



Introduction à
WordPress

Sommaire

Présentation

Installation

Types de contenus

L'interface back-end

Paramétrage

Utilisation d'un thème

Création d'un thème

Présentation

Historique



Michel Valdrighi

2001

Sortie de B2 cafelog en Open Source



Matt Mullenweg & Mike Little

2003

Sortie de WordPress en Open Source

2005

Création de la société Automattic

We are passionate about making the web a better place.

WordPress.com

Hassle-free blogging even with your own domain freemium model.

VaultPress

Disaster happens buy insurance for your site sleep soundly at night.

Gravatar

Identity is visually portable your face everywhere.

Simperium

An API for developers everywhere to stay on same page.

Jetpack

Power of the cloud right there in your own WordPress. Supercharge your site!

Akismet

Remember the days innocent inboxes gleam be spam-free again.

VideoPress

High-definition video for ev'ry need integrated well.

Code Poet

Resources for you, maker of things with WordPress. Code is poetry.

Simplenote

Synchronization. The bane of all notetakers is now simply solved.

Polldaddy

Your readers are smart easily find what they think polls and surveys, natch.

IntenseDebate

Blogging is social imagine better comments readers will thank you.

WordPress.com VIP

The best of the best sites need bulletproof service. We can handle it.

We also contribute to a number of non-profit and Open Source projects; here are a few:

WordPress.org

Community first
plugins and themes ev'rywhere
together we build.

P2 Theme

Collaborating?
Never send email again.
This will change your life.

WP for BlackBerry

"I'm not addicted;
I will quit any day now."
Your thumbs know the truth.

WP for iOS

Sometimes you have thoughts
while on-the-go in the world
your iPhone is there.

BuddyPress

The web is social,
our connections matter most.
Why isn't your site?

WP for Windows Phone

Real simplicity
in the app that brings WordPress
to your eager hands.

WP for Android

I, for one, welcome
our new Android overlords.
Robots can blog, too.

bbPress

Blogs sometimes restrain
community discussion.
Forums, the new black.

WordCamp SF

Hundreds gather to
share, learn, connect, celebrate
the first of [many](#).

De quoi parle t'on
lorsque l'on parle de WordPress ?

WordPress.co

Service Freemium
d'Automatic

WordPress.org

Fondation qui gère et met
gratuitement à disposition le
code de WordPress



WordPress.com ou WordPress.org ?



Résumé des 10 principales différences

WORDPRESS.ORG

PERSONNEL
monnomdomaine.fr

CELUI DE L'HÉBERGEMENT
(+ nom de domaine)

ILLIMITÉ

THÈME DE SON CHOIX
banque de données de thèmes

OUI

OUI

PERMIS

MANUELLE, ÊTRE RESPONSABLE
DE LA TOTALITÉ DU BLOG

OUI

OUI

POINTS

HÉBERGEMENT

COÛT

STOCKAGE

THÈME

EXTENSIONS

FTP / ACCÈS

MONÉTISATION

MAINTENANCE

BRANDING

CONTRÔLE

WORDPRESS.COM

SUR LE SITE - LIMITÉ sinon payant
adresse imposée : monsite.wordpress.com
publicité présente

GRATUIT mais pas sans nom de domaine
et selon capacité de stockage

LIMITÉ

GRATUIT MAIS LIMITÉ
AUX THÈMES PRÉSENTS
modification css payant

NON

NON

NON PERMIS OU TRÈS LIMITÉ

AUTOMATIQUE (par le logiciel Automattic)

NON

NON

Differences : wordpress.com et wordpress.org

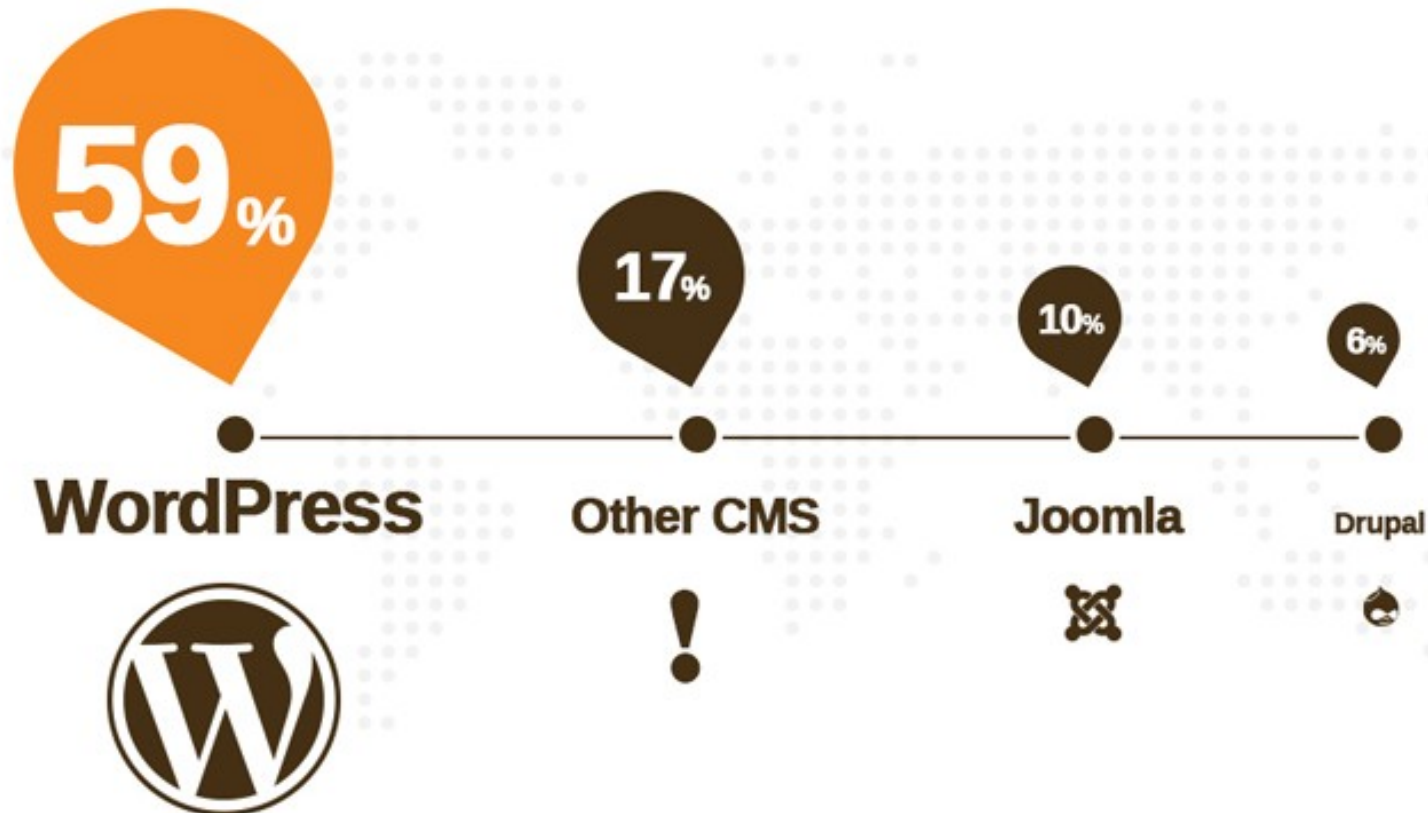
WordPress.com <i>Focus on your beautiful content, and let us handle the rest.</i>	WordPress.org <i>Get your hands dirty, and host your website yourself.</i>
Premium hosting, security, and backups are included. You can even upgrade to a custom domain , like YourGroovyDomain.com.	You'll need to find a host , and perform backups and maintenance yourself. We offer VaultPress for security and backups.
Choose from hundreds of beautiful themes . Make it your own with Custom Design .	Install custom themes. Build your own with PHP and CSS.
Integrate your site with Facebook, Twitter, Tumblr, and other social networks.	Install a plugin, like Jetpack , to enable sharing functionality on your site.
Popular features like sharing, stats, comments, and polls are included. There's no need to install plugins.	Install plugins to extend your site's functionality.
Personal support and the WordPress.com forums are always available.	Visit the WordPress.org support forums for assistance.
You must register for an account on WordPress.com and abide by our Terms of Service .	No registration with WordPress.org is required.

WordPress en chiffres en 2013



WordPress en chiffres en 2013

Websites with CMS



WordPress en chiffres (2014)

74.6 Million de sites sont basés sur WordPress dont 50 % sur wordpress.com

WordPress est traduit en 40 langues.

wordpress.com a plus de visiteurs uniques que Amazon US.

WordPress emploie 229 personnes réparties partout dans le monde.

29 000 plugins sont disponibles pour WordPress, il s'en développe un toutes les heures.

Les principaux préjugés concernant WordPress

« WordPress, c'est pour les
blogs ... »

Site magazine

SECTIONS

SEARCH

SUBSCRIBE NOW

SIGN IN

Register

U.S. INTERNATIONAL 中文网

InSIGHTFUL REPORTING
WHEREVER YOU ARE
TRY IT NOW >

The New York Times

Friday, February 7, 2014 | Today's Paper | Personalize Your Weather | f t

International New York Times
UNDER €1 FOR 12 WEEKS
SUBSCRIBE NOW >

WORLD U.S. NEW YORK BUSINESS OPINION SPORTS SCIENCE ARTS FASHION & STYLE VIDEO OLYMPICS

All Sections

International New York Times

GLOBAL COVERAGE THAT GOES EVERYWHERE YOU DO.
UNDER €1 FOR 12 WEEKS

GET IT NOW >

SOCHI 2014

A Triumph for Putin, Games Arrive as Russia Suffers a Slump

By STEVEN LEE MYERS

The Winter Olympics, whose opening ceremony is Friday, represents a chance for President Vladimir V. Putin to demonstrate anew his mastery of the global levers of power, but perhaps not for the country he governs.

24 Comments

- Latest Updates as Opening Ceremony Nears 6 minutes ago
- Designers Carve Fashion Runway at the Games
- Luger's Name Matches the One on His Waistband



INTERACTIVE FEATURE

What Gives McMorris the Edge at Slopestyle?

By MIKE BOSTOCK, BEDEL SAGET and CATHERINE SPANGLER

The snowboarding event, a new addition to the Games, has been dominated recently by Mark McMorris, a 20-year-old from the flat plains of Saskatchewan, Canada.

• White's Absence Looms Over Slopestyle | Photos: Day 1

Download our mobile apps to receive alerts.

Weakness Continues as 113,000 Jobs Are Added in January

By NELSON D. SCHWARTZ
8:51 AM ET

With the disappointing showing, some experts are

NEWS ANALYSIS

Complex Interplay in Retreat on Immigration

By CARL HULSE

Reaching any agreement has become appreciably harder because of a deep Republican reluctance to get caught up in

The Opinion Pages

Mr. Putin's Russia

By THE EDITORIAL BOARD

The Sochi Olympics should not be a reason to ignore the country's soul-crushing repression.

International New York Times



GLOBAL COVERAGE THAT GOES EVERYWHERE YOU DO.

UNDER €1 FOR 12 WEEKS

GET IT NOW >

Site corporate



ÊTRE UTILE AUX HOMMES

GSZ | 16,940€ (+0,98 %) | CAC 40 | 4 214,69 (+0,63 %) |    

Journalistes | Candidats | Actionnaires | Investisseurs | Experts RSE

» CONNAÎTRE LE GROUPE » COMPRENDRE NOS ENGAGEMENTS ▾ DÉCOUVRIR NOS ACTIVITÉS » TRAVAILLER CHEZ GDF SUEZ

FR 

PANORAMA
ÉLECTRICITÉ
GAZ NATUREL
SERVICES À L'ÉNERGIE

» Les offres de GDF SUEZ en France
» Trading d'énergie
» Développement urbain

» Les offres pour les particuliers
» Les offres pour les professionnels et les collectivités

Les offres de GDF SUEZ en France

Que vous soyez un particulier, un professionnel, une entreprise ou une collectivité en France, les marques de GDF SUEZ répondent à vos besoins grâce à l'expertise de multi-énergéticien du Groupe. Simples, claires et facilement reconnaissables, nos marques ont pour point commun de suivre un même objectif : «Etre utile aux hommes».

PARTICULIERS


PROFESSIONNELS


Site portfolio



WEB & GRAPHICS

A PORTFOLIO THEME FOR
WORDPRESS

HOME

PORTFOLIO

ABOUT

BLOG

SHORTCODES

ON THE SUBJECT OF ME

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud.

Fire me!

TWITTER

Our friends over @femmethemes designed this beautiful WordPress theme <https://t.co/QooubyADAK>

1 day ago

RT @tsiger: We have happily served 13000 happy CSSIgniters! Use the coupon code 13K and get 50% off! <http://t.co/qLokrhomCn>

8 days ago

Our 2nd theme for January is now available. Meet Tinos. A serious

MY NAME IS NICO ESCOBAR

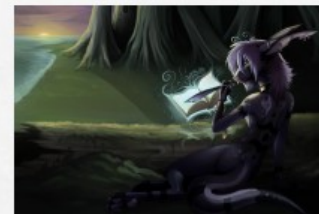
Stripped to our shirts and drawers, we sprang to the white-ash, and after several hours pulling were an almost disposed to renounce the chase, when a general pausing commotion among.

All Works

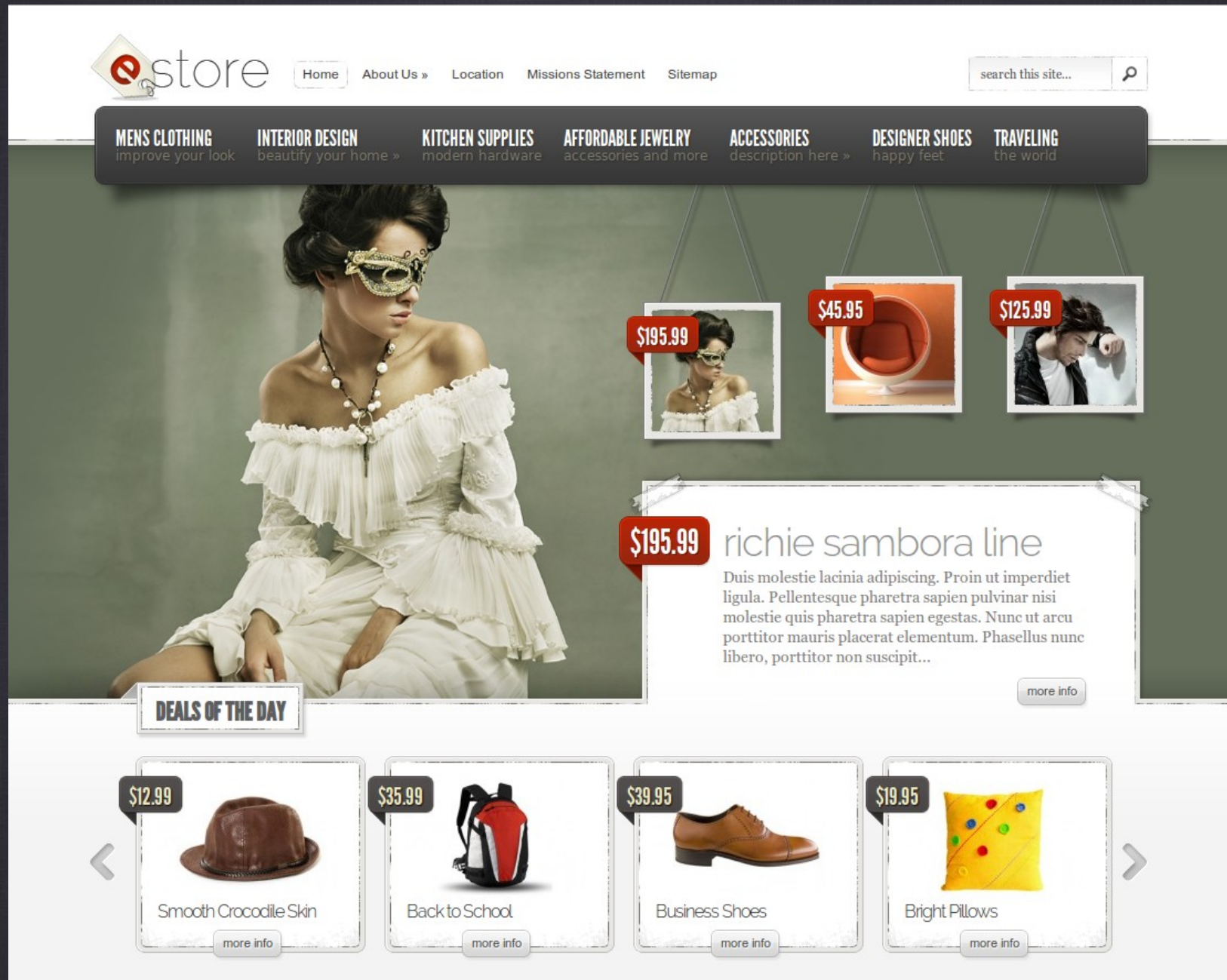
Illustrations

Photography

Typography




Site e-commerce



Site job-board

[About](#) [Blog](#) [Contact](#) [Resources](#) [FAQ](#) [Submit a Job](#) [Browse Resumes](#) [My Dashboard](#) [Logout](#)

 **JobRoller**
A Premium Job Listing Theme

[About](#) [Blog](#) [Contact](#) [Resources](#) [FAQ](#)

All Jobs

Location

Radius: Auto

Featured Jobs

Freelance

Sr. Finance Manager
[Charles Schwab](#) – Posted by [appthemedemo](#)

Los Angeles
California, United States

20 Nov
2010

Latest Jobs

☒ Freelance ☒ Full-Time ☒ Internship ☒ Part-Time ☒ Temporary [Filter](#)

Freelance

Hairsprayer
[Sublime Sally's Sassy Salon](#) – Posted by [Shannon](#)

New York
New York, United States

19 Apr
2013

Full-Time

Refrigeration Repair Technician
[Sears Corp](#) – Posted by [appthemedemo](#)

Austin
Texas, United States

23 Nov
2010

Freelance

Sr. Finance Manager
[Charles Schwab](#) – Posted by [appthemedemo](#)

Los Angeles
California, United States

20 Nov
2010

Temporary

Human Resources Manager
[Lowe's Hardware](#) – Posted by [appthemedemo](#)

Chicago
Illinois, United States

19 Nov
2010

Part-Time

Enterprise Deployment Specialist
[Google](#) – Posted by [appthemedemo](#)

Mountain View
California, United States

19 Nov
2010

Internship

Platform Validation Engineer
[Intel Corp](#) – Posted by [appthemedemo](#)

London
England, United Kingdom

19 Nov
2010

Full-Time

Solutions Consultant – Retail Sales
[Apple](#) – Posted by [appthemedemo](#)

New York
New York, United States

19 Nov
2010

Submit a Job

Starting at \$8.00 for 10 days

Browse by...

Tags

Job Type

Job Salary

Job Category

Date posted

Available Job Packs

Free

1 Jobs lasting 5 days, usable within 5 days

Free

Gold

10 Jobs lasting 15 days, usable within 15 days

\$15.00

IT Special

6 Jobs lasting 10 days, usable within 10 days

\$8.00

Buy Packs

Subscribe

Receive the latest job listings

Les principaux préjugés concernant WordPress

« WordPress pose des
problèmes
de sécurité ... »

WordPress est codé par des développeurs professionnels avec un souci permanent de sécurité. Le code de wordpress.com est mis à jour 70 fois par jour en moyenne.

WordPress est Open Source ce qui favorise la rapidité avec laquelle les failles de sécurité sont corrigées.

Un certain nombre de mesures peuvent renforcer la sécurité de WordPress.

Certains plugins et thèmes WordPress peuvent en revanche comporter des failles de sécurité. Mais ces développements ne sont pas effectués par l'équipe de développement de WordPress.

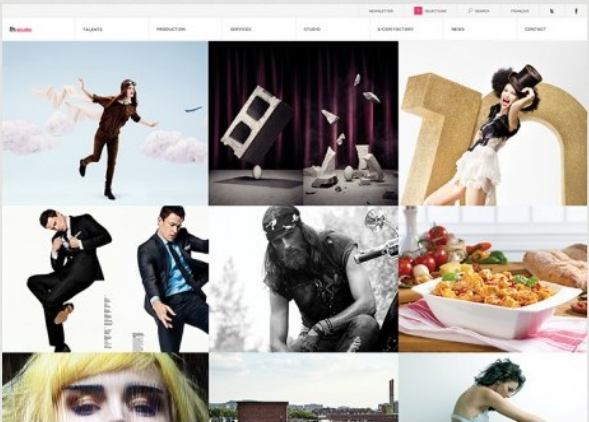
Les principaux préjugés concernant WordPress

« Tous les sites WordPress se
ressemblent ... »

Wordpress websites

398 Results for 'Wordpress websites'

WordPress is an Open Source application [Read more](#)



FH Studio

29

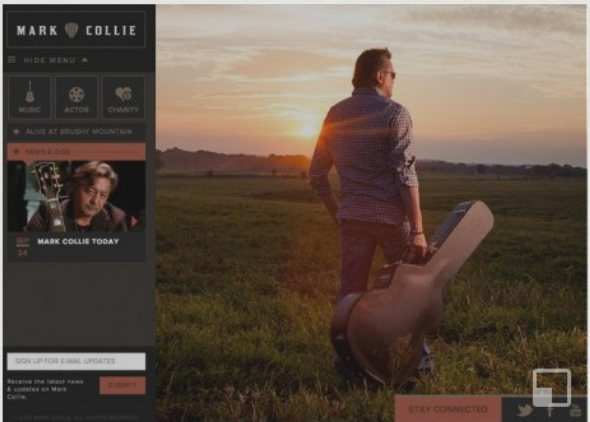
By [ANTRO](#) from [CANADA](#)
January 30, 2014 in [Honorable Mention](#)



Angle - Flat Responsive Bootstrap MultiPurpose Theme

9

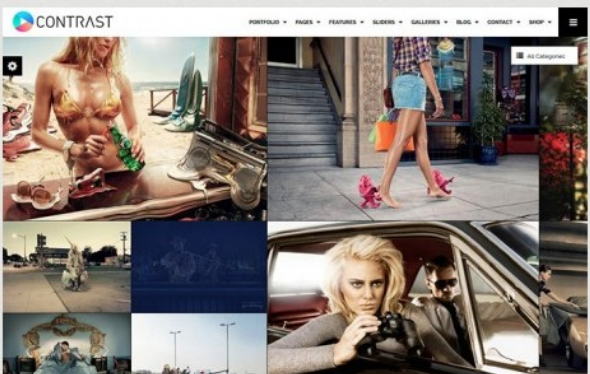
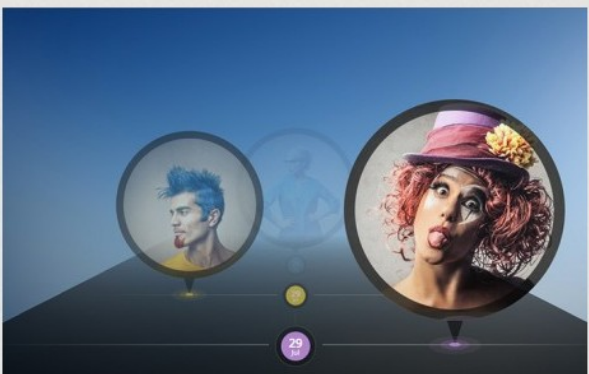
By [Oxygenna](#) from [GREECE](#)
January 29, 2014 in [Honorable Mention](#)



Mark Collie Website

8

By [Magnetic Creative](#) from [U.S.A.](#)
January 28, 2014 in [Honorable Mention](#)



Les principaux préjugés concernant WordPress

« L'avenir de WordPress
est incertain ... »

L'évolution du code est assuré par des développeurs professionnels et par une communauté mondiale.

WordPress est soutenu par une société de plusieurs centaines de salariés et un chiffre d'affaire de plusieurs millions de dollars.

WordPress est Open Source.

WordPress génère un écosystème business à travers le monde qui fait vivre des centaines de milliers de personnes.

Les principaux préjugés concernant WordPress

« WordPress n'est pas fait pour
les sites à fort trafic ... »

WordPress propulse des sites à très fort trafic comme TechCrunch, TheNextWeb, Wired, le New York Times.

Wordpress.com est lui même le 9 ème site le plus visité au monde.

WordPress n'est pas le facteur limitant à un fort trafic. Ce qui peut ralentir WordPress est une mauvaise optimisation du code des thèmes et des plugins, un mauvais paramétrage des serveurs ou le manque de ressources hardware attribuées aux serveurs.

Les qualités WordPress

Facile et rapide à installer.

Evolution constante et facile à mettre à jour.

Interface facile à prendre en main.

(pas toujours le cas pour certains plugins / thèmes)

Code sémantique et Google friendly.

Extensible et personnalisable (plugins / thèmes).

Très large communauté.

Documentation abondante.

Les défauts de WordPress

Très répandu donc plus souvent la cible d'attaques.

Qualité variable dans les plugins et les thèmes.

- Code
- Ergonomie des interfaces
- Logique de fonctionnement
- Performances

Moteur de recherche rudimentaire.

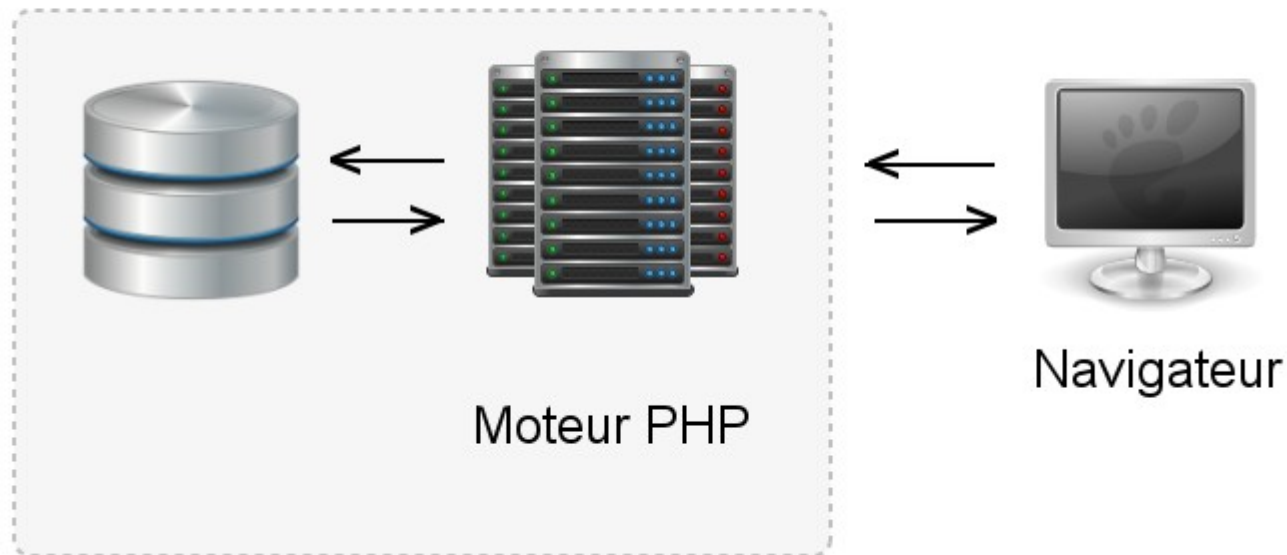
Ne gère pas le multilingue en natif.

Mises à jour fréquentes (compatibilité plugins / thèmes).

Installation

Environnement technique

Architecture Client Serveur



Base de données + Serveur web + Moteur PHP
(Mamp, Lamp, Wamp)

Etapes d'installation de WordPress

Etape 1 :

S'assurer que l'on dispose d'un serveur web et d'une base de données de type MySQL fonctionnel.

`http://localhost` ?

Présence d'un outil de type PhpMyAdmin ?

Répertoire racine des applications web ?

Etapes d'installation de WordPress

Etape 2 :

Créer une base « **bdd_wordpress** » dans MySQL grâce à un outil graphique tel que phpMyAdmin ou Adminer.

Création d'un utilisateur (requête sql)

```
CREATE USER 'user_wordpress' @ 'localhost' IDENTIFIED BY  
'f5g4_password' ;
```

Attributions de droits à cet utilisateur sur la base « **user_wordpress** »

```
GRANT ALL PRIVILEGES ON *.* TO 'user_wordpress' @ 'localhost'  
IDENTIFIED BY 'f5g4_password' ;
```

Pour que les changements soient pris en compte

```
FLUSH PRIVILEGES;
```

Etapes d'installation de WordPress

Etape 3 :

Télécharger les fichiers de WordPress et les placer à la racine du répertoire web du serveur.

http://fr.wordpress.org/latest-fr_FR.zip

- 1- Dézipper le fichier `wordpress-4.1.1-fr_FR.zip`
- 2- Mettre le contenu du répertoire « wordpress » dans le répertoire racine du serveur web.

Etapes d'installation de WordPress

Etape 4 :

Renseigner le fichier de configuration de WordPress « **wp-config.php** » avec les paramètres de connexion à la base de données « **bdd_wordpress** ».

```
// ** MySQL settings - You can get this info from your web host ** //  
/** The name of the database for WordPress */  
define('DB_NAME', 'bdd_wordpress');  
  
/** MySQL database username */  
define('DB_USER', 'user_wordpress');  
  
/** MySQL database password */  
define('DB_PASSWORD', 'f5g4_password');  
  
/** MySQL hostname */  
define('DB_HOST', 'localhost');  
  
/** Database Charset to use in creating database tables. */  
define('DB_CHARSET', 'utf8');  
  
/** The Database Collate type. Don't change this if in doubt. */  
define('DB_COLLATE', '');
```


Etapes d'installation de WordPress

Etape 5

Charger la page racine de votre serveur web (http://localhost) et suivez les étapes d'installation de WordPress.

Informations nécessaires

Merci de fournir les informations suivantes. Ne vous inquiétez pas, vous pourrez les modifier plus tard.

Titre du site	<input type="text" value="Nouveau site"/>
Identifiant	<input type="text" value="admin"/> <small>Les identifiants doivent contenir uniquement des caractères alphanumériques, espaces, tiret bas, tiret, points et le symbole @.</small>
Mot de passe, deux fois <small>Un mot de passe vous sera automatiquement généré si vous laissez ce champ vide.</small>	<div><input type="password" value="....."/> <input type="password" value="....."/> <div>Forte</div></div> <small>Conseil : votre mot de passe devrait faire au moins 7 caractères de long. Pour le rendre plus sûr, utilisez un mélange de majuscules, de minuscules, de chiffres et de symboles comme ! " ? \$ % ^ &).</small>
Votre adresse de messagerie	<input type="text" value="mail@francoisdubois.fr"/> <small>Vérifiez bien cette adresse de messagerie avant de continuer.</small>
Vie privée	<input checked="" type="checkbox"/> Autoriser mon site à apparaître dans les moteurs de recherche comme Google et Technorati.

Installer WordPress

Gestion du contenu

La gestion du contenu dans WordPress

Travail de groupe
(9 groupes de 2 personnes)

Décrire l'usage et le fonctionnement
d'un type de contenu de WordPress.

- 1- Pages - Sous Pages - Templates de page
- 2- Articles - Commentaires - Catégories et tags - Posts format
- 3- Menus personnalisés
- 4- Medias - Customizer (personnaliser)
- 5- Widgets - Zones widgetisables
- 6- Custom Post Type - Taxonomies
- 7- Hyperliens - Titre du site - Tagline du site
- 8- Thèmes - Child Thèmes
- 9- Plugins - Shortcodes

La gestion du contenu dans WordPress

Voir le guide de WPBRIX
« WordPress 3.6 for beginners »

L'interface backend

Front-end & Back-end

Front-end :

C'est la partie du site qui est visible par les internautes.

Back-end :

C'est la partie privée du site qui est accessible uniquement via un login/psw. On l'appelle aussi espace d'administration du site. C'est le lieu où sont effectuées les modifications du contenu et le paramétrage du site.

<http://mon-domaine.com/wp-login.php>

Dashboard

C'est le point d'entrée du back-end. On y trouve un ensemble de widgets qui donnent différentes informations sur le site ou la communauté WordPress.

Appearance

Ce menu gère certains types de contenus de WordPress et son apparence. On y trouve des entrées vers le changement de thème, la personnalisation du design, la gestion des menus personnalisés, la gestion des widgets ...

Users

Ce menu liste tous les utilisateurs du site. C'est l'endroit qui permet de gérer les droits utilisateurs et leurs paramètres (mot de passe, informations personnelles, préférences).

Les 5 rôles dans WordPress

Administrateur :

C'est le rôle qui permet de tout faire sur WordPress.

Editeur :

Un éditeur a le droit de créer et d'éditer un article, une page lui appartenant ou pas.

Auteur :

Un auteur a le droit d'écrire un article et de le publier.

Contributeur :

A contributeur a le droit d'écrire un article mais pas de le publier.

Abonné :

C'est un membre basique du site. Il peut réagir à un article sans avoir besoin de se loguer.

Plugins

Les plugins étendent les fonctionnalités de WordPress. Ce menu liste les plugins et permet de les activer/désactiver, de les supprimer, de les mettre à jour.

Tools

Ce menu donne accès à différents outils d'administration comme l'import/export de données sous forme de fichier xml.

Settings

C'est l'endroit où l'on va paramétrer les fonctionnalités générales de WordPress.

Paramétrage

Détails du menu « Settings »

General Settings

- Titre et baseline du site.
- Adresse mail de l'administrateur où seront envoyées les notifications.
- Formats des dates et des heures qui seront affichés sur le site.

Writing Settings

- Paramétrage d'interprétation des emoticons :-)
- Correction automatique d'une mauvaise syntaxe XHTML.
- Choix de la catégorie par défaut.
- Choix du post format par défaut.
- Bookmarklet « Press This »

Détails du menu « Settings »

Reading Settings

- Choix du type de page à afficher en page d'accueil.
- Nombre de posts à afficher par page.
- Paramétrage des fils de discussion RSS.
- Paramétrage de l'indexation par les moteurs de recherche.

Discussion Settings

- Paramétrage des options sur les articles
- Paramétrage des options sur les commentaires.
- Paramétrage des options sur les notifications par email.
- Paramétrage d'affichage des types d'avatar.

Détails du menu « Settings »

Media Settings

- Choix des tailles d'images générées lors de leur téléchargement.

Permalink Settings

- Choix du format de réécriture des URLs.

Page d'accueil

Afficher une page statique

Par défaut dans WordPress, la page d'accueil affiche la liste des derniers articles publiés.

Comment faire si on veut afficher une page statique en page d'accueil ?

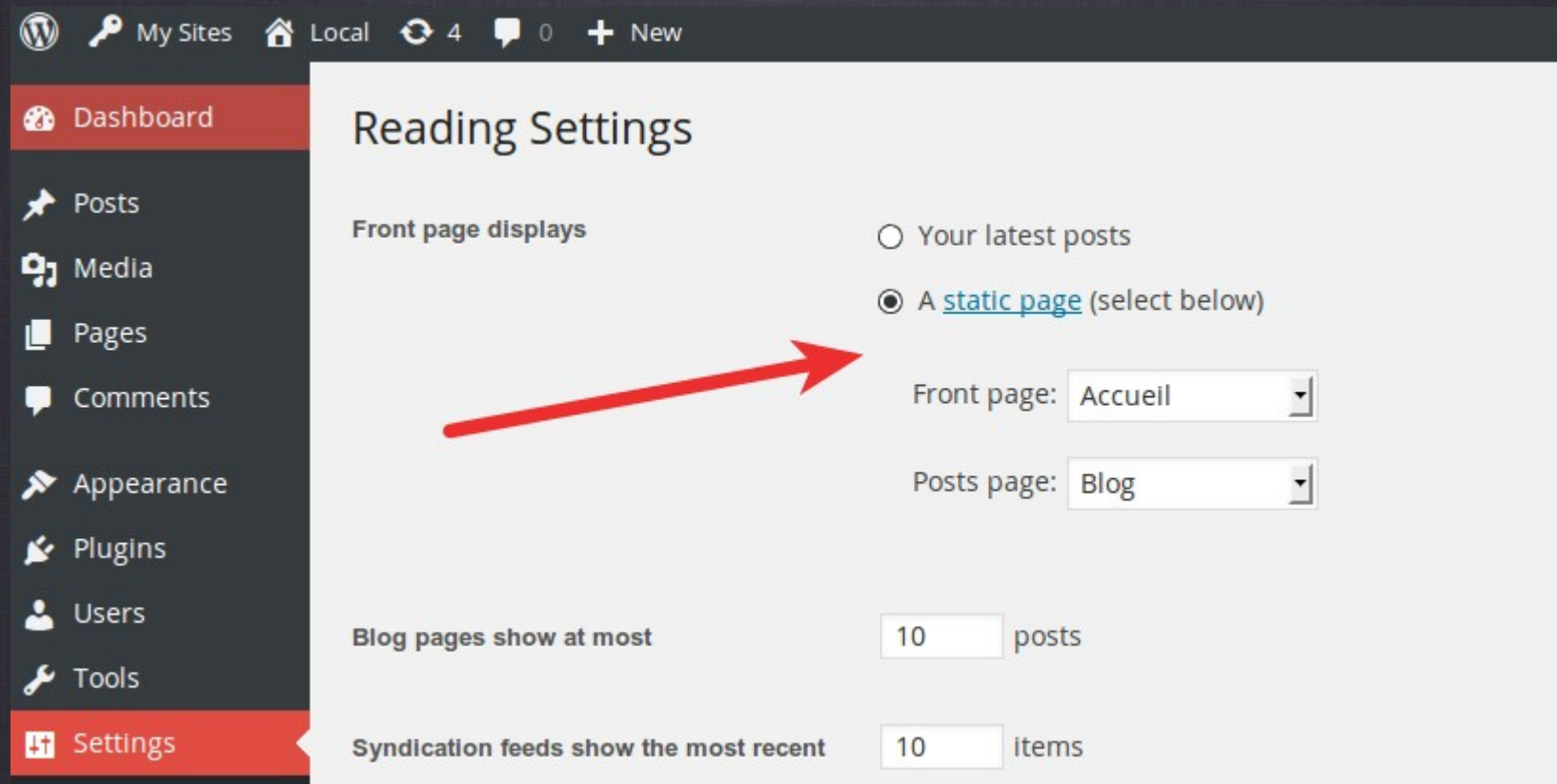
1- Dans la back-office de WordPress : ajouter une page « Accueil » et « Blog »

2- Dans la back-office de WordPress : Settings > Reading

Page d'accueil

Afficher une page statique

2- Dans la back-office de WordPress : Settings > Reading



Réalisation d'un mini-site à partir du backoffice

Le site devra comporter :

- Une page « Accueil »
- Une page « Actualités », articles avec des posts formats différents
- Une Page « Services » avec 3 sous pages « internet » « intranet » « extranet »
- Une page « A propos »

Un menu personnalisé :

« Accueil », « Services », « Actualités », « A propos »

L'item « Services » aura 3 sous-menus : internet, intranet, extranet

Les thèmes

Quelques mots sur les thèmes

Les thèmes sont composés d'un ensemble de fichiers html, css, js, php, image ...

Ils définissent la structure et le design front-end des pages web de WordPress.

Ils sont placés dans le répertoire [wordpress/wp-content/themes](#)

La qualité des thèmes trouvés sur internet est variable.

Où trouver des thèmes ?

« Dans le package de fichiers WordPress »

WordPress fournit dans ses fichiers des thèmes par défaut qui portent le nom des années :

2011 : twentyeleven,
2012 : twentytwelve,
2013 : twentythirteen,
2014 : twentyfourteen.

Ces thèmes sont développés par WordPress ou par des développeurs professionnels mandatés par WordPress.

Ce sont donc de bons exemples pour apprendre les bonnes pratiques de développement d'un thème WordPress.

Où trouver des thèmes ?

« Dans le dépôt des thèmes WordPress »

<http://wordpress.org/themes/>

WordPress met à disposition une galerie de thèmes en téléchargement (2300 thèmes).

Certains sont gratuits, d'autres sont payants.

Ces thèmes sont généralement de bonne qualité (code, sécurité) mais le design et les fonctionnalités minimalistes.

Où trouver des thèmes ?

« Les boutiques de thèmes payants »

Depuis 2008, des sociétés se sont spécialisées dans la création et la vente de thèmes WordPress.

Elles proposent des mises à jours pour une compatibilité avec les différentes versions de WordPress ainsi que le support utilisateur.

Les prix vont de \$20 à \$80 en moyenne.

Exemples :

Theme Foundry
Upthemes
Elegant Themes
Woothemes
Themify ...

Où trouver des thèmes ?

« Les galleries de thèmes gratuits »

On trouve également sur internet quelques galleries de thèmes gratuits.

Attention, certains thèmes gratuits sont de mauvaise qualité ou comportent des codes malicieux qui peuvent mettre en péril la sécurité de votre serveur/site.

Exemple de thèmes gratuits relativement sûre :

<http://www.wpexplorer.com/top-free-themes/>

Où trouver des thèmes ?

« Sur des sites de développeurs WordPress »

Certains développeurs distribuent leurs thèmes WordPress gratuitement sur internet.

On les trouve généralement sur leur propre site ou sur leur compte Github.

Attention, la qualité et la sécurité du code de ces thèmes peuvent varier.

Exemple :

<https://github.com/anthonydb/goshen-wordpress-theme>

Installer et activer un thème dans WordPress

Pour installer un nouveau thème dans WordPress, rien de plus simple.

1- Télécharger le thème (il est généralement contenu dans un répertoire portant son nom)

2- Placer le répertoire du thème dans le répertoire :
[wordpress/wp-content/themes/ ... ici ...](#)

3- Connectez-vous au back-office de votre site WordPress et activez le thème (menu : Apparence > Thèmes)

4- Chaque thème propose un layout et un design des pages différent. La richesse des options proposées dans le back-office dépendra également du thème.

Créer son propre thème

Pour répondre à 100% à vos besoins ou ceux de votre client, il peut être judicieux de créer son propre thème « from scratch ».

Créer un thème WordPress
nécessite des connaissances en :

HTML, CSS, JS, PHP

Passer en mode debug
wp-config.php

```
define('WP_DEBUG', true);
```

Créer son propre thème

« Trouver de la documentation »

La documentation officielle WordPress

- Codex : codex.wordpress.org

Les sites spécialisés WordPress

- Tutsplus : <http://code.tutsplus.com/categories/wordpress>
- Smashingmagazine : <http://wp.smashingmagazine.com/>
- WPRecipes : wprecipes.com
- WPExplorer : wpexplorer.com/blog/

Les sites de développeurs WordPress

- <http://kovshenin.com/>
- boiteaweb.fr/author/julio/
- wabeo.fr/

Créer son propre thème

« Trouver des réponses »

Les sites de « questions »

- wordpress.stackexchange.com/

Les podcasts

- WordPress TV : wordpress.tv
- Applyfilters : applyfilters.fm/

Les livres

- Wordpress : Toutes les clés pour créer, maintenir et faire évoluer votre site web de Xavier Borderie, Francis Chouquet et Amaury Balmer (Fr)
- Professional WordPress Plugin Development de Williams Brad, Richard Ozh, Tadlock Justin (En)

Créer son propre thème

« Anatomie d'un thème WordPress »

Un thème WordPress nécessite au minimum 2 fichiers pour fonctionner :

- style.css
- index.php

Mais pour réaliser un site plus élaboré, il faudra travailler avec davantage de fichiers. Par exemple pour des raisons d'optimisation, les parties récurrentes du site sont externalisées dans des fichiers php :

- header.php
- sidebar.php
- footer.php

Si le site nécessite des fonctions php, on a l'habitude de les regrouper dans un fichier spécifique :

- functions.php

header.php
get_header();

sidebar.php
get_sidebar();

footer.php *get_footer();*

Créer son propre thème

« Etape 1 »

Réaliser un thème WordPress nommé « lpatc » avec les fichiers suivant :

- style.css
- index.php
- screenshot.png (880X660)

Ligne de code html pour insérer la feuille de style « style.css » dans le fichier index.php :

```
<link rel="stylesheet" href="<?php echo get_template_directory_uri() ?>/style.css" type="text/css" />
```

Remarque : Ce n'est pas la bonne façon de faire pour insérer les fichiers CSS dans WordPress. Nous verrons plus loin la bonne méthode.

Créer son propre thème

« Etape 1 »

style.css

C'est la feuille de style CSS principale.

Elle contient en en-tête les informations obligatoires du thème sous forme de commentaires.

```
/*  
Theme Name: Your Theme (seul paramètre obligatoire)  
Theme URI: http://example.com/example/  
Description: Your thème description.  
Author: Your name  
Author URI: http://example.com/  
Version: 1.0  
Tags: Comma-separated tags that describe your theme  
  
Licence : GNU General Public License, version 2 (GPL).  
http://www.gnu.org/licenses/old-licenses/gpl-2.0.html  
*/
```

Créer son propre thème

« Etape 1 »

index.php

C'est le fichier template principal de WordPress. Il est obligatoire pour faire fonctionner un thème.

Ici le header ...

Ici mon bloc central ...

Ici, affichage du texte avec une fonction php "echo"

Ici le footer ...

OPEN FILES

index.php

FOLDERS

- wordpress
 - wp-content
 - themes
 - lpatc

index.php

screenshot.png

style.css

index.php

```
1  <?php
2  /**
3   * The main template file: index.php
4   *
5   * This is the most generic template file in a WordPress theme
6   * and one of the two required files for a theme (the other being style.css).
7   * It is used to display a page when nothing more specific matches a query.
8   *
9   * @link http://codex.wordpress.org/Template_Hierarchy
10  *
11  * @package WordPress
12  * @subpackage lpatc
13  * @since lpatc 1.0
14  */
15  ?>
16
17  <html>
18  <head>
19      <meta charset="utf-8" />
20      <title> Mon premier thème WordPress </title>
21      <link rel="stylesheet" href="<?php echo get_template_directory_uri() ?>/style.css" type="text/css" />
22  </head>
23
24  <body>
25
26      <header>
27          Ici le header ...
28      </header>
29
30      <div id="main">
31
32          <p>Ici mon bloc central ...</p>
33
34          <?php
35              echo '<p>Ici, affichage du texte avec une fonction php "echo"</p>';
36          ?>
37
38      </div>
39
40      <footer>
41          Ici le footer ...
42      </footer>
43
44  </body>
45
46  </html>
```

Créer son propre thème

« Etape 2 »

Externaliser dans des fichiers php distincts le code qui sera commun à toutes les pages du site.

Par exemple le header et le footer.

- style.css
- index.php
- screenshot.png (880X660)
- header.php
- footer.php

Dans la page index.php, appelez les fichiers header.php et footer.php à l'aide des fonctions WP :

```
get_header()  
get_footer()
```

Créer son propre thème

« Etape 2 »

Ici le header ... avec `"get_header()"`

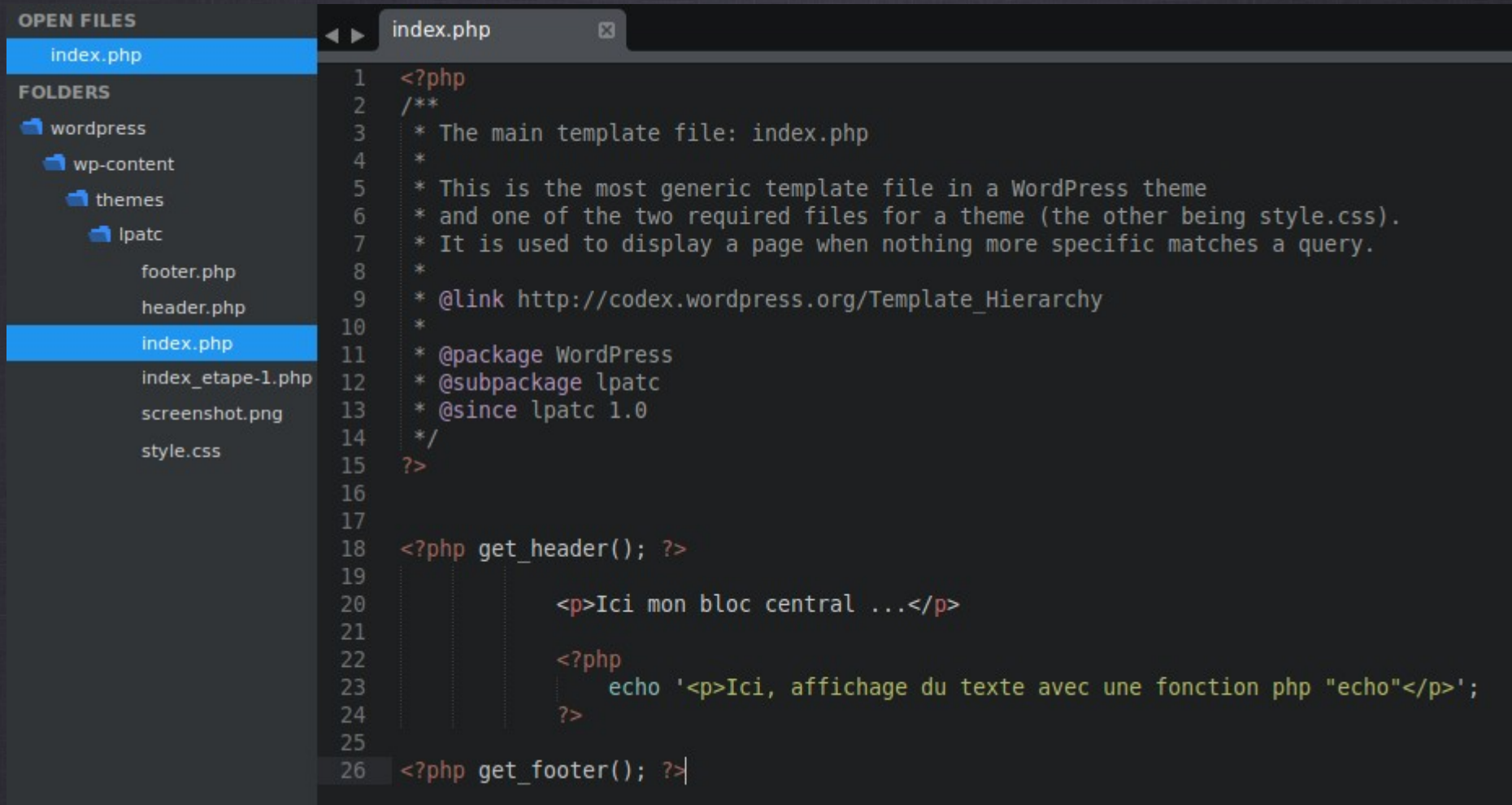
Ici mon bloc central ...

Ici, affichage du texte avec une fonction php `"echo"`

Ici le footer ... avec `"get_footer()"`

Créer son propre thème

« Etape 2 »



The image shows a code editor interface with a sidebar on the left and a main editing area on the right. The sidebar has two sections: 'OPEN FILES' and 'FOLDERS'. Under 'OPEN FILES', 'index.php' is listed and highlighted. Under 'FOLDERS', a tree structure shows 'wordpress' containing 'wp-content', which contains 'themes', which contains 'lpatc'. Below the folders, a list of files is shown: 'footer.php', 'header.php', 'index.php' (highlighted), 'index_etape-1.php', 'screenshot.png', and 'style.css'. The main editing area shows the content of 'index.php' with line numbers 1 through 26. The code is a PHP file for a WordPress theme, featuring a multi-line comment block with metadata and PHP calls to include header, content, and footer.

```
1 <?php
2 /**
3  * The main template file: index.php
4  *
5  * This is the most generic template file in a WordPress theme
6  * and one of the two required files for a theme (the other being style.css).
7  * It is used to display a page when nothing more specific matches a query.
8  *
9  * @link http://codex.wordpress.org/Template_Hierarchy
10 *
11 * @package WordPress
12 * @subpackage lpatc
13 * @since lpatc 1.0
14 */
15 ?>
16
17
18 <?php get_header(); ?>
19
20         <p>Ici mon bloc central ...</p>
21
22         <?php
23             echo '<p>Ici, affichage du texte avec une fonction php "echo"</p>';
24         ?>
25
26 <?php get_footer(); ?>
```

Créer son propre thème

« Etape 3 »

WordPress dispose d'un certain nombre de fonctions prédéfinies utilisables dans les fichiers php du thème. On parle aussi de « **Template tags** »

Codex : codex.wordpress.org/Function_Reference

Template tags : dbswebsite.com/design/wordpress-reference/v3/

Google : « Cheat Sheet WordPress »

Exemples déjà vus :

```
get_header()  
get_footer()  
get_template_directory_uri()
```

Autres exemples :

```
bloginfo('name')  
get_sidebar()  
get_calendar()  
...
```

Créer son propre thème

« Etape 3 »

Utilisez les fonctions WordPress « Template tags » pour :

codex.wordpress.org/Function_Reference

1- Afficher dans le header :

- Le titre du site
- La description du site

2- Afficher dans le footer :

- L'email de l'administrateur du site
- La version de WordPress utilisée

Créer son propre thème

« Etape 3 »

Local
Just another WordPress site

Ici mon bloc central ...

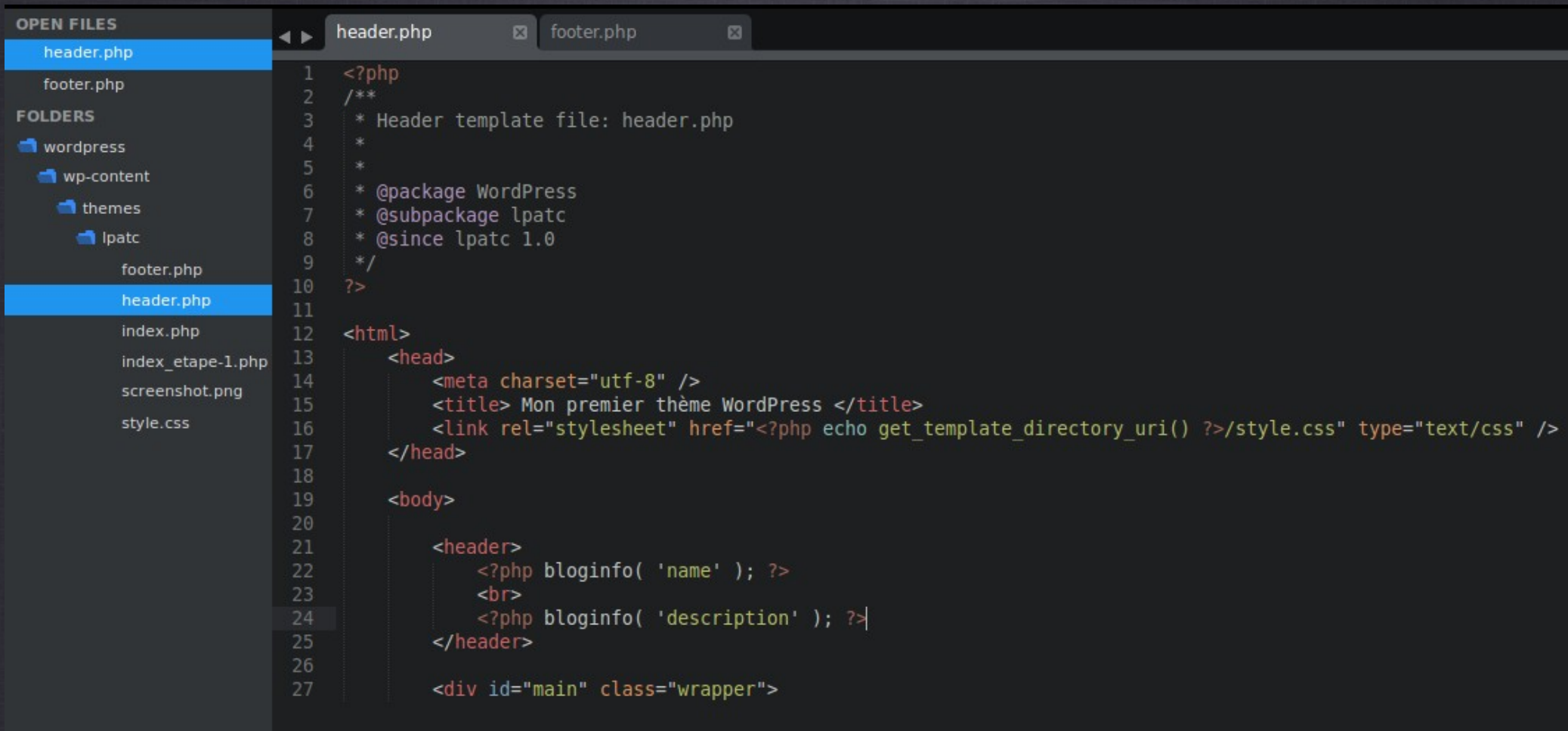
Ici, affichage du texte avec une fonction php "echo"

Contact administrateur : gillesvauvarin@gmail.com
Version WordPress : 3.8.1

Créer son propre thème

« Etape 3 »

header.php

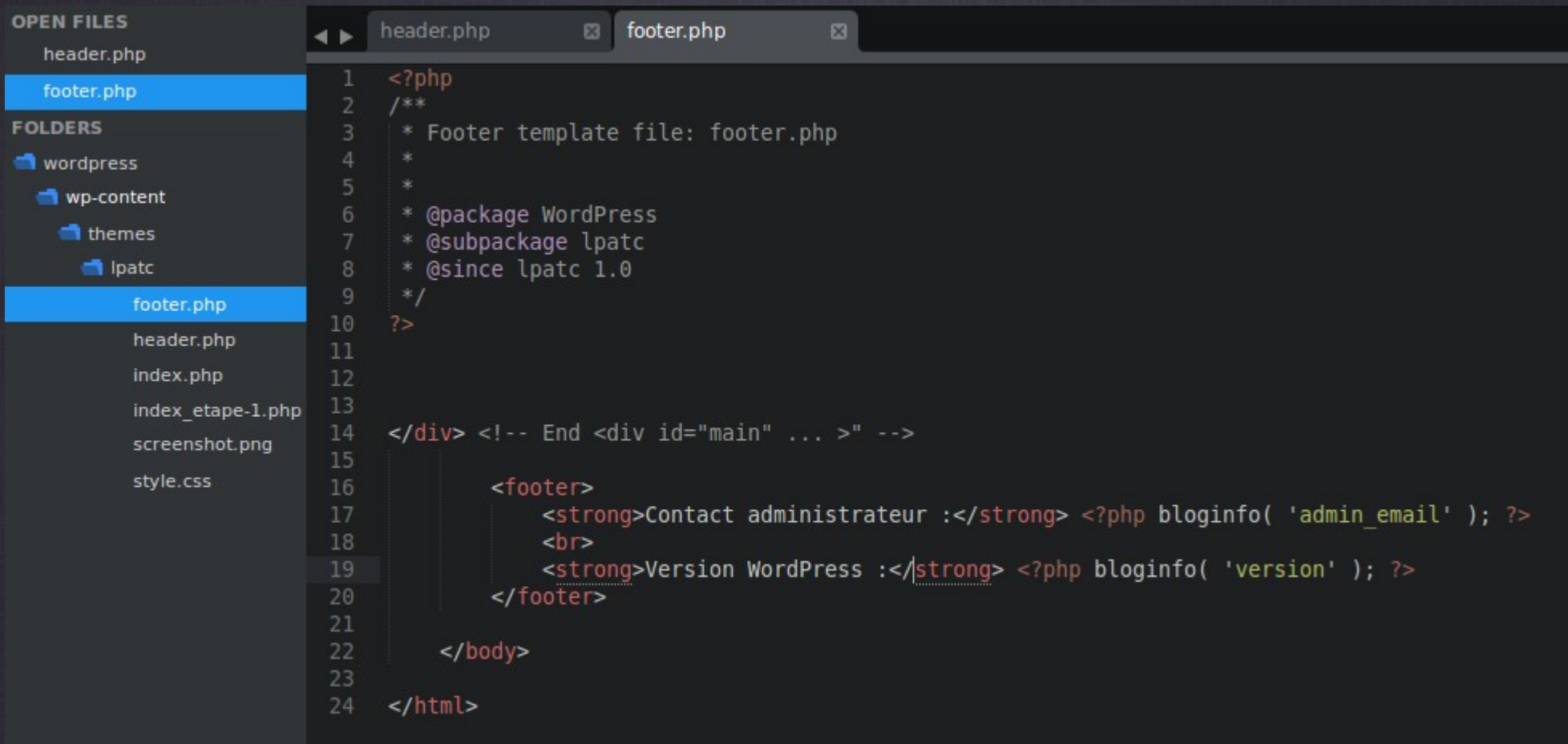


```
1 <?php
2 /**
3  * Header template file: header.php
4  *
5  *
6  * @package WordPress
7  * @subpackage lpatc
8  * @since lpatc 1.0
9  */
10 ?>
11
12 <html>
13     <head>
14         <meta charset="utf-8" />
15         <title> Mon premier thème WordPress </title>
16         <link rel="stylesheet" href="<?php echo get_template_directory_uri() ?>/style.css" type="text/css" />
17     </head>
18
19     <body>
20
21         <header>
22             <?php bloginfo( 'name' ); ?>
23             <br>
24             <?php bloginfo( 'description' ); ?>
25         </header>
26
27         <div id="main" class="wrapper">
```

Créer son propre thème

« Etape 3 »

footer.php

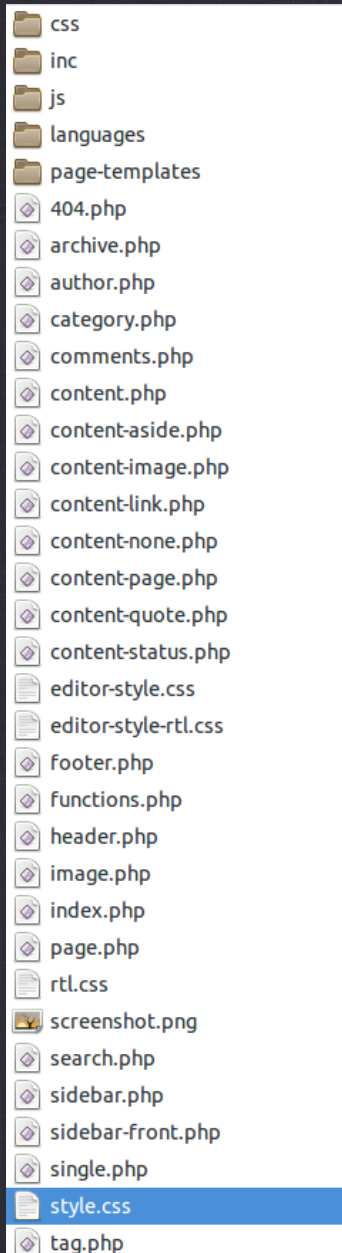


The screenshot shows a code editor with two tabs: 'header.php' and 'footer.php'. The 'footer.php' tab is active, displaying the following code:

```
1  <?php
2  /**
3   * Footer template file: footer.php
4   *
5   *
6   * @package WordPress
7   * @subpackage lpatc
8   * @since lpatc 1.0
9   */
10 ?>
11
12
13
14 </div> <!-- End <div id="main" ... >" -->
15
16     <footer>
17         <strong>Contact administrateur :</strong> <?php bloginfo( 'admin_email' ); ?>
18         <br>
19         <strong>Version WordPress :</strong> <?php bloginfo( 'version' ); ?>
20     </footer>
21
22 </body>
23
24 </html>
```

The left sidebar shows the 'OPEN FILES' and 'FOLDERS' panels. The 'FOLDERS' panel shows the directory structure: 'wordpress' > 'wp-content' > 'themes' > 'lpatc'. The 'OPEN FILES' panel shows the files in the 'lpatc' directory: 'header.php', 'footer.php' (selected), 'index.php', 'index_etape-1.php', 'screenshot.png', and 'style.css'.

Système de fichiers



Exemple avec le thème **Twentytwelve**

Remarque : les noms donnés aux fichiers sont spécifique à WordPress. On doit respecter des règles de nommage.

Exemples :

page.php

Fichier php de WordPress qui affiche les pages du site.

single.php

Fichier php de WordPress qui affiche un article.

404.php

Fichier php de WordPress qui affiche une page d'erreur 404 lorsque aucune page n'est trouvée.

Voir le Codex WordPress sur le développement des thèmes :
http://codex.wordpress.org/Theme_Development

Hiérarchie d'exécution des fichiers template de WordPress

Lorsque WordPress doit afficher une page du site, il utilise les fichiers template php présents dans le thème **en respectant une hiérarchie**.

Certains fichiers template sont utilisés sur toutes les pages :

- header.php
- footer.php
- sidebar.php

D'autres dans des cas spécifiques, c'est sur ces fichiers template que s'applique la hiérarchie :

- page.php
- single.php
- category.php

Hiérarchie d'exécution des fichiers template de WordPress

Exemple d'application de la hiérarchie WordPress sur l'affichage des « Pages »

Si WordPress doit afficher une « Page », il vérifiera dans le thème la présence des fichiers template suivant :

- > mon-template-page.php (priorité la plus haute)
- > page-{slug}.php
- > page-{id}.php
- > page.php
- > index.php (priorité la plus basse)

... et exécutera le fichier selon l'ordre de priorité affiché ci-dessus même si les autres fichiers sont présents.

https://codex.wordpress.org/Template_Hierarchy

Hiérarchie d'exécution des fichiers template de WordPress

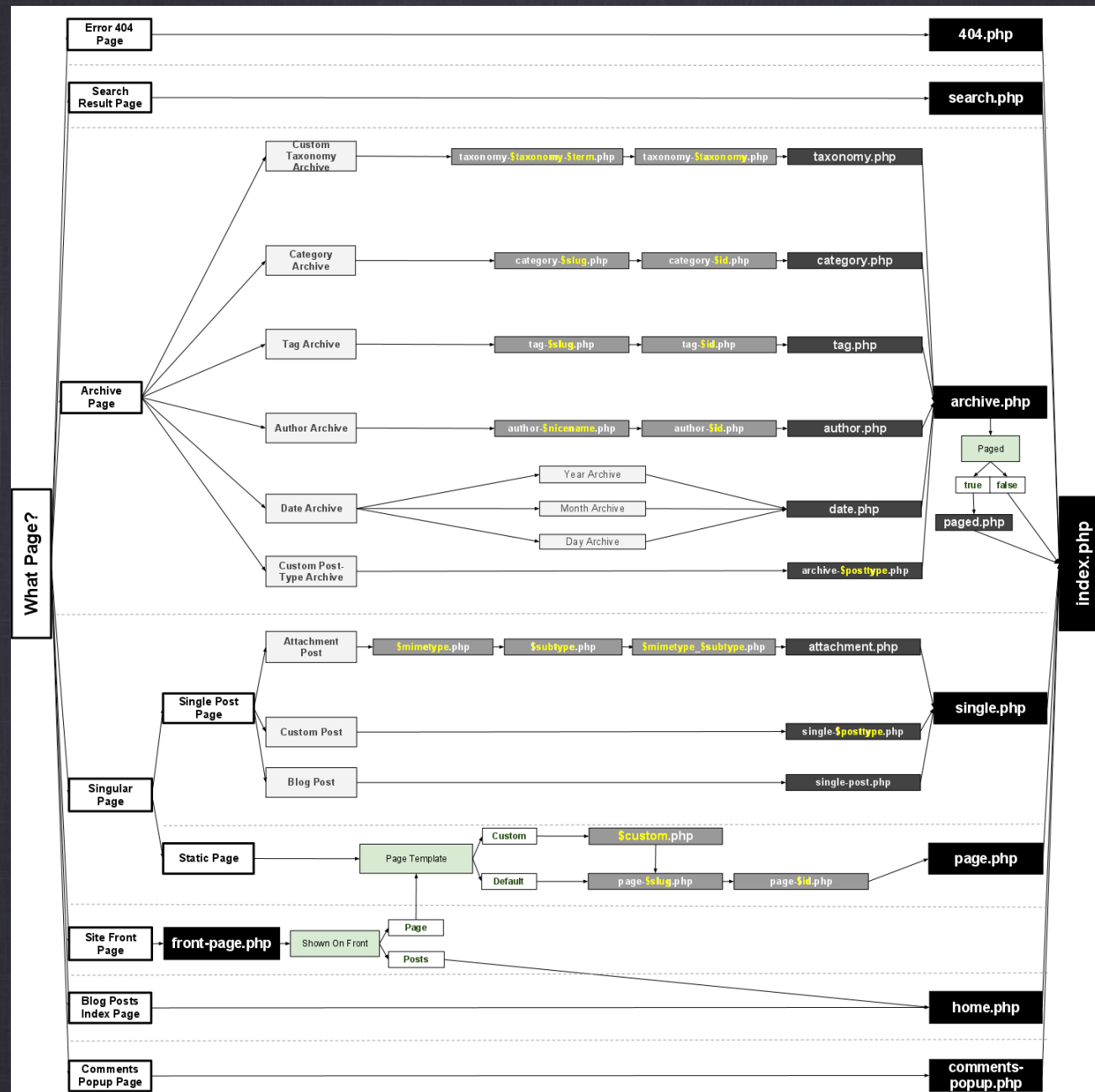
Exemple d'application de la hiérarchie WordPress sur l'affichage des « Pages »

`mon-template-page.php` (Template de Page personnalisé)
Utilisé pour afficher **un groupe** de pages de façon différente des
« Pages » classiques.

`page-{slug}.php`
`page-{id}.php`
Utilisé pour afficher **une page particulière** de façon différente des
« Pages » classiques.

`page.php`
Utilisé pour afficher **les pages classiques**

Hiérarchie d'exécution des fichiers template de WordPress



« Template de Page personnalisé »

Les « Pages » WordPress sont généralement affichées grâce à la présence du fichier page.php

Parfois vous aurez besoin **qu'un groupe de « Pages »** WordPress s'affichent et/ou fonctionnent différemment des autres pages.

Dans ce cas, il faut créer un/des « Template de Page personnalisé » et attribuer ce template de Page personnalisé, dans le backoffice, aux pages concernées (voir slide suivante).

Pour créer une « Template Page personnalisée » il faut créer un fichier php en déclarant en haut de page, dans un bloc de commentaire, le nom du template avec les mots clés
« Template Name : nom du template »

Exemple : themes/lptac/template-lptac.php

```
<?php
/*
Template Name : template lptac
*/
```

Code html, css, js, php ...

Dashboard

Posts

Media

Pages

All Pages

Add New

Comments

Appearance

Plugins

Users

Tools

Settings

Collapse menu

Edit Page [Add New](#)

Page exemple 2

Permalink: <http://127.0.0.1/wordpress/page-exemple-2/> [Edit](#) [View Page](#)

[Get Shortlink](#)

[Add Media](#)

[Add Template](#)

Visual

Text

b

i

[link](#)

b-quote

del

ins

img

ul

ol

li

code

more

[close tags](#)

[fullscreen](#)

Hello, I'm a WordPress page!
My name is "Page exemple 2"

**Attribution du
"Template de Page personnalisé"**



Publish

[Preview Changes](#)

Status: **Published** [Edit](#)

Visibility: **Public** [Edit](#)

Revisions: **2** [Browse](#)

Published on: **Mar 1, 2014 @ 18:26** [Edit](#)

[Move to Trash](#)

[Update](#)

Page Attributes

Parent

(no parent)

Template

template lptac

Order

0

Need help? Use the Help tab in the upper right of your screen.

Word count: 10

Last edited by admin0112 on March 1, 2014 at 6:34 pm

Revisions

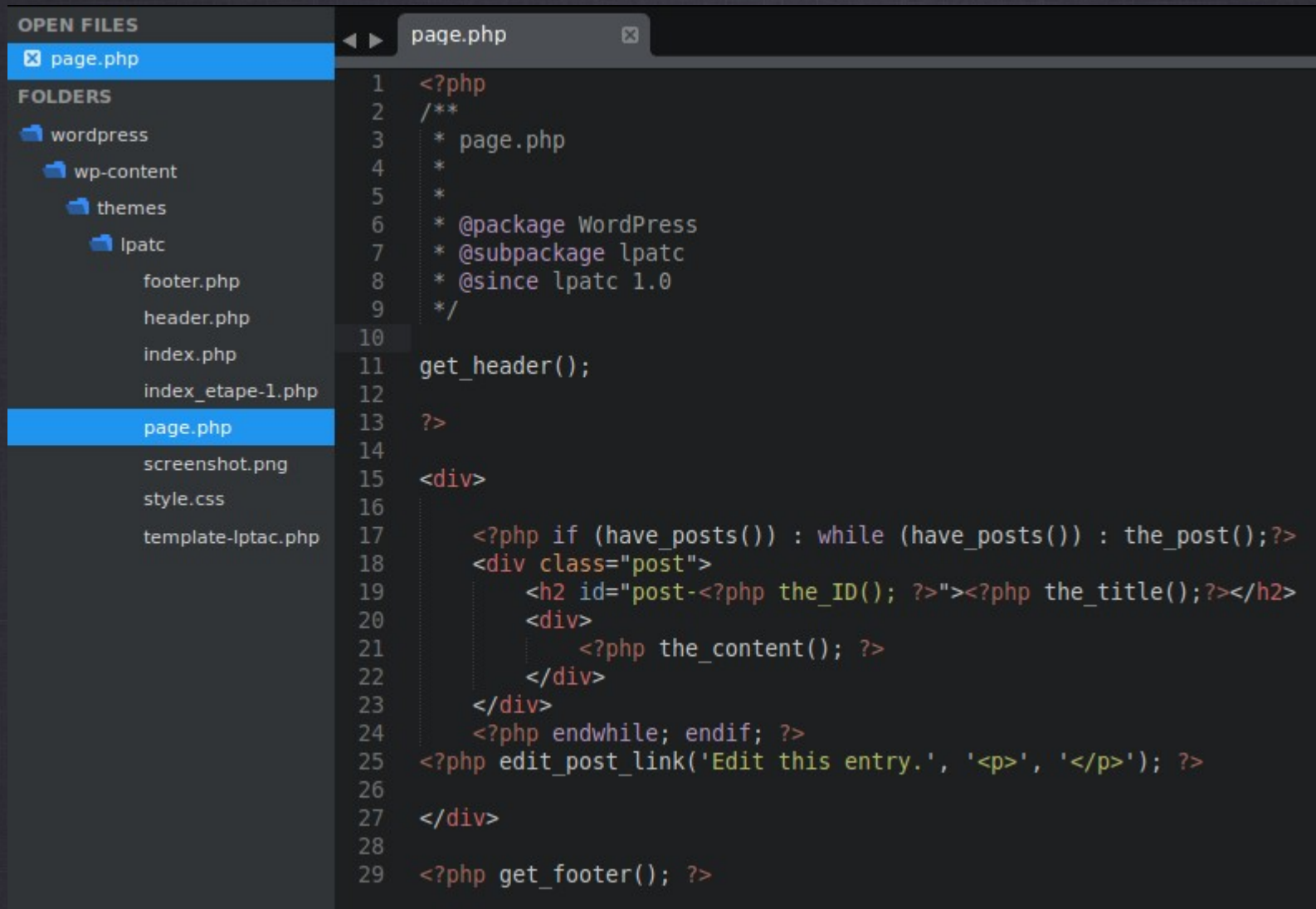
« Template de Page personnalisé »

Mise en pratique :

- 1- Créer une page dans le back-end de WordPress et afficher la page dans le navigateur.
> Qu'observe t'on ?
- 2- Créer un fichier page.php et afficher de nouveau la page dans le navigateur.
> Qu'observe t'on ?
- 3- Créer un fichier template-lpatc.php, attribuer ce template à la page dans le back-end et afficher de nouveau la page dans le navigateur.
> Qu'observe t'on ?

« Template de Page personnalisé »

page.php

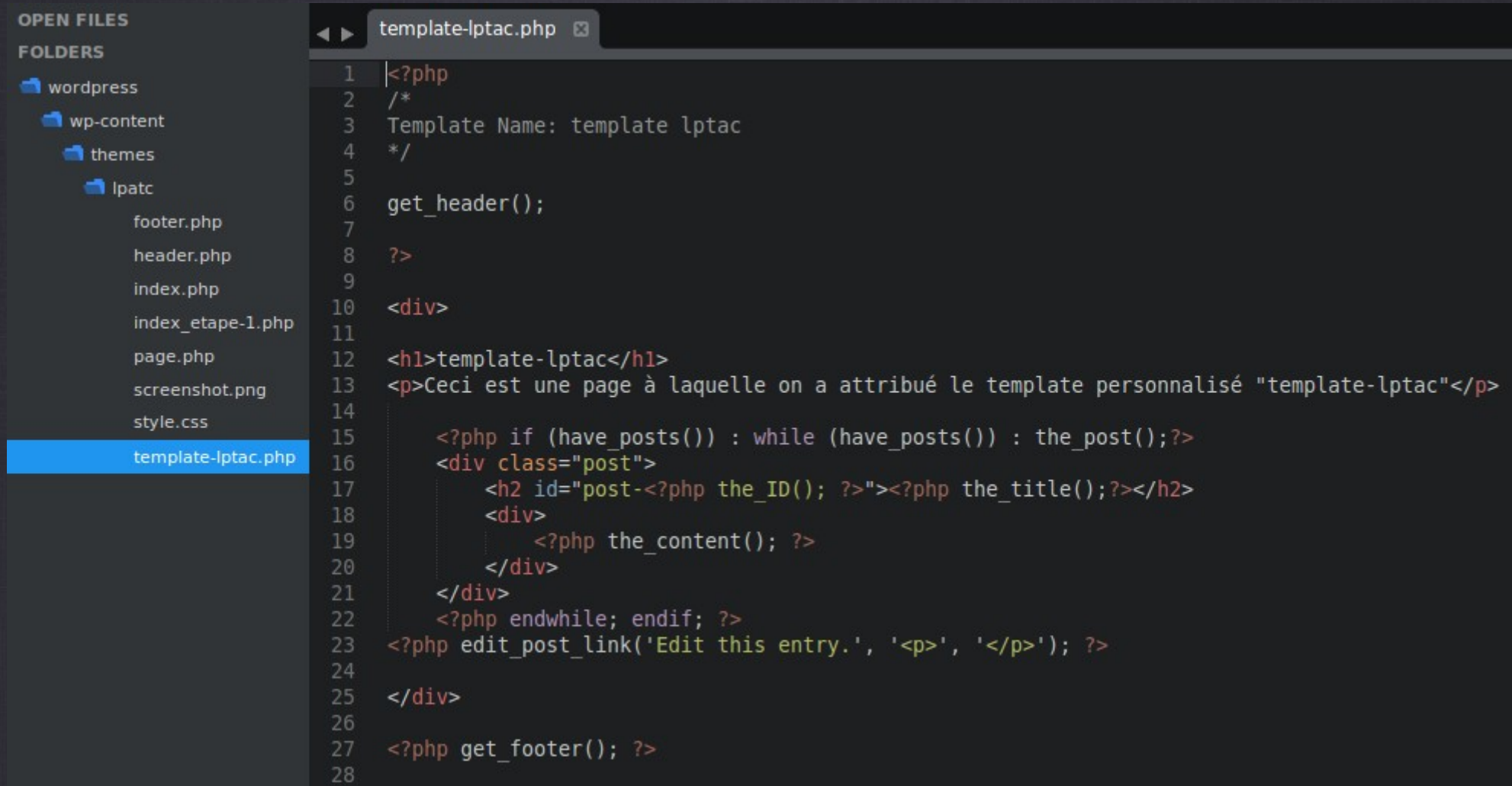


The image shows a code editor interface with a dark theme. On the left, there is a sidebar with two sections: 'OPEN FILES' and 'FOLDERS'. The 'OPEN FILES' section lists 'page.php' as the current file. The 'FOLDERS' section shows a tree structure: 'wordpress' > 'wp-content' > 'themes' > 'lpatc'. Below the folders, a list of files is shown, with 'page.php' highlighted. The main editor area displays the code for 'page.php'. The code starts with a PHP opening tag and a multi-line comment identifying the file as 'page.php' for the 'WordPress' package, 'lpatc' subpackage, and version '1.0'. It then calls 'get_header();'. A closing PHP tag follows. Then, a <div> tag is opened. Inside, a PHP conditional statement checks if there are posts and loops through them. For each post, it outputs an <h2> tag with the post ID and title, followed by a <div> containing the post content. After the loop, it calls 'edit_post_link()' to add an 'Edit this entry.' link. Finally, it closes the <div> and calls 'get_footer();'.

```
1 <?php
2 /**
3  * page.php
4  *
5  *
6  * @package WordPress
7  * @subpackage lpatc
8  * @since lpatc 1.0
9  */
10
11 get_header();
12
13 ?>
14
15 <div>
16
17     <?php if (have_posts()) : while (have_posts()) : the_post();?>
18     <div class="post">
19         <h2 id="post-<?php the_ID(); ?>"><?php the_title();?></h2>
20         <div>
21             <?php the_content(); ?>
22         </div>
23     </div>
24     <?php endwhile; endif; ?>
25 <?php edit_post_link('Edit this entry.', '<p>', '</p>'); ?>
26
27 </div>
28
29 <?php get_footer(); ?>
```


« Template de Page personnalisé »

template-lptac.php



The image shows a code editor interface. On the left, a sidebar displays the file structure under 'FOLDERS':

- wordpress
 - wp-content
 - themes
 - lpatc
 - footer.php
 - header.php
 - index.php
 - index_etape-1.php
 - page.php
 - screenshot.png
 - style.css
 - template-lptac.php (selected)

The main editor area shows the content of 'template-lptac.php' with line numbers 1 through 28:

```
1 <?php
2 /*
3 Template Name: template lptac
4 */
5
6 get_header();
7
8 ?>
9
10 <div>
11
12 <h1>template-lptac</h1>
13 <p>Ceci est une page à laquelle on a attribué le template personnalisé "template-lptac"</p>
14
15     <?php if (have_posts()) : while (have_posts()) : the_post();?>
16     <div class="post">
17         <h2 id="post-<?php the_ID(); ?>"><?php the_title();?></h2>
18         <div>
19             <?php the_content(); ?>
20         </div>
21     </div>
22     <?php endwhile; endif; ?>
23 <?php edit_post_link('Edit this entry.', '<p>', '</p>'); ?>
24
25 </div>
26
27 <?php get_footer(); ?>
28
```

Hiérarchie d'exécution des fichiers template de WordPress

Afficher **une page particulière** de façon différente des « Pages » classiques :

1- Repérer dans le back-end l'id de la page d'exemple : ID = 2

2- Créer le fichier « page-2.php » (copier coler le code de page.php et ajouter le code html

```
<h1>Page ID = 2</h1>
```

3- Afficher cette page (id=2) dans votre navigateur

> Qu'observe t'on ?

4- Attribuer dans le back-end WordPress le template « template-lptac » à la page d'id = 2 et afficher de nouveau la page dans votre navigateur.

> Qu'observe t'on ?

Les boucles

Les boucles sont le cœur de l'affichage des contenus dans WordPress.

Elles permettent de charger et d'afficher du contenu plus ou moins spécifique (tous les articles, tous les articles d'une catégorie, un seul article, une page, un ou des custom posts type etc.).

WordPress sait quoi afficher et comment l'afficher **grâce aux informations transmises par l'URL.**

Anatomie d'une boucle de base : (en bleue, le code spécifique WordPress)

```
<?php

// The Loop
if ( have_posts() ) :
    while ( have_posts() ) : the_post();
        // J'affiche mon contenu ...
    endwhile;
else:
    // Il n'y a pas de contenu ...
endif;

?>
```


Les boucles

```
<?php

// The Loop
if ( have_posts() ) : -----> Est qu'il y a du contenu ? Si oui ...
    while ( have_posts() ) : the_post(); -----> Boucle sur le contenu tant qu'il y en a
        // J'affiche mon contenu ...
    Endwhile; -----> Quand il n'y a plus de contenu, je ferme la boucle
Else: -----> Si il n'y a pas de contenu
    // Il n'y a pas de contenu ...
endif;

?>
```

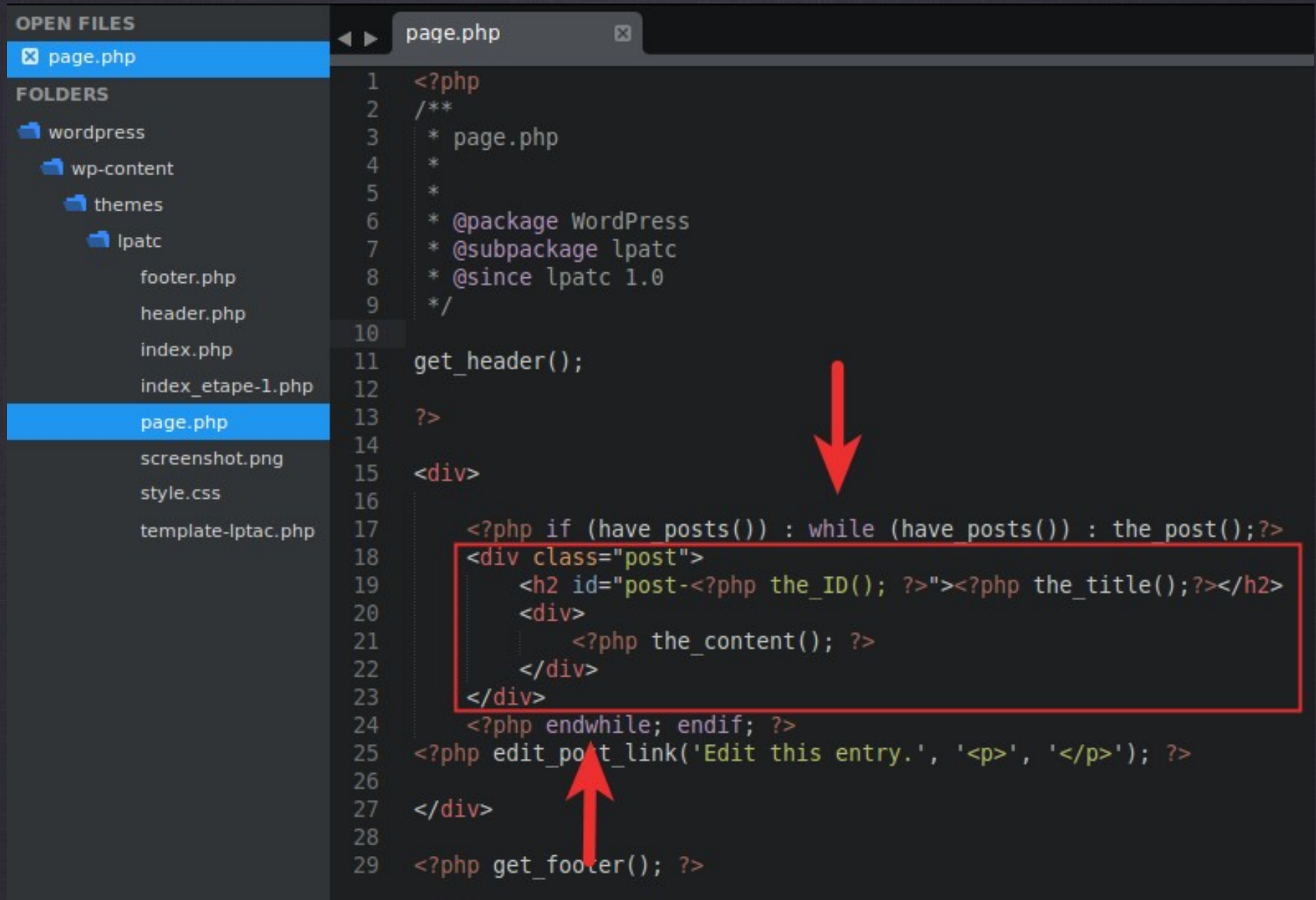
Remarque :

Pour les articles, le nombre affiché par la boucle est définie dans les réglages de WordPress (Settings > Reading).

Si le nombre d'articles a afficher est supérieur au nombre définie dans WordPress, il faut prévoir l'affichage d'une pagination pour accéder aux autres articles ou il faut augmenter ce nombre pour afficher davantage d'articles sur la page.

Les boucles

page.php



```
1  <?php
2  /**
3   * page.php
4   *
5   *
6   * @package WordPress
7   * @subpackage lpatc
8   * @since lpatc 1.0
9   */
10
11  get_header();
12
13  ?>
14
15  <div>
16
17      <?php if (have_posts()) : while (have_posts()) : the_post();?>
18      <div class="post">
19          <h2 id="post-<?php the_ID(); ?>"><?php the_title();?></h2>
20          <div>
21              <?php the_content(); ?>
22          </div>
23      </div>
24      <?php endwhile; endif; ?>
25      <?php edit_post_link('Edit this entry.', '<p>', '</p>'); ?>
26
27  </div>
28
29  <?php get_footer(); ?>
```

Boucles : les « template tags »

Voici quelques fonctions WordPress (Template tags) disponibles uniquement à l'intérieur d'une boucle :

the_title()
the_content()
the_ID()
the_time()
the_date()
the_category()
the_author()
the_permalink()
the_excerpt()
...

Ces fonctions **affichent** la donnée empactée ou pas dans du code HTML.

Exemple :

the_title() et the_ID() renvoient la donnée brute
the_content() renvoie la donnée + du HTML

La majorité de ces fonctions acceptent des paramètres, qui permettent en général de modifier le code HTML qu'elles génèrent.

Boucles : les « template tags »

Il est aussi possible de récupérer la donnée (et non de l'afficher directement) pour la stocker dans une variable ou l'afficher sans les balises HTML générées par WordPress :

```
<?php

// Récupération de la donnée « titre » (nous sommes à l'intérieur d'une boucle)
$titre = get_the_title() ;
// Affichage de la donnée « titre » brut
echo $titre ;

?>
```

Ces fonctions « get_ » peuvent être utilisées en dehors d'une boucle à condition d'indiquer en paramètre l'ID de l'article ou de la page.

Pour plus de détails sur les « template tags » :

- Codex http://codex.wordpress.org/Template_Tags#Post_tags
- Dans les WordPress Cheat Sheet

Personnaliser la boucle courrante

WordPress, à partir des paramètres indiqués dans l'URL, construit une requête SQL qui sera exécutée pour récupérer les bonnes données dans la base. Ces données seront ensuite affichées grâce à la boucle.

Pour des **besoins spécifiques**, il se peut que vous deviez personnaliser la boucle courrante pour afficher les données selon certains critères supplémentaires.

Exemples de besoins spécifiques que vous pourriez avoir :

- Afficher les articles d'une seule catégorie.
- Afficher les articles sauf ceux dont l'ID est 1, 2 et 10.
- Afficher seulement les 2 articles les plus récents.
- Afficher les articles dans un ordre aléatoire.
- Afficher les articles publiés entre telle date et telle date.
- etc ...

Personnaliser la boucle courrante

Utiliser le hook « `pre_get_posts` »

Grossièrement, voici les étapes pour afficher des données de la base lorsque l'on charge une page WordPress :

- 1- Chargement d'une page spécifique caractérisée par son URL.
- 2- WordPress analyse l'URL pour en extraire les paramètres.
- 3- WordPress construit à partir des paramètres une requête qui servira à extraire les bonnes données de la base de données.

Hook « `pre_get_posts` »

- 4- WordPress exécute la requête et stocke les données dans une variable.
- 5- Une boucle WordPress permet d'afficher les données dans un template.

Personnaliser la boucle courrante

Utiliser le hook « pre_get_posts »

Un hook est un « ancrage » (une porte d'entrée) dans le code de WordPress pour exécuter le code d'une fonction.

Dans le code de WordPress, il existe un « hook » entre l'étape 3 et 4, c'est à dire juste après la construction de la requête mais juste avant son exécution sur la base de données.

En utilisant ce « hook » nous pouvons intercepter la requête et la modifier (via une fonction) avant qu'elle s'exécute.

Remarques :

- Il faut s'assurer que l'on intercepte bien la requête principale.
`$query->is_main_query()` codex.wordpress.org/Function_Reference/is_main_query
- Il faut cibler les pages sur lesquelles on veut modifier la requête sinon la modification s'appliquera sur toutes les pages en front et back.
Utilisation des « conditionnal tags » codex.wordpress.org/Conditional_Tags

Personnaliser la boucle courrante

Utiliser le hook « pre_get_posts »

```
<?php
```

```
// Définition de la fonction « lptac_alter_query »
```

```
// « pre_get_posts » nous permet de passer l'objet $query en paramètre
```

```
function lptac_alter_query( $query ) {
```

```
    // Vérifie que la variable globale $query contient bien la requête principale et que  
    nous sommes bien sur le front-end sur la page désignée comme « blog »
```

```
    if ( $query->is_main_query() && ! is_admin() && is_home() ) {
```

```
        // Modification de la requête
```

```
        $query->set( 'paramètre' , valeur );
```

```
    }
```

```
}
```

```
// Execute la fonction « lptac_alter_query » sur le hook « pre_get_query »
```

```
add_action( 'pre_get_posts' , 'lptac_alter_query' );
```

```
?>
```

Personnaliser la boucle courrante

Utiliser le hook « pre_get_posts »

```
<?php

function lptac_alter_query( $query ) {

    if ( $query->is_main_query() && ! is_admin() && is_home() ) {
        $query->set( 'posts_per_page' , 2 );
    }

}

add_action( 'pre_get_posts' , 'lptac_alter_query' );

?>
```


Personnaliser la boucle courrante

Utiliser le hook « `pre_get_posts` »

Où trouver les paramètres/valeurs que l'on peut modifier dans la requête ?

```
$query->set( 'paramètre' , valeur );
```

Regarder du côté de la Classe « `WP_Query` » :

<https://gist.github.com/luetkemj/2023628>

http://codex.wordpress.org/Plugin_API/Action_Reference/pre_get_posts

http://codex.wordpress.org/Class_Reference/WP_Query

Personnaliser la boucle courrante

Utiliser le hook « `pre_get_posts` »

Mise en pratique :

1- Dans la back-end WordPress, ajouter 4 articles :

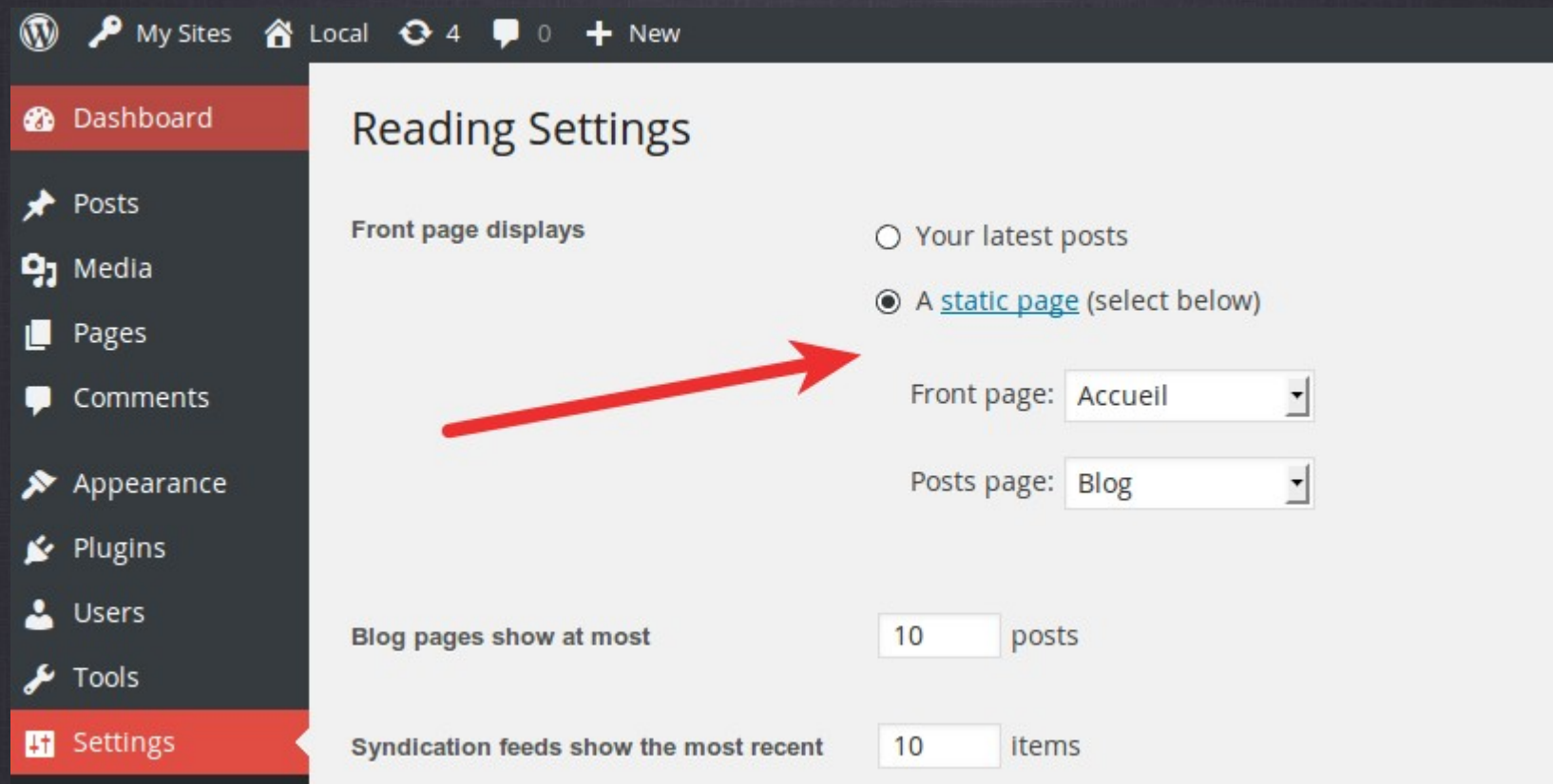
Titre = Article 1, 2, 3, 4 / Contenu = Lorem ipsum

2- Créer les fichiers php « `home.php` » et « `blog.php` » dans votre thème.
Faire une copie de `page.php` qui contient la boucle classique.

3- Dans la back-end WordPress : ajouter une page « Accueil » et « Blog »

Personnaliser la boucle courrante

4- Dans la back-end WordPress : Settings > Reading indiquer la page « Accueil » comme étant la page front et « blog » comme étant la page d'affichage des articles.



Personnaliser la boucle courrante

5- Afficher la page « blog » dans le navigateur

Qu'observez-vous ?

Le fichier « functions.php »

Le fichier functions.php se place à la racine du thème avec les autres templates php. On y place des instructions php ou des déclarations de fonctions.

Il est chargé automatiquement par WordPress et peut être utilisé dans plusieurs cas, parmi lesquels :

- Activation de fonctionnalités du thème prévues par WordPress (déclaration de menus personnalisé, déclaration de Custom Post Type, déclaration de widget area etc.),
- Définition des fonctions personnalisées pour les intégrer ensuite dans vos modèles de page.
- Appel des fichiers CSS et JS

Etc ...

Personnaliser la boucle courrante

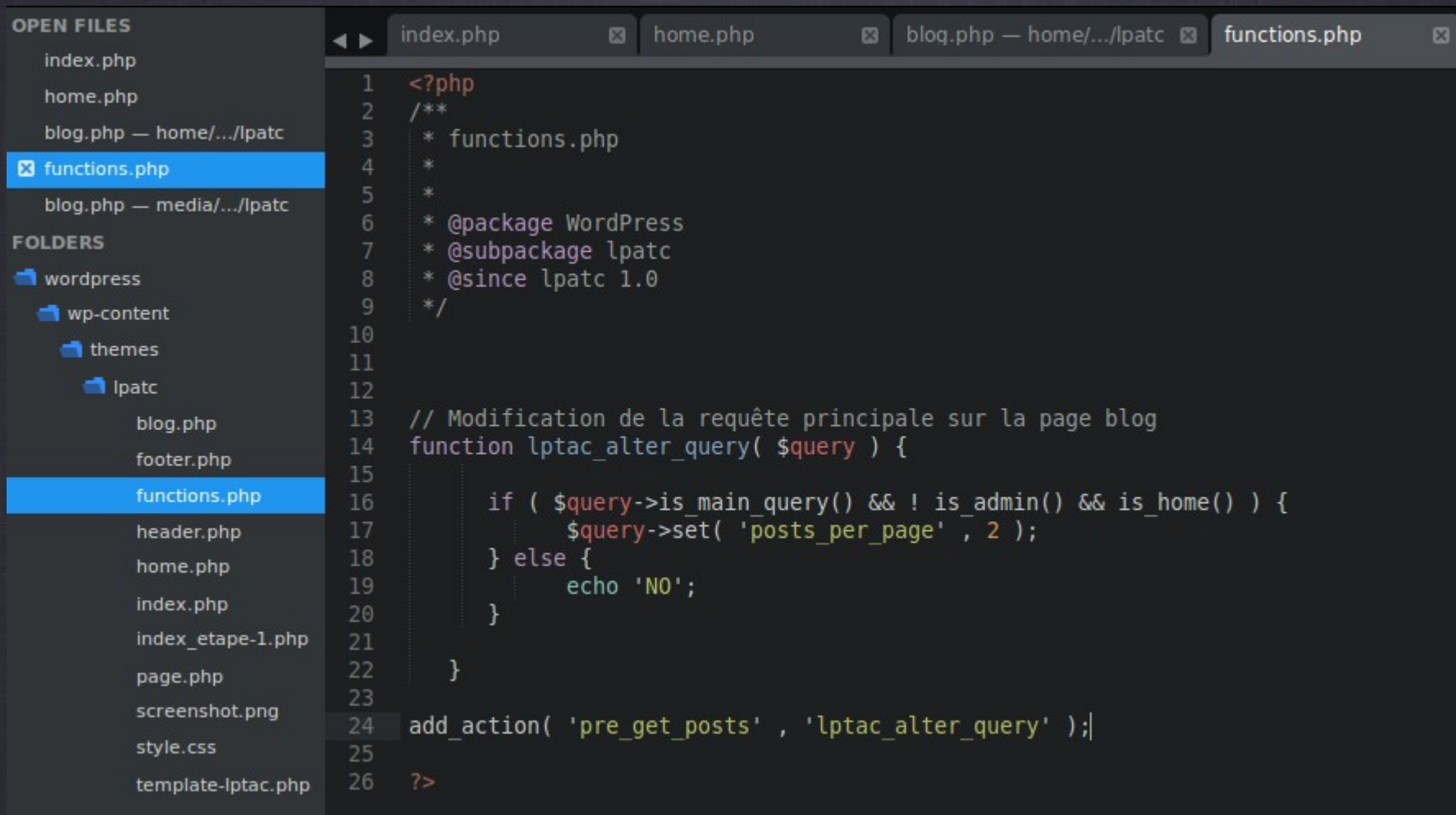
6- Modifier la requête principale uniquement sur la page « blog » pour n'afficher que 2 articles. Le code pour modifier la requête doit se trouver dans le fichier « functions.php »

7- Afficher la page « blog » dans le navigateur

Qu'observez-vous ?

Personnaliser la boucle courrante

functions.php



```
1  <?php
2  /**
3   * functions.php
4   *
5   *
6   * @package WordPress
7   * @subpackage lpatc
8   * @since lpatc 1.0
9   */
10
11
12
13  // Modification de la requête principale sur la page blog
14  function lptac_alter_query( $query ) {
15
16      if ( $query->is_main_query() && ! is_admin() && is_home() ) {
17          $query->set( 'posts_per_page' , 2 );
18      } else {
19          echo 'NO';
20      }
21
22  }
23
24  add_action( 'pre_get_posts' , 'lptac_alter_query' );
25
26  ?>
```

Boucles multiples

Généralement on utilise qu'une seule boucle par template php mais il se peut que vous ayez besoin de plusieurs boucles dans un même template php.

Cela peut être le cas par exemple si votre page inclut une « sidebar » ou un « footer » et que vous voulez afficher des données différentes dans votre sidebar/footer de celles affichées dans la boucle principale du corps de la page.

Les requêtes dans WordPress sont gérées par la Classe « **WP_Query** »
http://codex.wordpress.org/Class_Reference/WP_Query

Pour effectuer plusieurs boucles dans un template php WordPress, il faut créer un nouvel objet en instanciant la classe WP_Query.

Cette nouvelle requête sera définie par les paramètres que nous passerons en arguments à la classe WP_Query **et non plus par les paramètres présents dans l'URL.**

Boucles multiples

Création d'une nouvelle requête (en plus de la requête courrante) par instantiation de la classe WP_Query.

```
<?php
```

```
// 1- Définition des paramètres de sélection de la requête  
$args = array( 'post_type' => 'post' );
```

```
// 2- Instantiation de la classe WP_Query  
$the_query = new WP_Query( $args );
```

```
// 3- La boucle  
while ( $the_query->have_posts() ) :  
    $the_query->the_post();  
    echo '<h2>' . get_the_title() . '</h2>';  
endwhile;
```

```
// 4- Restaure la variable globale $post dans la requête principale  
wp_reset_postdata();
```

```
?>
```


Création de deux nouvelles requêtes (en plus de la requête courrante) par instantiation de la classe WP_Query.

```
<?php
```

```
// Requête 1
$args1 = array( 'post_type' => 'post' );
$query1 = new WP_Query( $args1 );
while ( $query1->have_posts() ) :
    $query1->the_post();
    echo '<h2>' . get_the_title() . '</h2>';
endwhile;
wp_reset_postdata();
```

```
// Requête 2
$args2 = array( 'post_type' => 'post' );
$query2 = new WP_Query( $args2 );
while ( $query2->have_posts() ) :
    $query2->the_post();
    echo '<h2>' . get_the_title() . '</h2>';
endwhile;
wp_reset_postdata();
```

```
?>
```

Boucles multiples

Mise en pratique :

1- Sur la page d'accueil, afficher le contenu de la Page « Accueil » et une liste de 3 articles dans le footer.

home.php
blog.php — home/.../lpatc
functions.php
footer.php
blog.php — media/.../lpatc

FOLDERS

- wordpress
 - wp-content
 - themes
 - lpatc**
 - blog.php
 - footer.php**
 - functions.php
 - header.php
 - home.php
 - index.php
 - index_etape-1.php
 - page.php
 - screenshot.png
 - style.css
 - template-lptac.php

```

1  <?php
2  /**
3   * Footer template file: footer.php
4   *
5   *
6   * @package WordPress
7   * @subpackage lpatc
8   * @since lpatc 1.0
9   */
10 ?>
11
12
13
14 </div> <!-- End <div id="main" ... >" -->
15
16     <footer>
17         <strong>Contact administrateur :</strong> <?php bloginfo( 'admin_email' ); ?>
18         <br>
19         <strong>Version WordPress :</strong> <?php bloginfo( 'version' ); ?>
20     </div>
21
22     <?php
23
24         // Définition des paramètres de sélection de la requête
25         $args = array(
26             'post_type'      => 'post',|
27             'posts_per_page' => 3
28         );
29
30         // Instanciation de la classe WP_Query
31         $the_query = new WP_Query( $args );
32
33         // La boucle
34         while ( $the_query->have_posts() ) :
35             $the_query->the_post();
36             echo '<h2>' . get_the_title() . '</h2>';
37             echo '<p>' . get_the_content() . '</p>';
38         endwhile;
39
40         // Restaure la variable globale $post dans la requête principale
41         wp_reset_postdata();
42
43     ?>
44
45     </div>
46 </footer>
47
48 </body>
49
50 </html>

```


get_template_part()

Nous avons déjà vu que pour factoriser des portions de code redondantes (header, footer, sidebar ...) WordPress offrait des fonctions qui nous permettait d'appeler dans nos template php des ces portions de code dans des fichiers externes.

`get_header()` > pour appeler le fichier header.php

`get_footer()` > pour appeler le fichier footer.php

La fonction « `get_template_part()` » nous permet de faire la même chose pour n'importe quel morceau de code que nous souhaiterions externaliser. Par exemple une boucle que nous utiliserions dans plusieurs fichiers template php.

codex.wordpress.org/Function_Reference/get_template_part

get_template_part()

Utilisation :

```
get_template_part( 'loop' );           // insertion de loop.php
get_template_part( 'loop' , '2' );      // insertion de loop-2.php
get_template_part( 'loop' , 'template' ); // insertion de loop-template.php
get_template_part( 'loop' , 'template-2' ); // insertion de loop-template-2.php
```

get_template_part()

Mise en pratique :

1- Dans le fichier de Page Template « template-lptac.php », utiliser la fonction « get_template_part() » pour externaliser la boucle dans un fichier « loop-template.php »

template-lptac.php

home.php
blog.php — home/.../lpatc
functions.php
footer.php
template-lptac.php
blog.php — media/.../lpatc

FOLDERS

wordpress
wp-content
themes
lpatc
blog.php
footer.php
functions.php
header.php

```
1  <?php
2  /*
3  Template Name: template lptac
4  */
5
6  get_header();
7
8  ?>
9
10 <div>
11
12 <h1>template-lptac</h1>
13 <p>Ceci est une page à laquelle on a attribué le template personnalisé "template-lptac"</p>
14
15 <?php get_template_part( 'loop' , 'template' ); ?>
16
17 </div>
18
19 <?php get_footer(); ?>
20
```

loop-template.php

home.php
blog.php — home/.../lpatc
functions.php
footer.php
template-lptac.php
blog.php — media/.../lpatc

FOLDERS

wordpress
wp-content
themes
lpatc
blog.php

```
1  <?php
2  /*
3  loop-template.php : Utilisation de la fonction "get_template_part()" voir fichier "template-lptac.php"
4  */
5  ?>
6
7  <h2>Utilisation de "get_template_part()"</h2>
8
9  <?php if (have_posts()) : while (have_posts()) : the_post();?>
10 <div class="post">
11     <h2 id="post-<?php the_ID(); ?>"><?php the_title();?></h2>
12     <div>
13         <?php the_content(); ?>
14     </div>
15 </div>
16 <?php endwhile; endif; ?>
```

Les « Hook »

Un « hook » (crochet, hameçon) est un point d'entrée dans le code de WordPress pour permettre aux développeurs de thèmes ou de plugins d'insérer leurs propres lignes de code.

Le code de WordPress 3.8 possède 2255 hooks !

Ces hooks permettent aux développeurs :

- De déclencher une action particulière à un moment précis du flux d'exécution du code de WordPress.

Utilisation des Hooks de type « Action »

- D'intercepter une valeur (destinée à être affichée à l'écran ou insérée dans la base de données), de la modifier et de la réinjecter dans le flux d'exécution du code de WordPress.

Utilisation des Hooks de type « Filtre »

Les « Hook »

Les hooks de type « Action »

(* arguments obligatoires)

Ajout de la fonction sur le hook :

```
add_action( $tag, $function_to_add, $priority, $accepted_args );
```



Nom du hook *



Nom de la
fonction *



Priorité
d'exécution
1 forte / 20 faible



Nombre
d'arguments
acceptés

Exécute la fonction qui a été spécifiée dans le « add_action » :

```
do_action( $tag, $args1, $args2 ... );
```



Nom du hook *



Argument

Les « Hook »

Exemple :

Executer une fonction qui affiche le message « Hello dude ! » sur le hook WordPress « wp_head »

```
function lptac_display_hello() {  
    echo 'Hello dude!';  
}  
  
add_action( 'wp_head' , 'lptac_display_hello' , 10 );
```

Nom du hook à cibler

Nom de la fonction à utiliser

Priorité d'exécution

Les « Hook »

Les hooks de type « Action »

```
add_action( ... );  
do_action( ... );  
has_action( ... ); // Vérifie si l'action a été ajouté à un hook  
remove_action( ... ); // Supprime l'action  
remove_all_actions( ... ); // Supprime toutes les actions  
did_action( ... ); // Nombre de fois que l'action est déclenchée
```

Les « Hook »

Mis en pratique :

- 1- Dans le fichier « `template-lptac.php` » créer un hook « `lptac-display-txt` » et le placer au dessus du titre « `template-lptac` »
- 2- Créer une fonction php « `lptac_display_hook()` » dans le fichier « `functions.php` » qui affichera le message « Texte de mon hook lptac »
- 3- Ajouter le fonction « `lptac_display_hook()` » sur le hook « `lptac-display-txt` »
- 4- Le texte « Texte de mon hook lptac » s'affiche t'il bien dans le fichier « `template-lptac.php` » au dessus du titre « `template-lptac` » ?

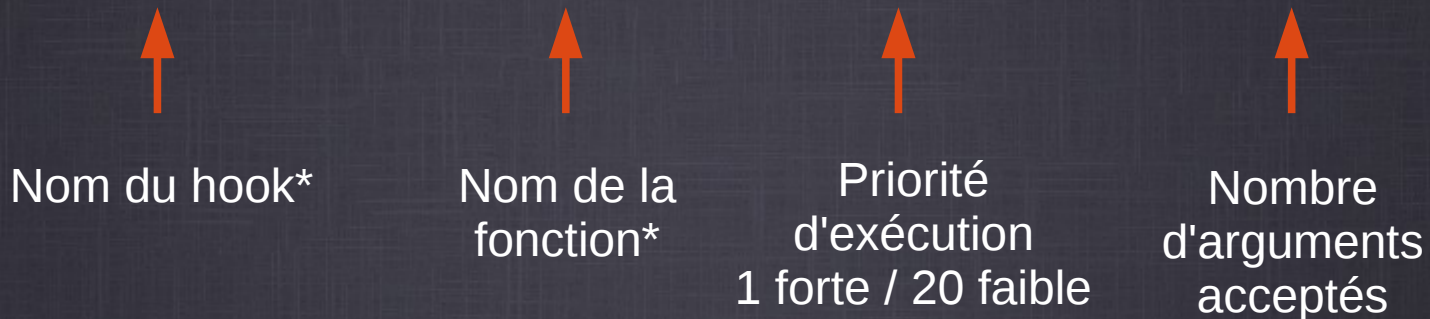
Les « Hook »

Les hooks de type « Filtre »

(* arguments obligatoires)

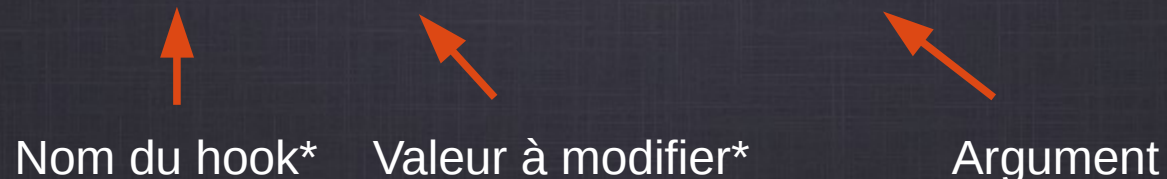
Ajout de la fonction sur le hook :

```
add_filter( $tag, $function_to_add, $priority, $accepted_args );
```



Exécute la fonction qui a été spécifiée dans le « `add_filter` » et retourne la valeur modifiée :

```
$val = apply_filter( $tag, $value, $args1, $args2 ... );
```



Les « Hook »

Exemple :

WordPress dispose d'un filtre pour les titres qui sont récupérés de la base de données : « `the_title` » Ce filtre s'applique juste avant l'affichage et accepte deux arguments : le titre et l'id de l'article ou de la page.

Nous allons intercepter les titres et les mettre en majuscules avant qu'ils s'affichent.

Code à placer dans le **functions.php**

```
// Filtre
function lptac_title_uppercase( $title ) {
    $title = strtoupper( $title ); // Converti le titre en majuscules
    return $title;
}

add_filter( 'the_title' , 'lptac_title_uppercase' , 10 , 1 );
```

Les « Hook »

Mis en pratique :

- 1- Dans le fichier « `template-lptac.php` » créer une variable php \$toto et lui affecter la chaîne de caractères « Hello, je m'appelle Toto ! »
- 2- Appliquer un hook de type filtre « `lptac_toto` » sur cette variable de sorte que l'on puisse la modifier par la suite à l'aide d'une fonction.
- 3- Afficher cette chaîne de caractère à l'écran.
- 4- Dans le fichier « `functions.php` » créer une fonction « `lptac_toto_uppercase()` » qui transforme une chaîne de caractères en majuscules.
- 4- Ajouter la fonction « `lptac_toto_uppercase()` » au hook de type filtre « `lptac_toto` » pour afficher la variable \$toto « Hello, je m'appelle Toto ! » en majuscules ?

template-lptac.php

```
<?php
/*
Template Name: template lptac
*/

get_header(); ?>

<div>

<h1>template-lptac</h1>
<p>Ceci est une page à laquelle on a attribué le template personnalisé "template-lptac"</p>

<?php get_template_part( 'loop' , 'template' ); ?>

<?php
$toto = "Hello, je m'appelle Toto !";
echo $toto . '<br>';

$toto = apply_filters( 'lptac_toto' , $toto );
echo $toto;
?>

</div>

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

functions.php

```
function lptac_toto_uppercase( $toto ) {
    $toto = strtoupper( $toto ); // Converti en majuscules
    return $toto;
}

add_filter( 'lptac_toto' , 'lptac_toto_uppercase' , 10 , 1 );
```

Les « Hook »

Comment trouver le bon hook :

- Lire le Codex

L'utilisation du hook adéquat pour une action donnée est souvent indiquée dans la documentation WordPress.

Le Codex liste les hooks « action » et « filtre » de WordPress les plus utilisés

http://codex.wordpress.org/Plugin_API/Action_Reference

http://codex.wordpress.org/Plugin_API/Filter_Reference

- Chercher dans le code de WordPress.

- Utiliser wpseek.com, un moteur de recherche de fonctions, constantes et hooks WordPress.

- Utiliser des plugins destinés aux développeurs WordPress.

 - « SF Admin bar tools »

 - « Debug Bar Action Hooks »

Comment insérer correctement les fichiers **css** et **js** dans un thème WordPress ?

WordPress met à disposition des développeurs une « méthode standard » pour charger les fichiers CSS et JavaScript.

Cette méthode permet d'éviter de charger plusieurs fois les mêmes scripts (cas qui peut se produire par exemple avec JQuery lorsque plusieurs plugins sont activés ou entre un plugin et un thème).

Cette méthode va nous permettre :

- d'éviter des conflits de code (par exemple chargement de différentes versions de JQuery).
- d'éviter de charger des scripts inutilement que se soit par doublons ou sur des pages qui ne le nécessitent pas (amélioration des performances).
- de charger les scripts/styles dans le bon ordre.
- de donner la possibilité aux autres développeurs de désactiver nos scripts facilement si besoin.

Comment insérer correctement les fichiers **css** et **js** dans un thème WordPress ?

La méthode standard de WordPress pour charger les styles (CSS) et les scripts (JS) se déroule en deux étapes :

Etape 1 : Enregistrement des styles/scripts dans WordPress « **register** »

L'enregistrement permet d'indiquer à WordPress où trouver le fichier et ses dépendances (si dépendances il y a). Cette étape n'est pas obligatoire mais conseillée si le styles/script est utilisé plusieurs fois dans le thème ou le plugin. En effet, une fois l'enregistrement effectué, la mise en fil d'attente avec les fonctions « enqueue » sera plus simple.

Etape 2 : Mise en fil d'attente des styles/scripts « **enqueue** »

La mise en fil d'attente permet de lier les fichiers de styles/scripts à la page demandée et donc de charger ces fichiers lorsque la page est affichée dans le navigateur.

Comment insérer correctement les fichiers **css** et **js** dans un thème WordPress ?

Etape 1 : Enregistrement des styles/scripts dans WordPress « **register** »

Style CSS : `wp_register_style($handle, $src, $deps, $ver, $media)`

Scripts JS : `wp_register_script($handle, $src, $deps, $ver, $in_footer)`

Etape 2 : Mise en fil d'attente des styles/scripts « **enqueue** »

Si il y a eu enregistrement préalable des styles/scripts :

Style CSS : `wp_enqueue_style($handle)`

Scripts JS : `wp_enqueue_script($handle)`

Si il n'y a pas eu d'enregistrement préalable des styles/scripts :

Style CSS : `wp_enqueue_style($handle, $src, $deps, $ver, $media)`

Scripts JS : `wp_enqueue_script($handle, $src, $deps, $ver, $in_footer)`

Comment insérer correctement les fichiers **css** et **js** dans un thème WordPress ?

Description des paramètres :

\$handle	= nom que l'on va donner au style/script. Pourra être utilisé dans le « enqueue ».
\$src	= indique le chemin du script/style.
\$deps	= indique, si elle existe, la dépendance du scrit/style. Si une dépendance existe, elle sera chargée avant le script/style.
\$ver	= permet d'indiquer un nombre de version.
\$media	= permet de spécifier pour quel type de média charger ce style/script (all, screen, print or handheld).
\$in_footer	= Paramètre boolean (true/false) qui permet d'indiquer si on souhaite charger le scriupt dans le footer plutôt que dans le header (permet de ne pas rallonger le délais de chargement de l'arbre DOM HTML).

Comment insérer correctement les fichiers **css** et **js** dans un thème WordPress ?

wp_head()

```
<?php
/**
 * Header template file: header.php
 *
 *
 * @package WordPress
 * @subpackage lpatc
 * @since lpatc 1.0
 */
?>
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8" />
    <title> Mon premier thème WordPress </title>
    <?php wp_head();| ?>
  </head>

  <body>

    <header>
      <?php bloginfo( 'name' ); ?>
      <br>
      <?php bloginfo( 'description' ); ?>
    </header>

    <div id="main" class="wrapper">
```

Activation du hook « wp_head »

Ce hook est utilisé par WordPress pour charger d'autres hooks dont « wp_enqueue_scripts »

Dans les thèmes WordPress il est donc nécessaire d'ajouter juste avant la balise fermante </head> la fonction « wp_head() » et dans le footer la fonction « wp_footer() » juste avant la balise fermante </body>.

Chargement des CSS

```
/* Register et Enqueue styles -----  
  
// Styles css : register  
function lpatc_register_style() {  
    wp_register_style(  
        'lpatc_style_css',  
        get_template_directory_uri() . '/style.css',  
        '1.0',  
        '',  
        'all' );  
  
    wp_register_style(  
        'lpatc_css',  
        get_template_directory_uri() . '/css/lpatc-style.css',  
        '1.0',  
        '',  
        'all' );  
}  
  
// Styles css : enqueue  
function lpatc_enqueue_style() {  
    wp_enqueue_style( 'lpatc_style_css' );  
    if ( is_page_template( 'template-lpatc.php' ) ) {  
        wp_enqueue_style( 'lpatc_css' );  
    }  
}  
  
// FRONT  
add_action( 'wp_enqueue_scripts' , 'lpatc_register_style' );  
add_action( 'wp_enqueue_scripts' , 'lpatc_enqueue_style' );  
  
// BACK  
/*  
add_action( 'admin_enqueue_scripts' , 'lpatc_register_style' );  
add_action( 'admin_enqueue_scripts' , 'lpatc_enqueue_style' );  
*/
```

Chargement des JS

```
/* Register et Enqueue scripts */

// Script js : register
function lpatc_register_script() {
    wp_register_script(
        'lpatc_js',
        get_template_directory_uri() . '/js/lpatc-script.js',
        '1.0',
        'jquery',
        false );
}

// Script js : enqueue
function lpatc_enqueue_script() {
    if ( is_page_template( 'template-lpatc.php' ) ) {
        wp_enqueue_script( 'lpatc_js' );
    }
}

// FRONT
add_action( 'wp_enqueue_scripts' , 'lpatc_register_script' );
add_action( 'wp_enqueue_scripts' , 'lpatc_enqueue_script' );

// BACK
/*
add_action( 'admin_enqueue_scripts' , 'lpatc_register_script' );
add_action( 'admin_enqueue_scripts' , 'lpatc_enqueue_script' );
*/
```


Comment insérer correctement les fichiers **css** et **js** dans un thème WordPress ?

Mise en pratique

- 1- Dans « header.php » ajouter `<?php wp_head(); ?>` juste avant la balise `</head>`
- 2- Créer dans le répertoire du thème un dossier « **css** » et « **js** »
- 3- Créer dans le dossier « **css** » un fichier « **lptac-style.css** »

```
h1 { color : #FF52E1 ; }
```

- 4- Créer dans le dossier « **js** » un fichier « **lptac-script.js** » qui dépend de JQuery

```
jQuery( document ).ready( function( $ ) {  
    $( '.wrapper' ).css( 'background-color' , '#FFF8DC' );  
} );
```

- 5- Enregistrez et mettez en file les fichiers style/script ci-dessus pour qu'ils se chargent sur la page « **template-lptac.php** » uniquement.

Comment ajouter **le support des menus** dans le back-end de WordPress de votre thème ?

Le support des menus s'effectue en 3 étapes :

1- Enregistrement du/des menus et définition des paramètres. S'effectue dans le fichier « functions.php » en se branchant sur le hook « **init** ».

```
register_nav_menus( $locations )
```

2- Affichage du menu sur le front-end
S'effectue dans un fichier template php du thème.

```
wp_nav_menu( $args )
```

3- Gestion des menus dans le back-end de WordPress

Comment ajouter **le support des menus** dans le back-end de WordPress de votre thème ?

Enregistrer un ou plusieurs menus personnalisé(s) dans WordPress

```
function lptac_register_menu() {  
  
    $locations = array(  
        'menu_principal'    => 'Menu principal', // Clé => Valeur  
        'menu_footer'      => 'Menu footer'  
    );  
  
    register_nav_menus( $locations );  
}  
  
add_action( 'init' , 'lptac_register_menu' );
```

Remarque :

On utilisera « `register_nav_menu($locations, $description)` » pour l'enregistrement d'un seul menu.

http://codex.wordpress.org/Function_Reference/register_nav_menus

Comment ajouter le **support des menus** dans le back-end de WordPress de votre thème ?

Mise en pratique :

Est ce qu'un item « Menus » est présent dans le menu « Appearance » du back-end de votre thème WordPress ? (normalement non)

Dans le fichier « functions.php » enregistrer 2 menus :
« menu_principal » et « menu_footer »
à l'aide de la fonction « **register_nav_menus(\$locations)** »

Rendez-vous dans le back-end de WordPress et cliquez sur le menu
« Appearance »

Est ce qu'un item « Menus » est présent ?
(normalement oui !)

Est ce que mes « theme locations » sont visibles ?

Créer en back-end un nouveau menu « menu1 » et l'affecter au
theme_location « menu_principal » un nouveau menu « menu2 » et l'affecter
au theme_location « menu_footer »

Comment activer un **menu personnalisé** dans un thème WordPress ?

Afficher le menu en front-end

```
wp_nav_menu( )
```

http://codex.wordpress.org/Function_Reference/wp_nav_menu

Comment ajouter **le support des menus** dans le back-end de WordPress de votre thème ?

Contrôle plus fin de l'affichage des menus avec les arguments

```
$args_nav_menu = array(  
    'theme_location' => "",  
    'menu'           => "",  
    'container'      => "",  
    'container_class' => "",  
    'container_id'    => "",  
    'menu_class'      => "",  
    'menu_id'         => "",  
    'echo'            => true,  
    'fallback_cb'     => "",  
    'before'          => "",  
    'after'           => "",  
    'link_before'     => "",  
    'link_after'      => "",  
    'items_wrap'      => "",  
    'depth'           => 0,  
    'walker'          => ""  
);
```

Voir codex sur « wp_nav_menu »
pour des explications sur le rôle
des arguments.

```
wp_nav_menu( $args_nav_menu ) ;
```


Comment ajouter **le support des menus** dans le back-end de WordPress de votre thème ?

Mise en pratique :

Afficher les 2 menus en front-end à l'aide de la fonction WordPress « `wp_nav_menu()` ».

Fait en sorte que le « menu1 » s'affiche sur toutes les pages dans le header et le « menu2 » s'affiche sur toutes les pages dans le footer.

Comment ajouter le **support des menus** dans le back-end de WordPress de votre thème ?

functions.php

```
/* Menus */
function lptac_register_menu() {
    $args_register_menu = array(
        'menu_principal' => 'Menu principal', // clé (= location) => valeur (= description)
        'menu_footer'    => 'Menu footer'
    );

    register_nav_menus( $args_register_menu );
    // Si un seul menu à enregistrer
    // register_nav_menu( 'menu_principal' , 'Menu principal' );
}

add_action( 'init', 'lptac_register_menu' );
```

Comment ajouter le support des menus dans le back-end de WordPress de votre thème ?

header.php

```
<?php
$args_nav_menu = array(
    'theme_location' => 'menu_principal', // Designe l'emplacement du menu
    'menu'            => '', // Designe le menu que l'on souhaite afficher (id, slug, nom dans le back-end). Prend la main sur 't
    'container'       => 'nav', // Default '<div>', mettre 'false' pour aucun, accepte <div> ou <nav>
    'container_class' => 'menu-principal', // Default: menu-{menu slug}-container
    'container_id'    => 'menu-principal', // Default: none
    'menu_class'      => 'toto', // Classe(s) de la balise <ul>, default: menu
    'menu_id'         => 'toto', // Id de la balise <ul>, default: menu-{menu slug} ou menu-{menu slug}-n si dupliqué
    'echo'            => true, // true = affiche, false = renvoie, default: true
    'fallback_cb'     => 'wp_page_menu', // Fonction à utiliser si wp_nav_menu non utilisé. Default: wp_page_menu
    'before'          => '', // Afficher du texte avant la balise <a>, default: none
    'after'           => '', // Afficher du texte après la balise <a>, default: none
    'link_before'     => '', // Afficher du texte avant le texte du lien, default: none
    'link_after'      => '', // Afficher du texte après le texte du lien, default: none
    'items_wrap'      => '<ul id="%1$s" class="%2$s">%3$s</ul>', // Default: <ul id="%1$s" class="%2$s">%3$s</ul> %1$s reprend 'm
    'depth'           => 0, // Profondeur des sous-menus, default: 0 (pas de limite)
    'walker'          => '' // Objet pour une customisation plus poussée du menu
);

wp_nav_menu( $args_nav_menu ); |
?>
```


Comment ajouter le support des menus dans le back-end de WordPress de votre thème ?

footer.php

```
<?php
$args_nav_menu_footer = array(
    'theme_location' => 'menu_footer', // Designe l'emplacement du menu
    'menu'            => '', // Designe le menu que l'on souhaite afficher (id, slug, nom dans le back-end). Prend la main sur
    'container'       => 'nav', // Default '<div>', mettre 'false' pour aucun, accepte <div> ou <nav>
    'container_class' => 'menu-footer', // Default: menu-{menu slug}-container
    'container_id'    => 'menu-footer', // Default: none
    'menu_class'      => 'toto', // Classe(s) de la balise <ul>, default: menu
    'menu_id'         => 'toto', // Id de la balise <ul>, default: menu-{menu slug} ou menu-{menu slug}-n si dupliqué
    'echo'            => true, // true = affiche, false = renvoie, default: true
    'fallback_cb'     => 'wp_page_menu', // Fonction à utiliser si wp_nav_menu non utilisé. Default: wp_page_menu
    'before'          => '', // Afficher du texte avant la balise <a>, default: none
    'after'           => '', // Afficher du texte après la balise <a>, default: none
    'link_before'     => '', // Afficher du texte avant le texte du lien, default: none
    'link_after'      => '', // Afficher du texte après le texte du lien, default: none
    'items_wrap'      => '<ul id="%1$s" class="%2$s">%3$s</ul>', // Default: <ul id="%1$s" class="%2$s">%3$s</ul> %1$s reprend
    'depth'           => 0, // Profondeur des sous-menus, default: 0 (pas de limite)
    'walker'          => '' // Objet pour une customisation plus poussée du menu
);

wp_nav_menu( $args_nav_menu_footer );

?>
```

Créer un Custom Post Type (CPT)

La création d'un CPT s'effectue en 2 étapes :

Enregistrement et Affichage

1- **Enregistrement** du CPT et définition des paramètres. S'effectue dans le fichier « functions.php » en se branchant sur le hook « **init** ».

```
function lpatc_register_cpt_movie() {  
  
    $cpt = 'movie' ; // Nom du CPT : minuscule, pas d'espace  
    $args = array(); // Choix des paramètres  
    register_post_type( $cpt , $args );  
  
}  
  
add_action( 'init' , 'lpatc_register_cpt_movie' );
```

Créer un Custom Post Type (CPT)

Définition des « labels » dans les paramètres

```
$labels = array(  
    'name'                => 'Movies',  
    'singular_name'       => 'Movie',  
    'add_new'             => 'Add New',  
    'add_new_item'        => 'Add New Movie' ,  
    'edit_item'           => 'Edit Movie' ,  
    'new_item'            => 'New Movie' ,  
    'all_items'           => 'All Movies' ,  
    'view_item'           => 'View Movie' ,  
    'search_items'        => 'Search Movies' ,  
    'not_found'           => 'No movies found' ,  
    'not_found_in_trash'  => 'No movies found in the Trash' ,  
    'parent_item_colon'   => '' ,  
    'menu_name'           => 'Movies'  
);
```


Créer un Custom Post Type (CPT)

Les paramètres d'enregistrement d'un CPT

```
$args = array(  
    'labels'           => $labels,  
    'description'      => 'Japan movies',  
    'public'           => true,  
    'menu_position'     => 5,  
    'supports'          => array(  
        'title',  
        'editor',  
        'excerpt',  
        'comments'  
    ),  
    'has_archive'       => true,  
);
```

Pour plus de détails sur les paramètres, voir le Codex :
codex.wordpress.org/Function_Reference/register_post_type

Créer un Custom Post Type (CPT)

2- Affichage en front-end des CPT :

Il y a deux solutions

1- Soit on utilise la hiérarchie des templates de WordPress avec une boucle classique. WordPress se chargera de sélectionner les bons CPT en analysant le nom du template php dans l'URL.

Pour utiliser cette méthode le paramètre d'enregistrement du CPT « `has_archive` » doit être à « `true` ».

Le template php doit être nommé `archive-[post_type].php`

2- Soit on utilise un template php quelconque dans lequel on fait une requête à partir de la classe « `WP_Query` »

Créer un Custom Post Type (CPT)

2- **Affichage** en front-end des CPT à l'aide de la hiérarchie WordPress et d'une boucle simple.

archive-movie.php

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

    <div class="post">
        <h2 id="post-<?php the_ID(); ?>"><?php the_title();?></h2>
        <div>
            <?php the_content(); ?>
        </div>
    </div>

<?php endwhile; endif; ?>
```

Le paramètre d'enregistrement de votre CPT « `has_archive` » doit être à « `true` »

Vos CPT s'afficheront en tapant l'url : <http://votresite.com/movie>

Créer un Custom Post Type (CPT)

2- **Affichage** en front-end des CPT à l'aide d'une boucle et d'une requête construite avec la classe « WP_Query ».

```
<?php

$args = array( 'post_type' => 'movie' ); // Voir WP_Query

$products = new WP_Query( $args );

if ( $products->have_posts() ) {
    while( $products->have_posts() ) { // Début de la boucle
        $products->the_post();

        // Affichage du titre et du contenu des CPT ?>
        <h1><?php the_title() ?></h1>
        <div class='content'>
            <?php the_content() ?>
        </div>

    <?php
        } // end while, fin de la boucle
    } // end if
?>
```

Créer un Custom Post Type (CPT)

Documentation :

Codex

codex.wordpress.org/Function_Reference/register_post_type

Article de Smashing Magazin

wp.smashingmagazine.com/2012/11/08/complete-guide-custom-post-types/

Créer un Custom Post Type (CPT)

Mise en pratique :

Créer un CPT « Book » avec les caractéristiques suivantes :

Créer quatre CPT « Book » dans le back-end.

Afficher ces 4 CPT « Book » sur le front-end en utilisant **la hiérarchie de WordPress** (archive-[post_type].php) et une boucle classique.

Afficher ces 4 CPT « Book » sur le front-end en utilisant **un template de page** que vous nommerez « template book » à l'aide d'un fichier « template-cpt-book.php » et une requête utilisant la classe WP_Query.

Vous devrez pour chaque CPT afficher : le titre, l'excerpt, le contenu.

Créer un Custom Post Type (CPT)

archive-book.php (Utilisation de la hiérarchie WordPress)

```
functions.php  archive-book.php  template-cpt-book.php  lptac-style.css  index.php

<?php
/*
archive-book.php : CPT "book"
*/

get_header(); ?>

<h1>Japan books</h1>

<?php if ( have_posts() ) : while ( have_posts() ) : the_post();?>
    <div class="post">
        <h2 id="post-<?php the_ID(); ?>"><?php the_title();?></h2>
        <div>
            <?php the_excerpt(); ?>
            <?php the_content(); ?>
        </div>
    </div>
</div>

<?php endwhile; endif; ?>

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

Créer un Custom Post Type (CPT)

Template-cpt-book.php

(utilisation d'une template page + requête WP_Query)

```
functions.php  archive-book.php  template-cpt-book.php  lptac-style.css

<?php
/*
Template Name: template book
*/

get_header(); ?>

<h1>Japan books</h1>

<h2>Utilisation d'une page template</h2>

<?php
    $args = array(
        'post_type' => 'book',
    );
    $products = new WP_Query( $args );
    if( $products->have_posts() ) {
        while( $products->have_posts() ) {
            $products->the_post();
            ?>
            <h2><?php the_title() ?></h2>
            <?php the_excerpt(); ?>
            <div class='content'>
                <?php the_content() ?>
            </div>
            <?php
        }
    }
?>

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

Utiliser un **Child-theme**

Pourquoi utiliser un child-theme ?

- Pour accélérer le temps de développement d'un site en s'appuyant sur le code d'un thème existant.
- Pour apporter des modifications à un thème sans perdre ces modifications lors d'une mise à jour du thème parent.

Utiliser un **Child-theme**

Comment créer un child-theme ?

- Créer un répertoire « child-lpatc » dans le répertoire « themes/ »
- Dans le répertoire « child-lpatc » créer un fichier « style.css »
- Commencer le fichier « style.css » avec les lignes de commentaires suivantes (attention, pas d'espace avant le « : »)

```
/*  
Theme name: LPATC Child  
Theme URI: http://monsite.com  
Description: Ma description ...  
Author: Gilles Vauvarin  
Author URI: http://monsite.com  
Template: lpatc  
Version: 1.0.0  
Tags: light  
*/
```

Utiliser un **Child-theme**

La ligne de commentaire « **Template: lpatc** » indique le thème parent.

Un thème enfant s'active comme un thème classique dans le back-end de WordPress.

Menu > Appearance > Themes

Utiliser un **Child-theme**

Remarques importantes :

CSS

- Par défaut, c'est le « style.css » du thème enfant qui est chargé.
- Il est possible d'hériter des styles du thème parent si ils ont été importés.
- Il est possible de surcharger (redéfinir) les styles du thème parent, à condition que le fichier « style.css » enfant se charge **après** le fichier « style.css » parent.
- Il y a deux façons de charger le « style.css » enfant après le fichier « style.css » parent :

Avec une instruction « @import » dans le « style.css » enfant.
Avec la fonction « wp_enqueue_style » (recommandé)

Utiliser un Child-theme

Importation des styles du thème parent

La méthode « @import »

```
1  /*
2  Theme name: lpatcchild
3  Theme URI: http://monsite.com
4  Description: Child thème du thème lpatc
5  Author: Gilles Vauvarin
6  Author URI: http://monsite.com
7  Template: lpatc
8  Version: 1.0.0
9  Tags: light
10
11  This theme, like WordPress, is licensed under the GPL.
12  Use it to make something cool, have fun, and share what you've learned with others.
13  */
14
15
16  @import url("../lpatc/style.css");
17
18
19  body {
20      background-color: yellow;
21  }
```

Utiliser un Child-theme

Importation des styles du thème parent

La méthode « wp_enqueue_style »

```
<?php
/**
 * functions.php: lpatc Child theme
 *
 *
 * @package WordPress
 * @subpackage lpatc
 * @since lpatc 1.0
 */

/* CSS */

function lpatc_enqueue_parent_style() {
    wp_enqueue_style( 'parent-style' , get_template_directory_uri() . '/style.css' );
}
add_action( 'wp_enqueue_scripts' , 'lpatc_enqueue_parent_style' , 10 );

function lpatc_enqueue_child_style() {
    wp_enqueue_style( 'child-style' , get_stylesheet_uri() );
}
add_action( 'wp_enqueue_scripts' , 'lpatc_enqueue_child_style' , 11 );
```

Utiliser un **Child-theme**

Remarques importantes :

Templates php

- Par défaut, un thème enfant hérite des templates php de son parent. Il est possible de surcharger (redéfinir) ces templates en créant un template php **portant le même nom** dans le répertoire du thème enfant.
- Il est possible d'ajouter des nouveaux templates php dans le thème enfant qui n'existent pas dans le thème parent .

Utiliser un **Child-theme**

Remarques importantes :

functions.php

- Si le thème enfant dispose d'un fichier « functions.php » il **ne surcharge pas** celui du parent mais s'y ajoute.
- Le « functions.php » enfant se charge juste **avant** celui du parent.
- Pour ajouter une nouvelle fonction au thème, il suffit donc juste de l'ajouter dans le « functions.php » enfant.
- Il est possible, dans certain cas, de redéfinir des fonctions du thème parent dans le « functions.php » enfant.

Utiliser un **Child-theme**

Remarques importantes :

Il est possible de redéfinir une fonction du thème parent à partir du « functions.php » du thème enfant dans les deux cas suivant :

1- Les fonctions du thème parent ont été déclarées avec une condition d'existence.

```
if ( ! function_exists( 'lpatc_ma_fonction_parent' ) ) {  
    function lpatc_ma_fonction_parent() {  
        ...  
    }  
}
```

Utiliser un **Child-theme**

2- Les fonctions du thème parent ont été ajoutées sur des hook. On utilise alors le hook « `after_setup_theme` » qui est chargé une fois le « `functions.php` » enfant et parent chargé.

`functions.php` enfant :

```
add_action ( ' after_setup_theme ' , ' lpatc_ma_fonction_enfant ' ) ;

function lpatc_ma_fonction_enfant() {
    remove_action( ' init ' , ' fonction_parent ' );
    add_action( ' init ' , ' fonction_enfant ' );
}

function fonction_enfant() {
    ...
}
```


Utiliser un **Child-theme**

Mise en pratique :

- Créer un thème enfant du thème « lpatc » et l'activer dans le back-end.
- Modifier la couleur de background du site depuis le thème enfant.
- A partir du « functions.php » enfant, désactiver la fonction parent qui affiche le message « Hello dude ! » en haut de page.
- Surcharger le fichier « template-cpt-book.php » pour retirer l'affichage du footer.