5 Comprendre le fonctionnement d'un thème WordPress

Le thème est une partie incontournable de WordPress. C'est le thème qui donne son identité au blog, et c'est la première chose que voit le visiteur. Le thème habille le contenu du blog et prend en charge tous les aspects de mise en page et de design : polices, organisation, couleurs…

Dans ce chapitre dédié au développement d'un thème, vous apprendrez à mieux connaître tous les aspects d'un thème WordPress. Afin de ne pas sortir des sentiers battus, tous nos exemples de code seront basés sur le thème par défaut de WordPress, nommé Kubrick, et qui se trouve dans le dossier /wp-content/themes/default de votre installation de WordPress.

Dans le chapitre suivant, vous partirez d'une page blanche et concevrez de A à Z un thème pour votre blog. Si vous êtes pressé, vous pouvez passer directement à ce chapitre.

Analyse de la structure d'un thème pour WordPress

Vous avez pu voir dans la partie précédente que les thèmes ont un dossier qui leur est propre au niveau de l'installation de WordPress : /wp-content/themes.

Par défaut, deux thèmes sont disponibles : Kubrick et Classic. Chacun de ces thèmes est stocké dans un dossier, respectivement /default et /classic, dans lequel on trouve les différents fichiers liés au thème et nommés selon une convention précise :

- la feuille de style du blog : style.css ;
- les fichiers composant le thème, appelés "fichiers de modèle" : index.php, comments.php, header.php, footer.php, single.php, etc. ;
- un fichier pour les fonctions liées au thème, agissant comme une extension : functions.php.

Le thème par défaut, Kubrick, contient par ailleurs un sous-dossier, /images, où sont stockées toutes les images qu'il utilise.

Chaque fichier joue un rôle bien précis, dans une hiérarchie souple et cohérente.

La feuille de style

Le fichier style.css contient tous les aspects de mise en page du thème de votre blog, écrit à l'aide du langage CSS (*Cascading Style Sheet*). C'est ici que sont enregistrés notamment tous les choix relatifs à l'emplacement, la couleur et la typographie. C'est également ce fichier qui contient les informations que WordPress affichera au sujet du thème : nom du thème, version, nom de l'auteur…

Les fichiers de modèle

L'appellation "fichiers de modèle" (*template files*) regroupe tous les fichiers au format .php du thème, hormis les fichiers spéciaux comme functions.php.

Il existe deux types de fichiers de modèle :

- **Les fichiers qui vont composer le thème.** Reliés entre eux, ils vont créer et afficher la page web.
- **Les fichiers qui vont afficher les différentes pages du blog.**

Seuls deux fichiers sont obligatoires pour qu'un thème puisse être considéré comme valide par WordPress : index.php pour le code PHP, et style.css pour la mise en forme du blog. Notez qu'il suffit que le fichier style.css soit absent ou mal formaté pour que WordPress ne prenne pas le thème en compte.

Avec un thème qui n'utiliserait que ces deux fichiers, le fichier index.php doit contenir l'ensemble des appels PHP qui récupèrent et agencent les informations, pour les afficher sur le blog en fonction des requêtes du visiteur : en-tête, pied de page, textes, commentaires, colonne latérale, moteur de recherche… Cependant, si vous souhaitez personnaliser votre thème, il est préférable de le segmenter en différents fichiers qui correspondront notamment à différentes parties du blog : header.php pour l'en-tête, footer.php pour le pied de page, sidebar.php pour la colonne… Nous verrons très bientôt les différents noms de fichier possibles.

Le fichier functions.php

Ce fichier agit un peu comme une extension de WordPress et n'est pas obligatoire. Grâce à ce fichier, vous pouvez présenter différentes options et paramètres du thème à l'utilisateur *via* l'interface d'administration de WordPress, sans pour autant faire des modifications directement dans le thème. Il est ainsi possible, avec un peu de travail et de prévoyance, de laisser l'utilisateur choisir le nombre de colonnes, l'image en haut du blog, la taille des textes, etc. Les possibilités sont infinies.

Le dossier images

Ce dossier contient toutes les images qui servent à la présentation de votre blog. Il n'est pas obligatoire, mais à partir de trois images, il est préférable de les ranger dans un dossier à part. Le nom de ce dossier n'est pas non plus fixé, et vous pouvez le baptiser à votre gré : images, img, pix…

Notez bien que ce n'est pas dans ce dossier que vous devez ranger les images que vous avez l'intention d'utiliser dans vos articles. En effet, si vous les insérez ici, elles disparaîtront le jour où vous changerez de thème.

Structure et convention de nomenclature

Ce ne sont là que quelques exemples de la structure "classique" que vous trouverez sur de nombreux thèmes WordPress. Cette structure se présente le plus souvent comme suit :

- **header.php.** Affiche les informations de l'en-tête du blog.

- **index.php.** Affiche le contenu du blog.
- **sidebar.php.** Affiche la colonne latérale du blog.
- **footer.php.** Affiche le pied de page du blog.

Notez que trois fichiers ont été extraits du header.php de base, pour gérer respectivement trois zones du thème indépendantes du contenu : l'en-tête, le pied de page et la colonne latérale.

Sur le thème default, la page se présentera comme à la Figure 5.01.

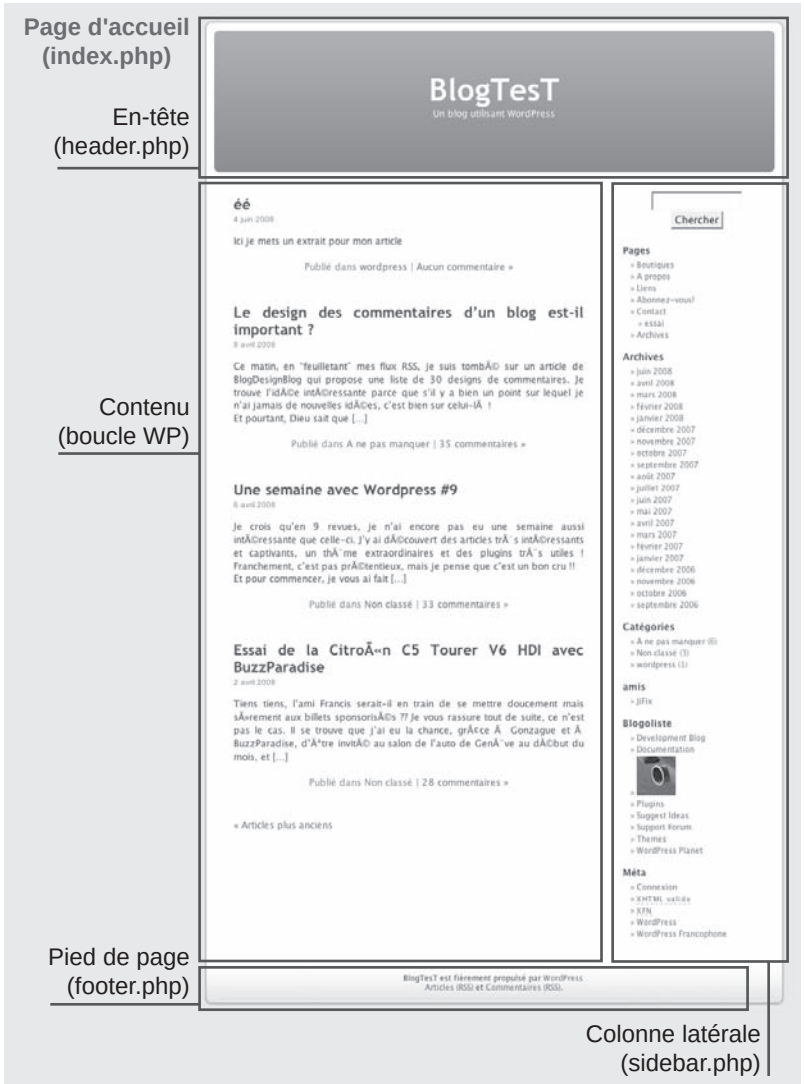

Figure 5.01
Page d'accueil default.

Lors de l'affichage d'un seul article du blog, WordPress fera appel aux fichiers suivants, liés entre eux par des appels PHP et les conventions de WordPress :

- **single.php.** Affiche le contenu de l'article.
- **header.php.** Affiche l'en-tête du blog.
- **sidebar.php.** Affiche la colonne latérale du blog.
- **footer.php.** Affiche le pied de page du blog.
- **comments.php.** Affiche les commentaires et le formulaire de commentaire.

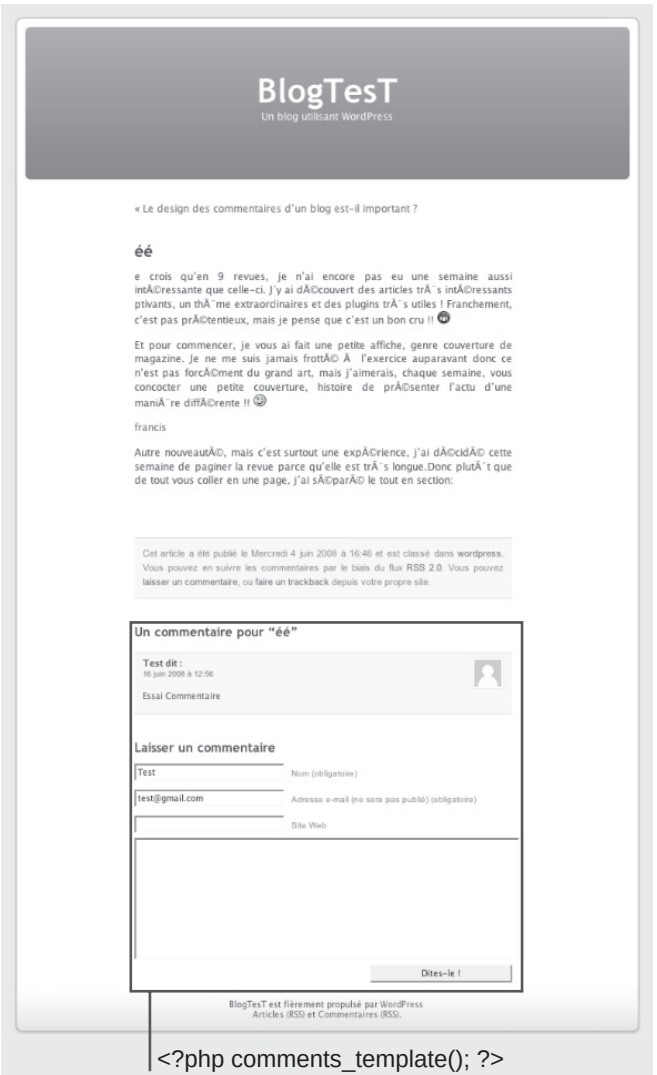

Figure 5.02
Page de l'article avec mise en relief des commentaires.

En séparant le thème en plusieurs fichiers, chacun s'occupant d'une partie précise de l'affichage, on simplifie les fichiers, et leur développement devient bien plus aisé. Le fichier HTML final est construit à l'aide d'appel de méthodes PHP internes à WordPress, qui assument le chargement des fichiers nécessaires selon leurs noms et la hiérarchie existante, ce que nous verrons dans la section qui suit.

Sur les autres types de pages, comme les pages de catégories, de labels (tags), d'archives ou encore de recherche, nous avons une combinaison tournant autour de ces fichiers :

- index.php, category.php, 404.php ou search.php : fichiers de modèle spécifiques à un affichage ;
- header.php ;
- sidebar.php ;
- footer.php.

Le thème reprend généralement la même organisation que pour la page d'accueil, mais il est également possible de créer un fichier typique pour chacune de ces pages, comme category.php pour les pages de catégories.

Différents fichiers pour différentes pages

Nous venons de voir qu'il est préférable de scinder une page web en différents fichiers et de les appeler par des fonctions PHP de WordPress. Nous avons également abordé la structure HTML de base qui sera utilisée pour créer un thème WordPress. Celle-ci sera à déployer sur les différents fichiers qui afficheront différents types de pages du blog. Il reste à définir les fichiers que nous pouvons utiliser.

Voici la liste complète des fichiers permettant de gérer séparément des parties du blog :

- **single.php.** Pour afficher un article seul.
- **page.php.** Pour afficher une page seule (on peut également utiliser pagename.php qui affichera directement le nom de la page).
- **home.php.** Pour afficher une page d'accueil spécifique.
- **tag.php.** Pour afficher les pages de tags (on peut aussi employer tag-slug.php qui affichera les articles correspondant à un tag en ayant son nom directement dans l'URL sous la forme "tag-wordpress.php" par exemple, avec ici "wordpress" comme nom de tag).
- **archive.php.** Pour afficher les archives (par catégorie, par label, par auteur, etc.).
- **category.php.** Pour afficher une catégorie seule. Il peut être utilisé sous une déclinaison encore plus précise, category-X.php, pour gérer l'affichage des articles de la catégorie dont le numéro d'identifiant est X (par exemple, category-7-.php).
- **search.php.** Pour afficher les résultats d'une recherche.
- **404.php.** Pour afficher la page d'erreur, quand un article n'est pas disponible par exemple (ce qu'on appelle une erreur 404).

- **[nom de la page].php.** Pour afficher une page précise, à partir de son permalien. Par exemple, si nous voulons un affichage spécifique pour la page À Propos, nous utiliserons a-propos.php.
- **author.php.** Pour afficher une page présentant les données liées à un auteur en particulier (articles, pages, liens, autres informations).
- **date.php.** Pour afficher les articles par date précise (année, mois ou jour).
- **image.php, video.php, audio.php et application.php.** Pour afficher les pages correspondant à ces différents types de médias qui ont leur propre URL. En créant un marqueur de modèle particulier, vous pourrez générer des pages spécifiques pour chacun d'eux.

Hiérarchie des différents types de pages

Les noms de fichiers présentés ci-dessus servent pour afficher les différentes vues disponibles du blog, sans être pour autant obligatoires. En effet, les modèles de WordPress fonctionnent autour d'une hiérarchie précise, et si un fichier n'existe pas dans le thème, WordPress ira chercher le fichier supérieur dans cette hiérarchie pour afficher le contenu.

La Figure 5.03 représente l'organigramme de hiérarchie des modèles dans WordPress.

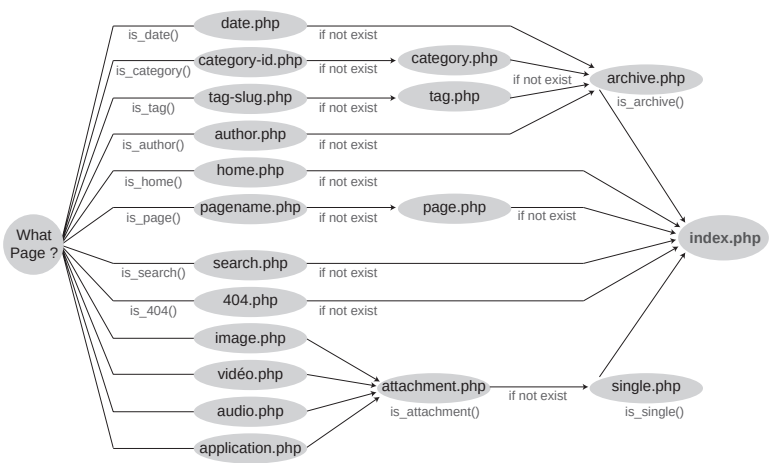

Figure 5.03
Hiérarchie des modèles.

Prenons un exemple pour illustrer la logique de cette hiérarchie. Un visiteur passe sur un blog et clique sur le lien pour voir tous les articles de l'année 2008. Cette URL a la forme suivante : http://www.monblog.com/2008 .

Si nous reprenons les règles de nommage des fichiers citées ci-dessus, la page devrait s'afficher à l'aide du fichier date.php. Or, le thème employé ne dispose pas d'un fichier de ce nom. En utilisant la hiérarchie des modèles, nous pouvons voir que dans ce cas WordPress ira trouver le fichier archive.php pour afficher les articles de l'année 2008. Et s'il n'existe pas non plus, il remontera jusqu'au fichier index.php qui, lui, est obligatoire.

Il en est de même pour l'ensemble des fichiers présentés plus haut. Ils suivent une hiérarchie qui permettra toujours d'afficher une page web avec le contenu demandé, mais peut-être pas