

# **Introduction aux CMS et présentation de WordPress**

## Gestionnaire de contenu: définition

CMS (Content Management System) ou SGC (Système de Gestion de Contenu)

= Logiciels destinés à la conception et à la mise à jour dynamique de site web.

Un CMS permet de disposer d'un site web, que l'on peut administrer:

- ✓ pour y ajouter des contenus rédactionnels et/ou multimédias,
- ✓ pour mettre en place de nouvelles fonctionnalités.

## Gestionnaire de contenu: caractéristiques communes

- ✓ Gratuits et téléchargeables
- ✓ Tableau de bord accessible en ligne avec identifiant et mot de passe
- ✓ Mise en ligne de contenu
- ✓ Gestions séparées de la forme et du contenu
- ✓ Structuration et archivage du contenu par date, par auteur, par mots clés

## Principaux gestionnaire s de contenu



30 % des sites - 60 % des SGC



3 % des sites - 6 % des SGC



2 % des sites - 4 % des SGC

[https://w3techs.com/technologies/overview/content\\_management/all](https://w3techs.com/technologies/overview/content_management/all)  
<https://websitesetup.org/cms-comparison-wordpress-vs-joomla-drupal/>

# WordPress

- ✓ Lancé en 2003 par Matt Mullenweg et Mike Little
- ✓ Libre (open source) et gratuit
- ✓ Distribué sous la licence GPL (General Public Licence)
- ✓ Géré aujourd'hui par la Fondation WordPress
- ✓ Toujours le logiciel de SGC le plus populaire
- ✓ Offert sous le modèle d'affaires Freemium (Free + Premium)
- ✓ Est disponible en ± 40 langues
- ✓ **Personnalisable grâce à de nombreux thèmes et extensions**

## WordPress: utilisation

- Site traditionnel sans blogue, comme des millions de sites WordPress  
<https://www.cmaisonneuve.qc.ca/>
- Blogue attaché à votre site principal  
<https://www.starwars.com/news>
- Site transactionnel pour le commerce électronique  
<https://woocommerce.com/>
- Site de membres, avec une extension pour le créer et le gérer  
<https://fr.wordpress.org/plugins/wp-members/>
- Site pour gérer des réservations  
<https://www.checkfront.com>
- Site de gestion d'événements  
<https://wpeventsplanner.com>
- Site pour courtier immobilier  
<https://wordpress.org/plugins/wp-property/>
- Site pour financement participatif (crowdfunding)  
<https://www.ignitiondeck.com>

## WordPress: avantages (selon WordPress)

- ✓ Respect des standards du web
- ✓ Changement rapide de l'aspect du site avec les thèmes
- ✓ Gestion des pages statiques
- ✓ Blogues: outils de communication inter-blogs,  
commentaires,  
protection d'article par mot de passe  
Auteurs multiples
- ✓ Protection contre le spam
- ✓ Gestion des utilisateurs, du simple contributeur à l'administrateur
- ✓ Facilité d'installation et de mise à jour
- ✓ Importer du contenu depuis d'autres plateformes
- ✓ Bibliothèque de média (images, vidéos, audios,...)
- ✓ Gestion des caractères spéciaux (apostrophe, guillemets, esperluette, etc.)

## WordPress: installation

- Téléchargement sur <https://fr.wordpress.org/> (version française)
- Copier les répertoires dans votre site Web
- Aller à index.php et suivez les instructions
  - ☞ Pour des raisons de sécurité, changer le nom de la BDD, par exemple n41wp  
et ne pas utiliser wp\_ comme préfixe des tables mais par exemple n41\_
  - ☞ Donner un nom au site,  
votre identifiant et votre mot de passe de connexion en tant qu'administrateur,  
ainsi qu'une adresse courriel pour votre site



# WordPress: structures

- Répertoires à la racine du site :

wp-admin

wp-includes

wp-content

- Fichiers à la racine du site : wp-login.php,...

wp-config.php (paramétrage du site)

en bleu, les structures du noyau qui ne doivent pas être modifiées.

en vert, le dossier des ajouts par l'utilisateur ou le développeur

- Contenu HTML (pages et articles) et métadonnées\* sont stockés dans la BDD.

\* Une métadonnée est une donnée qui sert à définir ou à décrire une autre donnée.

# WordPress: créer un premier article

The screenshot shows the WordPress dashboard interface. In the top left corner, the user is logged in as 'n41'. The top navigation bar includes a 'Créer' button and a 'Bonjour, n41admin' greeting. The left sidebar contains a menu with 'Articles' selected, and the 'Ajouter' button is highlighted with a red box. The main content area displays the title 'Mon premier Post' and the text 'Bonjour, Ceci est mon premier article dans le cadre du cours: création de site Web 2. Bonne lecture.' The right sidebar shows the 'Document' tab selected, with the 'Publier...' button highlighted in a red box. The 'État et visibilité' section includes options for 'Visibilité' (Public), 'Publier' (Immédiatement), and checkboxes for 'Épingler en haut du blog' and 'En attente de relecture'.

WordPress dashboard interface showing the 'Ajouter' button highlighted in the left sidebar and the 'Publier...' button highlighted in the top right.

The main content area displays the title 'Mon premier Post' and the text 'Bonjour, Ceci est mon premier article dans le cadre du cours: création de site Web 2. Bonne lecture.'

The right sidebar shows the 'Document' tab selected, with the 'Publier...' button highlighted in a red box. The 'État et visibilité' section includes options for 'Visibilité' (Public), 'Publier' (Immédiatement), and checkboxes for 'Épingler en haut du blog' and 'En attente de relecture'.

# WordPress: créer une première page

The screenshot displays the WordPress admin interface. On the left sidebar, the 'Pages' menu is selected, and the 'Ajouter' (Add) button is highlighted with a red rectangle. The main content area shows a draft of a new page titled 'Ma première page statique' (My first static page). Below the title, the text reads: 'Ceci est ma première page statique dans le cadre du cours: création de site Web 2.' (This is my first static page in the context of the course: creation of Web site 2). In the top right corner, the 'Publier...' (Publish) button is also highlighted with a red rectangle. The right-hand sidebar contains the 'Document' settings panel, which includes options for 'État et visibilité' (Status and visibility), 'Visibilité' (Public), 'Publier' (Publish), and 'En attente de relecture' (Pending review).

Bonjour, n41admin

Créer

Tableau de bord

Articles

Médias

**Pages**

Toutes les pages

**Ajouter**

Commentaires

Apparence

Extensions

Utilisateurs

Outils

Réglages

Réduire le menu

Enregistrer le brouillon

Prévisualiser

**Publier...**

Document

Bloc

État et visibilité

Visibilité [Public](#)

Publier [Immédiatement](#)

☐ En attente de relecture

Image mise en avant

Discussion

Attributs de page

# Ma première page statique

Ceci est ma première page statique dans le cadre du cours:  
création de site Web 2.

# WordPress: supprimer les commentaires de la page, en modification (1)

The screenshot shows the WordPress editor interface. On the left is the sidebar with the 'Pages' menu item highlighted. The main content area shows the title 'Ma sta' and the text 'Ceci est création'. A dialog box titled 'Options' is open in the center, showing three sections: 'Général', 'Options de clavier', and 'Panneaux de documentation'. In the 'Panneaux de documentation' section, the 'Commentaires' checkbox is checked. A red arrow points from the 'Options' button in the right sidebar to the 'Comments' checkbox in the dialog box. Another red arrow points from the 'Options' button in the right sidebar to the 'Options' button in the right sidebar. The text 'Commentaires doit être coché.' is written in red above the 'Options' button in the right sidebar.

Options

Général

- ☒ Vérifications de pré-publication
- ☐ Activer la catégorie des blocs les plus utilisés dans la bibliothèque de blocs

Options de clavier

- ☐ Contenir le curseur de texte à l'intérieur du bloc actif

Panneaux de documentation

- ☒ Permalien
- ☒ Image mise en avant
- ☒ Commentaires

Commentaires doit être coché.

Options

## WordPress: supprimer les commentaires de la page, en modification (2)

The screenshot shows the WordPress page editor interface. The main content area displays the title "Ma première page statique" and the text "Ceci est ma première page statique dans le cadre du cours: création de site Web 2." Below this, a prompt says "Commencez à écrire ou saisissez « / » pour choisir un bloc".

The right sidebar, titled "Document", contains several sections: "État et visibilité", "2 révisions", "Permalien", "Image mise en avant", and "Commentaires". The "Commentaires" section is expanded, showing a checkbox labeled "Autoriser les commentaires", which is currently unchecked. A red arrow points to this checkbox.

At the top of the sidebar, the "Mettre à jour" button is highlighted with a red box. The top navigation bar shows "Cours n41", "Créer", "Voir la page", and "Bonjour, n41admin".

**Autoriser les commentaires doit être décoché.** →

# WordPress: créer un menu de navigation entre les pages

The screenshot shows the WordPress 'Appearance' menu editor. The left sidebar contains the 'Apparence' menu item (1) and the 'Menus' sub-item (2). The main content area is divided into two panels: 'Ajouter des éléments de menu' and 'Structure du menu'.

**Ajouter des éléments de menu:**

- Under the 'Pages' tab, there are checkboxes for 'Ma première page statique' and 'Page d'exemple', both of which are checked.
- A button 'Ajouter au menu' (5) is located at the bottom of this panel.

**Structure du menu:**

- The 'Nom du menu' field is set to 'Principal' (3).
- An 'Enregistrer le menu' button (4) is at the top right of this panel.
- Below the instructions, the menu structure is shown with two items: 'Ma première page statique' and 'Page d'exemple', each with a 'Page' dropdown menu.

**Réglages du menu:**

- The 'Ajoutez automatiquement des pages' section has a checkbox 'Ajouter automatiquement les pages de premier niveau à ce menu' (6), which is unchecked.
- The 'Afficher l'emplacement' (7) section has two checked options: 'Menu horizontal grand écrans' and 'Menu étendu grands écrans'.
- Other options include 'Menu sur mobile', 'Menu du pied de page', and 'Menu réseaux sociaux', all of which are unchecked.

At the bottom of the 'Structure du menu' panel, there is a link 'Supprimer le menu' and another 'Enregistrer le menu' button (8).

# WordPress: documentation

Toute la documentation officielle à l'adresse :

<https://codex.wordpress.org/>

# Gestion de WordPress



# WordPress: configuration de base (1)

## Dans le fichier wp-config.php

- Peut être déplacé à la racine du site `www/` au lieu de `www/mon-site-wp/`  
c'est une bonne pratique sécuritaire
- Les options sont définies avec l'instruction php:  

```
define( 'OPTION_NAME', 'value');
```
- Si besoin pour déboguer, récupérer les constantes WP en utilisant la fonction PHP:  

```
var_dump( @get_defined_constants() );
```

## WordPress: configuration de base (2)

### Dans le fichier wp-config.php

- Renforcement de la sécurité avec des clés aléatoires et des clés de salage associées pour crypter les informations dans les cookies:

```
define('AUTH_KEY',         '~TKYz3I}|7HH+[IP[=?^O+9z]Gh|1gG[6 ^tEIW6 (DZok|. +vV>oDxOIA. (/ -J1y');  
define('SECURE_AUTH_KEY', 'Qq=vqd;3[2&u~w6ky$wHj`uL|-q6E=R>3Z`Z/ E :9zH%wX6Q]Zq8+b/R.kK[;[');  
define('LOGGED_IN_KEY',    '*%81t`aDT>N*J3ZjWUb92r=+ogCUn}>M*L _~_VRW2O=Lzy73@T#jNWolP)M?,z(');  
define('NONCE_KEY',       ' F8%AVYIkNF+3 (&ADF{VeU56|IjW9vZ:7}v]{sc(_i?di[]d6gNNu0^gFAL:<lw)');  
define('AUTH_SALT',       'uIDp!vWLFVH3v5})U)/RHoI;={a<x1kr7y-fcb74~8fjB*1@SifaqcRq|h[E<>7W');  
define('SECURE_AUTH_SALT', '7;aFofVhh)W)Zxp7CeT2^fz~:IyPP^mX`_?:$6q]y%P23=[+@|_FkgA/e> {'!V{[');  
define('LOGGED_IN_SALT',  'Bn !8y[w=d*F:H8WW**{1a3d1d>G3z`]7FNRhy;v~c=W3GNKtsRbA0cc!/N^k+(E');  
define('NONCE_SALT',      '^+y:0+00K*.s~ m!Cje_W@fQPhMa].AK~>_t!tG^y!PN?QVBV1` |xb:eDAMBhN}');
```

- Peuvent être régénérées en utilisant <https://api.wordpress.org/secret-key/1.1/salt/> ce qui invalide les cookies en cours d'utilisation et force les utilisateurs à se reconnecter.

## WordPress: configuration de base (3)

### Dans le fichier wp-config.php

- Choix de la langue:

version < 4.0	define ( 'WPLANG', 'fr_FR' );
version ≥ 4.0	Réglage -> Général -> langue du site via la console d'administration stockage dans la table options

- Débogage pour les développeurs d'extensions:

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

## WordPress: configuration de base (4)

### Dans le fichier wp-config.php, options avancées non présentes

- Révisions:

Par défaut WP sauvegarde toutes les versions d'un article ou d'une page.

`define( 'WP_POST_REVISIONS', false );` -> pas de sauvegarde

`define( 'WP_POST_REVISIONS', 3 );` -> 3 dernières versions

- Intervalle de sauvegardes automatiques (60 secondes par défaut):

`define( 'AUTOSAVE_INTERVAL', 300 );`

## WordPress: configuration de base (5)

### Dans le fichier wp-config.php, options avancées non présentes

- Augmenter la mémoire allouée à PHP (si autorisé par l'hébergeur):  
define( 'WP\_MEMORY\_LIMIT', '96M' );

Traitement dans /wp-includes/default-constants.php :

```
// Define memory limits.
if ( ! defined( 'WP_MEMORY_LIMIT' ) ) {
    if ( false === wp_is_ini_value_changeable( 'memory_limit' ) ) {
        define( 'WP_MEMORY_LIMIT', $current_limit );
    } elseif ( is_multisite() ) {
        define( 'WP_MEMORY_LIMIT', '64M' );
    } else {
        define( 'WP_MEMORY_LIMIT', '40M' );
    }
}
```

# WordPress: liens permanents (permaliens/permalinks)

## Réglages -> Permaliens

### Réglages des permaliens

WordPress vous offre la possibilité de créer une structure personnalisée d'adresses web pour vos permaliens et archives. Ceci peut améliorer l'esthétique, l'utilisabilité et la pérennité de vos liens. De [nombreux marqueurs sont disponibles](#), et nous vous donnons quelques exemples pour commencer.

#### Réglages les plus courants

☐ Simple

`http://localhost/n41wp/?p=123`

☒ Date et titre

`http://localhost/n41wp/2019/08/27/exemple-article/`

☐ Mois et titre

`http://localhost/n41wp/2019/08/exemple-article/`

☐ Numérique

`http://localhost/n41wp/archives/123`

☐ Titre de la publication

`http://localhost/n41wp/exemple-article/`

☐ Structure personnalisée

`http://localhost/n41wp`

Étiquettes disponibles :

%year%	%monthnum%	%day%	%hour%	%minute%	%second%	%post_id%	%postname%
%category%	%author%						

# WordPress: liens permanents, fichier .htaccess

```
# BEGIN WordPress
<IfModule mod_rewrite.c>
RewriteEngine On
RewriteBase /wordpress/
RewriteRule ^index\.php$ - [L]
RewriteCond %{REQUEST_FILENAME} !-f
RewriteCond %{REQUEST_FILENAME} !-d
RewriteRule . /wordpress/index.php [L]
</IfModule>

# END WordPress
```

①

②

- ① Si URL index.php, exécution d'index.php sans examiner les règles qui suivent (car [L])
- ② Toutes les URL qui ne correspondent pas à un fichier ou un répertoire existant sont traitées par index.php

# WordPress: utilisateurs

Utilisateurs -> Ajouter

## Ajouter un utilisateur

Créer un nouvel utilisateur et l'ajouter à ce site.

Identifiant *(nécessaire)*

Adresse de messagerie  
*(nécessaire)*

Prénom

Nom

Site web

Mot de passe

Afficher le mot de passe

Envoyer une notification à  
l'utilisateur

☒ Envoyer un message au nouvel utilisateur à propos de son compte.

Rôle

Abonné	▼
Abonné	
Contributeur	
Auteur	
Éditeur	
Administrateur	

Ajouter un utilisateur



## WordPress: utilisateurs, rôles

Actions	Abonné	Contributeur	Auteur	Éditeur	Admin
Modifier son profil utilisateur	X	X	X	X	X
Lire et commenter	X	X	X	X	X
Éditer et supprimer ses articles		X	X	X	X
Publier ses articles			X	X	X
Téléverser des fichiers			X	X	X
Éditer et supprimer les articles des autres				X	X
Publier les articles des autres				X	X
Gérer les catégories				X	X
Modérer les commentaires				X	X
Éditer, supprimer et publier les pages				X	X
Administrer: options, utilisateurs, thèmes, extensions,...					X


# WordPress: utilisateurs, inscription directement en tant qu'abonné

**Réglages -> Général**

**Inscription** ☒ Tout le monde peut s'enregistrer

Rôle par défaut de tout nouvel utilisateur: **Abonné**

localhost/wordpress/wp-login.php

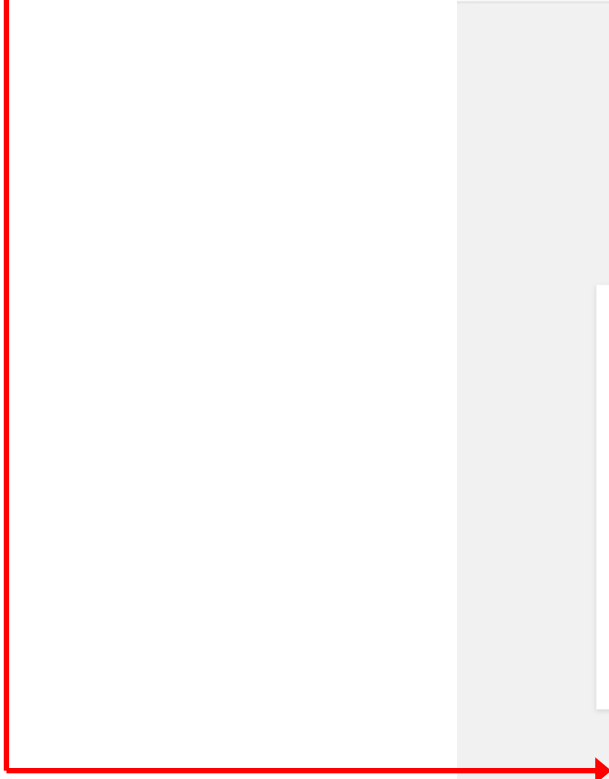


Nom d'utilisateur ou adresse e-mail

Mot de passe

☐ Se souvenir de moi **Se connecter**

**Inscription** Mot de passe oublié ?



## WordPress: thèmes

- Un Thème WP est un ensemble de fichiers qui fonctionnent ensemble pour produire un design unifié du site web.
- Un Thème modifie la façon dont le site est affiché, sans changer le logiciel WP.
- Les pages html du site sont affichées à partir de fichiers modèles (template files) qui contiennent un mélange de code html et de code php.
- Les Thèmes contiennent :
  - des fichiers modèles personnalisés,
  - des fichiers images,
  - des feuilles de style (\*.css),
  - ainsi que tous les fichiers de programmation requis (\*.php, \*.js).
- La personnalisation d'un thème avec la console d'administration est plus ou moins grande selon le thème choisi.

Le thème peut contenir un constructeur de pages (page builder) ou bien s'interfacer avec un "page builder" externe (Elementor par exemple).

La version 5 de WP introduit le constructeur de pages intégré Gutenberg.

## WordPress: extensions (plugins)

- Une extension WP est une application qui s'interface avec WP pour lui apporter des fonctionnalités supplémentaires.
- Les extensions WP sont disponibles à partir de plusieurs sources.  
La plus populaire est la source officielle: <https://wordpress.org/plugins/>
- 2 extensions sont installées avec le noyau WP:
  - Akismet , analyse les commentaires pour détecter les spams.
  - Hello Dolly, extension très simple dans un seul fichier php (hello.php).

<https://wordpress.org/support/article/plugins/>

<https://codex.wordpress.org/Akismet>

## WordPress: widgets

- Un widget WP est une zone autonome d'une page web qui remplit une fonction spécifique. C'est aussi le code qui génère cette zone.
- WP a des widgets intégrés sous [/wp-includes/widgets](#), comme:
  - le calendrier des articles du blogue,
  - l'affichage des commentaires récents,
  - l'affichage d'une galerie d'images,
  - etc.
- Les Thèmes et les extensions (plugins) peuvent fournir des widgets additionnels.

<https://wordpress.org/support/article/glossary/#widget>

<https://wordpress.org/support/article/wordpress-widgets/>

## WordPress: widgets, installation

- Le thème doit être programmé pour accueillir les widgets, sinon il faut insérer du code pour cela.
- Pour installer un widget:
  - Apparence -> Widgets,
  - puis choisir le widget à installer en cliquant sur son bouton "Ajouter"
  - puis le configurer et l'affecter à une zone d'accueil de widgets:  
colonne latérale (sidebar), pied de page,....

## WordPress: widgets, exemples d'installation

- Ajouter le widget "calendrier" et le widget des commentaires récents à une zone d'accueil prévue par le thème.
  - ☞ Remarquer que chaque widget a ses propres paramètres de configuration.
- Ajouter le widget "menu de navigation".
- Ajouter le widget de texte, donner un titre et un contenu.

# Noyau de WordPress



## Noyau WP

- Le noyau WP contient tous les fichiers nécessaires au fonctionnement de WP: fichiers PHP, JavaScript, CSS, XML, HTML et images.
- Les ajouts tels que données, plugins et thèmes n'en font pas partie et ne sont pas écrasés lors des mises à jour de WP.
- Le noyau WP comporte un **grand nombre de fonctions PHP** (programmation procédurale) **et de classes PHP (POO) dont la plupart peuvent être utilisées par les développeurs de thèmes et de plugins.**

# Noyau WP: fonctions

- Des fonctions spécifiques pour les développeurs de thèmes :  
les marqueurs de modèles (template tags).  
[https://codex.wordpress.org/Template\\_Tags](https://codex.wordpress.org/Template_Tags)
- Des fonctions pour les développeurs d'extensions (plugins) :  
les hooks (hameçons/crochets) qui permettent d'intégrer des fonctions du plugin au fonctionnement du noyau de WP.  
[https://codex.wordpress.org/Plugin\\_API](https://codex.wordpress.org/Plugin_API)
- Des fonctions utilisées par le noyau WP dont la plupart peuvent être utilisées également par les développeurs de thèmes et de plugins.  
[https://codex.wordpress.org/Function\\_Reference](https://codex.wordpress.org/Function_Reference) (répertoriées par catégories)

Index de toutes les fonctions par ordre alphabétique: <https://developer.wordpress.org/reference/functions/>

# Noyau WP: fonctions, recherche d'après leur classement par catégories

Répertoire des fonctions classées par catégories : [https://codex.wordpress.org/Function\\_Reference](https://codex.wordpress.org/Function_Reference)

Catégories de fonctions	Exemples
Articles, pages, contenus personnalisés, médias, ...	<b>Récupération des données d'un élément de la table "posts" (article, page, média,...):</b> <code>get_post</code>
Catégories, mots-clés, taxinomies	<b>Récupération des données d'une catégorie (définie pour des articles):</b> <code>get_category</code>
Utilisateurs	<b>Gestion de métadonnées (données additionnelles) pour un utilisateur:</b> <code>add_user_meta</code> , <code>get_user_meta</code> , <code>delete_user_meta</code>
Crochets ("hooks") d'actions et de filtres	<b>Ajout d'une fonction à un crochet d'action WP, cette fonction sera exécutée dans un contexte WP précis ou quand un évènement spécifique survient:</b> <code>add_action</code>
Thèmes	<b>Inclusion de modèles (templates):</b> <code>get_header</code> , <code>get_sidebar</code> , <code>get_footer</code> ,...
Formatage (données de formulaires,...)	<b>Assainissement (sanitization) des chaînes de caractères:</b> <code>esc_html</code> , <code>esc_url</code> , <code>sanitize_text_field</code>
Diverses: options, dates,...	<b>Gestion d'une option dans la table "options":</b> <code>add_option</code> , <code>get_option</code> , <code>update_option</code> , <code>delete_option</code> <code>set_transient</code> , <code>get_transient</code> , <code>delete_transient</code> (transient = option avec une date d'expiration)

## Noyau WP: fonctions diverses

Les fonctions sont regroupées dans des fichiers,  
comme par exemple dans ce fichier [functions.php](#) sous wp-includes :

[absint\(\)](#) <https://developer.wordpress.org/reference/functions/absint/>  
convertit une valeur en un entier positif.

[current\\_time \(\)](#) [https://developer.wordpress.org/reference/functions/current\\_time/](https://developer.wordpress.org/reference/functions/current_time/)  
donne l'heure courante dans différents formats.

[wp\\_nonce\\_field\(\)](#) [https://developer.wordpress.org/reference/functions/wp\\_nonce\\_field/](https://developer.wordpress.org/reference/functions/wp_nonce_field/)  
affiche un champ caché dans un formulaire avec un 'nonce' (number used once)  
pour se prémunir des attaque de type CSRF (cross-site request forgery).  
Utilisation d'un nonce : <https://developer.wordpress.org/themes/theme-security/using-nonces/>  
etc.

Index de toutes les fonctions par ordre alphabétique: <https://developer.wordpress.org/reference/functions/>

## Noyau WP: classes

Il comporte également des classes, comme par exemple ces classes usuelles dans ces fichiers sous wp-includes :

[wp-db.php](#) <https://developer.wordpress.org/reference/classes/wpdb/>

classe "wpdb" pour accéder aux tables de la base de données,  
aussi bien les tables WP que d'autres tables créées par les développeurs.

[class-wp-query.php](#) [https://developer.wordpress.org/reference/classes/wp\\_query/](https://developer.wordpress.org/reference/classes/wp_query/)

classe "WP\_Query" pour récupérer des informations sur les articles et les pages.

[class-wp-widget.php](#) [https://developer.wordpress.org/reference/classes/wp\\_widget/](https://developer.wordpress.org/reference/classes/wp_widget/)

classe "WP\_Widget"  
un widget est créé avec une classe enfant qui hérite de cette classe parent (extends).

Index de toutes les classes par ordre alphabétique: <https://developer.wordpress.org/reference/classes/>

## Noyau WP: interactions avec le noyau WP

En plus des liens précédents vers les fonctions et les classes, les interactions avec le noyaux WP sont regroupées par sujet sur cette page :

[https://codex.wordpress.org/WordPress\\_API's](https://codex.wordpress.org/WordPress_API's)

avec des liens vers les API (interfaces de programmation) pour par exemple :

- développer un plugin: [https://codex.wordpress.org/Plugin\\_API](https://codex.wordpress.org/Plugin_API)
- développer un widget: [https://codex.wordpress.org/Widgets\\_API](https://codex.wordpress.org/Widgets_API)
- gérer les options dans la table options: [https://codex.wordpress.org/Options\\_API](https://codex.wordpress.org/Options_API)

# Noyau WP: documentation dans les fichiers PHP (1)

Les fichiers PHP sont documentés au format [PHPDoc](#),  
faire de même pour les développements externes au noyau.

👉 Tous les fichiers ont un en-tête, par exemple pour wp-includes/plugin.php:

```
<?php
/**
 * The plugin API is located in this file, which allows for creating actions
 * and filters and hooking functions, and methods. The functions or methods will
 * then be run when the action or filter is called.
 *
 * The API callback examples reference functions, but can be methods of classes.
 * To hook methods, you'll need to pass an array one of two ways.
 *
 * Any of the syntaxes explained in the PHP documentation for the
 * {@link https://secure.php.net/manual/en/language.pseudo-types.php#language.types.callback 'callback'}
 * type are valid.
 *
 * Also see the {@link https://codex.wordpress.org/Plugin\_API Plugin API} for
 * more information and examples on how to use a lot of these functions.
 *
 * This file should have no external dependencies.
 *
 * @package WordPress
 * @subpackage Plugin
 * @since 1.5.0
 */
```

## Noyau WP: documentation dans les fichiers PHP (2)

Les fichiers PHP sont documentés au format [PHPDoc](#),  
faire de même pour les développements externes au noyau.

👉 Toutes les fonctions ont un en-tête, par exemple pour wp-includes/plugin.php:

```
/**
 * Check if any filter has been registered for a hook.
 *
 * @since 2.5.0
 *
 * @global array $wp_filter Stores all of the filters.
 *
 * @param string      $tag          The name of the filter hook.
 * @param callable|bool $function_to_check Optional. The callback to check for. Default false.
 * @return false|int If $function_to_check is omitted, returns boolean for whether the hook has
 *                  anything registered. When checking a specific function, the priority of that
 *                  hook is returned, or false if the function is not attached. When using the
 *                  $function_to_check argument, this function may return a non-boolean value
 *                  that evaluates to false (e.g.) 0, so use the === operator for testing the
 *                  return value.
 */
function has_filter($tag, $function_to_check = false) {
    global $wp_filter;

    if ( ! isset( $wp_filter[ $tag ] ) ) {
        return false;
    }

    return $wp_filter[ $tag ]->has_filter( $tag, $function_to_check );
}
```



## **Boucle de WordPress**

# Boucle WP: étapes du processus de génération d'une page HTML

E1 - Analyse et passage de l'URL d'un permalien à /index.php (.htaccess)

E2 - Instanciation des objets WP et WP\_Query dans wp-settings.php

E3 - Exécution de la méthode parse\_request() de l'objet WP.

Cette méthode prépare les paramètres qui vont permettre de construire à l'étape 4 la requête MySQL de récupération des 'posts' (articles ou pages), en analysant l'URL du permalien.

E4 - Exécution de la méthode query\_posts() de l'objet WP.

Cette méthode exécute la méthode query() de l'objet WP\_Query avec les paramètres créés à l'étape 3.

Cette méthode query() construit et exécute la requête MySQL de récupération des 'posts' dans la table posts de la base de données, elle enregistre le **résultat dans la propriété \$posts de l'objet WP\_Query, cette propriété sera utilisée dans la "boucle"** à l'étape 5.

E5 - WP charge un modèle (template) du thème en fonction du type de l'URL et du nombre de posts (articles ou pages) retournés.

Ce modèle implémente "la boucle" (while PHP) pour afficher la page HTML résultat.

# Boucle WP: exemple dans le modèle index.php du thème twentytwenty

```
if ( have_posts() ) {  
    $i = 0;  
    while ( have_posts() ) {  
        $i++;  
        if ( $i > 1 ) {  
            echo '<hr class="post-separator styled-separator is-style-wide section-inner" aria-hidden="true" />';  
        }  
        the_post();  
        get_template_part( 'template-parts/content', get_post_type() );  
    }  
}
```

①

```
} elseif ( is_search() ) {  
    ?>  
    <div class="no-search-results-form section-inner thin">  
        <?php  
        get_search_form(  
            array(  
                'label' => __( 'search again', 'twentytwenty' ),  
            )  
        );  
        ?>  
    </div><!-- .no-search-results -->  
    <?php  
}  
?>
```

1 - boucle

2 - itère l'index des posts dans la boucle

3 - charge une partie de modèle pour ce post (content-xxx.php sous template-parts)

4 - template partiel de pagination

④ <?php get\_template\_part( 'template-parts/pagination' ); ?>

## Boucle WP: les marqueurs de modèles (template tags)

= Fonctions PHP utilisables dans les modèles ou les parties de modèles d'un thème pour afficher directement des informations , quand préfixées par "the\_", ou pour récupérer des informations dans une variable, quand préfixées par "get\_").

Fonctions d'affichage "the_" courantes	Résultats
<code>the_author()</code>	affiche l'auteur du 'post' courant
<code>the_category(\$separator = ", \$parents = ", \$post_id = false)</code>	affiche les catégories du 'post' courant
<code>the_content(\$more_link_text = null, \$strip_teaser = false)</code>	affiche le contenu du 'post' courant
<code>the_excerpt()</code>	affiche un extrait du 'post' courant, peut être généré automatiquement par WP avec les 55 premiers mots ( <code>wp_trim_excerpt()</code> )
<code>the_ID()</code>	affiche l'id du 'post' courant
<code>the_permalink(\$post = 0)</code>	affiche le permalien du 'post' courant
<code>the_tags(\$before = null, \$sep = ', ', \$after = "")</code>	affiche les balises du 'post' courant
<code>the_time(\$d = "")</code>	affiche la date et l'heure du 'post' courant
<code>the_title(\$before = ", \$after = ", \$echo = true)</code>	affiche le titre du 'post' courant

## Boucle WP: les marqueurs de modèles (template tags)

Autres fonctions d'affichage non préfixées par 'the\_'

Autres fonctions d'affichage courantes	Résultats
<code>comments_link()</code>	affiche l'URL vers les commentaires du 'post' courant, ou vers le formulaire de saisie d'un commentaire pour ce 'post' s'il n'y en a pas.
<code>comments_popup_link(\$zero = false, \$one = false, \$more = false, \$css_class = "", \$none = false)</code>	affiche un libellé cliquable associé au lien vers les commentaires du 'post' courant, ou associé au lien vers le formulaire de saisie d'un commentaire pour ce 'post' s'il n'y en a pas.
<code>edit_post_link(\$text = null, \$before = "", \$after = "", \$id = 0, \$class = 'post-edit-link')</code>	affiche un lien pour éditer le 'post' courant, si l'utilisateur est connecté et autorisé à le faire.

# Boucle WP: les marqueurs de modèles (template tags)

Fonctions préfixées par "get\_"

utilisées pour la plupart par les fonctions préfixées par "the\_" correspondantes.

Fonctions "get_" courantes	Fonctions "the_" ou autres qui les utilisent
<code>get_the_author()</code>	<code>the_author()</code>
<code>get_the_category_list()</code>	<code>the_category()</code>
<code>get_the_content()</code>	<code>the_content()</code>
<code>get_the_excerpt()</code>	<code>the_excerpt()</code>
<code>get_the_ID()</code>	<code>the_ID()</code>
<code>get_permalink()</code>	<code>the_permalink()</code>
<code>get_the_tag_list()</code>	<code>the_tags()</code>
<code>get_the_time()</code>	<code>the_time()</code>
<code>get_the_title()</code>	<code>the_title()</code>
<code>get_comments_link()</code>	<code>comments_link()</code>

## Boucle WP: exemple d'utilisation des marqueurs de modèles

Par exemple avec le thème **twentytwenty**,  
extrait du modèle partiel **template-parts/content.php** d'affichage d'un article  
sur la page du template `index.php` (page du blogue du site avec la liste des articles) :

```
<div class="entry-content">

    <?php
    if ( is_search() || ! is_singular() && 'summary' === get_theme_mod( 'blog_content', 'full' ) ) {
        the_excerpt();
    } else {
        the_content( __( 'Continue reading', 'twentytwenty' ) );
    }
    ?>

</div><!-- .entry-content -->
```

## Boucle WP: pratique

Modifier la page qui affiche la liste des articles du blogue, en affichant pour chaque article :

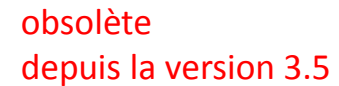
- son titre cliquable avec son permalien et dans une balise <h3>,
- sa date précédée de la mention 'créé le ',
- son auteur, précédé de la mention 'auteur : ',
- ses catégories, précédé de la mention 'catégorie(s) : ',
- un résumé de l'article, précédé de la mention 'résumé : ',
- un lien vers les commentaires.

Pour cela dupliquer index.php pour conserver l'original, puis mettre en commentaire la ligne `get_template_part( 'template-parts/content', get_post_type() );` si vous utilisez le thème twentytwenty et insérer votre code dans la boucle WP.



# **Base de données de WordPress**

## BDD WP: schéma relationnel



## BDD WP: tables SQL

Il n'y a pas de contraintes d'intégrité MySQL (clés étrangères,...) pour décrire les relations entre les tables.

Tables	Descriptions
options	Contient les paramètres de configuration du site, ces paramètres sont mis à jour par l'installation de WP et par l'interface d'administration. Contient également les paramètres de configuration des extensions (plugins).
users, usermeta	Contient les données utilisateurs (identifiant de connexion, mot de passe, courriel,...). usermeta contient des informations complémentaires dont le rôle (user level).
<b>posts</b> , postmeta	La plus importante, contient tout le contenu du site: articles, pages, médias, items de menus,... postmeta contient des informations complémentaires aux éléments de la table posts.
comments, commentmeta	Contient les commentaires des articles et des pages. commentmeta contient des informations complémentaires (utilisée par Akismet,...).
terms, termmeta, term_taxonomy	Contient les catégories et les mots-clés (tags) des articles , les noms des menus,... termmeta contient des informations complémentaires aux éléments de la table terms. term_taxonomy contient la classification pour chaque élément de la table terms.
term_relationships	Relie les éléments de la table terms (via term_taxonomy) à ceux de la table posts.

## BDD WP: tables SQL

- Exemples de requêtes directement sur la BDD :

Réinitialisation du mot de passe d'un utilisateur:

```
UPDATE n41_users SET user_pass = MD5("nouveau mot de passe")  
WHERE user_login = "nom"
```

Vidage de la table posts :

```
SELECT * FROM n41_posts  
WHERE post_type = "post" AND post_status = "publish"  
ORDER BY post_date DESC
```

post\_type: post, page, revision, attachment (média), nav\_menu\_item,...

post\_status: publish, inherit (révision, sauvegarde automatique,...), private, trash,...

- Les développeurs d'extensions peuvent ajouter des nouvelles tables aux tables WP, si les fonctionnalités qu'ils rajoutent le nécessitent.

## BDD WP: programmation, utilisation de la classe wpdb

- Tous les accès à la BDD se font en utilisant les méthodes de la classe wpdb.

La classe wpdb se trouve dans le fichier /wp-includes/wp-db.php

Elle est instanciée dans la phase d'initialisation du traitement d'une URL, par le fichier /wp-includes/load.php :

```
/**
 * Load the database class file and instantiate the `$wpdb` global.
 *
 * @since 2.5.0
 *
 * @global wpdb $wpdb The WordPress database class.
 */
function require_wp_db() {

    global $wpdb;

    require_once( ABSPATH . WPINC . '/wp-db.php' );
    if ( file_exists( WP_CONTENT_DIR . '/db.php' ) )
        require_once( WP_CONTENT_DIR . '/db.php' );
    if ( isset( $wpdb ) ) {
        return;
    }

    $wpdb = new wpdb( DB_USER, DB_PASSWORD, DB_NAME, DB_HOST );
}
```

## BDD WP: programmation, utilisation de la classe wpdb

Pour utiliser les méthodes de l'objet instancié avec la classe wpdb, il faut déclarer la variable globale \$wpdb qui est la référence de cet objet, comme par exemple dans cette séquence de code :

```
global $wpdb;

// $wpdb->prefix renvoie le préfixe décidé à l'installation (par ex. n41_)
$nomTable = $wpdb->prefix."ma_table";

// préparation de la requête pour se protéger contre les injections SQL
// malveillantes
$col1 = 100; // numérique
$col2 = "je suis une variable textuelle"; // chaîne de caractères
$req = $wpdb->prepare(
    "INSERT INTO $nomTable (col1, col2) VALUES (%d, %s)",
    $col1,
    $col2 );

// exécution de la requête avec la méthode générique query
$wpdb->query($req);
```

<https://developer.wordpress.org/reference/classes/wpdb>

<https://developer.wordpress.org/reference/classes/wpdb/prepare/>

<https://developer.wordpress.org/reference/classes/wpdb/query/>

## BDD WP: programmation, requêtes d'interrogation (1)

`$wpdb->get_row()` : récupère une ligne de résultat

```
// $wpdb->posts renvoie posts avec le préfixe décidé à l'installation,  
// soit par exemple n41_posts  
$resultat = $wpdb->get_row( "SELECT * FROM $wpdb->posts WHERE ID = 1" );  
echo $resultat->post_title;  
  
$resultat = $wpdb->get_row( "SELECT * FROM $wpdb->posts WHERE ID = 1" ),  
                        ARRAY_A );  
echo $resultat['post_title'];  
  
$resultat = $wpdb->get_row( "SELECT * FROM $wpdb->posts WHERE ID = 1" ),  
                        ARRAY_N );  
echo $resultat[5];
```

La requête doit être préparée avec `$wpdb->prepare()` si des éléments qui la constituent proviennent par exemple d'un formulaire, pour se prémunir contre les injections SQL.

## BDD WP: programmation, requêtes d'interrogation (2)

**\$wpdb->get\_results()** : récupère un tableau de plusieurs lignes de résultat

```
$resultat = $wpdb->get_results(
    "SELECT * FROM $wpdb->posts WHERE post_status = 'publish'",
    OBJECT ); // valeur par défaut
// ARRAY_A ;
// ARRAY_N ;

foreach ( $resultat as $postPublish ) {
    ?>
    <p><?php echo $postPublish->post_title ?></p>
    <p><?php // echo $postPublish['post_title'] ?></p>
    <p><?php // echo $postPublish[5] ?></p>
    <?php
}
```

La requête doit être préparée avec `$wpdb->prepare()` si des éléments qui la constituent proviennent par exemple d'un formulaire, pour se prémunir contre les injections SQL.



# BDD WP: programmation, requêtes d'insertion

**\$wpdb->insert()** : exécute une requête SQL INSERT

```
// $col1 numérique (par ex. 100)
// $col2 chaîne de caractères ( par ex. "je suis une variable textuelle")

$nbLignes = $wpdb->insert(
    $wpdb->prefix.'ma_table',
    array(
        'col1' => $col1,
        'col2' => $col2
    ),
    array(
        '%d',
        '%s'
    )
);
```

Récupération de l'id d'auto-incrément dans la propriété **\$wpdb->insert\_id**

Cette méthode intègre une préparation de la requête pour se prémunir contre les injections SQL.

<https://developer.wordpress.org/reference/classes/wpdb>

# BDD WP: programmation, requêtes de mise à jour

**\$wpdb->update()** : exécute une requête SQL UPDATE

```
// $col1 numérique (par ex. 100)
// $col2 chaîne de caractères ( par ex. "je suis une variable textuelle")
// $col3 numérique (par ex. valeur de la clé primaire col3)

$nbLignes = $wpdb->update(
    $wpdb->prefix.'ma_table',
    array(                                     // champs modifiés
        'col1' => $col1,
        'col2' => $col2
    ),
    array( 'col3' => $col3 ), // champs de la clause where
    array(                                     // formats des champs modifiés
        '%d',
        '%s'
    ),
    array( '%d' ) // format champs de la clause where
);
```

Cette méthode intègre une préparation de la requête pour se prémunir contre les injections SQL.

<https://developer.wordpress.org/reference/classes/wpdb>

## BDD WP: programmation, requêtes de suppression

**\$wpdb->delete()** : exécute une requête SQL DELETE

```
// $col1 numérique (par ex. valeur de la clé primaire col1)

$nbLignes = $wpdb->delete(
    $wpdb->prefix.'ma_table',
    array(                                     // champs de la clause where
        'col1' => $col1
    ),
    array(                                     // formats champs de la clause where
        '%d'
    )
);
```

Cette méthode intègre une préparation de la requête pour se prémunir contre les injections SQL.

# BDD WP: programmation, créer une table ou changer sa structure

**fonction dbDelta()** : n'est pas dans l'API WP (charger /wp-admin/includes/upgrade.php)

```
global $wpdb

$sql = "CREATE TABLE $wpdb->prefix"."ma_table (
    ID    bigint(20) UNSIGNED NOT NULL AUTO_INCREMENT,
    date  datetime NOT NULL DEFAULT '0000-00-00 00:00:00',
    text  varchar(255) NOT NULL DEFAULT '',
    PRIMARY KEY(ID)
) ".$wpdb->get_charset_collate();

require_once( ABSPATH . 'wp-admin/includes/upgrade.php' );

dbDelta( $sql );
```

# BDD WP: pratique

## historique des actions utilisateurs (connexions,...)

### Création d'une table personnalisée "history"

```
CREATE TABLE n41_history (  
    id          bigint(20) UNSIGNED NOT NULL AUTO_INCREMENT,  
    date_mes    datetime NOT NULL,  
    user_id     bigint(20),  
    level_mes   varchar(20) NOT NULL DEFAULT '',  
    message     varchar(255) NOT NULL DEFAULT '',  
    PRIMARY KEY(id)  
) DEFAULT CHARACTER SET utf8mb4 COLLATE utf8mb4_unicode_ci;
```

### Chargement du contenu de la table "history"

```
INSERT INTO n41_history (id, date_mes, user_id, level_mes, message) VALUES  
(1, '2020-02-07 10:10:00', 1, 'info', 'Connecté'),  
(2, '2020-02-07 12:25:32', NULL, 'avertissement', 'Échec de connexion avec  
l\'identifiant: administrateur'),  
(3, '2020-02-07 13:15:22', 2, 'info', 'Connecté'),  
(4, '2020-02-07 13:20:06', 2, 'info', 'Déconnecté'),  
(5, '2020-02-07 14:52:44', 1, 'info', 'Déconnecté');
```

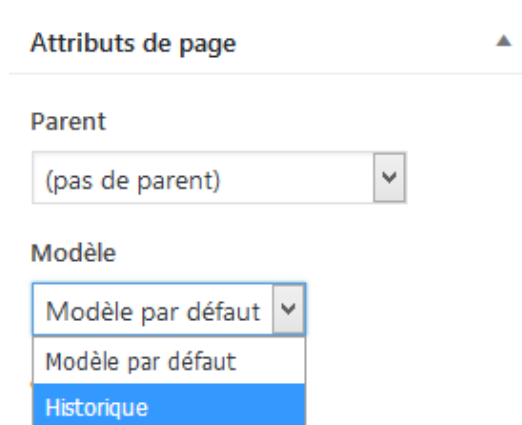
# BDD WP: pratique

## affichage de la table "history" dans un modèle de page

- Créer un modèle de page personnalisé dans le thème "Twenty Twenty" :
  - Dupliquer le fichier `index.php` en un fichier **template-history.php**
  - Remplacer les commentaires en tête du fichier par :

```
/**  
 * Template Name: Historique  
 *  
 */
```

pour que son nom apparaisse dans la liste déroulante de sélection du modèle, dans la fenêtre d'administration de création d'une page :



Attributs de page

Parent

(pas de parent)

Modèle

Modèle par défaut

Modèle par défaut

Historique

# BDD WP: pratique

## affichage de la table "history" dans un modèle de page

- Créer une page "Historique" via l'interface d'administration, en sélectionnant le modèle "Historique" :

The screenshot shows the WordPress administration interface. On the left is a dark sidebar with a menu. The 'Pages' menu item is highlighted in blue. Below it, the 'Ajouter' (Add) button is visible. The main content area is a light beige color and displays the title 'Historique' in a large, bold, black font, with a red circle containing the number '1' next to it. Below the title, there is a prompt: 'Commencez à écrire ou saisissez « / » pour choisir un bloc'. On the right side of the interface is a sidebar with various settings. At the top of this sidebar, there are buttons for 'Prévisualiser' (Preview) and 'Publier...' (Publish), with a red circle containing the number '3' next to the 'Publier...' button. Below these buttons is a tabbed interface with 'Document' and 'Bloc' tabs. The 'Document' tab is active. Under the 'Document' tab, there are sections for 'État et visibilité' (Status and visibility), 'Image mise en avant' (Featured image), 'Discussion', and 'Attributs de page' (Page attributes). The 'Attributs de page' section is expanded, showing a 'Modèle :' (Template) dropdown menu. A red circle containing the number '2' is next to this dropdown menu. The dropdown menu is open, showing several options: 'Historique' (which is highlighted in blue), 'Modèle par défaut' (Default template), 'Modèle avec bannière' (Template with banner), and 'Modèle pleine largeur' (Full-width template).

## BDD WP: pratique

### affichage de la table "history" dans un modèle de page

- Vérifier le bon affichage de la page "Historique", pour cela :
  - Remplacer le code de la boucle WP

```
while ( have_posts() ) {  
    $i++;  
    if ( $i > 1 ) {  
        echo '<hr class="post-separator styled-separator is-style-wide section-inner" aria-hidden="true" />';  
    }  
    the_post();  
  
    get_template_part( 'template-parts/content', get_post_type() );  
}
```

par le code HTML :

?><p>Vous utilisez le modèle de page "Historique"</p><?php

- Saisir le permalien de la page dans la barre d'adresse du navigateur ,  
par exemple si le site est installé dans le dossier n41 :  
<http://localhost/n41/historique>



## BDD WP: pratique

### affichage de la table "history" dans un modèle de page

- Complétez le modèle de page `template-history.php` pour successivement :
  - Récupérer le contenu de la table "history" dans un tableau `$historyList` et l'afficher avec la fonction PHP `print_r()`.
  - Améliorer la présentation avec une mise en forme HTML et CSS pour chaque ligne du tableau `$historyList`.
  - Limiter l'affichage aux seuls utilisateurs enregistrés et connectés.
  - Restreindre l'affichage aux actions de cet utilisateur, s'il n'est pas administrateur. L'identifiant de l'utilisateur dans la table "history" est le champ "user\_id" et correspond au champ "ID" de la table `n41_users`.

# BDD WP: pratique

## affichage de la table "history" dans un modèle de page

### ➤ Accès aux informations des utilisateurs :

#### ✓ Récupération d'un objet WP\_User par :

```
$oCurrentUser = wp_get_current_user() // utilisateur courant connecté  
$oUser = get_userdata($user_id)      // utilisateur d'ID $user_id
```

#### puis accès aux propriétés comme :

```
$oUser->ID  
$oUser->display_name
```

#### et aux méthodes comme :

```
$oUser->has_cap($cap) // par exemple $cap = 'administrator'  
                    // retourne un booléen
```

#### ✓ Des fonctions comme :

```
is_user_logged_in() // retourne un booléen  
current_user_can($cap) // alternative à $oCurrentUser->has_cap($cap)
```

# **Extension simple de WordPress avec création d'un formulaire**

## Extension WP: cas pratique, saisie de recettes de cuisine

- Il s'agit de créer une extension pour saisir des recettes de cuisine en stockant les informations dans une table 'recipes' décrite ainsi :

```
CREATE TABLE n41_recipes(  
  id          bigint(20)    UNSIGNED NOT NULL AUTO_INCREMENT,  
  title       varchar(255) NOT NULL,  
  ingredients  text         NOT NULL,  
  instructions text        NOT NULL,  
  prep_time   int(11)       UNSIGNED NOT NULL,  
  cook_time   int(11)       UNSIGNED NOT NULL,  
  PRIMARY KEY(id)  
) DEFAULT CHARACTER SET utf8mb4 COLLATE utf8mb4_unicode_ci;
```

## Extension WP: cas pratique, étapes

- E1- Créer le dossier et le fichier principal de l'extension, puis initialiser l'en-tête du fichier principal pour afficher l'extension dans l'administration.
- E2- Écrire une fonction qui affiche le formulaire.
- E3- Écrire une fonction qui insère les informations saisies, dans la table 'recipes'.
- E4- Écrire une fonction pour traiter le formulaire, qui regroupe les deux fonctions précédentes, et l'associer à un "code court" (shortcode).  
Ce code court devra être ajouté manuellement dans une page "Saisie d'une recette" à créer via l'administration.
- E5- Écrire le code pour créer la table 'recipes' lors de l'activation de l'extension.
- E6- Écrire le code pour supprimer la table 'recipes' lors de la suppression de l'extension.

## Extension WP: cas pratique, étape 1

E1- Créer le dossier et le fichier principal de l'extension, puis initialiser l'en-tête du fichier principal pour afficher l'extension dans l'administration.

`/wp-content/plugins/n41-recipes/n41-recipes.php`

```
<?php
/*
Plugin Name: N41 Recipes (1)
Plugin URI: https://n41.plugins.com (2)
Description: Gestion de recettes de cuisine (3)
Version: 1.0 (4)
Author: N41 (5)
Author URI: https://n41.com (6)
*/
```

`http://localhost/wordpress/wp-admin/plugins.php`

(1)	(3)
<input type="checkbox"/> N41 Recipes	Gestion de recettes de cuisine
<a href="#">Activer</a>   <a href="#">Supprimer</a>	Version 1.0   Par N41   <a href="#">Aller sur le site de l'extension</a>
(4)	(5) (2)
	(6)

## Extension WP: cas pratique, étape 2

E2- Écrire une fonction qui affiche le formulaire.

```
/**
 * Création du formulaire de saisie d'une recette
 *
 * @param none
 * @return echo html form recipe code
 */
function html_form_recipe_code() {
?>
    <form action="<?php echo esc_url( $_SERVER['REQUEST_URI'] ) ?>" method="post">
        <label>Nom de la recette</label>
        <input type="text" name="title" required>
        <label>Ingrédients</label>
        <textarea name="ingredients" placeholder="une ligne par ingrédient" required></textarea>
        <label>Instructions</label>
        <textarea name="instructions" required></textarea>
        <label>Temps de préparation</label>
        <input type="text" name="prep_time" placeholder="nombre de minutes" required
            pattern="[1-9][0-9]*">
        <label>Temps de cuisson</label>
        <input type="text" name="cook_time" placeholder="nombre de minutes" required
            pattern="[1-9][0-9]*">
        <input type="submit" name="submitted" value="Envoyez">
        <?php wp_nonce_field( 'add_a_recipe', 'nonce_recipe' ); ?>
    </form>
<?php
}
```

## Extension WP: cas pratique, étape 3

E3- Écrire une fonction qui insère les informations saisies, dans la table 'recipes'.

```
/**
 * Insertion d'une recette dans la table recipes
 * @param none
 * @return none
 */
function insert_recipe() {
    // si le bouton submit est cliqué
    if ( isset( $_POST['submitted'] ) ) {
        if ( wp_verify_nonce($_POST['nonce_recipe'], 'add_a_recipe') ) {
            // assainir les valeurs du formulaire
            $title          = sanitize_text_field( $_POST["title"] );
            $ingredients     = sanitize_textarea_field( $_POST["ingredients"] );
            $instructions    = sanitize_textarea_field( $_POST["instructions"] );
            $prep_time       = sanitize_text_field( $_POST["prep_time"] );
            $cook_time       = sanitize_text_field( $_POST["cook_time"] );
            // insertion dans la table
            global $wpdb;
            $wpdb->insert( $wpdb->prefix.'recipes',
                array('title' => $title, 'ingredients' => $ingredients,
                    'instructions' => $instructions, 'prep_time' => $prep_time,
                    'cook_time' => $cook_time),
                array('%s','%s','%s','%d','%d')
            );
        }
        ?> <p>La recette a été enregistrée.</p><?php
    } else {
        ?> <p>La recette n'a pas été enregistrée.</p><?php
    }
}
```



## Extension WP: cas pratique, étape 4

E4- Écrire une fonction pour traiter le formulaire, qui regroupe les deux fonctions précédentes, et l'associer à un "code court" (shortcode).

```
/**
 * Exécution du code court (shortcode) de saisie d'une recette
 *
 * @param none
 * @return the content of the output buffer (end output buffering)
 */
function shortcode_input_form_recipe() {
    ob_start(); // temporisation dans un buffer (mémoire tampon) de l'envoi du code HTML
    insert_recipe();
    html_form_recipe_code();
    return ob_get_clean(); // fin de la temporisation, retour du buffer au programme appelant
}

// créer un shortcode pour afficher et traiter le formulaire
add_shortcode( 'saisie_recette', 'shortcode_input_form_recipe' );
```

La fonction WP `add_shortcode()` associe le code court `saisie_recette` à la fonction `shortcode_input_form_recipe()`.

L'insertion du shortcode `[saisie_recette]` dans le contenu d'une page provoquera l'exécution de la fonction `shortcode_input_form_recipe()` et par conséquent l'affichage du formulaire et son traitement.

## Extension WP: cas pratique, étape 5

E5- Écrire le code pour créer la table 'recipes' lors de l'activation de l'extension.

- ☞ Commencer par écrire la fonction `n41_recipes_create_table()` qui crée la table en utilisant la fonction WP `dbDelta()`

```
/**
 * Création de la table recipes
 *
 * @param none
 * @return none
 */
function n41_recipes_create_table() {
    global $wpdb;
    $sql = "CREATE TABLE $wpdb->prefix"."recipes (
        id            bigint(20)    UNSIGNED NOT NULL AUTO_INCREMENT,
        title         varchar(255)  NOT NULL,
        ingredients   text          NOT NULL,
        instructions  text          NOT NULL,
        prep_time     int(11)        UNSIGNED NOT NULL,
        cook_time     int(11)        UNSIGNED NOT NULL,
        PRIMARY KEY   (id)
    ) ".$wpdb->get_charset_collate();
    require_once( ABSPATH . 'wp-admin/includes/upgrade.php' );
    dbDelta( $sql );
}
```

## Extension WP: cas pratique, étape 5 (suite)

E5- Écrire le code pour créer la table 'recipes' lors de l'activation de l'extension.

- ➡ Puis intégrer cette fonction dans une fonction `n41_recipes_activate()` qui regroupe les fonctions à exécuter lors de l'activation de l'extension.

```
/**
 * Traitements à l'activation de l'extension
 *
 * @param none
 * @return none
 */
function n41_recipes_activate() {
    n41_recipes_create_table();
}
```

- ➡ Enfin ajouter l'exécution de la fonction WP `register_activation_hook()` qui ajoute la fonction `n41_recipes_activate()` à la liste des fonctions à exécuter lorsque le crochet d'action (hook) associé à l'activation de cette extension, est déclenché par le noyau WP.

```
register_activation_hook( __FILE__, 'n41_recipes_activate' );
```

Le premier paramètre indique le chemin vers le fichier courant qui est le fichier principal de l'extension.

## Extension WP: cas pratique, étape 6

E6- Écrire le code pour supprimer la table 'recipes' lors de la suppression de l'extension.

- ☞ Commencer par écrire la fonction `n41_recipes_drop_table()` qui supprime la table en utilisant la méthode `query` de la classe WP `$wpdb`

```
/**
 * Suppression de la table recipes
 *
 * @param none
 * @return none
 */
function n41_recipes_drop_table() {
    global $wpdb;
    $sql = "DROP TABLE $wpdb->prefix"."recipes";
    $wpdb->query($sql);
}
```

- ☞ Puis intégrer cette fonction dans une fonction `n41_recipes_uninstall()` qui regroupe les fonctions à exécuter lors de la suppression de l'extension.

```
/**
 * Traitements à la désinstallation de l'extension
 *
 * @param none
 * @return none
 */
function n41_recipes_uninstall() {
    n41_recipes_drop_table();
}
```

## Extension WP: cas pratique, étape 6 (suite)

E6- Écrire le code pour supprimer la table 'recipes' lors de la suppression de l'extension.

- ✎ Enfin ajouter l'exécution de la fonction WP `register_uninstall_hook()` qui ajoute la fonction `n41_recipes_uninstall()` à la liste des fonctions à exécuter lorsque le crochet d'action (hook) associé à la désinstallation de cette extension, est déclenché par le noyau WP.

```
register_uninstall_hook(__FILE__, 'n41_recipes_uninstall');
```

Le premier paramètre indique le chemin vers le fichier courant qui est le fichier principal de l'extension.

## **Extension de WordPress :**

- **Création et suppression de pages**
- **Activation, désactivation et désinstallation**
- **Crochet (hook) de filtres**
- **Intégration de médias**

# Extension WP: création d'une page à l'activation

Compléter l'extension "n41-recipes" pour créer automatiquement une page qui va contenir le formulaire de saisie d'une recette.

👉 Créer la fonction `n41_recipes_create_pages()`

```
/**
 *Création des pages de l'extension
 *
 * @param none
 * @return none
 */
function n41_recipes_create_pages() {
    $n41_recipes_page = array(
        'post_title'      => "Saisie d'une recette",
        'post_name'       => "saisie-recette",
        'post_content'    => "[saisie_recette]",
        'post_type'       => 'page',
        'post_status'     => 'publish',
        'comment_status'  => 'closed',
        'ping_status'     => 'closed',
        'meta_input'      => array('n41_recipes' => 'form')
    );
    wp_insert_post($n41_recipes_page);
}
```

La valeur 'form' de cette métadonnée 'n41\_recipes' permet d'identifier de manière unique ce post de page.



La fonction WP `wp_insert_post()` insère un post (ici de type page) dans la table posts, et une métadonnée 'n41\_recipes' dans la table postmeta, avec les paramètres du tableau `$n41_recipes_page`. La métadonnée 'n41\_recipes' est utilisée par l'extension pour identifier les posts qu'elle gère.

## Extension WP: création d'une page à l'activation

Compléter l'extension "n41-recipes" pour créer automatiquement une page qui va contenir le formulaire de saisie d'une recette. (suite)

- ➡ Puis ajouter cette fonction dans la fonction `n41_recipes_activate()` déjà créée qui regroupe les fonctions à exécuter lors de l'activation de l'extension.

```
/**
 * Traitements à l'activation de l'extension
 *
 * @param none
 * @return none
 */
function n41_recipes_activate() {
    n41_recipes_create_table();
    n41_recipes_create_pages();
}
```

Rappel: la fonction `n41_recipes_activate()` est enregistrée dans le crochet (hook) d'activation de l'extension par `register_activation_hook()`.



# Extension WP: suppression d'une page à la désactivation

Compléter l'extension "n41-recipes" pour supprimer automatiquement les pages qui ont été créées à l'activation de l'extension.

☞ Créer la fonction `n41_recipes_delete_pages()`

```
/**
 * Suppression des pages de l'extension
 *
 * @param none
 * @return none
 */
function n41_recipes_delete_pages(){
    global $wpdb;
    $postmetas = $wpdb->get_results(
        "SELECT * FROM $wpdb->postmeta WHERE meta_key = 'n41_recipes'");
    $force_delete = true;
    foreach ($postmetas as $postmeta) {
        wp_delete_post( $postmeta->post_id, $force_delete );
    }
}
```

La fonction WP `wp_delete_post()` supprime définitivement (paramètre `$force_delete` à "true") la page à partir de son ID, dans la table `posts`, et également les métadonnées attachées à cette page dans la table `postmeta`.

# Extension WP: suppression d'une page à la désactivation

Compléter l'extension "n41-recipes" pour supprimer automatiquement les pages qui ont été créées à l'activation de l'extension. (suite)

- ➡ Puis intégrer cette fonction dans une fonction `n41_recipes_deactivate()` qui regroupe les fonctions à exécuter lors de la désactivation de l'extension.

```
/**
 * Traitements à la désactivation de l'extension
 *
 * @param none
 * @return none
 */
function n41_recipes_deactivate() {
    n41_recipes_delete_pages();
}
```

- ➡ Enfin ajouter l'exécution de la fonction WP `register_deactivation_hook()` qui ajoute la fonction `n41_recipes_deactivate()` à la liste des fonctions enregistrées par WP pour être exécutées lors de la désactivation de l'extension.

```
register_deactivation_hook( __FILE__, 'n41_recipes_deactivate' );
```

# Extension WP: pratique, ajout/suppression de la page de liste

Compléter l'extension "n41-recipes" :

- Créer une fonction `n41_recipes_html_list_code()` qui produit le code html d'affichage de la liste des recettes (récupérer le code du template correspondant dans le thème courant).
- Créer la fonction `n41_recipes_shortcode_list()` qui exécute la fonction précédente puis envoie le tampon (buffer) de sortie au navigateur.
- Créer le shortcode `[n41_recipes_list]` associé à la fonction précédente.
- Créer automatiquement la page "Recettes" avec pour contenu le shortcode précédent et la métadonnée 'n41\_recipes' => 'list', lors de l'activation de l'extension.

La suppression de cette page lors de la désactivation de l'extension, sera prise en charge par la fonction `n41_recipes_delete_pages()`, parce qu'elle a la métadonnée 'n41\_recipes' comme toutes les pages de l'extension.

# Extension WP: contrôle de la version de WP à l'activation

Si l'extension utilise des fonctionnalités qui nécessitent une version minimale de WP :

👉 Ajouter la fonction `n41_recipes_check_version()`

```
/**
 * Vérification de la version WP
 *
 * @param none
 * @return none
 */
function n41_recipes_check_version() {
    global $wp_version;
    if ( version_compare( $wp_version, '4.9', '<' ) ) {
        wp_die( 'Cette extension requiert WordPress version 4.9 ou plus.' );
    }
}
```

👉 Puis ajouter cette fonction dans la fonction `n41_recipes_activate()`

```
/**
 * Traitements à l'activation de l'extension
 *
 * @param none
 * @return none
 */
function n41_recipes_activate() {
    n41_recipes_check_version();
    n41_recipes_create_table();
    n41_recipes_create_pages();
}
```

## Extension WP: définition et principe du crochet (hook) de filtres

le crochet (hook) de filtres est un mécanisme de WP qui permet d'exécuter des fonctions de filtrage externes au noyau WP, pour modifier le contenu d'une variable du noyau WP.

Ce mécanisme permet de modifier le comportement du noyau WP sans modifier son code.

Les étapes chronologiques pour la bonne exécution d'un crochet sont :

1. Dans l'extension, créer une fonction de filtrage '`ma_fonction_de_filtrage`', qui reçoit la variable WP à modifier avec des paramètres complémentaires si besoin, et retourne la variable modifiée.

Puis ajouter cette fonction aux filtres du crochet associé à cette variable de WP par l'instruction `add_filter( 'nom_crochet_WP', 'ma_fonction_de_filtrage' );`

Cette instruction s'exécute au moment du chargement de l'extension.

Par conséquent ce filtre sera bien pris en compte au moment de l'exécution du crochet dans le noyau, moment qui intervient plus tard.

2. Dans le noyau WP, le crochet est exécuté (dans une des fonctions du noyau) par l'instruction `apply_filters( 'nom_crochet_WP', $nom_variable_WP)`

# Extension WP: définition et principe du crochet (hook) de filtres

Quelques crochets de filtres couramment utilisés :

- `the_title` appliqué au titre d'un article ou d'une page avant son affichage
- `the_content` appliqué au contenu d'un article ou d'une page avant son affichage
- `comment_text` appliqué au texte d'un commentaire avant son affichage
- `the_permalink` appliqué au permalien d'un article ou d'une page avant son affichage

Liste plus complète des crochets de filtres :

[https://codex.wordpress.org/Plugin\\_API/Filter\\_Reference](https://codex.wordpress.org/Plugin_API/Filter_Reference)

## Extension WP: pratique, crochet de filtres `the_title`

Dans l'extension "n41-recipes", compléter le titre d'une page "générique" d'affichage d'une recette en ajoutant un filtre au crochet de filtres `the_title`.

1. Au préalable, ajouter la fonction qui génère le code html d'affichage de cette page générique :

```
function n41_recipes_html_single_code() {  
    ?> <section style="margin: 0 auto; width: 80%; max-width: 100%; padding: 0"> <?php  
        global $wpdb;  
        $recipe_id = isset($_GET['id']) ? $_GET['id'] : null;  
        $sql = "SELECT * FROM $wpdb->prefix"."recipes WHERE id =%d";  
        $recipe = $wpdb->get_row($wpdb->prepare($sql, $recipe_id));  
        if ($recipe !== null) :  
            ?>  
            <div style="display: flex"><p style="width:250px; padding: 5px; color: #777">Ingrédients:</p>  
                <p style="padding: 5px"><?= stripslashes(nl2br($recipe->ingredients)) ?></p></div>  
            <div style="display: flex"><p style="width:250px; padding: 5px; color: #777">Instructions:</p>  
                <p style="padding: 5px"><?= stripslashes(nl2br($recipe->instructions)) ?></p></div>  
            <div style="display: flex"><p style="width:250px; padding: 5px; color: #777">Temps de préparation:</p>  
                <p style="padding: 5px"><?= $recipe->prep_time ?> minutes</p></div>  
            <div style="display: flex"><p style="width:250px; padding: 5px; color: #777">Temps de cuisson:</p>  
                <p style="padding: 5px"><?= $recipe->cook_time ?> minutes</p></div>  
            <?php  
            else :  
            ?>  
            <p>Cette recette n'est pas enregistrée.</p>  
            <?php  
            endif;  
        ?> </section> <?php  
    }
```

## Extension WP: pratique, crochet de filters the\_title (suite)

Dans l'extension "n41-recipes", compléter le titre d'une page "générique" d'affichage d'une recette en ajoutant un filtre au crochet de filters `the_title` (suite).

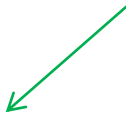
### 2. Puis ajouter la fonction qui affiche ce code et l'associer à un shortcode :

```
/**
 * Exécution du code court (shortcode) d'affichage d'une recette
 *
 * @param none
 * @return the content of the output buffer (end output buffering)
 */
function n41_recipes_shortcode_single() {
    ob_start(); // temporisation de sortie
    n41_recipes_html_single_code();
    return ob_get_clean(); // fin de la temporisation de sortie pour l'envoi au navigateur
}
// créer un shortcode pour afficher une recette
add_shortcode( 'n41_recipes_single', 'n41_recipes_shortcode_single' );
```

### 3. Enfin, insérer le code de création de la page dans `n41_recipes_create_pages()`

```
$n41_recipes_page = array(
    'post_title'      => "Recette",
    'post_content'    => "[n41_recipes_single]",
    'post_type'       => 'page',
    'post_status'     => 'publish',
    'comment_status'  => 'closed',
    'ping_status'     => 'closed',
    'meta_input'      => array('n41_recipes' => 'single')
);
wp_insert_post($n41_recipes_page);
```

La valeur 'single' de cette métadonnée 'n41\_recipes' permet d'identifier de manière unique ce post de page.





## Extension WP: pratique, crochet de filtres the\_title (suite)

Dans l'extension "n41-recipes", compléter le titre d'une page "générique" d'affichage d'une recette en ajoutant un filtre au crochet de filtres `the_title` (suite).

4. Pour permettre l'accès à cette page générique, associer le nom de chaque recette à un lien cliquable vers cette page en passant l'id de la recette en "query string", dans la page de liste des recettes.

Insérer pour cela les éléments en rouge, comme dans cet extrait :

```
if (count($recipes) > 0) :
```

-> récupération du lien vers la page générique d'affichage d'une recette :

```
$postmeta = $wpdb->get_row(
    "SELECT * FROM $wpdb->postmeta WHERE meta_key = 'n41_recipes' AND meta_value = 'single'");
$single_permalink = get_permalink($postmeta->post_id);
```

```
foreach ($recipes as $recipe) :
```

```
?>
```

```
<hr>
<article style="display: flex">
  <h4 style="margin: 0; width: 300px;">
```

-> le titre est un lien cliquable vers la page d'affichage d'une recette :

```
  <a href="<?php echo $single_permalink.'?page='.stripslashes($recipe->title).'&id='.$recipe->id?>">
    <?= stripslashes($recipe->title) ?>
  </a>
</h4>
<div>
  <div style="display: flex">
    <p style="width:250px; padding: 5px; color: #777">Ingrédients:</p>
    ...
```

## Extension WP: pratique, crochet de filtres the\_title (suite)

Dans l'extension "n41-recipes", compléter le titre d'une page "générique" d'affichage d'une recette en ajoutant un filtre au crochet de filtres `the_title` (suite).

### 5. Ajouter la fonction `n41_recipes_hook_the_title` ()

```
/**
 * Ajoute le nom de la recette dans le titre via le crochet de filtres the_title
 * pour la page "Recette" uniquement, identifiée par la métadonnée n41_recipes = single
 *
 * @param bool $title The current title
 * @param int $post_id The current post ID
 * @return bool false
 */
function n41_recipes_hook_the_title($title, $post_id) {
    $single = true;
    $n41_recipes = get_post_meta($post_id, 'n41_recipes', $single);
    if ($n41_recipes === 'single' && isset($_GET['id'])) {
        global $wpdb;
        $sql = "SELECT title FROM $wpdb->prefix"."recipes WHERE id =%d";
        $recipe = $wpdb->get_row($wpdb->prepare($sql, $_GET['id']));
        $title .= ':<br>'.stripslashes($recipe->title);
    }
    return $title;
}
```

Utilisation de la fonction `get_post_meta()` pour déterminer si le filtre s'applique au post courant, en testant la métadonnée 'n41\_recipes' à la valeur 'single' qui identifie la page générique d'affichage d'une recette. Puis récupération du nom de la recette dans la table `recipes` à partir de la variable `$_GET['id']` et concaténation à la variable WP `$title` avant de la retourner.

## Extension WP: pratique, crochet de filtres the\_title (suite)

Dans l'extension "n41-recipes", compléter le titre d'une page "générique" d'affichage d'une recette en ajoutant un filtre au crochet de filtres `the_title` (suite).

### 6. Ajouter cette fonction aux filtres du crochet `the_title`

```
add_filter( 'the_title', 'n41_recipes_hook_the_title', 10, 2 );
```

Le 3<sup>ème</sup> paramètre indique la priorité de prise en compte du filtre (10 par défaut).

Le 4<sup>ème</sup> paramètre indique le nombre de variables qui seront passées par le crochet à la fonction `n41_recipes_hook_the_title()`, ici 2 variables :

la première `$title` est la variable WP à modifier et la deuxième `$post_id` est un paramètre complémentaire utilisable dans le filtre.

### 7. Pour information, ce crochet est exécuté dans la fonction `get_the_title()` du fichier `wp-includes/post-template.php`, par l'instruction :

```
return apply_filters( 'the_title', $title, $id );
```

où `$id` est l'ID du post courant.

[https://developer.wordpress.org/reference/functions/add\\_filter/](https://developer.wordpress.org/reference/functions/add_filter/)  
[https://developer.wordpress.org/reference/functions/get\\_the\\_title/](https://developer.wordpress.org/reference/functions/get_the_title/)

## Extension WP: pratique, exemple d'intégration de médias

Dans l'extension "n41-recipes", ajouter le téléversement (upload) d'une image associée à chaque recette, et afficher cette image dans la page d'affichage d'une recette.

1. Modifier le formulaire de saisie d'une recette, comme dans cet extrait :

```
function html_form_recipe_code() {  
    ?>  
  
    <form action="<?php echo esc_url( $_SERVER['REQUEST_URI'] ) ?>"  
        method="post" enctype="multipart/form-data">  
  
    <label>Nom de la recette</label>  
  
    <input type="text" name="title" required>  
  
    <label>Image de la recette</label>  
  
    <input type="file" name="image" required>  
  
    <label>Ingrédients</label>  
  
    <textarea name="ingredients" placeholder="une ligne par ingrédient" required></textarea>
```

[https://www.w3schools.com/tags/att\\_form\\_enctype.asp](https://www.w3schools.com/tags/att_form_enctype.asp)

# Extension WP: pratique, exemple d'intégration de médias (suite)

Dans l'extension "n41-recipes", ajouter le téléversement (upload) d'une image associée à chaque recette, et afficher cette image dans la page d'affichage d'une recette (suite).

## 2. Modifier la fonction d'insertion d'une recette, comme dans cet extrait :

```
// insertion dans la table
global $wpdb;
$wpdb->insert( $wpdb->prefix.'recipes',
    array('title' => $title, 'ingredients' => $ingredients,
        'instructions' => $instructions, 'prep_time' => $prep_time,
        'cook_time' => $cook_time),
    array('%s', '%s', '%s', '%d', '%d'));
// générer le titre de l'image avec l'id de la recette insérée dans la table recipes
$recipe_image_title = "recipe-".$wpdb->insert_id;
// echo "<pre>".print_r($_FILES, true)."</pre>"; exit;
// chargement des fichiers nécessaires à l'exécution de la fonction media_handle_upload
require_once( ABSPATH . 'wp-admin/includes/image.php' );
require_once( ABSPATH . 'wp-admin/includes/file.php' );
require_once( ABSPATH . 'wp-admin/includes/media.php' );
// déplacement du fichier image dans le dossier wp-content/uploads
// et création d'un post de type attachment dans la table posts
// le premier paramètre 'image' est le nom du champ input qui suit dans $_FILES['image']
$recipe_image_post_id = media_handle_upload(
    'image', 0, array('post_title' => $recipe_image_title));
// ajouter une métadonnée n41_recipes dans la table postmeta, associée au post précédent,
// pour rattacher ce post à l'extension
$unique = true;
add_post_meta($recipe_image_post_id, 'n41_recipes', 'img', $unique);
?>
<p>La recette a été enregistrée.</p>
```

## Extension WP: pratique, exemple d'intégration de médias (suite)

Dans l'extension "n41-recipes", ajouter le téléversement (upload) d'une image associée à chaque recette, et afficher cette image dans la page d'affichage d'une recette (suite).

3. Modifier la fonction `n41_recipes_html_single_code()`, qui génère le code html de la page d'affichage d'une recette, comme dans cet extrait :

```
if ($recipe !== null) :  
    // récupération du post de l'image associée à la recette, dans la table posts,  
    // à partir du post_name qui contient l'id de la recette  
    $recipe_image_post = $wpdb->get_row(  
        "SELECT * FROM $wpdb->posts WHERE post_name = 'recipe-".$recipe->id.  
        "' AND post_type='attachment'");  
?  
<!-- l'url de ce média est obtenue avec wp_get_attachment_url -->  
  
    alt="<?php echo $recipe->title ?>">  
<div style="display: flex">  
...  

```

# Extension WP: pratique, exemple d'intégration de médias (suite)

Dans l'extension "n41-recipes", supprimer les images lors de la désinstallation.

1. Ajouter une fonction `n41_recipes_delete_images()` de suppression des images de l'extension :

```
/**
 * Suppression des images (posts et fichiers) de l'extension
 *
 * @param none
 * @return none
 */
function n41_recipes_delete_images() {
    global $wpdb;
    $postmetas = $wpdb->get_results(
        "SELECT * FROM $wpdb->postmeta WHERE meta_key = 'n41_recipes'");
    // booléen pour indiquer à la fonction wp_delete_post, la suppression des fichiers
    // et la suppression des informations dans les tables posts et postmeta
    $force_delete = true;
    foreach ($postmetas as $postmeta) {
        if ($postmeta->meta_value === 'img') {
            wp_delete_post( $postmeta->post_id, $force_delete );
        }
    }
}
```

# Extension WP: pratique, exemple d'intégration de médias (suite)

Dans l'extension "n41-recipes", supprimer les images lors de la désinstallation (suite).

## 2. Intégrer la fonction précédente dans la fonction n41\_recipes\_uninstall ():

```
/**
 * Traitements à la désinstallation de l'extension
 *
 * @param none
 * @return none
 */
function n41_recipes_uninstall() {
    n41_recipes_drop_table();
    n41_recipes_delete_images();
}
```



## Extension WP: pratique, exemple d'intégration de médias (suite)

Dans l'extension "n41-recipes", supprimer les images lors de la désinstallation (suite).

3. Modifier la fonction `n41_recipes_delete_pages()` pour ne pas supprimer les posts des images (et tout le contexte de ces images) pendant une désactivation de l'extension :

```
/**
 * Suppression des pages de l'extension, exceptés les posts des images
 *
 * @param none * @return none
 */
function n41_recipes_delete_pages() {
    global $wpdb;
    $postmetas = $wpdb->get_results(
        "SELECT * FROM $wpdb->postmeta WHERE meta_key = 'n41_recipes'");
    $force_delete = true;
    foreach ($postmetas as $postmeta) {
        // suppression lorsque la métadonnée n41_recipes n'est pas celle d'un post image
        if ($postmeta->meta_value !== 'img') {
            wp_delete_post( $postmeta->post_id, $force_delete );
        }
    }
}
```

## **Extension de WordPress :**

- **Création, modification et suppression d'options**
- **Crochet (hook) d'actions**
- **Gestion des réglages dans l'interface d'administration WP**

# Extension WP: création, modification, suppression d'options

La table "options" contient les paramètres du site.

Une extension WP peut l'utiliser pour y stocker ses propres paramètres de configuration.

Une option est caractérisée par :

- son nom qui doit être unique,
- sa valeur qui peut être une valeur simple de type chaîne de caractères ou numérique :

option_id	option_name	option_value
1	siteurl	http://localhost/wordpress

option_id	option_name	option_value
74	comments_per_page	50

un tableau (associatif ou pas) de valeurs, stocké sérialisé dans la table :

option_id	option_name	option_value
33	active_plugins	a:1:{i:0;s:19:"akismet/akismet.php";}

# Extension WP: création, modification, suppression d'options

Fonctions de base:

- `add_option( $name, $value = '', $deprecated = '', $autoload = 'yes' )`  
Crée l'option si elle n'existe pas et retourne le booléen 'true' dans ce cas, 'false' si elle n'a pas été créée.
- `update_option( $name, $value = '', $autoload = null )`  
Modifie l'option si elle existe ou la crée si elle n'existe pas et retourne le booléen 'true' en cas de succès, 'false' sinon.
- `get_option( $name )`  
Récupère l'option et retourne sa valeur si elle existe ou retourne le booléen 'false' sinon.
- `delete_option( $name )`  
Supprime l'option si elle existe et retourne le booléen 'true' en cas de succès.

Lorsqu'une option est créée avec le paramètre `autoload = 'yes'`, WP la chargera automatiquement dans sa phase d'initialisation.

## Extension WP: pratique, création des options pendant l'activation

Dans l'extension "n41-recipes", gérer une option "n41\_recipes\_settings" pour afficher ou pas chaque rubrique sur la page de liste.

L'option "n41\_recipes\_settings" doit être créée pendant l'activation de l'extension avec des valeurs par défaut. Ces valeurs pourront être ajustées ultérieurement via l'administration.

### 1. Pour cela créer la fonction `n41_recipes_default_settings()`

```
/**
 * Initialisation de l'option n41_recipes_settings,
 * qui regroupe un tableau de réglages pour l'affichage des rubriques sur la page de liste
 *
 * @param none
 * @return none
 */
function n41_recipes_default_settings() {
    add_option(
        'n41_recipes_settings',
        array(
            view_ingredients => 'yes',
            view_instructions => 'yes',
            view_prep_time => 'yes',
            view_cook_time => 'yes'
        )
    );
}
```

### 2. Puis l'ajouter dans la fonction `n41_recipes_activate()`

## Extension WP: pratique, création des options pendant l'activation

L'option "n41\_recipes\_settings" doit être créée pendant l'activation de l'extension avec des valeurs par défaut. Ces valeurs pourront être ajustées ultérieurement via l'administration (suite).

3. Vérifier l'insertion dans la table 'options' avec phpMyAdmin :

option_id	option_name	option_value	autoload
1514	n41_recipes_settings	a:4:{s:16:"view_ingredients";s:3:"yes";s:17:"view_instructions";s:3:"yes";s:14:"view_prep_time";s:3:"yes";s:14:"view_cook_time";s:3:"yes";}	yes

Le tableau de valeurs a été sérialisé par la fonction `add_option()` pour être stocké dans la table 'options'.

## Extension WP: pratique, suppression des options pendant la désinstallation

L'option "n41\_recipes\_settings" doit être supprimée pendant la désinstallation de l'extension.

1. Pour cela créer la fonction `n41_recipes_delete_settings ()`

```
/**
 * Suppression de l'option n41_recipes_settings
 *
 * @param none
 * @return none
 */
function n41_recipes_delete_settings() {
    delete_option('n41_recipes_settings');
}
```

2. Puis l'ajouter dans la fonction `n41_recipes_uninstall ()`

# Extension WP: pratique, exploiter les réglages dans la page de liste

Modifier la fonction `n41_recipes_html_list_code()`, en insérant le code en rouge :

- pour récupérer l'option "`n41_recipes_settings`" par la fonction `get_option()`,
- puis pour exploiter chacune des valeurs de l'option "`n41_recipes_settings`".

```
$single_permalink = get_permalink($postmeta->post_id);

$settings = get_option('n41_recipes_settings');

foreach ($recipes as $recipe) :
?>
    <hr>
    <article style="display: flex">
        <h4 style="margin: 0; width: 300px;">
            <a href="<?php echo $single_permalink.'?page='.stripslashes($recipe->title).'&id='.$recipe->id?>">
                <?= stripslashes($recipe->title) ?></a>
            </h4>
            <div>
<?php
    if (isset($settings['view_ingredients']) && $settings['view_ingredients'] === 'yes') :
?>
        <div style="display: flex">
            <p style="width:250px; padding: 5px; color: #777">Ingrédients:</p>
            <p style="padding: 5px"><?= stripslashes(nl2br($recipe->ingredients)) ?></p>
        </div>
<?php
    endif;
    if (isset($settings['view_instructions']) && $settings['view_instructions'] === 'yes') :
?>
        <div style="display: flex">
            <p style="width:250px; padding: 5px; color: #777">Instructions:</p>
            <p style="padding: 5px"><?= stripslashes(nl2br($recipe->instructions)) ?></p>
        </div>
    ...
```



# Extension WP: définition et principe du crochet (hook) d'actions

Le crochet (hook) d'actions est un mécanisme de WP qui permet d'exécuter des fonctions externes au noyau WP, à des moments autorisés du noyau WP.

Ce mécanisme permet de modifier le comportement du noyau WP sans modifier son code.

Les étapes chronologiques pour la bonne exécution d'un crochet sont :

1. Dans l'extension, créer une fonction d'action 'ma\_fonction\_d\_action'.  
Contrairement à une fonction de filtrage, cette fonction s'exécutera sans retourner de valeur.  
Puis ajouter cette fonction aux actions du crochet nécessaire pour son exécution, par l'instruction `add_action( 'nom_crochet_WP', 'ma_fonction_d_action' );`  
La programmation de cette instruction devra faire en sorte qu'elle s'exécute avant le moment de l'exécution du crochet dans le noyau.
2. Dans le noyau WP, le crochet est exécuté dans un programme ou une fonction du noyau, par l'instruction `do_action( 'nom_crochet_WP' )`.

# Extension WP: définition et principe du crochet (hook) d'actions

Quelques crochets d'actions couramment utilisés :

- `init` WP est presque totalement chargé à ce stade, l'utilisateur est authentifié et certaines extensions utilisent ce crochet pour initialiser des traitements nécessitant la connaissance de l'utilisateur,...  
Les headers ne sont pas encore envoyés.  
Il est possible d'intercepter les variables `$_GET` et `$_POST`.
- `wp_loaded` quand WP est complètement chargé (extensions et thème compris).
- `wp_head` pour intervenir dans la section `<head>` du thème.
- `wp_footer` pour intervenir dans la section `<footer>` du thème.

Liste plus complète des crochets d'actions :

[https://codex.wordpress.org/Plugin\\_API/Action\\_Reference](https://codex.wordpress.org/Plugin_API/Action_Reference)

# Extension WP: pratique, crochets d'actions pour gérer les réglages dans l'interface d'administration

Dans l'extension "n41-recipes", ajouter un fichier `n41-recipes-settings.php` qui sera chargé via `include` au début du fichier principal. Ce fichier va gérer les réglages de l'extension via une page des réglages dans l'interface d'administration. Sa structure est :

```
// l'exécution du hook 'admin_menu' sert à compléter le panneau d'administration,  
// pour les extensions et les thèmes  
add_action( 'admin_menu', 'n41_recipes_add_menu_page' );  
  
// ajout de la page formulaire des réglages dans le panneau d'administration  
→ function n41_recipes_add_menu_page() {  
    add_menu_page(<paramètres>);  
  
    // l'exécution du hook 'admin_init' sert à initialiser le traitement de la page des réglages,  
    // avant l'affichage du panneau d'administration  
    add_action( 'admin_init', 'n41_recipes_register_setting' );  
}  
  
// initialisation du traitement de la page formulaire des réglages  
→ function n41_recipes_register_setting() {  
    register_setting(<paramètres>);  
}
```

Codes des couleurs: actions de hook, fonctions de l'extension, fonctions WP des "Settings API"

# Extension WP: pratique, gestion des réglages dans l'interface d'administration

Dans l'extension "n41-recipes", compléter la structure du fichier `n41-recipes-settings.php` :

```
// l'exécution du hook 'admin_menu' sert à compléter le panneau d'administration,
// pour les extensions et les thèmes
add_action( 'admin_menu', 'n41_recipes_add_menu_page' );

// ajout de la page formulaire des réglages dans le panneau d'administration
function n41_recipes_add_menu_page() {
    add_menu_page('Réglages de l\'extension N41 Recipes', // balise title de la page des réglages
                  'N41 Recipes',                        // texte de menu de la page des réglages
                  'administrator',                      // dans le menu latéral gauche
                  'n41-recipes-settings-page',          // capacité pour afficher cette page
                  'n41_recipes_settings_page');         // slug dans l'url de la page
    // l'exécution du hook 'admin_init' sert à initialiser le traitement de la page des réglages,
    // avant l'affichage du panneau d'administration
    add_action( 'admin_init', 'n41_recipes_register_setting' );
}

// initialisation du traitement de la page formulaire des réglages
function n41_recipes_register_setting() {
    register_setting('n41_recipes_option_group', // nom de la zone des réglages, associée
                    'n41_recipes_settings',      // à la saisie des valeurs de l'option
                    'n41_recipes_sanitize_option'); // nom de l'option des réglages
    // fonction pour assainir les valeurs
    // de l'option des réglages
}
```

Codes des couleurs: actions de hook, fonctions de l'extension, fonctions WP des "Settings API"

[https://codex.wordpress.org/Settings\\_API](https://codex.wordpress.org/Settings_API)

# Extension WP: pratique, gestion des réglages dans l'interface d'administration

Dans l'extension "n41-recipes", compléter la structure du fichier `n41-recipes-settings.php` :

```
/**
 * Affichage de la page du formulaire des réglages ①
 *
 * @param none
 * @return none
 */
function n41_recipes_settings_page() {
    ?>
    <div class="wrap">
        <h2>Réglages de N41 Recipes</h2>
        <form method="post" action="options.php">

        <!-- génération de balises input cachés pour faire le lien
             avec la fonction register_setting par le paramètre option_group -->
        <?php settings_fields( 'n41_recipes_option_group' ); ?>

        <?php $n41_recipes_settings = get_option( 'n41_recipes_settings' ); ?>
        <h3>Visibilité des rubriques sur la page de liste</h3>
    </div>
}
```

Codes des couleurs: valeur de l'attribut action obligatoire, fonctions de l'extension,  
fonctions WP des "Settings API"

Balises input cachées, générées par la fonction **settings\_fields** :

```
<input type='hidden' name='option_page' value='n41_recipes_option_group' />
<input type="hidden" name="action" value="update" />
<input type="hidden" id="_wpnonce" name="_wpnonce" value="666da12dcd" />
<input type="hidden" name="_wp_http_referer" value="/wordpress/wp-admin/admin.php?page=n41-recipes-settings-page" />
```

# Extension WP: pratique, gestion des réglages dans l'interface d'administration

Dans l'extension "n41-recipes", compléter la structure du fichier `n41-recipes-settings.php` :

```
<table class="form-table"> ① suite
  <tr>
    <th scope="row">Ingrédients</th>
    <td><p>
      <input type="radio" name="n41_recipes_settings[view_ingredients]" value="yes"
      <?php checked( !isset( $n41_recipes_settings['view_ingredients']) ||
        $n41_recipes_settings['view_ingredients'] === 'yes' ) ?>>
      Oui<br>
      <input type="radio" name="n41_recipes_settings[view_ingredients]" value="no"
      <?php checked( isset( $n41_recipes_settings['view_ingredients']) &&
        $n41_recipes_settings['view_ingredients'] === 'no' ) ?>>
      non
    </p></td>
  </tr>
```

```
<!-- ainsi de suite pour les réglages view_instructions, view_prep_time, view_cook_time -->
```

```
</table>
<pre><?php // print_r($n41_recipes_settings); ?></pre>
<p class="submit">
  <input type="submit" class="button-primary" value="Enregistrer les modifications">
</p>
</form>
</div>
```

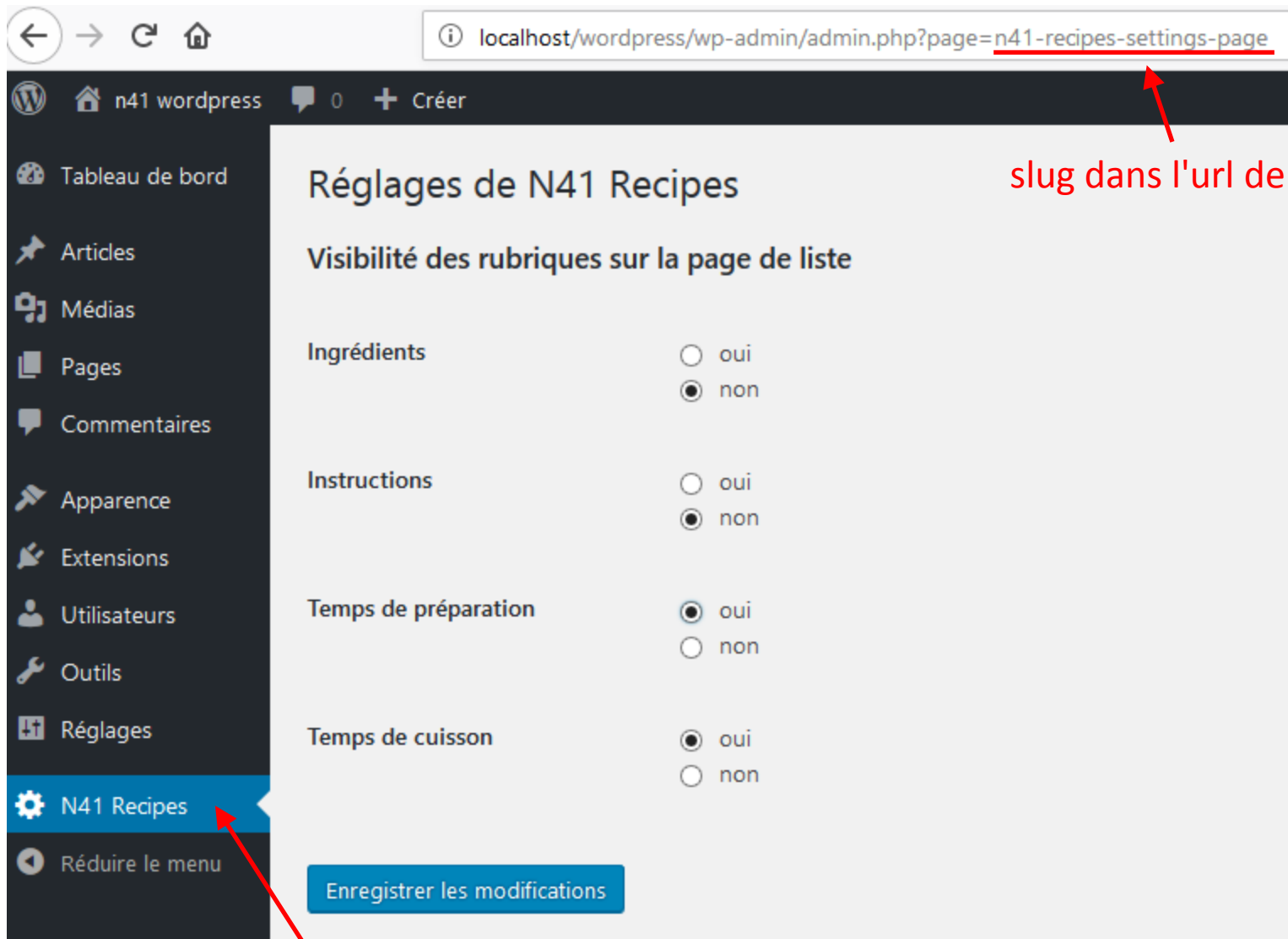
# Extension WP: pratique, gestion des réglages dans l'interface d'administration

Dans l'extension "n41-recipes", compléter la structure du fichier `n41-recipes-settings.php` :

```
/**
 * Assainissement des valeurs de l'option renvoyées par le formulaire des réglages ②
 *
 * @param none
 * @return none
 */
function n41_recipes_sanitize_option( $input ) {
    $input['view_ingredients'] = sanitize_text_field( $input['view_ingredients'] );
    $input['view_instructions'] = sanitize_text_field( $input['view_instructions'] );
    $input['view_prep_time']    = sanitize_text_field( $input['view_prep_time'] );
    $input['view_cook_time']    = sanitize_text_field( $input['view_cook_time'] );
    return $input;
}
```

Codes des couleurs: fonctions de l'extension

# Extension WP: pratique, gestion des réglages dans l'interface d'administration



slug dans l'url de la page

texte de menu de la page des réglages