemple dans le fichier **footer.php** :
 - `get_template_part('template-parts/footer/site', 'info')`
 - Le chemin d'accès au dossier des templates concernés est **template-parts/footer/** et le préfixe du template à utiliser est **site**.
 - Le suffixe du template est **info**. Donc le template à utiliser est **site-info.php** et il se trouve dans le dossier **template-parts/footer/**.
 - Le tiret entre le préfixe et le suffixe est automatiquement ajouté par la fonction.
- Dans le fichier **header.php**, nous avons l'appel au template qui gère l'affichage de la barre de navigation :
 - `get_template_part('template-parts/navigation/navigation', 'top')`
 - Le chemin d'accès au dossier est **template-parts/navigation/**,
 - le préfixe est **navigation** et le suffixe est **top**.
 - C'est donc le fichier **navigation-top.php**, dans le dossier **template-parts/navigation/**.



La création de son propre thème

L'appel aux formats d'article



```
add_theme_support.php
1 add_theme_support( 'post-formats', array(
2     'aside',
3     'image',
4     'video',
5     'quote',
6     'link',
7     'gallery',
8     'audio',
9 ) );
```

- L'appel aux formats d'article

- Appel aux templates des différents types de format des articles.
- En effet, les articles peuvent utiliser des formats qui permettent une mise en forme spécifique.
- Il faut donc que l'appel aux templates prenne en compte ces formats. Le nom des formats d'article est fixe et ne peut pas être modifié :
 - **content.php** pour le format **Par défaut**,
 - **content-aside.php** pour le format **En passant**,
 - **content-image.php** pour le format **Image**,
 - **content-video.php** pour le format **Vidéo**,
 - **content-quote.php** pour le format **Citation**,
 - **content-link.php** pour le format **Lien**,
 - **content-gallery.php** pour le format **Galerie**,
 - **content-audio.php** pour le format **Son**,
- La déclaration des formats disponibles se fait dans le fichier **functions.php** du thème

L'appel aux formats d'article

```
add_theme_support.php
1 add_theme_support( 'post-formats', array(
2     'aside',
3     'image',
4     'video',
5     'quote',
6     'link',
7     'gallery',
8     'audio',
9 ) );
```

- La déclaration des formats disponibles se fait dans le fichier **functions.php** du thème.
- Bien sûr, chaque concepteur de thème peut faire ce qu'il veut quant au nombre de format proposé.
- Pour appeler le template correspondant au bon format, la fonction **get_template_part()** utilise comme deuxième paramètre la fonction **get_post_format()**, dont voici l'URL : https://developer.wordpress.org/reference/functions/get_post_format/
- Nous avons donc cette structure imbriquée :
 - **get_template_part('template-parts/post/content', get_post_format())**
- Le chemin d'accès est **template-parts/post/**.
 - Le préfixe est **content**.
- Le suffixe est déterminé automatiquement par la fonction **get_post_format()** qui utilise le nom des formats : **aside**, **image**, **video**, **quote**...

La création de son propre thème

La hiérarchie des templates

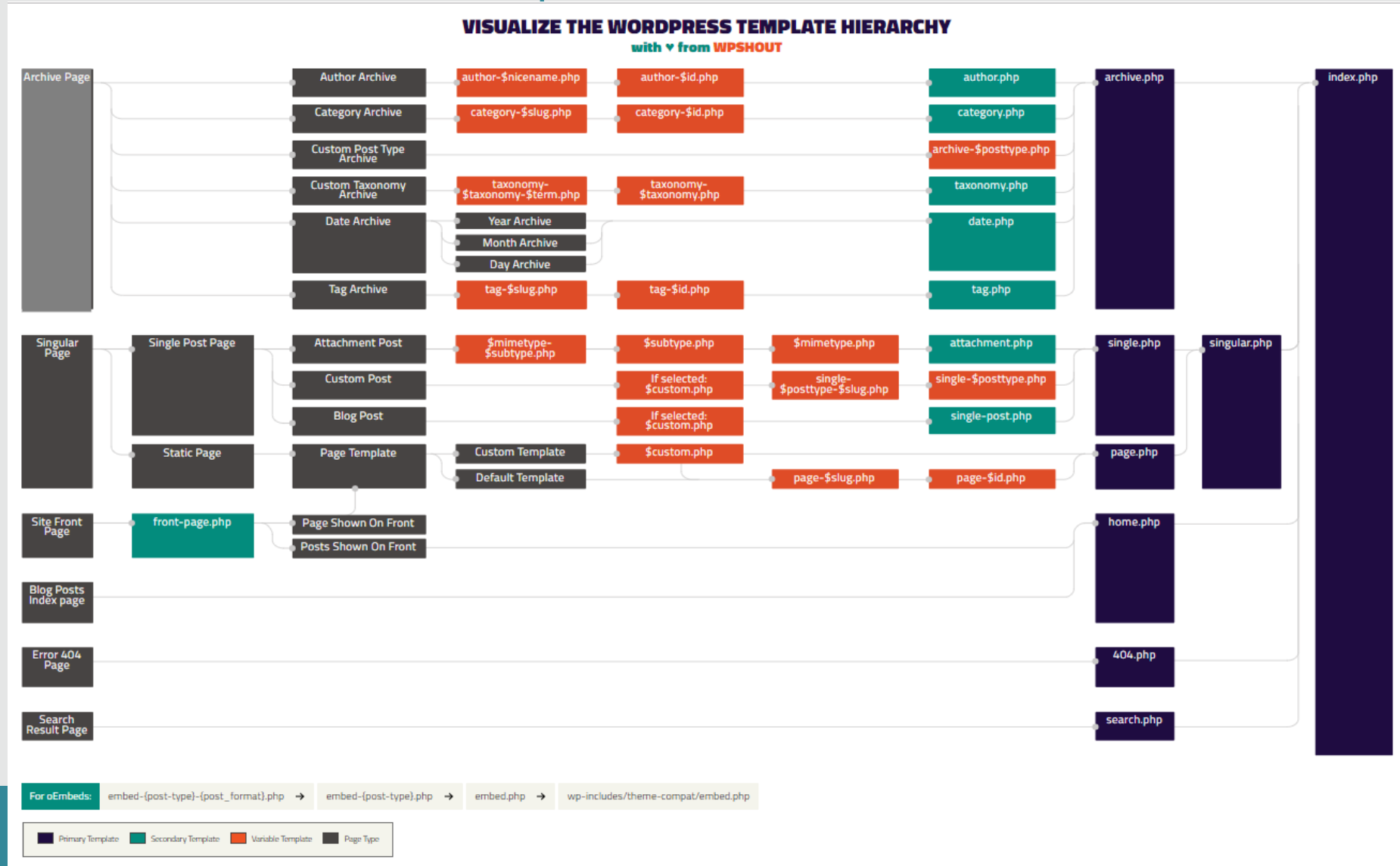
- **L'objectif**

- Les thèmes pour WordPress sont constitués de nombreux fichiers appelés **template** qui prennent en charge l'affichage des pages demandées par les visiteurs. Il n'y a pas qu'un seul fichier de mise en page pour l'ensemble du site WordPress.
- À chaque type de page du site correspond un fichier de mise en page, un template qui s'utilisent de manière précise selon une hiérarchie parfaitement définie.
- Voici le lien sur le codex de WordPress, expliquant cette hiérarchie des templates : http://codex.wordpress.org/Template_Hierarchy



La création de son propre thème

La hiérarchie des templates

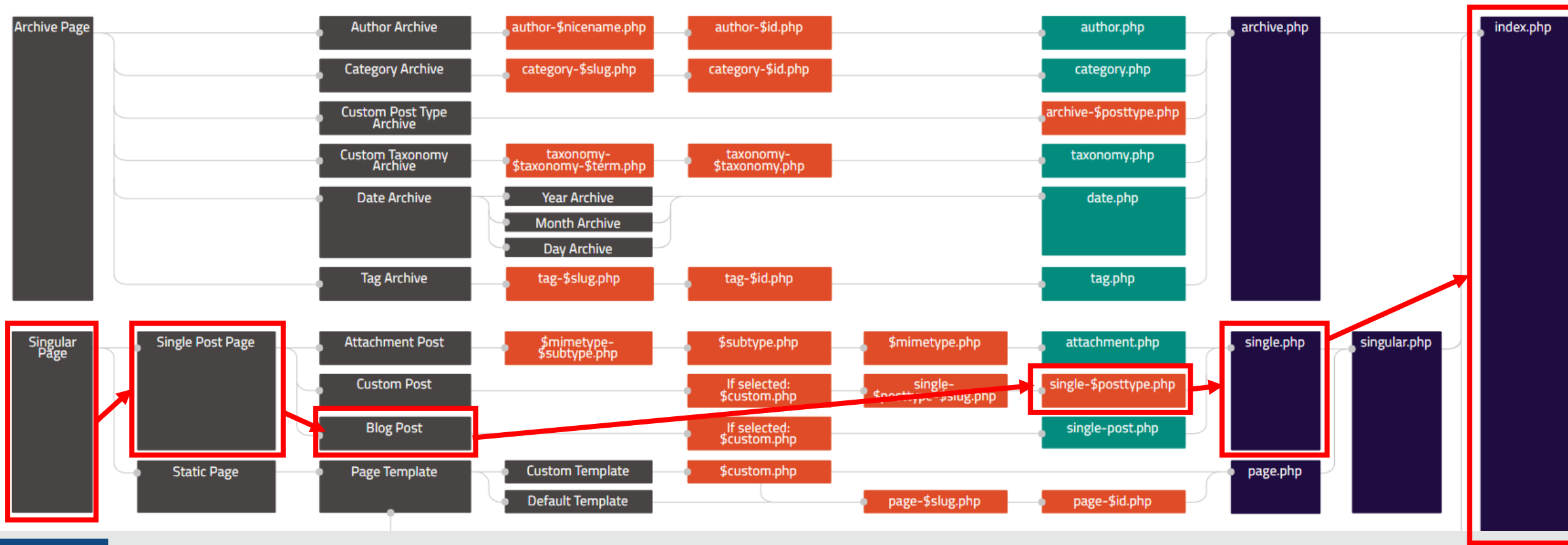


La création de son propre thème

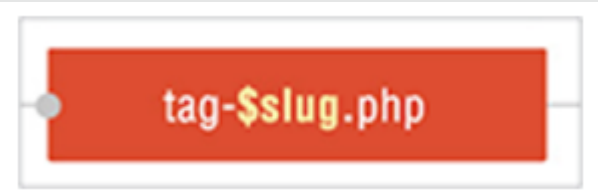
La hiérarchie des templates

VISUALIZE THE WORDPRESS TEMPLATE HIERARCHY

with ♥ from WPSHOUT



Les templates



- Dans ce schéma de la hiérarchie des templates, plusieurs types de templates existent.
- Les templates de variable
 - Les templates de variable (en rouge sur le schéma du codex) sont à créer par le concepteur du thème, lorsqu'il souhaite avoir une mise en page spécifique pour un contenu spécifique.
 - Prenons un premier exemple : le concepteur du thème veut avoir une mise en page unique pour afficher la liste des contenus rédactionnels de l'auteur **denis**. Le concepteur devra créer un template nommé **author-denis.php**. Denis étant le **nicename**, le pseudonyme dans le profil de l'utilisateur.
 - Deuxième exemple : le concepteur du thème veut avoir une mise en page unique pour l'affichage des contenus utilisant le mot-clé **voyage**, il faut qu'il crée un template nommé **tag-voyage.php**.
- Vous comprenez la règle de nommage : dans ces templates la partie variable, **\$variable**, doit utiliser comme libellé, le nom du type de contenu à afficher :
- **author-\$nicename.php**, donne **author-denis.php** pour le pseudonyme **denis**.
- **tag-\$slug.php**, donne **tag-voyage.php** pour le mot-clé **voyage**.

La création de son propre thème

Les templates

- Les templates secondaires

- Les templates secondaires (en bleu clair sur le schéma du codex) ne sont pas obligatoires et sont donc souvent absents des thèmes. Ils permettent de définir des mises en page plus spécifiques.
- Le template secondaire **author.php** permet de définir la mise en page pour l'affichage des contenus rédactionnels de tous les auteurs. Le template secondaire **tag.php** permet de définir la mise en page pour l'affichage de la liste des articles pour les mots-clés cliqués.



La création de son propre thème

Les templates

- Les templates primaires

- Les templates primaires (en bleu foncé sur le schéma du codex) ne sont pas strictement obligatoires (puisque seul le template **index.php** l'est), mais ils sont généralement tous présents dans les thèmes actuels.
 - Le template primaire **archive.php** gère l'affichage de toutes les archives (par auteurs, catégories, mots-clés et calendaires).
 - Le template **single.php** gère l'affichage des articles en page seule.
 - Le template **page.php** gère l'affichage des pages.
 - Le template **home.php** gère un des affichages possibles de la page d'accueil.
 - Le template **comments-popup.php** gère l'affichage des commentaires en page seule.
 - Le template **404.php** gère l'affichage des erreurs des pages non trouvées.
 - Le template **search.php** gère l'affichage du résultat des recherches.
- Le seul template obligatoire est **index.php**. C'est lui qui clôt tous les chemins des choix des templates.

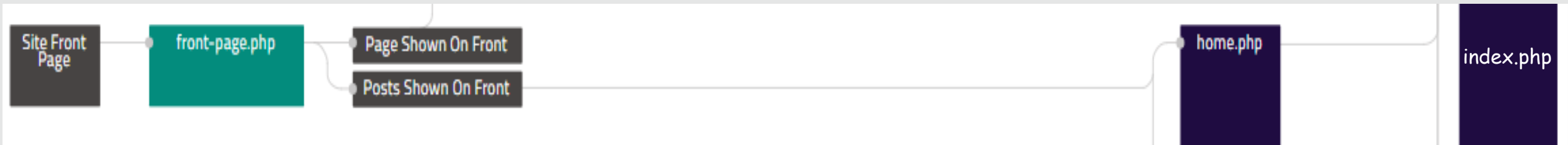


La création de son propre thème

Les templates : les pages d'accueil

- **Les pages d'accueil**

- WordPress permet de gérer plusieurs types de page d'accueil, en fonction de ce que vous souhaitez y afficher.



- La page d'accueil de type blog

- Par défaut, WordPress affiche dans la page d'accueil, la liste des derniers articles publiés. C'est le template **index.php** qui est utilisé.
- Dans le menu **Réglages**, choisissez **Lecture**. Dans les options **La page d'accueil affiche**, laissez cochée l'option **Les derniers articles**.

- La page d'accueil de type site classique

- Pour la page d'accueil, vous pouvez aussi afficher une page spécifique précédemment créée.
- Dans les mêmes réglages, choisissez l'option **une page statique**. Puis dans les deux listes déroulantes (**Page d'accueil** et **Page des articles**), choisissez la page qui doit être utilisée pour la page d'accueil et celle pour l'affichage des articles.
- Avec ce réglage, c'est le fichier **front-page.php** qui est utilisé pour l'affichage de la page d'accueil, si ce template existe bien sûr. Et lorsque vous affichez la liste des derniers articles de votre site, la partie blog de celui-ci, c'est le fichier **home.php** qui est utilisé, là encore, si ce template existe.

La création de son propre thème

Les templates : des articles

- **Les templates des articles**

- Plusieurs templates permettent de prendre en charge l'affichage des articles.

- **Les types d'affichage**

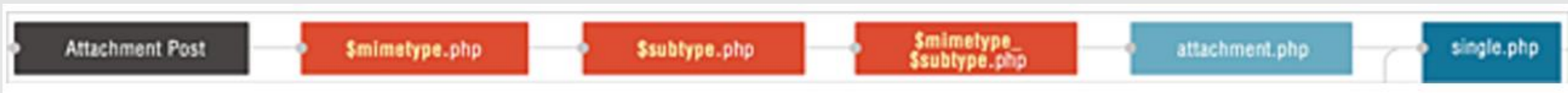
- Une fois que WordPress a déterminé que le type de page à afficher est une page seule (**Singular Page**) et qu'il s'agit d'un article (**Single Post Page**), il doit déterminer quel est le contenu à afficher.
- Nous avons trois possibilités :
 - si l'affichage concerne un fichier joint (comme les images incluses dans les articles), c'est le type **Attachment Post** qui est choisi.
 - si l'affichage concerne un contenu personnalisé (**Custom Post Type**), c'est le type **Custom Post** qui est choisi.
 - si l'affichage concerne un article, c'est le type **Blog Post** qui est choisi.



La création de son propre thème

Les templates des fichiers joints

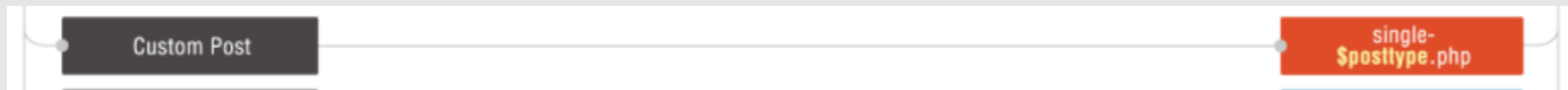
- Les fichiers joints sont par exemple les images que vous insérez dans les articles.
- Les templates peuvent être personnalisés en fonction du type MIME des fichiers joints.
- Voici le lien vers la page Wikipedia sur les types MIME : http://fr.wikipedia.org/wiki/Type_MIME
- Vous pouvez créer un template spécifique selon le type MIME du fichier. Les templates dédiés seront nommés avec la syntaxe **\$typemime.php**.
 - Si c'est une image, le template sera nommé **image.php**.
 - Si c'est un fichier bureautique, le template dédié devra être nommé **application.php**.
- Le deuxième template utilisable est spécifique au format du fichier attaché.
- Le template est nommé selon cette syntaxe : **\$subtype.php**. La variable **\$subtype** est le type MIME : JPEG, PDF, MPEG, PLAIN... Ce qui donne des templates nommés : **jpeg.php**, **pdf.php**, **mpeg.php**, **plain.php**...
- Nous pouvons aussi associer le type MIME et le format avec la syntaxe **\$typemime_\$subtype.php**. Pour les images au format JPEG, le template dédié devra être nommé **image_jpeg.php**. Pour les fichiers bureautiques au format PDF, le template dédié devra être nommé **application_pdf.php**.
- Si ces templates variables n'existent pas dans le dossier du thème, c'est le template secondaire **attachment.php** qui sera utilisé, s'il est présent bien sûr.
- Si ce dernier n'est pas présent, les templates **single.php** et **index.php** sont utilisés en dernier recours.



La création de son propre thème

Les templates des contenus personnalisés

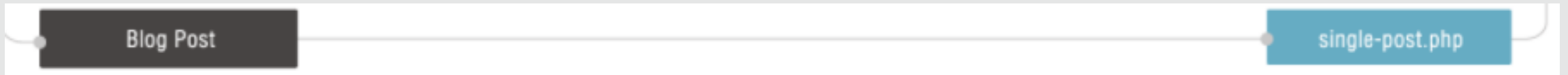
- WordPress ne propose nativement que des types de contenu de type **Article** et **Page**.
- Cependant nous pouvons créer nos propres types de contenu avec les **Custom Post Type** (http://codex.wordpress.org/Post_Types).
- Si par exemple on crée un contenu nommé **livre**, ces contenus pourront être affichés de manière spécifique avec le template utilisant la syntaxe **single-\$posttype.php**. Dans notre cas: **single-livre.php**.
- Si ce template n'est pas présent, ce sera le template **single.php** qui sera utilisé, puis **index.php** en cas d'absence de ce dernier.



La création de son propre thème

Les templates des articles

- Avec le type de contenu **Article**, c'est le template secondaire nommé **simple-post.php** qui sera utilisé. S'il n'est pas présent, ce sera **single.php**, puis **index.php** en cas d'absence de ce dernier.



La création de son propre thème

Les templates des pages

- Vous avez à votre disposition deux types de page pour gérer l'affichage des pages statiques de vos sites WordPress : les pages standards et les modèles de page.
- **Les types de page**
 - Une fois que WordPress a déterminé que la page à afficher est une page seule (**Singular Page**), qu'il s'agit d'une page statique (**Static Page**) et qu'il s'agit d'un modèle de page (**Page Template**), WordPress doit déterminer le type de template : soit c'est un template standard (**Default Template**), soit c'est un template personnalisé (**Custom Template**).

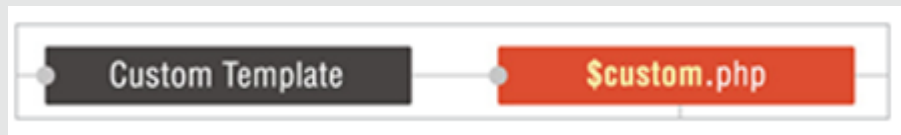


La création de son propre thème

Les templates des pages : Les modèles de page

- Les modèles de page

- WordPress permet de créer et d'utiliser des modèles de page personnalisés, afin d'avoir une mise en page et une mise en forme pour des pages spécifiques.
- Tous ces modèles sont accessibles lorsque vous créez vos pages, dans le module **Attributs de la page**, dans la liste déroulante **Modèle**.
- Attention, cela dépend du thème parfois, il n'y a pas de modèle de page statique, donc la liste déroulante **Modèle** n'est pas affichée.
- Mais dès que vous créerez des modèles de page, la liste déroulante sera affichée dans le module.
- Le nom des templates des pages est laissé libre, ils utilisent la syntaxe **\$custom.php**.



Attributs de la page

Parent

(pas de parent)

Modèle

Modèle par défaut

Ordre

0

Besoin d'aide ? Utilisez l'onglet Aide présent dans le coin supérieur droit de votre écran.

Attributs de la page

Parent

(pas de parent)

Ordre

0

Besoin d'aide ? Utilisez l'onglet Aide présent dans le coin supérieur droit de votre écran.

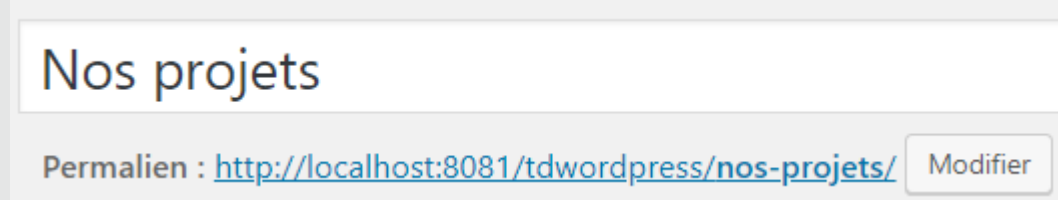
La création de son propre thème

Les templates spécifiques

- Possibilité de créer et d'utiliser des templates spécifiques pour certaines pages statiques de votre site. Ces templates variables utilisent soit le slug, soit l'identifiant de ces pages dans leur nom.
- Vous pouvez utiliser soit la syntaxe **page-\$slug.php**, soit la syntaxe **page-\$id.php**.



- Si le titre de la page est **Projets** (sachant que par défaut le titre donne le slug du contenu), le template spécifique doit s'appeler **page-projets.php**.



- Si l'identifiant unique de la page est **23**, le template spécifique doit s'appeler **page-23.php**.
- **Les templates standards**
 - Si vous n'avez pas de modèle de page et si vous n'avez pas créé de template spécifique, c'est le template **page.php** qui est utilisé.
 - Si ce dernier n'est pas présent, c'est le template **index.php** qui est utilisé.

La création de son propre thème

Les templates des archives

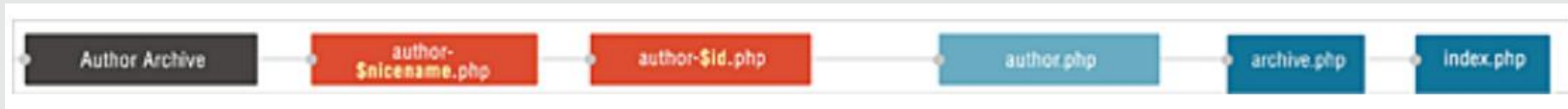
- WordPress nous permet d'utiliser des archives selon différents types :
 - archive par catégorie d'article,
 - archive par mot-clé,
 - archive par auteur et
 - archive calendaire.
- Nous allons pouvoir utiliser des templates spécifiques pour chaque type d'archive.
 - **Les types de page**
 - Une fois que WordPress a déterminé qu'il doit afficher une page d'archive (**Archive Page**), il détermine le type d'archive :
 - archive des auteurs : **Author Archive**,
 - archive des catégories des articles : **Category Archive**,
 - archive pour les types de contenus personnalisés : **Custom Post Type Archive**,
 - archive pour la taxinomie personnalisée : **Custom Taxonomy Archive**,
 - archive calendaire : **Date Archive**,
 - archive pour les mots-clés : **Tag Archive**.



La création de son propre thème

Les templates des archives des auteurs

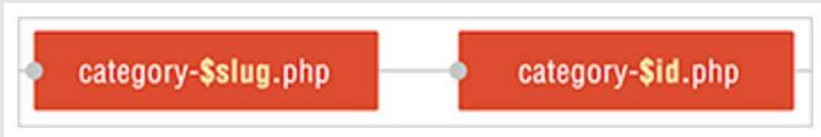
- Lorsque le visiteur clique sur le nom d'un auteur d'un contenu, il affiche la liste des contenus rédigés par cet auteur et affiche éventuellement des renseignements du profil de ce rédacteur.
- Dans ce cas, ce sont les templates des archives des auteurs qui sont utilisés.
 - Vous pouvez créer et utiliser des templates spécifiques pour tel ou tel auteur.
 - Pour le nom de ces templates, utilisez la syntaxe **author-\$nickname.php** ou **author-\$id.php**.
 - La variable **\$nickname** correspond au pseudonyme de l'utilisateur et **\$id** correspond à son identifiant unique.
 - Si le pseudonyme de l'auteur est **denis**, son template spécifique devra être nommé **author-denis.php**. Si son identifiant est **42**, son template spécifique devra être nommé **author-42.php**.
- S'il n'y a pas de template spécifique présent, c'est le template secondaire **author.php** qui est utilisé. Ensuite, c'est **archive.php**, puis comme toujours **index.php** qui clôt le choix du template à utiliser.



La création de son propre thème

Les templates des archives des catégories

- Lorsque vous avez mis en place une taxinomie des articles basée sur l'utilisation de catégories, les visiteurs peuvent cliquer sur le nom d'une de ces catégories pour afficher la liste de tous les articles qui sont classés dans celle-ci.



- Vous pouvez créer et utiliser un template spécifique pour chaque catégorie voulue. Le nom de ces templates doit utiliser la syntaxe **category-\$slug.php** ou **category-\$id.php**.

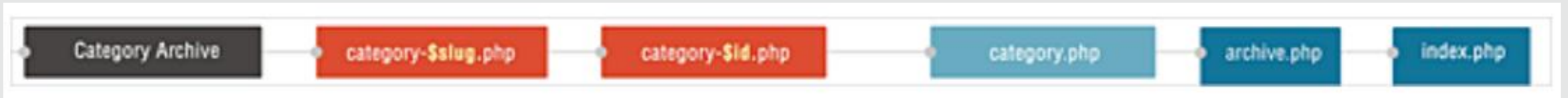
Nom	<input type="text" value="france"/>
	<small>Ce nom est utilisé un peu partout sur votre site.</small>
Identifiant	<input type="text" value="france"/>
	<small>L'identifiant est la version normalisée du nom. Il ne peut contenir que des lettres minuscules non accentuées, des chiffres et des tirets.</small>

- La variable **\$slug** correspond au nom de la catégorie. Par défaut le nom de la catégorie donne le slug de celle-ci. C'est le champ **Identifiant** dans l'interface de WordPress.

La création de son propre thème

Les templates des archives des catégories

- La variable **\$id** correspond à l'identifiant unique de la catégorie.
- Si le slug de la catégorie est **france**, son template spécifique sera nommé **category-france.php**.
- Si son identifiant est **32**, son template spécifique sera nommé **category-32.php**.
- Si aucun template spécifique n'est présent, c'est le template secondaire **category.php** qui est utilisé.
- Ensuite, c'est **archive.php**, puis comme toujours **index.php** qui clôt le choix du template à utiliser.



La création de son propre thème

Les templates des archives des types de contenu personnalisé

- Lorsque vous créez de nouveaux types de contenu personnalisé (**Custom Post Type**), ces derniers peuvent fonctionner comme les articles.
- Les visiteurs pourront donc afficher leurs archives.
- Le template de ces archives doit être nommé selon cette syntaxe : **archive-\$posttype.php**, où la variable **\$posttype** est le nom du type de contenu personnalisé.
- Si le nouveau type de contenu personnalisé est **livres**, le template des archives doit être nommé **archive-livres.php**.
- Comme toujours, nous avons ensuite **archive.php** et **index.php**.



La création de son propre thème

Les templates de la taxinomie des types de contenu personnalisé

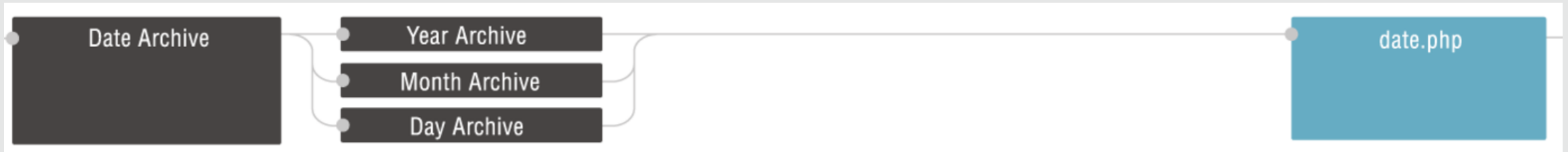


- Lorsque vous créez de nouveaux types de contenu personnalisé (**Custom Post Type**), vous pouvez leur associer une taxinomie, comme les catégories pour les articles.
- Si vous souhaitez créer des templates spécifiques pour cette taxinomie personnalisée, vous devez nommer ces templates avec cette syntaxe : **taxonomy-\$taxonomy-\$term.php** ou **taxonomy-\$taxonomy.php**.
- La variable **\$taxonomy** indique le nom de la taxinomie. Si nous avons une taxinomie nommée **genre**, le template sera nommé **taxonomy-genre.php**.
- La variable **\$term** indique une valeur de cette taxinomie. Si nous avons une valeur nommée **roman**, le template sera nommé **taxonomy-genre-roman.php**.
- S'il n'y a pas de template spécifique, c'est le template secondaire **taxonomy.php** qui est utilisé. Comme toujours, nous avons ensuite **archive.php** et **index.php**.

La création de son propre thème

Les templates des archives calendaires / mots-clés

- Tous les articles de vos sites WordPress sont classés de manière calendaire : par année, mois et jour.
 - Vous allez pouvoir utiliser le template **date.php** pour personnaliser cet affichage. Comme toujours, nous avons ensuite l'utilisation des templates **archive.php** et **index.php**.

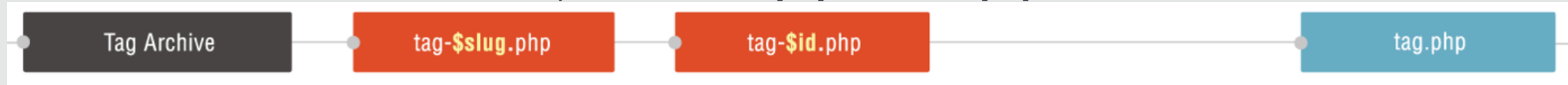


La création de son propre thème

Les templates des archives calendaires / mots-clés

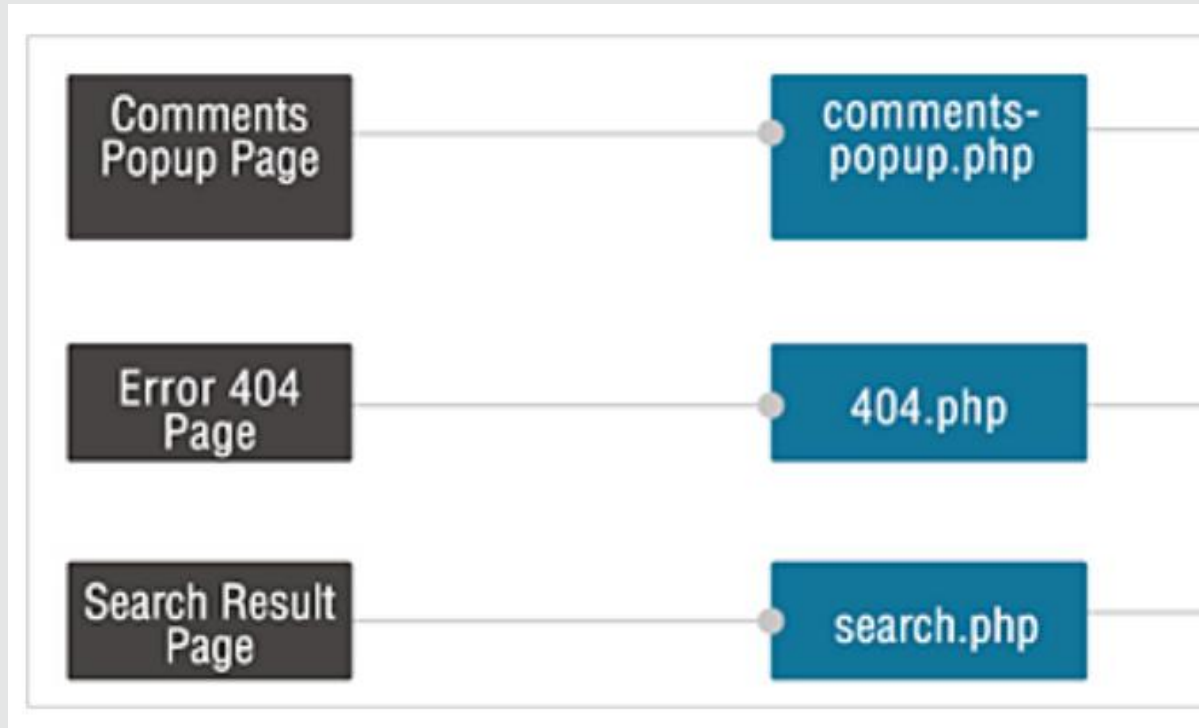
- **Les templates des mots-clés**

- Vous avez la possibilité d'associer des mots-clés à vos articles.
- Lorsqu'un visiteur clique sur l'un de ces mots-clés, il affiche tous les contenus utilisant ce mot-clé.
- Une fois que WordPress a déterminé que le type de page est celui d'un mot-clé (type **Tag Archive**), vous pouvez personnaliser les templates utilisés pour l'affichage de la liste des articles contenant le mot-clé cliqué.
- Le template **tag- $\$slug$.php** permet l'affichage de la page d'un mot-clé spécifié.
- La variable **$\$slug$** est le nom du mot-clé. Par défaut, le slug reprend le libellé du mot-clé. Dans l'interface de WordPress, le slug est indiqué dans le champ **Identifiant** dans la gestion des mots-clés.
- Si le mot-clé est **voyage**, le template doit s'appeler **tag-voyage.php**.
- Si l'identifiant du mot-clé est **12**, son template spécifique devra être nommé **tag-12.php**.
- S'il n'y a pas de template spécifique, c'est le template secondaire **tag.php** qui est utilisé. Comme toujours, nous avons ensuite l'utilisation des templates **archive.php** et **index.php**.



La création de son propre thème

Les autres templates

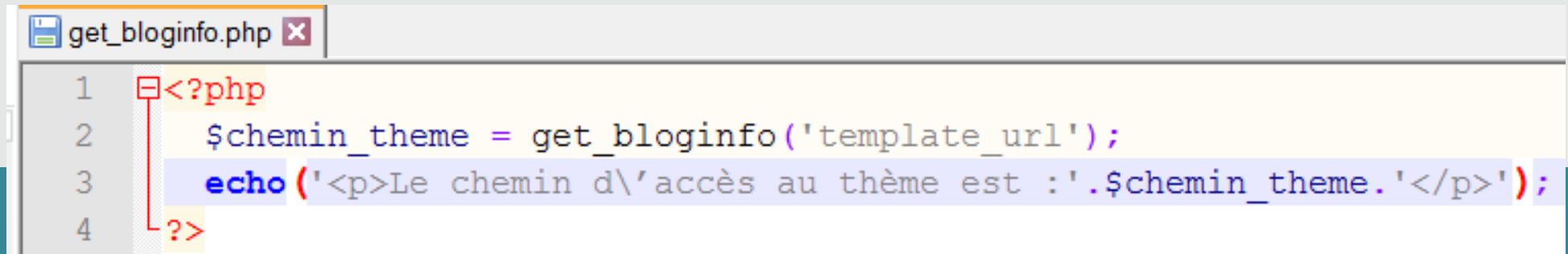


- Vous avez à votre disposition trois autres types de page :
- les pages de commentaires en nouvelle fenêtre : **Comments Popup Page**,
- la page d'erreur pour les pages non trouvées : **Error 404 Page**,
- la page des résultats d'une recherche : **Search Result Page**.
- Chacun de ces types de page possède son template : **comments-popup.php**, **404.php** et **search.php**.

La création de son propre thème

Les marqueurs et les fonctions

- Pour afficher les contenus de votre site WordPress, les templates utilisent des marqueurs de modèle, des **template tags** en anglais.
- Les marqueurs sont des fonctions PHP spécialement dédiées à WordPress qui vont chercher des informations dans la base de données pour permettre l'affichage de leurs valeurs dans les templates.
 - Par exemple, le marqueur `bloginfo('url')` permet d'afficher l'URL de votre site.
 - Le code pourrait être celui-ci : `<p><?php bloginfo('url'); ?></p>`
 - Le template va afficher dans un paragraphe `<p>`, l'URL de votre site.
- Les fonctions de référence de WordPress (**functions reference** en anglais) ressemble aux marqueurs.
- Cependant, les fonctions n'affichent rien, elles renvoient des valeurs que vous pouvez stocker dans des variables PHP. Ensuite, vous pouvez effectuer un traitement sur ces variables.
- Par exemple, la fonction `get_bloginfo('template_url')` va renvoyer le chemin d'accès au thème actif.
- Cet exemple va permettre de stocker le chemin d'accès du thème actif dans une variable nommée `$chemin_theme`. Ensuite, nous allons afficher la valeur de cette variable dans un paragraphe `<p>` à l'aide de la fonction PHP `echo()` :



```
get_bloginfo.php
1  <?php
2      $chemin_theme = get_bloginfo('template_url');
3      echo ('<p>Le chemin d\'accès au thème est : '.$chemin_theme.'</p>');
4  ?>
```


La création de son propre thème

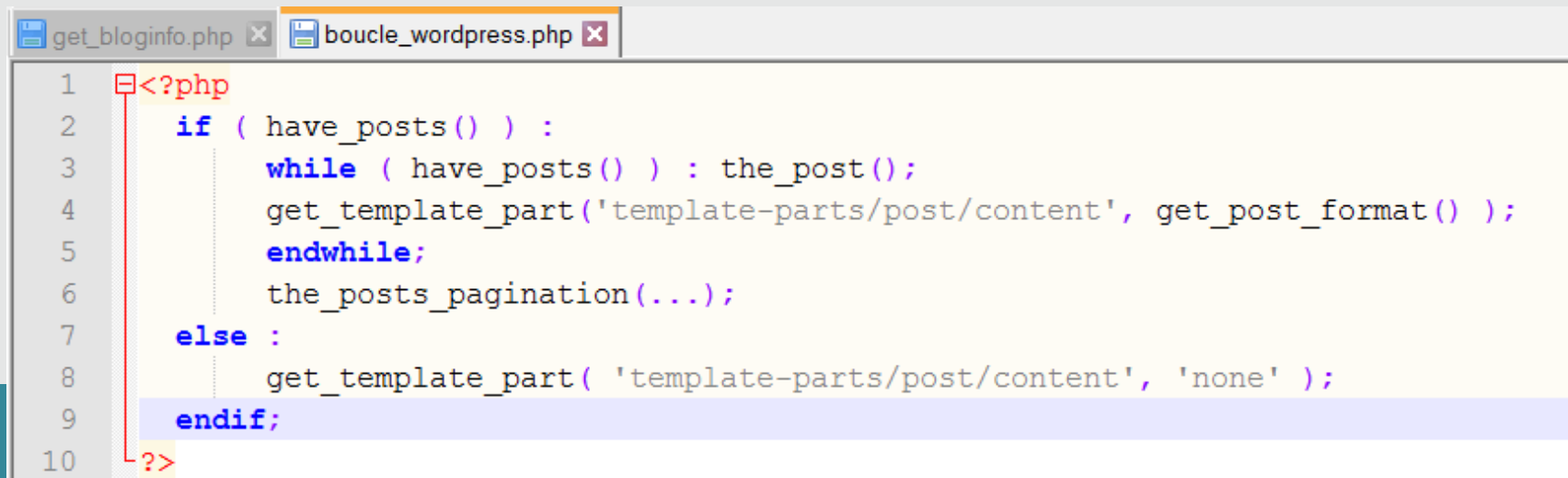
Les marqueurs et les fonctions

- Les **marqueurs** sont principalement destinés à la création des **thèmes**.
- Les **fonctions** sont principalement destinées à la création de **plug-ins** et d'applications personnalisées.
- Les marqueurs et les fonctions se trouvent dans les thèmes.
- Les marqueurs sont des fonctions de référence.
- Documentation technique sur les marqueurs et les fonctions à cette URL : <https://codex.wordpress.org>.

La création de son propre thème

La boucle WordPress

- Dans les thèmes, => la "fameuse" boucle WordPress (*loop* en anglais) qui est le cœur de l'affichage des sites WordPress.
- C'est par cette boucle **while()** que sont affichés les contenus rédactionnels dans toutes les pages de vos sites WordPress.
- La boucle va interroger la base de données pour savoir s'il y a des données à afficher.
- Si la réponse est oui, la boucle va afficher les données que nous lui demandons d'afficher.
- Voici la boucle standard simplifiée du thème **Twenty Seventeen** :



```
get_bloginfo.php x boucle_wordpress.php x
1 <?php
2     if ( have_posts() ) :
3         while ( have_posts() ) : the_post();
4             get_template_part('template-parts/post/content', get_post_format() );
5         endwhile;
6         the_posts_pagination(...);
7     else :
8         get_template_part( 'template-parts/post/content', 'none' );
9     endif;
10 ?>
```

Créer de nouvelles boucles

- Les objectifs

- La boucle standard de WordPress permet d'afficher les contenus dans toutes les pages de votre site.
- Dans la page d'accueil, la boucle permet d'afficher la liste des X derniers articles créés, avec une configuration standard de type blog
- Pour personnaliser l'affichage en page d'accueil, vous pouvez souhaiter afficher en plus d'autres contenus selon d'autres critères.
 - Par exemple, vous souhaiteriez afficher les X derniers articles d'une catégorie donnée, ou d'un auteur précis, ou d'un mois spécifié. Vous pouvez aussi vouloir afficher les articles d'un type de contenu personnalisé (*Custom Post Type*).
- Si tel est votre objectif, vous devez alors créer une deuxième boucle WordPress à l'aide de la fonction **WP_Query()**, dont voici l'URL de référence sur le Codex de WordPress : https://codex.wordpress.org/Class_Reference/WP_Query.

La création de son propre thème

Créer une boucle sur une catégorie

- 1^{er} exemple :
- Nous allons créer une boucle sur une catégorie donnée.
- La première étape consiste à connaître l'identifiant de la catégorie voulue.
- Pour ce faire vous pouvez utiliser un plug-in comme **Catch IDs** (<https://wordpress.org/plugins/catch-ids/>)
- Ou repérer cet identifiant au survol de la catégorie, dans la barre d'état de l'administration de votre site.

La création de son propre thème

Créer une boucle sur une catégorie

- Dans cet exemple, l'identifiant de la catégorie ciblée est 3.

Actions groupées ▾		Appliquer			9 éléments
<input type="checkbox"/>	ID	Nom	Description	Slug	Total
<input type="checkbox"/>	3	Afrique	Pour tous les voyages en Afrique	voyage-afrique	0
<input type="checkbox"/>	8	— Algérie	Pour tous les voyages en Algérie	voyage-algerie	0
<input type="checkbox"/>	6	— Maroc	Pour tous les voyages au Maroc	voyage-maroc	1

- Nous allons afficher le résultat de cette nouvelle boucle dans la page d'accueil. C'est donc dans le fichier **index.php** que nous allons insérer ce code, sous la boucle principale.

La création de son propre thème

Créer une boucle sur une catégorie

- La première étape consiste à définir les arguments de la nouvelle requête, avec les critères de sélection.
- Dans cet exemple, les critères seront d'afficher les contenus de type **Article** de la catégorie **Voyage** dont l'identifiant est **2** et d'afficher les **10** derniers.
- Voici la déclaration de cette requête :

```
requete_boucle_voyage.php x
1 // Définition des arguments de la requête
2 $args_voyage=array(
3     'post_type' => 'post',
4     'cat' => 2,
5     'posts_per_page' => 10,
6 );
```

La création de son propre thème

Créer une boucle sur une catégorie

- Ensuite, dans la deuxième étape, nous allons définir une nouvelle requête :

```
requete_voyage.php x
1 // Définition de la nouvelle requête
2 $query_voyage = new WP_Query($args_voyage);
```

- La troisième étape consiste à créer une boucle WordPress avec cette nouvelle requête :

```
4 //Exécution de la boucle avec la nouvelle requête
5 if($query_voyage->have_posts()) : while ($query_voyage->
6 have_posts() ) : $query_voyage->the_post();
7 the_title('<h4><a href="'.get_permalink().'">', '</a></h4>');
8 endwhile;
9 endif;
```

- Pour terminer, la dernière étape permet de réinitialiser la requête principale afin de remettre à zéro le compteur de contenu post :

```
11 // Réinitialisation de la requête principale
12 wp_reset_postdata();
```

- https://developer.wordpress.org/reference/functions/wp_reset_postdata/

La création de son propre thème

Créer une boucle sur un mois

- Dans cet exemple, nous allons afficher les 10 derniers articles du mois d'août.
- Voici la déclaration du tableau d'arguments :

```
requete_voyage.php x boucle_sur_un_mois.php x
1 //paramètre monthnum pour définir le numéro du mois d'août : 8.
2 $args_aout=array(
3     'post_type' => 'post',
4     'monthnum' => 8,
5     'posts_per_page' => 10,
6 );
7
8 //nouvelle requête :
9 $query_aout = new WP_Query($args_aout);
10
11 //Dans cette boucle, nous affichons le titre des articles avec leur lien et la date de publication, the_date().
12 if($query_aout->have_posts()) : while ($query_aout->
13 have_posts() ) : $query_aout->the_post();
14     the_title('<h4><a href="'.get_permalink().'">', '</a></h4>');
15     the_date();
16 endwhile;
17 endif;
18
19 //Réinitialisation de la requête
20 wp_reset_postdata();
```


La création de son propre thème

Créer une boucle sur un auteur

```
boucle_auteur.php
1 // Définition des arguments de la requête
2 $args_christine=array(
3     'post_type' => 'post',
4     'author' => 2,
5     'posts_per_page' => 10,
6 );
7
8 // Définition de la requête
9 $query_christine = new WP_Query($args_christine);
10
11 // Exécution de la boucle avec la nouvelle requête
12 if($query_christine->have_posts()) : while ($query_christine->have_posts() ) : $query_christine->the_post();
13     the_title('<h4><a href="'.get_permalink().'">', '</a></h4>');
14 endwhile;
15 endif;
16
17 // Réinitialisation de la requête principale
18 wp_reset_postdata();
19
20 // Affichage de l'URL de sa page d'auteur
21 $url_auteur = get_author_posts_url(2);
22 echo ('<p>URL de la page de l\'auteur : <a href="'. $url_auteur. '">'.get_the_author_meta('first_name',2). '</a>.</p>');
```

- Vous pouvez aussi afficher tous les articles d'un auteur donné. Comme pour les catégories, il faut connaître l'identifiant de l'auteur concerné. Dans cet exemple, l'identifiant est 2.
- Voici immédiatement le code complet de cet exemple qui utilise le paramètre 'author' => 2 pour identifier l'auteur :

Créer une boucle multicritère

- Dans la déclaration du tableau des arguments de la requête, vous pouvez indiquer tous les critères de recherche que vous voulez.
- Vous pouvez spécifier des critères sur les auteurs, les catégories, les étiquettes, les types de contenu (article ou page), les droits des utilisateurs, les dates de publication, les statuts des publication...
- Reportez-vous à la page du Codex pour visualiser tous ces critères : https://codex.wordpress.org/Class_Reference/WP_Query
- Voici un exemple d'un tableau d'arguments portant sur le type de contenu, la catégorie, l'auteur et la date de publication :

```
boucle_multicritere.php x
1  $args = array(
2      'post_type' => 'post',
3      'cat' => 6,
4      'author' => 2,
5      'monthnum' => 5,
6      'posts_per_page' => 10,
7  );
```

La création de son propre thème

Créer une boucle sur un type de contenu

- WordPress propose de manière standard que deux types de contenu : les articles et les pages.
- À l'aide d'un plug-in ou directement en mode code dans le fichier **functions.php**, vous pouvez créer d'autres types de contenu (**Custom Post Type**) et leur taxinomie associée.
- Dans cet exemple, le nouveau type de contenu est nommé livre.
- Voici la déclaration des arguments :

```
$args_livre=array(  
    'post_type' => 'livre',  
    'posts_per_page' => 5,  
);
```

- C'est avec le paramètre `post_type` que nous spécifions quel est le type de contenu sur lequel nous recherchons des contenus. Dans cet exemple, c'est **livre**.

La création de son propre thème

Les inclusions des fichiers et des templates

- Insérer les fichiers de structure
 - Les thèmes WordPress sont constitués de plusieurs fichiers de structure qu'il faut associer afin de constituer l'affichage complet des pages.
 - De manière usuelle nous avons quatre fichiers de structure :
 - la page principale : **index.php** qui est le fichier maître, celui qui assemble l'ensemble des autres fichiers, des autres templates.
 - l'en-tête des pages est construit avec le fichier **header.php**.
 - la colonne latérale avec le fichier **sidebar.php**.
 - le pied de page avec le fichier **footer.php**.
- Depuis le fichier **index.php** (et d'autres fichiers de structure, comme **archive.php**, **page.php**...), pour intégrer ces trois fichiers, WordPress utilise trois fonctions dédiées :
 - **get_header()** permet d'intégrer l'en-tête, le fichier **header.php**.
 - https://developer.wordpress.org/reference/functions/get_header/
 - **get_sidebar()** permet d'intégrer la colonne latérale, le fichier **sidebar.php**.
 - https://developer.wordpress.org/reference/functions/get_sidebar/
 - **get_footer()** permet d'intégrer le pied de page, le fichier **footer.php**.
 - https://developer.wordpress.org/reference/functions/get_footer/

Les inclusions des fichiers et des templates

- Insérer les templates

- Pour afficher les contenus des pages demandées, les thèmes peuvent utiliser des templates qui sont appelés avec la fonction `get_template_part()`
(https://developer.wordpress.org/reference/functions/get_template_part/)
- Cette fonction, `get_template_part(string $slug, string $name = null)`, possède deux paramètres :
 - **\$slug** indique le chemin d'accès au dossier qui contient les templates concernés et le préfixe du nom du template. Ce chemin est indiqué depuis la racine du dossier du thème.
 - **\$name** indique le suffixe du nom du template à utiliser, sans l'extension **.php**.
- Voici un exemple dans le fichier **footer.php**, avec le thème **Twenty Seventeen** :
 - `get_template_part('template-parts/footer/site', 'info');`
 - Le chemin d'accès au dossier des templates concernés est **template-parts/footer/** et le préfixe du template à utiliser est **site**.
 - Le suffixe du template est **info**.
 - Donc le template à utiliser est **site-info.php** et il se trouve dans le dossier **template-parts/footer/**.
 - Le tiret entre le préfixe et le suffixe est automatiquement ajouté par la fonction.

La création de son propre thème

Les inclusions des fichiers et des templates

- **L'appel aux formats d'article**

- Les articles peuvent utiliser des formats qui permettent de concevoir une mise en forme spécifique afin de personnaliser l'affichage selon le type de contenu de l'article.
- L'appel aux templates doit tenir compte de ces formats.
- Le nombre de formats est fixe, comme leur nom qui ne peut pas être modifié :
 - **content.php** pour le format **Par défaut**,
 - **content-aside.php** pour le format **En passant**,
 - **content-image.php** pour le format **Image**,
 - **content-video.php** pour le format **Vidéo**,
 - **content-quote.php** pour le format **Citation**,
 - **content-link.php** pour le format **Lien**,
 - **content-gallery.php** pour le format **Galerie**,
 - **content-audio.php** pour le format **Son**.

```
add_theme_support( 'post-formats', array(  
    'aside',  
    'image',  
    'video',  
    'quote',  
    'link',  
    'gallery',  
    'audio',  
) );
```

La création de son propre thème

Les inclusions des fichiers et des templates

```
add_theme_support( 'post-formats', array(
    'aside',
    'image',
    'video',
    'quote',
    'link',
    'gallery',
    'audio',
) );
```

- La déclaration des formats disponibles se fait dans le fichier **functions.php** du thème :
- Pour appeler le template correspondant au bon format, la fonction **get_template_part()** utilise comme deuxième paramètre la fonction **get_post_format()**
- <https://developer.wordpress.org/reference/functions/get-template-part/> et [get post format/](https://developer.wordpress.org/reference/functions/get-post-format/).
- Avec le thème **Twenty Seventeen**, nous avons donc cette structure imbriquée : **get_template_part('template-parts/post/content', get_post_format())**.
- Le chemin d'accès est **template-parts/post/**.
- Le préfixe est **content**.
- Le suffixe est déterminé automatiquement par la fonction **get_post_format()** qui utilise le nom des formats : **aside, image, video, quote...**

La création de son propre thème

Les marqueurs de site

- Les marqueurs de site permettent d'afficher toute une série d'informations concernant les paramètres du site :
 - le titre du site, son slogan, son URL, le chemin d'accès aux fichiers CSS...
 - C'est le marqueur **bloginfo(\$show)** qui permet d'afficher les informations voulues par l'intermédiaire du paramètre \$show.
 - <https://developer.wordpress.org/reference/functions/bloginfo/>.
- Voici des exemples d'utilisation avec différents arguments :
 - **bloginfo('name')** permet d'afficher le titre du site tel qu'il est indiqué dans **Réglages - Général**, dans le champ **Titre du site**.
 - **bloginfo('description')** permet d'afficher le slogan du site tel qu'il est indiqué dans **Réglages - Général**, dans le champ **Slogan**.
 - **bloginfo('url')** permet d'obtenir l'URL du site actif, tel qu'il est indiqué dans **Réglages - Général**, dans le champ **Adresse web de WordPress (URL)**.
 - **bloginfo('admin_email')** permet d'afficher l'adresse e-mail de contact du site, tel qu'il est indiqué dans **Réglages - Général**, dans le champ **Adresse de messagerie**.
 - **bloginfo('charset')** permet d'obtenir l'encodage des caractères utilisé dans le site.
 - **bloginfo('stylesheet_url')** permet d'obtenir le chemin d'accès au fichier standard des styles CSS, **style.css**.
 - **bloginfo('template_url')** permet d'obtenir le chemin d'accès au dossier racine du thème actif.

La création de son propre thème

Les marqueurs de site

- Ces marqueurs affichent directement la valeur voulue, sans ajouter de code HTML.
 - Par exemple le code `<?php echo(bloginfo('admin_email')); ?>`
 - affiche directement l'adresse mail de contact.
- Si vous souhaitez afficher cette adresse mail avec du code HTML et CSS pour une éventuelle mise en forme, il est plus pratique d'utiliser la fonction `get_bloginfo('admin_email')` dans une variable :

```
$adresseMailContact = get_bloginfo('admin_email');  
echo ('<p style="adresse-mail-contact">L\'adresse mail de contact:'. $adresseMailContact. '</p>');
```

L'adresse mail de contact:dsanchez@eni-ecole.fr

- Vous pourrez ensuite créer une règle CSS `.adresse-mail-contact` pour appliquer une mise en forme.



La création de son propre thème

Les injections de code

- La fonction **wp_head()**, généralement placée dans l'élément HTML <head>, permet d'y injecter du code.
 - Le code en question peut être des liaisons à des fichiers CSS, JavaScript, des scripts, des règles CSS...
 - https://developer.wordpress.org/reference/functions/wp_head/.
- La fonction **wp_footer()**, généralement placée avant la balise de fermeture de l'élément <body> fonctionne sur le même principe.
 - Le code injecté sera exécuté en toute fin de chargement de la page.
 - https://developer.wordpress.org/reference/functions/wp_footer/.
- Ces deux fonctions ne font qu'insérer le code injecté dans la page HTML.
- C'est dans le fichier **functions.php** qu'est indiqué le code à injecter à l'aide, par exemple, de la fonction **wp_enqueue_scripts()**.

Les fonctions pour les liens

- Les liens dans WordPress

- Dans tout site web, le visiteur doit pouvoir afficher le contenu qu'il souhaite par l'intermédiaire de liens.
- WordPress ne déroge pas à la règle et nous propose des marqueurs et des fonctions pour créer plusieurs types de liens.

- Le lien vers la page d'accueil

- Pour revenir à la page d'accueil, c'est la fonction `home_url()` qu'il faut utiliser.
- https://developer.wordpress.org/reference/functions/home_url/

- Cette fonction propose deux paramètres : `home_url(string $path = "", string|null $scheme = null)`.

- l'argument **\$path** permet de préciser le caractère de fin de l'URL du site.
 - `home_url()` donnera `http://www.mon-site`.
 - `home_url('/')` donnera `http://www.mon-site/`.
- l'argument **\$scheme** permet de préciser le protocole : 'http', 'https' ou 'relative'.

- Cette fonction `home_url()` s'utilise avec la fonction `esc_url()` qui permet de "nettoyer" l'URL des caractères interdits.

- https://developer.wordpress.org/reference/functions/esc_url/.

- Ces deux fonctions imbriquées s'utilisent dans un lien `<a>`.

- Voici la syntaxe usuelle utilisée : `<a href="<?php echo esc_url(home_url('/')); ?>"> <?php bloginfo('name'); ?> `.

La création de son propre thème

Les URL du site

- Avec une installation standard de WordPress, le dossier racine du site publié contient tous les fichiers de WordPress.
- Dans ce cas, dans les réglages généraux,
 - les champs **Adresse web de WordPress (URL)**
 - et **Adresse web du site (URL)** sont identiques.
- Si vous ne voulez pas avoir tous les fichiers de WordPress à la racine de votre dossier de publication, mais plutôt dans un sous-dossier nommé **wordpress** par exemple.
 - Dans ce cas, le champ **Adresse web de WordPress (URL)** indique l'URL d'accès au sous-dossier qui contient tous les fichiers de WordPress
 - et le champ **Adresse web du site (URL)** indique l'URL d'accès à la page d'accueil du site.
 - Dans ce cas :
 - la fonction `home_url()` indique l'URL d'accès à la page d'accueil du site, c'est-à-dire le champ **Adresse web du site (URL)**.
 - la fonction `site_url()` indique l'URL d'accès au sous-dossier des fichiers de WordPress, c'est-à-dire le champ **Adresse web de WordPress (URL)**. Cette fonction s'utilise exactement de la même manière que `home_url()`.



Les URL du site

- **Le lien vers un contenu**

- La fonction `get_permalink()` permet de créer un lien vers un contenu en page seule, que ce soit un article ou une page.
- Voici l'URL de référence : https://developer.wordpress.org/reference/functions/get_permalink/.
- Cette fonction permet de récupérer le permalien des contenus.
- La fonction propose deux paramètres : `get_permalink(int|WP_Post $post, bool $leavename = false)`.
 - Le premier paramètre optionnel, **\$post**, permet de préciser l'identifiant du contenu à afficher. Cela permet de faire un lien vers un contenu précis et spécifié. Par défaut, c'est le paramètre global WP_Post qui est utilisé et qui permet de cibler tous les contenus se trouvant dans la boucle WordPress.
 - Le deuxième paramètre optionnel, **\$leavename**, permet de conserver une structure de permalien sous la forme %postname%, au lieu du permalien complet. C'est un booléen, par défaut la valeur est false.
- Voici un exemple usuel dans une boucle WordPress.
 - Le lien est placé sur le titre des contenus, `the_title()`, dans un élément `<h2>` :
 - `the_title(' <h2> ' </h2>')`.

La création de son propre thème

Les URL du site

- Voici un exemple pour créer un lien vers un article dont l'identifiant est 62, en dehors de la boucle :

```
<?php
    $dimanche_pl = get_permalink(62);
    echo ('<p>Voir l\'article de <a href="'. $dimanche_pl.'">
Dimanche</a>.</p>')
?>
```

- L'URL d'un contenu

- La fonction **the_permalink()** permet d'afficher et d'utiliser l'URL d'un contenu, article ou page.
- Voici son URL de référence : https://developer.wordpress.org/reference/functions/the_permalink/

La création de son propre thème

Les URL du site

ARTICLES

11 OCTOBRE 2018 MODIFIER

Dakhla, la petite Essaouira aux portes du désert

Paradis des surfeurs, l'ancienne colonie espagnole jouit de sa localisation entre le Sahara et l'Atlantique pour offrir des séjours bien-être et découvertes. Trekkings dans le désert, sources chaudes et sports nautiques sont au rendez-vous dans cette ville stratégique pour le Maroc, dans la conquête du Sahara occidental.



• Le lien d'édition

- Certains thèmes vous proposent, dans l'affichage des contenus, un lien qui permet d'éditer ce contenu, si l'utilisateur est connecté et s'il possède les droits suffisants.
- Voici l'exemple avec le thème **Twenty Seventeen**, avec le lien **Modifier** :
- C'est la fonction `edit_post_link()` qui est utilisée
 - https://developer.wordpress.org/reference/functions/edit_post_link/
 - Cette fonction propose plusieurs paramètres optionnels :
`edit_post_link(string $text = null, string $before = "", string $after = "", int $id, string $class = 'post-edit-link')`.
 - Le premier paramètre, **\$text**, est le libellé affiché de ce lien. Il dépend naturellement du thème et des éventuelles traductions disponibles. Par défaut, c'est **Edit this** qui est utilisé.
 - Les deux paramètres suivants, **\$before** et **\$after**, permettent d'insérer du code HTML/CSS devant et après ce lien. Cela permet d'insérer ce lien dans un élément HTML de votre choix.
 - Le quatrième paramètre, **\$id**, est l'identifiant de l'article. Mais utilisé dans la boucle WordPress, ce paramètre prend par défaut la valeur courante de la variable globale **\$post**, c'est-à-dire l'identifiant de l'article affiché.
 - Le dernier paramètre, **\$class**, permet d'ajouter une classe CSS au lien. Par défaut cette classe est **.post-edit-link**.

Les URL du site

MÉTA

Admin. du site

Déconnexion

Flux RSS des articles

RSS des commentaires

Site de WordPress-FR

- **Le lien vers l'administration**

- Le widget **Méta** contient par défaut un lien de connexion vers l'administration. Mais il n'est pas toujours très pratique car il contient d'autres liens, pas toujours très utilisés.
- Voici ce widget affiché dans le thème WordPress **Twenty Seventeen** :
- Si vous ne souhaitez pas utiliser ce widget, vous pouvez utiliser la fonction `get_admin_url()` pour créer un lien vers l'administration.
- Voici son URL : https://developer.wordpress.org/reference/functions/get_admin_url/
- Voici un exemple de code qui permet d'afficher un lien nommé **Administration** :

```
<?php
    $admin = get_admin_url();
    echo ('<p><a href="'. $admin. '>Administration</a></p>');
?>
```


Les marqueurs des contenus : Afficher le titre

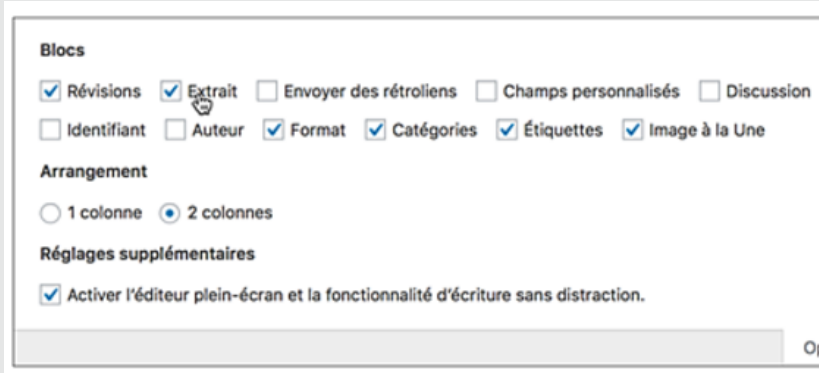
- Pour afficher tout le contenu rédactionnel des articles et des pages, WordPress nous propose plusieurs marqueurs.
- Afficher le titre
 - le marqueur `the_title()` affiche le titre de l'article ou de la page.
 - https://developer.wordpress.org/reference/functions/the_title/.
 - Ce marqueur doit être utilisé dans la boucle WordPress.
 - Dans un template donné, le marqueur `<?php the_title(); ?>`, sans paramètre, affiche le titre du contenu.
 - Ce marqueur propose plusieurs paramètres optionnels : `the_title(string $before = "", string $after = "", bool $echo = true)` :
 - **\$before** permet d'insérer du code HTML/CSS avant l'affichage du titre.
 - **\$after** permet d'insérer du code HTML/CSS après l'affichage du titre.
 - **\$echo** indique ce que doit retourner ce marqueur. La valeur par défaut **true** indique que le titre doit être affiché. La valeur **false** indique que le marqueur renvoie la valeur, le titre donc, sans l'afficher, afin de l'utiliser dans du code PHP.
 - Par exemple, ce code :
 - `<?php the_title('<h2 class="titre-contenu">';</h2>'); ?>`
 - affichera le titre du contenu dans un élément HTML `<h2>`, avec la classe CSS **.titre-contenu**.
 - Voici le code qui sera généré : `<h2 class="titre-contenu">Voyage à Venise</h2>`.

Les marqueurs des contenus : Afficher le contenu

- Afficher le contenu

- Le marqueur **the_content()** permet d'afficher le contenu rédactionnel des articles et des pages, ce marqueur doit être utilisé dans la boucle WordPress.
- https://developer.wordpress.org/reference/functions/the_content/
- Dans un template donné, `<?php the_content(); ?>`, sans paramètre, affiche le contenu rédactionnel dans des éléments HTML `<p>`.
- Ce marqueur propose deux paramètres : **the_content**(string \$more_link_text = null, bool \$strip_teaser = false).
 - Le paramètre optionnel **\$more_link_text** permet de personnaliser le libellé affiché lorsque le contenu contient la balise **Lire la suite** dans l'interface de saisie :
- Voici l'affichage par défaut :
 - Le libellé **Continuer la lecture** permet d'afficher l'article complet.
 - Le code **the_content ('Lire la suite.')** affichera sous forme de lien le libellé **Lire la suite**
 - Pour ne pas afficher le lien **Continuer la lecture**, il vous suffit de ne rien mettre entre les cotes : `the_content(`).
 - Le paramètre optionnel **\$strip_teaser** à true permet de masquer le contenu qui se trouve avant l'insertion de la balise **Lire la suite**.

Les marqueurs des contenus : Afficher l'extrait



- Afficher l'extrait
 - WordPress vous permet d'utiliser la fonction d'extrait associée à l'article. Cela permet par exemple d'afficher un résumé de l'article sur la page d'accueil.
 - Dans l'administration, il faut afficher le champ pour saisir l'extrait en utilisant les **Options de l'écran** et cocher l'option **Extrait**.
 - Ensuite, saisissez le contenu du champ **Extrait**.
 - Pour afficher cet extrait, il faut utiliser le marqueur `the_excerpt()`, à la place du marqueur `the_content()` si vous le souhaitez.
 - https://developer.wordpress.org/reference/functions/the_excerpt/
 - Si le champ **Extrait** de l'article n'est pas renseigné, ce sont les 55 premiers mots de l'article qui sont utilisés.
 - Dans cet exemple, l'article **Mercredi** possède un extrait, mais pas l'article **Mardi**. Voici l'affichage obtenu :
 - Le contenu s'affiche dans un élément `<p>`.
- Si vous le souhaitez, vous pouvez tester l'utilisation de l'extrait avec le marqueur conditionnel `has_excerpt()` (https://developer.wordpress.org/reference/functions/has_excerpt/) :

```
<?php
if (has_excerpt() ) {
    the_excerpt();
}
?>
```

La création de son propre thème

Les marqueurs de date et d'heure de rédaction

- Il est d'usage d'indiquer a minima la date de création des articles. C'est une indication précieuse pour les visiteurs des sites et pour le référencement. Vous avez aussi la possibilité d'y ajouter l'heure.
- **Afficher la date de création**
 - Le marqueur `the_date()` affiche la date de création de l'article et donc pas forcément la date de publication sur le site, puisqu'un article peut être enregistré en tant que brouillon avant sa publication.
 - https://developer.wordpress.org/reference/functions/the_date/.
 - Ce marqueur doit être utilisé dans la boucle WordPress.
 - L'utilisation du marqueur `the_date()` sans paramètre affiche la date sans aucun conteneur HTML.
- Les paramètres d'affichage de la date
 - Ce marqueur nous propose plusieurs paramètres : `the_date(string $d = "", string $before = "", string $after = "", bool $echo = true)`.
 - Le premier paramètre `$d` permet de personnaliser la date affichée. Par défaut, c'est le format de la date réglée dans **Réglages - Général - Format de date**, qui est utilisé.

Format de date		
<input checked="" type="radio"/> 14 juin 2017	j F Y	
<input type="radio"/> 2017-06-14	Y-m-d	
<input type="radio"/> 06/14/2017	m/d/Y	
<input type="radio"/> 14/06/2017	d/m/Y	
<input type="radio"/> Personnalisé :	j F Y	14 juin 2017

La création de son propre thème

Les marqueurs de date et d'heure de rédaction

- Voici une URL pour déterminer le format de date que vous souhaitez : <http://php.net/manual/fr/function.date.php>
- Voici un exemple d'un format de date personnalisé : `the_date('l d F Y')`
- Voici l'affichage obtenu :

lundi 02 janvier 2017

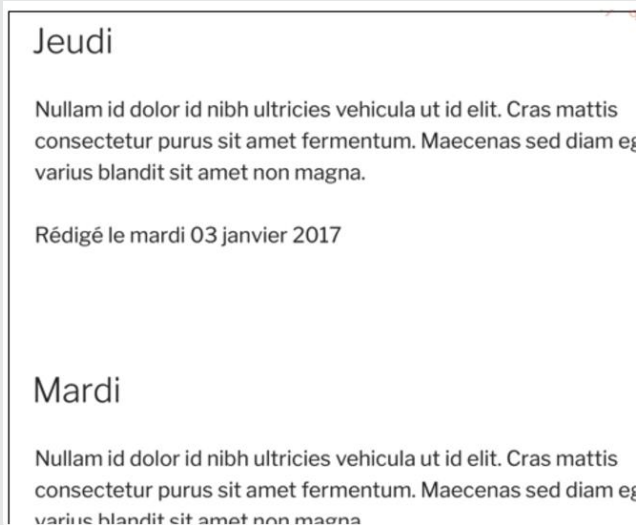
- Nous avons la possibilité d'insérer du code avant et après la date, avec les paramètres **\$before** et **\$after**.
- Voici un exemple : `the_date('l d F Y', '<p class="date">Rédigé le ', ' </p>');`
- Voici l'affichage obtenu :

Rédigé le lundi 02 janvier 2017

- Voici le code généré : `<p class="date">Rédigé le lundi 02 janvier 2017</p>`.
- Le paramètre **\$echo** indique si la date est affichée : true (par défaut) ou non, false. Dans ce cas, la date peut être utilisée dans du code PHP.

La création de son propre thème

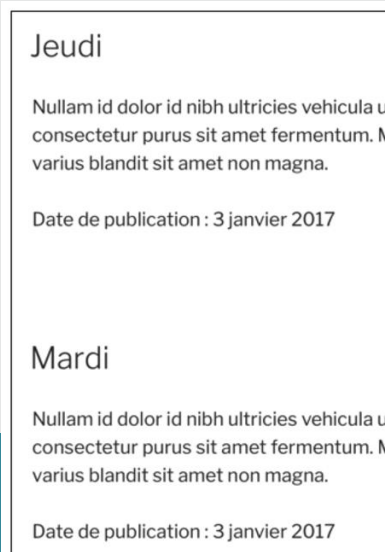
Les marqueurs de date et d'heure de rédaction



- Afficher des dates de création similaires
- Attention, si plusieurs articles sont créés le même jour et affichés dans la page d'accueil par exemple, seule la date du premier article est affichée. Dans cet exemple, la date affichée par le thème **Twenty Seventeen** a été masquée pour ne conserver que la date de création personnalisée avec le marqueur **the_date**
- Les deux articles ont été créés le même jour, seul le premier affiche la date.
 - Pour forcer l'affichage de toutes les dates, il faut utiliser le marqueur **get_the_date()** dans une variable.
 - https://developer.wordpress.org/reference/functions/get_the_date/
 - Voici un exemple de code :

```
<?php
$date_publication = get_the_date( );
echo '<p>Date de publication :
'.$date_publication.'</p>';
?>
```

- Voici l'affichage obtenu :



La création de son propre thème

Les marqueurs de date et d'heure de rédaction

Jeudi

Nullam id dolor id nibh ultricies vehicula ut id elit
consectetur purus sit amet fermentum. Maecen
varius blandit sit amet non magna.

Date de publication : 3 janvier 2017

Modifié le lundi 09 janvier 2017

- Afficher la date de modification
- Pour parfaire les informations de rédaction des articles, vous pouvez afficher la date de modification avec le marqueur `the_modified_date()`.
- https://developer.wordpress.org/reference/functions/the_modified_date/.
- Ce marqueur reprend les mêmes paramètres que `the_date()`.
- Voici un exemple d'utilisation de ce marqueur :
 - `the_modified_date('l d F Y','<p class="date">Modifié le '</p>');`.
- Voici l'affichage obtenu :

La création de son propre thème

Les marqueurs de date et d'heure de rédaction

- Afficher l'heure de création

- C'est le marqueur `the_time()` qui affiche l'heure de création de l'article ou de la page.
- Voici son URL de référence : https://developer.wordpress.org/reference/functions/the_time/
- Le format de l'heure est par défaut celui indiqué dans les réglages généraux :

Format d'heure

☒ 16 h 06 min `G \h i \m\i\n`

☐ 4:06 `g:i A`

☐ 16:06 `H:i`

☐ Personnalisé : `G \h i \` 16 h 06 min

[Documentation sur le format des dates.](#)

- Vous pouvez personnaliser ce format avec le seul paramètre `$d` qui s'utilise comme celui des dates.
- Vous pouvez bien sûr utiliser une variable afin d'insérer l'heure de création dans l'affichage de la date :

```
$heure_creation = get_the_time();  
the_date('l d F Y', '

Rédigé le ', '  
à '.$heure_creation.'

');
```

- Voici l'affichage obtenu :

Rédigé le mardi 03 janvier 2017 à 18 h 21 min



La création de son propre thème

Les marqueurs de date et d'heure de rédaction

- Afficher l'heure de modification
 - Le marqueur `the_modified_time()` affiche la date de dernière modification de l'article.
 - https://developer.wordpress.org/reference/functions/the_modified_time/.
 - Ce marqueur ne propose qu'un seul paramètre, `$d`, pour personnaliser l'heure affichée.
- Voici un exemple d'utilisation avec une variable pour un affichage commun avec la date de modification :

```
$heure_modification = get_the_modified_time();  
the_modified_date('l d F Y', '<p class="date">Modifié le ', '  
à '.$heure_modification.'</p>');
```

- Voici l'affichage obtenu :

Modifié le lundi 09 janvier 2017 à 11 h 54 min