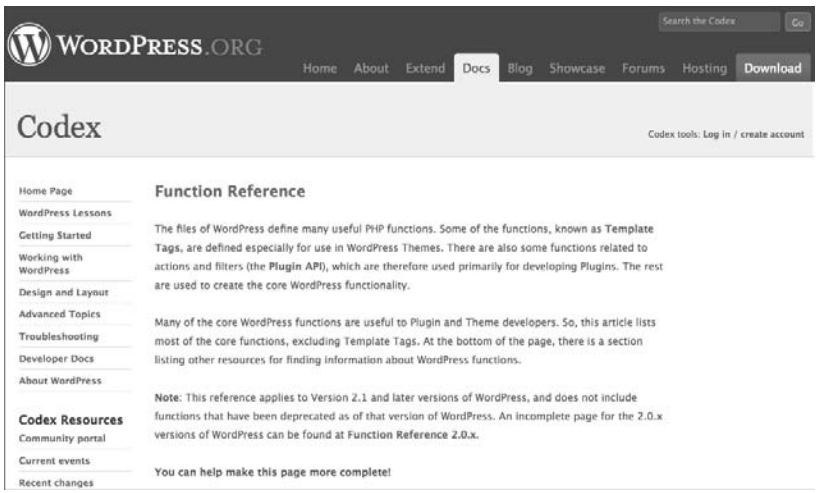
8 Philosophie des extensions WordPress

Le concept

L'une des forces de WordPress dans la jungle CMS, c'est la simplicité de création d'extensions. Il y a plusieurs raisons à cela. Des raisons techniques tout d'abord : WordPress est développé en PHP 4, sur un modèle classique. Il est très peu orienté POO (programmation objet) ; il est donc très simple de débuter la programmation PHP avec WordPress.

Ensuite, il y a des raisons qui lui sont propres : une documentation technique très complète sur le wiki de WordPress, le codex, **http://codex.wordpress.org/Developer_Documentation** (voir Figure 8.01), et un ensemble d'API mises à disposition par les développeurs de WordPress pour étendre et modifier facilement les fonctionnalités.

Figure 8.01
Le codex est un wiki collaboratif détaillant les entrailles de WordPress.



Notez que depuis la version 2.5 du logiciel un énorme travail est réalisé par l'équipe de WordPress pour documenter au fur et à mesure le cœur du logiciel. Cette documentation incluse directement dans les fichiers de WordPress peut être extraite *via* des logiciels tels que PHPdoc.

Cette documentation décrit chaque fonction et propose un résumé fonctionnel, l'auteur, la version d'ajout dans WordPress, sans oublier les paramètres que la fonction peut prendre, ainsi que la valeur de retour.

Les prérequis techniques

Créer une extension WordPress demande quelques connaissances techniques… En plus de bien connaître les rouages de WordPress d'un point de vue utilisateur, il est indispensable de maîtriser ou de comprendre plusieurs langages informatiques dont :

- le PHP ;
- le JavaScript ;
- le HTML ;
- le CSS.

Autrement dit les quatre langages utilisés pour le développement de WordPress en lui-même.

Le premier d'entre eux et le plus important ici est le langage PHP.

PHP est un langage de script libre ; il est principalement utilisé pour générer des pages Internet dynamiques *via* un serveur HTTP. Ce qu'il faut retenir ici, c'est que WordPress utilise PHP de façon très classique ; c'est l'ancienne école, diront certains. Il n'est pas question d'une programmation orientée objet mais d'une utilisation classique des fonctions, des variables globales et de quelques classes PHP 4. Car il ne faut pas l'oublier, une des forces de WordPress réside également dans une compatibilité PHP 4 qui lui permet un maximum de compatibilité sur le marché de l'hébergement !

Les trois autres langages sont communs à tous les projets Internet ; le JavaScript permet une meilleure approche utilisateur et une interactivité dans les fonctionnalités d'un site web, tandis que le code HTML et le CSS permettent la mise en page sur les navigateurs web.

L'API des hooks

Présentation

L'API la plus importante de WordPress est celle des hooks. En français, cette notion de hook peut être traduite par "crochet". Cette traduction a donné lieu à un débat enflammé sur le blog de WordPress Francophone : **http://www.wordpress-fr.net/blog/comment-traduiriez-vous-hook**.

Dans ce livre, nous utiliserons le terme de crochet. Sachez cependant qu'il existe d'autres traductions : marqueur, point d'ancrage…

Ces crochets sont parsemés dans tout le code de WordPress par l'équipe de développement. Ils ne sont pas placés aléatoirement, ce qui n'aurait aucun intérêt, mais suivant une logique propre à WordPress. En effet, pour chaque action, chaque événement de WordPress, les développeurs ont placé stratégiquement des crochets susceptibles d'être utilisés par des développeurs d'extensions.

D'ailleurs, dans chaque fonctionnalité implémentée dans WordPress, les développeurs vont penser, vont structurer le code pour y placer un ou plusieurs crochets, permettant ainsi à une extension de modifier ou étendre le comportement initial.

Vous avez bien lu, modifier ou étendre le comportement de WordPress. En effet, il est possible de différencier deux familles de crochets : les actions, et les filtres. La différence entre les deux est subtile, mais très importante.

Les filtres vont permettre de modifier le comportement de WordPress, alors que les actions vont étendre une fonctionnalité. Paradoxalement, on peut généralement réaliser la même chose avec les deux types de crochets… Pour comprendre, voici un petit exemple.

L'objectif de l'extension est d'ajouter un copyright à la fin du texte de chaque article WordPress.

Pour inclure un copyright avec une action, la première méthode consiste à ajouter une action lors de l'enregistrement de l'article, en automatisant l'ajout du copyright au texte au moment de l'insertion en base de données.

La seconde méthode, qui serait plus intéressante ici, consiste à ajouter un filtre lors de l'affichage du texte de l'article sur la partie visiteur de votre blog et à intégrer dynamiquement le fameux copyright au contenu de l'article.

Le filtre est ici la meilleure solution, car vous pourrez modifier le texte du copyright sur chaque article à la volée, chose impossible *via* la méthode de l'action et l'enregistrement dans la base de données de WordPress. Néanmoins, les deux méthodes présentent des inconvénients : ainsi, avec la technique du filtre, il sera impossible pour un auteur d'éditer le texte du copyright.

En conclusion, avec les filtres et les actions vous disposez de deux méthodes pour étendre ou modifier le comportement de WordPress. Il vous faudra peser le pour et le contre afin de choisir la meilleure solution pour répondre correctement et facilement à votre besoin.

Les filtres

Nous allons maintenant étudier un exemple pour décrire les différentes fonctions liées aux filtres ; ici nous souhaitons supprimer un mot spécifique lors de l'affichage.

Grâce à différentes listes de hooks, nous allons trouver le crochet lié à cette action. Dans notre exemple, le crochet utilisé à l'affichage de l'article est `the_content`.

Nous pouvons le trouver dans WordPress 2.6.1, à la ligne 79 du fichier wp-includes/post-template.php :

```
$content = apply_filters('the_content', $content);
```

Nous voyons que les développeurs utilisent la fonction `apply_filters()` pour donner à une extension la possibilité de filtrer le contenu de la variable PHP `$content`.

Généralement la fonction `apply_filters()` ne prend que deux paramètres : le nom du crochet, suivi de la variable à filtrer. Cependant, il n'est pas impossible de voir trois, quatre, voire cinq paramètres permettant de mieux cibler l'événement.

Pour ajouter un filtre, nous allons utiliser la fonction `add_filter()` dans le code de notre extension.

```
add_filter('the_content', 'je_filtre');
```

Cette fonction prend généralement deux paramètres. Le premier est le crochet ciblé, le deuxième est le nom de la fonction PHP qui sera exécutée lors de l'application du filtre. La fonction `add_filter()` peut également prendre un troisième paramètre permettant de modifier la priorité d'exécution de la fonction sur le filtre donné.

Par défaut, la priorité est de 10. Dans certains cas, il peut être intéressant d'exécuter une fonction PHP avant les filtres par défaut de WordPress. Dans ce cas, il suffit de préciser une priorité inférieure à 10. En effet, plus la priorité est faible, plus l'exécution aura lieu tôt, et inversement.

Exemple :

```
add_filter('the_content', 'je_filtre', 1);
```

Pour l'anecdote, sachez qu'il existe un quatrième paramètre permettant de définir le nombre de paramètres que peut recevoir la fonction exécutée par le filtre, mais il est très rarement utilisé dans le développement d'extension. Personnellement, nous ne l'avons jamais croisé, et nous ne l'avons jamais utilisé.

Revenons à l'exemple initial où nous souhaitons remplacer certains mots dynamiquement à l'affichage.

```
function je_filtre( $texte ) {
   $texte = str_replace( 'payant', 'gratuit', $texte );
   return $texte;
}
```

La fonction `je_filtre()` sera appelée lors de l'événement `the_content` ; elle doit posséder un paramètre pour pouvoir accepter la variable PHP `$content`. Nous procédons ensuite au traitement de la variable. Ici, nous remplaçons le mot payant par gratuit.

Il est très important de retourner une valeur avec la fonction PHP dans le cadre d'un filtre. Si aucune valeur n'est retournée par la fonction, la variable filtrée sera nulle.

La raison est simple et agit comme une fonction formatant une valeur avant d'être stockée dans une variable. Pour s'en convaincre, il suffit de regarder la fonction qui applique le filtre :

```
$content = apply_filters('the_content', $content);
```

Si la fonction `apply_filters()` ne retourne pas de valeur, la variable `$content` sera nulle. Or, cette fonction renvoie comme valeur celle qui est retournée par la fonction appelée par le filtre : ici il s'agit de `je_filtre()`.

Les actions

Maintenant que nous avons étudié les filtres, nous allons voir les différences avec les actions. Pour cela, nous allons prendre un exemple relativement simple.

Nous allons créer une alerte e-mail survenant lors de l'effacement d'un article.

Grâce aux bases de données des hooks, nous allons voir le crochet lié à cette action. Nous pouvons trouver dans le fichier wp-includes/post.php les actions suivantes :

```
À la ligne 1036 : do_action('delete_post', $postid);
Et à la ligne 1088 : do_action('deleted_post', $postid);
```

Deux actions pour la même chose ?

Non pas exactement. Ici le premier crochet est exécuté avant la suppression de l'article par WordPress, alors que le second est exécuté une fois que WordPress a terminé son travail d'effacement.

Dans notre cas, nous allons employer le premier crochet afin d'envoyer le contenu de l'article dans l'e-mail avant sa suppression de la base de données. Pour cela, nous utilisons la fonction `add_action()` sur l'événement `delete_post` :

```
add_action('delete_post', 'mail_article_efface');
```

Cette fonction se comporte en tout point de la même façon que `add_filter()`. Les arguments sont exactement les mêmes, le nom du crochet, le nom de la fonction PHP exécutée, ainsi que sa priorité.

Voici le détail de la fonction :

```
function mail_article_efface( $post_id ) {
   // Je récupère les données de l'article
   $post = get_post( $post_id );

   // Je construis le sujet de mon email avec le titre de l'article
   $sujet = "Effacement de l'article : " . $post->post_title;

   // Je construis le sujet de mon email avec le titre de l'article
   $message = "Contenu de l'article : " . $post->post_content;

   // J'envoi l'email !
   wp_mail( 'monemail@wordpress.fr', $sujet, $message );
}
```

Contrairement à un filtre, nous n'avons pas besoin de retourner de valeur. Et même si nous en retournons une, cela n'aura aucun impact sur le fonctionnement de WordPress.

Supprimer une action ou un filtre

Nous venons de voir comment ajouter un filtre dans WordPress, sachez qu'il est également possible d'en supprimer. Pour cela, il existe deux fonctions, `remove_action()` et `remove_filter()`, permettant de désactiver un filtre ou une action de WordPress, voire d'une extension tierce.

La fonction se présente de la façon suivante :

```
remove_action('delete_post', 'mail_article_efface');
```

La fonction `remove_filter()` fonctionne de la même façon.

Le premier argument est le nom du crochet, le second est le nom de la fonction à exécuter. Les deux paramètres doivent être complétés pour que la suppression fonctionne.

Ici il est très important de comprendre le concept de priorité. En effet, si l'on souhaite supprimer un filtre ou une action, il est impératif d'utiliser la fonction `remove_action()` ou `remove_filter()`, entre l'enregistrement du filtre et son exécution.

Les filtres et les actions par défaut

La façon la plus simple de trouver les filtres et les actions de WordPress consiste à effectuer une recherche dans le code source avec comme mots-clefs : `add_filter` et `add_action`.

Cependant, depuis la version 2.1 de WordPress, la plupart des filtres et actions par défaut se trouvent dans le fichier wp-includes/default-filters.php.

Vous pouvez néanmoins en trouver quelques autres dans les fichiers :

- wp-admin/admin-ajax.php ;
- wp-admin/admin-functions.php ;
- wp-admin/custom-header.php ;
- wp-admin/edit.php ;
- wp-admin/index.php ;
- wp-admin/options-permalink.php ;
- wp-admin/upload-functions.php ;
- wp-admin/upload.php ;
- wp-includes/bookmark.php ;
- wp-includes/general-template.php ;
- wp-includes/kses.php ;
- wp-includes/plugin.php ;
- wp-includes/rewrite.php ;
- wp-includes/template-loader.php ;
- wp-includes/theme.php.

Les fonctions amovibles de WordPress

Les fonctions pluggeables, que l'on peut traduire par "amovibles", sont des fonctions PHP de WordPress lancées après le chargement des extensions. De ce fait, elles peuvent être très facilement remplacées par une extension.

Les fonctions en question sont principalement celles qui sont liées à l'authentification et à la gestion utilisateur de WordPress. On retrouve néanmoins la fonction d'envoi d'e-mail de WordPress ainsi que des fonctions de redirection HTTP.

Techniquement, ces fonctions sont juste entourées de la condition :

```
if ( !function_exists('wp_mail') ) :
   function wp_mail( ... ) {
   ...
   }
endif;
```

Il suffit donc à une extension de posséder une fonction `wp_mail()` pour remplacer la fonction d'envoi d'e-mail de toutes les fonctionnalités de WordPress.

Dans le cadre de la fonction `wp_mail()`, cela est très pratique si l'on souhaite utiliser un serveur SMTP particulier à la place de la fonction PHP `mail()`.

Les fonctions d'authentification et de gestion d'utilisateurs permettront l'utilisation d'une base externe à WordPress, par exemple un annuaire LDAP ou une base d'utilisateurs tierce.

> Un chapitre intitulé « Présentation de la base de données de WordPress » est présent sur le CD-ROM offert avec cet ouvrage.