CMS MADE SIMPLE
Guide du développeur

Table des matières

1.Information document

1.1.Licence

1.2.Avertissements

Ce document a pour but de fournir à tous les moyens de bien commencer avec CMS Made Simple. Nous utiliserons donc volontairement un vocabulaire et une méthode de travail accessibles à un public de non informaticiens, le tout abondamment illustré par des captures d'écran.

Ce document ne se substitue en aucune manière aux documents officiels CMS Made Simple, mais se veut plutôt comme un document de synthèse des questions de base que se posent les nouveaux venus dans l'univers CMS Made Simple.

Les auteurs du présent document ne sauraient être tenus pour responsables des erreurs ou dysfonctionnements constatés lors des phases de tests ou de mise en production d'un site CMS Made Simple consécutifs à la lecture de ce document.

Les auteurs se réservent le droit de mettre à jour le présent document ou d'en modifier le contenu à tout moment. Vous trouverez la version la plus à jour de ce document sur l'espace boutique.

Vous êtes par ailleurs invités à signaler tous les problèmes liés au fond ou la forme de ce document sur le forum documentation.

www.cmsmadesimple.fr

Ensemble, faisons un geste pour l'environnement : n'imprimons que si nécessaire.

1.3.Remerciements

- Aux initiateurs du projet CMS Made Simple.

- Aux équipes de développeurs et de designers qui de par le monde contribuent à rendre CMS Made Simple toujours plus convivial, simple, performant et évolutif.

- A la communauté des utilisateurs.

- A nos familles.

1.4.Auteurs

Auteurs	Eric Menczynski - Pseudo Eric
Correcteurs	Alain Thebault - pseudo f5jmh Jean Claude Etiemble – pseudo JCE76350

1.5.Versions

Date	Version	Commentaires
23/11/08	0.1	Version origine

Notes :

- Etant donné les changements prévus pour la v2 (ORM activeRecord, framework Silk ...), la création de module n'est pas traité en détail dans ce document. Cette première version est seulement la traduction du wiki.

1.6.Conventions et signalétique

Afin de mieux vous guider, des indicateurs visuels sont utilisés pour attirer votre attention sur des points particuliers. Cette signalétique est expliquée ici.

Conseil :

Un conseil est illustré par ce symbole. Le suivre est laissé à l'appréciation de l'utilisateur .

Attention :

Vous rencontrerez ce symbole afin d'indiquer une remarque importante à laquelle vous devez apporter toute votre attention.

Astuce :

Ceci est une astuce, un moyen d'arriver au même résultat de manière plus simple ou plus directe.

Important:

Ce signal pour ce que vous ne devez pas faire, sous peine de problèmes.

Code :

```
Ceci indique qu'il faut modifier ou créer un code Php, Html, Css,
signale une intervention sur la base de données ou encore au niveau
du système.
```

2.Environement

2.1.Liens utiles

Télécharger CMS Made Simple:
http://www.cmsmadesimple.fr/telecharger-cms

Télécharger des modules et plugins, boutique francophone:
http://www.cmsmadesimple.fr/boutique-cms

Télécharger des modules:
http://dev.cmsmadesimple.org/project/list/module

Télécharger des plugins:
http://dev.cmsmadesimple.org/project/list/plugin

Chat développeur: Le chat est disponible sur *irc.freenode.net*
sur le canal #cms
Client : http://www.pidgin.im, https://addons.mozilla.org/fr/firefox/addon/16

Forge: http://dev.cmsmadesimple.org

Translation Center: http://translations.cmsmadesimple.org/login.php

Roadmap: http://www.cmsmadesimple.org/development/roadmap

API: http://api.cmsmadesimple.fr

SVN: http://viewsvn.cmsmadesimple.org

WIKI: http://wiki.cmsmadesimple.org

2.2.Participez !

CMS Made Simple est développé par une communauté mondiale de bénévoles. Comment y participer, comment contribuer ?

- Vous pouvez participer pour créer des modules, des plugins, des templates, de la documentation etc...
- Vous rejoindre l'équipe des développeurs sur IRC
- Vous pouvez tester les versions candidates, signaler des bugs.
- Vous pouvez également fournir des traductions.

Pour faire parti de l'équipe franchophone, il vous suffit lire la discussion sur le forum

2.3.Signaler un bug

Sur la forge, sélectionner le projet, puis cliquer sur l'onglet « Bug Tracker », et sur le lien « submit a new bug ».

2.4.Pre-Requis

Ce document suppose que vous êtes déjà familier avec :

- la programmation PHP
- le concept MVC (Modèle, Vue, Controller)
- les concepts de base de la programmation orientée objet (encapsulation, héritage, notions design patterns etc.)
- MySQL, syntaxe SQL, etc.
- PhpMyAdmin pour examiner la base de CMSMS.
- le fonctionnement de CMSMS *d'un point de vue administrateur* : vous connaissez les templates, les stylesheets, les pages et les blocs de contenu ; vous savez que l'on peut gérer les permissions pour de nombreuses fonctionnalités et vous savez inclure des {tag} dans les templates pour afficher des contenus.

2.5.Récupérer et utiliser la version SVN de CMSMS

Pour information consulter la documentation de Subversion (appelé aussi "SVN") ici.

Utilisation de TortoiseSVN avec Windows

1. Télécharger et installer TortoiseSVN - configurer TortoiseSVN en Français
2. Créer un nouveau dossier, avec l'explorateur Windows, pour télécharger la version SVN version de CMSMS.
3. Se placer sur le dossier créé
4. Avec le bouton gauche de la souris choisir : "TortoiseSVN -> Extraire..."
5. La boîte de dialogue s'affiche, dans "URL du référentiel" copier : http://svn.cmsmadesimple.org/svn/cmsmadesimple/trunk
6. Cliquer sur le bouton "OK".
7. Cela va télécharger les fichiers ...
8. Vous êtes maintenant en possession de la dernière version SVN de CMSMS. Merci de reporter les bugs.

Pour les postes sous Linux

Vous pouvez, bien sûr, utiliser la ligne de commande ou un client graphique comme par exemple rapidsvn.

Dernière version "nightly"

Vous pouvez télécharger la dernière version "nightly" ici .

3.Foire aux questions

Cette FAQ contient les réponses aux questions posées régulièrement par les développeurs qui débutent CMS Made Simple. Les développeurs doivent d'abord rechercher les réponses ici avant de déposer leurs questions dans les forums.

3.1.Inclure du PHP dans une page ou un gabarit

J'ai un script ou un bout de code PHP provenant d'un ancien site, et je veux l'utiliser dans ma nouvelle page CMS. Comment faire cela ?

R1: Utilisez les Tags Utilisateurs *(recommandé)*

Vous pouvez créer un Tag utilisateur *(Extensions >> Tags Utilisateurs)* et y inclure votre code PHP. Exemple : créez un Tag appelé my_phpinfo placer le code suivant à l'intérieur :

```
phpinfo();  //(les mots clés <?php et ?> ne sont pas nécessaires).
```

Maintenant, ajoutez le code suivant dans votre page pour activer ce script :

```
{my_phpinfo}
```

Vous pouvez utiliser n'importe quel code PHP dans un Tag utilisateur, y compris la commande "include" pour inclure du code externe. Vous avez aussi accès aux API de CMS Made Simple et aux variables globales. C'est la manière recommandée d'intégrer du code PHP, et vous pouvez contrôler les personnes qui ont accès à cette fonctionnalité grâce à la gestion des droits d'accès.

R2: Utilisez {php} and {/php} Tags

Cette option n'est pas recommandée car elle offre la possibilité à toutes les personnes habilitées à éditer une page d'y inclure du code PHP et d'introduire par la même occasion des failles de sécurité.

La seconde manière d'inclure du code PHP se fait en utilisant les tags **{php}** and **{/php}** encapsulant votre code PHP dans votre page ou votre template *(les mots clés <?php et ?> ne sont pas nécessaires)*.

Attention : cette fonctionnalité est désactivée par défaut sur les nouvelles installations.

Pour activer cette fonctionnalité, vous devez modifier une ligne dans votre fichier de configuration : Mettre la paramètre "use_smarty_php_tags" à true.
Exemple :

```
//Autorise le tag {php}.
// Option à éviter si vous ne faites pas totalement
//confiance à vos utilisateurs.
$config['use_smarty_php_tags'] = true;
```

Vous pouvez ensuite placer votre code PHP à l'intérieur de vos pages ou de vos gabarits comme vous le souhaitez. Exemple :

```
{php}phpinfo();{/php}
```

3.2.Mon "modifieur" Smarty ne fonctionne pas

J'ai utilisé le Tag suivant :

```
{content | lower}
```

dans mon gabarit, et cela ne fonctionne pas. Où est l'erreur ?

R: Parce que smarty est très sensible aux espaces

Smarty utilise les espaces comme séparateur pour indiquer les attributs d'un Tag ; il utilise le caractère "pipe" | pour indiquer le début d'un "modifieur". Vous devez retirer les espaces :

{content|lower}

3.3.Supprimer des images, modules en mode FTP

Certains utilisateurs, en particulier ceux qui utilisent l'environnement *Windows* peuvent avoir une surprise quand ils se connectent en mode FTP : ils n'arrivent pas à supprimer certains fichier ; particulièrement ceux qui sont situés dans le répertoire uploads.

R: Un utilisateur ne peux supprimer les fichiers d'autres utilisateurs

MAIS CE SONT MES FICHIERS ! Non, désolé, ils ne vous appartiennent pas !

La plupart des plateformes d'hébergement utilisent essentiellement linux. Et généralement, le process du serveur web est exécuté depuis un compte utilisateur spécial (nobody, apache, httpd ou autre). Par défaut, les fichiers générés par ce process ne peuvent pas être détruits par d'autre utilisateurs. De la même façon, le process httpd ne peut écrire dans les répertoires d'autres utilisateurs. C'est une bonne mesure de sécurité mais cela peut être parfois déroutant.

Quand un fichier est téléchargé dans le répertoire uploads, ou un module ou un thème est extrait depuis un fichier XML que vous avez téléchargé (ou récupéré depuis le Gestionnaire de modules), ce fichier appartient au process httpd et ses propriété d'accès sont définies aux valeurs par défaut. Et la plupart du temps, les autres utilisateurs peuvent les lires mais ne peuvent pas les modifier ou les supprimer.

Alors, comment on règle ce problème ?

Et bien, la réponse est en deux parties :

1) Vous pouvez résoudre le problèmes pour les prochains fichiers en ajustant le masque de création de fichier (file creation mask) dans les préférences du site. Cela ajuste le "httpd processes umask" (masque utilisateur), et cela affectera les permissions des nouveaux répertoires et fichiers. En résumé, cela fonctionne de la manière suivante :

```
File Permissions = 0777 - umask
```

Exemple : Si votre masque de création de fichier est égal à 0022, les permissions sur un nouveau répertoire seront 0777 - 0022 = 0755 Les fichiers auront le même niveau de permission, sauf pour le niveau exécution et obtiendront le niveau 644.

2) Pour les fichiers que vous avez déjà téléchargé et que vous ne pouvez pas supprimer, parcequ'ils appartiennent au process httpd, il faut exécuter un script spécial.

3.4.Accéder à un module depuis un plugin

Je développe un plugin (Tag utilisateur) et j'aimerais pouvoir l'interfacer avec le module XYZ. Est-ce possible ?

R: Oui, via $gCms->modules['TheModule']

La liste de tous les modules installés (au minimum, ceux qui ont des fonctionnalités de type frontend) est disponible dans l'objet global $gCms. Vous pouvez accéder à cet objet de la manière suivante :

```
global $gCms;
  if( !isset( $gCms->modules['TheModuleIWant'] ) || !isset( $gCms-
>modules['TheModuleIWant']['object'] ) )
    {
       return;
    }
  $themoduleiwant = $gCms->modules['TheModuleIWant']['object'];
  $themoduleiwant->DoThis($somedata);
  $themoduleiwant->DoThat($somemoredata);
```

Ce n'est pas la meilleure façon de faire, si vous n'êtes pas sûr que le module soit installé ou non, mais c'est déjà très bien pour démarrer. Vous pouvez toujours utiliser var_dump et print_r sur les différentes portions de $gCms pour voir ce qui est disponible.

3.5.Modifier une chaine de caractères

Occasionnellement, vous pouvez vouloir modifier un message affiché par un module. Et ce message est présent dans le fichier <install dir>/modules/<ModuleName>/lang/xx_YY.php qui est facile à modifier mais vous voulez savoir ce qui se passera lors de la prochaine mise à jour. Est-ce que vos modifications seront conservées ?

La gestion des messages de certains modules n'est pas toujours très propre et certains messages sont codés en dur ; Ce n'est pas très bon de devoir modifier le code source et dans ce cas, prenez contact avec l'administrateur du module concerné et invitez le à corriger le problème.

R: Ajouter votre propre fichier de langue

IL est possible de surcharger n'importe quel message en créant votre propre fichier de langue qui ne contiendrait que les messages que vous souhaitez modifier.

Le nouveau fichier de langue devra être installé à l'endroit suivant :

```
<install dir>/module_custom/<ModuleName>/lang/xx_YY.php
```

Reprenez les chaînes de caractères que vous souhaitez modifier et remplacez les libellés par leurs nouvelles valeurs :

```
$lang['logout'] = 'Se déconnecter';
$lang['prompt_changesettings'] = 'Mes préférences';
```

3.6.Comment je peux débuguer mon code ?

R: Vous pouvez utiliser ces outils :

Pour IE:

- fiddler (http://www.fiddlertool.com) est un proxy local qui intercepte et modifie les trame envoyées et reçues par le serveur ; parfait pour tester certains scripts (javascript) ou codes CSS avant de les intégrer dans votre code.
- AIS Web Accessibility Toolbar permet (parmi d'autres superbes outils)de voir exactement quels attributs CSS sont appliqués sur l'élément de page sélectionné.

  (http://www.visionaustralia.org.au/info.aspx?page=614)

Pour Firefox:

- Firebug: consulter et débugguer les requêtes AJAX et les "page elements", éditer le CSS à la volée, etc.
- Web Developer: éditer le CSS à la volée, debuguer HTML, ect.

4.Tags utilisateurs

4.1.Version simplifiée

Les tags personnalisés (appelés tags utilisateurs) sont un moyen très simple d'insérer du code PHP dans votre site. Pour les utiliser, Il suffit de créer un nouveau tag et d'y coller votre code PHP (en retirant "<?php" et "?>"). Vous pouvez alors utiliser les tags smarty en utilisant la syntaxe suivante : {tag_name}

L'installation par défaut permet l'utilisation d'un tag prédéfini et que vous pouvez exploiter en utilisant {user_agent}

4.1.1.Paramètres pour les tags

Vous pouvez passer des paramètres aux tags par exemple

```
"{image_link src='hedgehog'}"
```

et le code du tag correspondant :

```
echo '<a href="' . $params['src'] . '.jpg" target="_blank">';
echo '<img src="t_' . $params['src'] . '.jpg"></a>';
```

4.2.Version détaillée

4.2.1.Comment créer un tag personnalisé

Comme beaucoup de choses dans un CMS, ajouter un nouveau plugin est simple. Pour ajouter votre propre plugin, suivez ces instructions :

1. L'éditeur de plugin est accessible dans la partie administration et vous devez vous authentifier en tant qu'admin, ou bien en tant qu'un utilisateur disposant des permissions adéquates.
2. Dans le panneau d'administration, cliquez sur 'Extensions -> tags utilisateurs' en haut du menu déroulant.
3. En bas de la page, cliquez sur 'Ajouter un tag utilisateur'.
4. Dans le champ 'Nom' saisissez le nom de votre tag (ce champ ne peut contenir que des lettres, des chiffres ou des underscores). C'est ce que vous devrez saisir à l'intérieur des parenthèses pour ajouter un tag dans une page et donc choisissez un nom représentatif mais pas trop long.
5. Dans le champ 'Code' saisissez le code PHP qui correspondra au tag quand la page sera appelée. (Voir partie suivante pour plus d'infos)

6. Cliquez sur le bouton 'Envoyer'.

4.2.2.Votre premier tag personnalisé

Au lieu de se focaliser sur chaque détail d'un plugin ou de PHP, un simple exemple devrait vous aider à démarrer. Comme toutes les introductions à la programmation, nous allons suivre pas à pas les étapes pour essayer d'écrire en premier lieu un script permettant d'afficher "hello world".

Suivez les étapes ci-dessus pour créer un nouveau plugin dont le nom est "helloworld" (sans guillemets) et dans le champs 'Code' saisissez ou copier/coller le code ...

```
echo 'Hello World!';
```

Cliquez sur envoyer. Pour tester le module, créez une nouvelle page de contenu, saisissez "{helloworld}" (sans guillemets) quelquepart dans la partie contenu et cliquez sur 'apperçu'. Au lieu de voir afficher {helloworld} vous devriez voir afficher "Hello World!".

La commande echo écrit simplement ce qu'il y à l'intérieur des guillemets et donc nous pouvons aussi utiliser cela pour ajouter du code HTML, ou même du DHTML ou du Javascript.

Essayez maintenant ce code (passez juste le Plugin en édition)

```
echo '<h4>Hello World!</h4>';
```

Testez le de la même façon et vous deviez voir apparaître quelque chose qui ressemble à ceci :

Hello World!

"Mais on peut faire cela directement dans une zone de texte riche !" me direz-vous. Bien sûr, mais les Tags sont vraiment très utiles quand vous voulez ajouter des paramètres. Par exemple, si vous essayez de dire bonjour à quelqu'un qui s'appelle Bob, essayez le code suivant :

```
echo '<h4>Hello ' . $params['name'] . '!</h4>';
```

Ceci permet d'ajouter le paramètre "name" à ce plugin ; le contenu du paramètre sera placé ici quand le plugin sera appelé grâce au tag correspondant. Testez le de la même façon que précédemment en remplaçant "{helloworld}" par "{helloworld name='Bob'}". Vous devriez voir apparaître quelque chose qui ressemble à ceci :

Hello Bob!

C'est juste pour présenter les bases de l'écriture d'un Plugin mais vous pouvez faire des choses vraiment utiles avec la commande echo, les paramètres et un peu d'imagination alors, pensez à ce que vous pourriez faire si vous appreniez le langage PHP ...

4.3.Passer le contenu de la page en paramètre

Vous pouvez accéder au contenu de la page au sein d'un tag personnalisé en l'utilisant comme paramètre: Dans votre template:

```
{content assign=pagecontent}
{table_of_contents thepagecontent="$pagecontent"}
```

Dans votre Tag personnalisé nommé "table_of_contents":

```
echo $params['thepagecontent']; // Affiche la page de contenu.
```

4.4.Récupérer l'url à partir du content_id

Ceci suppose que votre identifiant de contenu (content_id) est dans la variable $page_content_id :

```
global $gCms;
$hm =& $gCms->GetHierarchyManager();
$curnode =& $hm->getNodeById($page_content_id);
$curcontent =& $curnode->GetContent();
echo 'Page URL: ' . $curcontent->GetURL();
```

4.5.Exécuter Smarty à partir d'un tag utilisateur

En faisant comme cela :

```
global $gCms;
$smarty = &$gCms->GetSmarty();
$smarty_data = "{menu}";
$smarty->_compile_source('temporary template', $smarty_data,
$_compiled );
@ob_start();
$smarty->_eval('?>' . $_compiled);
$_contents = @ob_get_contents();
@ob_end_clean();
echo $_contents;
```

Le code suivant devrait vous permettre de gérer plusieurs domaines avec une seule installation de CMSMS :

```
global $gCms;
$smarty = &$gCms->GetSmarty();
$url = $_SERVER['REQUEST_URI'];
if(eregi('domain1',$url))
{
    $smarty_data = "{menu template='cssmenu.tpl'}";
}
else if(eregi('domain2',$url))
{
    $smarty_data = "{menu template='cssmenu2.tpl'}";
}
$smarty->_compile_source('temporary template', $smarty_data,
$_compiled );
@ob_start();
$smarty->_eval('?>' . $_compiled);
$_contents = @ob_get_contents();
@ob_end_clean();
echo $_contents;
```

Cette partie consiste à créer des scripts et à les appeler à tous moments.

4.6.Créer un tag utilisateur

En bas de la page Tags utilisateurs, cliquer sur Ajouter un Tag utilisateur. Nommer le par exemple "beta" puis saisissez le code Par exemple :

```
//Vérifie si on a choisit une valeur ici NomImmeuble
if(isset($_POST['NomImmeuble']) and $_POST['NomImmeuble']!="")
{
//Création de la table avec bordure, centré dans la fenêtre
echo "<table border=1 width='10'>";

$valeur = $_POST['NomImmeuble'];
            $reqt = "SELECT * FROM agence WHERE NomImmeuble =
'$valeur'";
            $rest = mysql_query($reqt) or die ("Exécution de la
requête impossible");

            while ($ligne = mysql_fetch_array($rest))
            {
              ....
```

Pour terminer, il suffit d'appuyer sur « ENVOYER ». Maintenant que le tag utilisateur est terminé, il suffit d'insérer {beta} dans une page pour appeler le code correspondant.

4.7.Supprimer un tag utilisateur

Dans la page des tags utilisateurs, il suffit de cliquer sur la corbeille du tag en question, pour le supprimer.

5.Tutoriel pour la création d'un module

Ce tutoriel vous montre comment créer un nouveau module pour CMS Made Simple.

5.1.Ce que nous construisons

Nous allons construire ensemble un module destiné à la gestion d'un catalogue de produits. L'interface d'administration devrait permettre les opérations suivantes :

- ajouter de nouveaux produits et saisir ou modifier les caractéristiques ;
- afficher l'ensemble des produits présents dans la base de données ;
- sélectionner différentes options d'affichage (nombre de colonnes par page par exemple) ;
- catégoriser les produits selon leur type ou leur fournisseur ;
- gérer la pagination (paramétrage du nombre de produits par page) ;
- associer une photo à chaque produit ;
- permettre de retrouver les produits par leurs caractéristiques (moteur de recherche) ;

Mais pour que ce tutoriel reste simple, nous allons limiter les informations sur le produit à un nom, une description et un prix et tous les produits seront affichés sur une simple liste sur une seule page.

Nous allons appeler ce module "Catlist" (pour éviter la confusion avec le module existant Cataloger).

5.2.La structure d'un module CMSMS

Tout d'abord, tous les fichiers relatifs à votre module doivent se trouver dans un répertoire qui porte le même nom que votre module et qui est situé sous le répertoire Modules soit dans notre cas : "modules/Catlist" Si vous regarder un module existant (par exemple le module Skeleton), vous verrez de nombreux fichiers et sous-répertoires qui peuvent être regroupés de la manière suivante :

- les fichiers action.___.php : ces fichiers contiennent du code qui gère différentes "actions" gérée depuis CMSMS. C*es fichiers ne sont pas obligatoires, le code concerné peut être placé dans le fichier "ModuleName.module.php".*

*Dans ce tutoriel, **nous n'utiliserons pas** les fichiers "action.___.php", mais pensez à les utiliser pour obtenir un code plus facilement maintenable.*

- le répertoire "images" : Toutes les images qui sont affichées par ce module doivent être placées dans ce répertoire, qu'elles soient utilisées pour l'interface d'administration ou destinées à l'utilisateur final.

- le fichier "index.html" : c'est un fichier vierge, placé dans le dossier au cas ou l'adresse de ce répertoire soit saisie directement dans le navigateur, afin d'éviter d'afficher la liste des fichiers qu'il contient.

- le répertoire "lang" : il est destiné au support du multi-langues ; vous devez mettre vos fichiers de ressources dans ce répertoire (un fichier par langue).

- les fichiers method.___.php : ils contiennent différentes méthodes appelées par CMSMS -- normalement celle utilisée pour l'installation, la desinstallation et la mise à jour du module. C*omme pour les fichiers "action.___.php", leur utilisation est facultative.*

- le fichier ModuleName.module.php file: c'est le cœur du module. Tout le code qui permet à l'administrateur d'ajouter ou de modifier du contenu ainsi que celui qui permet l'affichage dans la partie publique se trouve dans ce fichier.

- le répertoire "templates" : tous les gabarits qui sont utilisés pour afficher le contenu de votre module se trouvent dans ce répertoire ; ces gabarits peuvent être aussi bien utilisés pour afficher le contenu depuis l'interface d'administration (Back end) que depuis la partie publique du site (Front end).

5.3.Comment CMSMS affiche le contenu

Le concept le plus difficile à intégrer au début est de comprendre comment CMSMS affiche les contenus dans la partie publique. Il semble qu'il y ait deux façons de réaliser cette opération :

- **Inclure le tag {module} dans une page :**

L'éditeur crée une nouvelle page générique de type "content" et inclus un tag (comme "{cms_module module='Catlist'}"), ce qui a pour effet d'appeler la fonction DoAction de notre module (fonction présente dans le fichier Catlist.module.php), ce qui génère une requête qui va récupérer les données dans la base de données puis les affiche au travers d'un des template "Smarty" qui sera inclu dans la page.

- **Création d'un nouveau type de contenu:**

Le nouveau type de contenu sera ajouté à la liste déroulante « Type de contenu » lorsque que vous ajouter un nouveau contenu. Lorsque ce type de contenu est sélectionné, un formulaire apparaît avec les différents paramètres définis dans votre module.

Pour ce tutoriel, nous allons utiliser la première méthode.

5.4.Création du répertoire et de l'objet

Créer un nouveau répertoire appelé "Catlist". Faites attention à l'orthographe et aux majuscules. Maintenant, lancez votre éditeur de texte et de créer un nouveau fichier nommé "Catlist.module.php" . Nous allons commencer par les bases.

```
<?php
/* Your initial Class declaration. This file's name must
   be "[class's name].module.php", or, in this case,
   Catlist.module.php
*/
class Catlist extends CMSModule
{

    /*---------------------------------------------------------
       GetName()
       must return the exact class name of the module.
       If these do not match, bad things happen.

       This is the name that's shown in the main Modules
       page in the Admin.
      ---------------------------------------------------------*/
    function GetName()
    {
        return 'Catlist';
    }

    /*---------------------------------------------------------
       GetFriendlyName()
       This can return any string.
       This is the name that's shown in the Admin Menus and section pages
          (if the module has an admin component).
      ---------------------------------------------------------*/
    function GetFriendlyName()
    {
        return 'Simple Catalog Product List';
    }


    /*---------------------------------------------------------
       GetVersion()
       This can return any string, preferably a number or
       something that makes sense for designating a version.
       The CMS will use this to identify whether or not
       the installed version of the module is current, and
       the module will use it to figure out how to upgrade
       itself if requested.
      ---------------------------------------------------------*/
    function GetVersion()
    {
        return '0.1';
    }


    /*---------------------------------------------------------
        IsPluginModule()
        This function returns true or false, depending upon
        whether users can include the module in a page or
        template using a smarty tag of the form
        {cms_module module='Skeleton' param1=val param2=val...}
        If your module does not get included in pages or
        templates, return "false" here.

        (Required if you want to use the method DoAction later.)
        ---------------------------------------------------------*/
    function IsPluginModule()
    {
        return true;
    }
}
?>
```

Bon, ce n'est pas beaucoup, mais c'est un début. Les commentaires explique ce que fait chaque fonction.

5.5.Gestion des langues

Pour vos gérer vos libellés dans plusieurs langues, au lieu de mettre les chaînes de texte en dur votre code, vous pouvez utiliser le « $this->Lang ()» pour récupérer le texte selon la langue.

Créez un nouveau répertoire à l'intérieur du répertoire de votre module "Catlist" appelé répertoire "lang". Revenez à votre éditeur de texte et de créer un nouveau fichier, dans le répertoire "lang", appelé "en_US.php". Nous voulons mettre dans ce fichier est le "Friendly Name".

Ajouter ce code pour le nouveau fichier:

```
<?php
    $lang['friendlyname'] = 'Simple Catalog Product List';
?>
```

Pour vos développement, utiliser toujours en priorité l'anglais puis ensuite et seulement après traduisez vos modules avec un fichier de langue en français dans le fichier « fr_FR.php », par exemple:

```
<?php
    $lang['friendlyname'] = 'Liste du catalogue produits';
?>
```

Il suffit maintenant d'implémenter la gestion des langues dans votre fichier Catlist.module.php comme ci-dessous:

```
<?php
    return $this->Lang('friendlyname');
?>
```

5.6.Scripts d'installation et de désintallation

Pour la plupart des modules, il faudra écrire et lire des enregistrements dans la base de données de CMSMS (pour sauvegarder et récupérer nos données), l'autorisation de nouveaux rôles (les administrateurs peuvent autoriser ou interdire les utilisateurs d'effectuer certaines opérations), et quelques préférences. Ces choses doivent être faites au moment de l'installation du module.

A l'installation c'est la fonction "Install ()" qui est appelée. Pour ce module, nous allons créer une seule table dans la base de données pour stocker des informations sur les produits, et une seule autorisation qui permet ou interdit aux utilisateurs d'ajouter / modifier / supprimer des produits dans le catalogue.

La table SQL sera très simple - nous aurons des champs pour ID de produit, nom du produit, la description du produit et le prix. CMSMS utilise la bibliothèque ADODB pour interagir avec la base de données (voir le manuel ADODB pour plus de détails).

Entrer le code suivant dans le fichier Catlist.module.php

```
function Install()
{
   //Get a reference to the database
   $db = $this->cms->db;

   // mysql-specific, but ignored by other database
   $taboptarray = array('mysql' => 'TYPE=MyISAM');

   //Make a new "dictionary" (ADODB-speak for a table)
   $dict = NewDataDictionary($db);

   //Add the fields as a comma-separated string.
   // See the ADODB manual for a list of available field types.
   //In our case, the id is an integer, the name is a varchar(100) field,
   // the description is a text field, and the price is a float.
   $flds = "id I KEY,
         name C(100),
         description X,
         price F";

      //Tell ADODB to create the table called "module_catlist_products",
      // using the our field descriptions from above.
      //Note the naming scheme that should be followed when adding tables to the database,
      // so as to make it easy to recognize who the table belongs to, and to avoid conflict with other
modules.
      $sqlarray = $dict->CreateTableSQL(cms_db_prefix().'module_catlist_products', $flds,
$taboptarray);
      $dict->ExecuteSQLArray($sqlarray);

      //Now create a "sequence", which is used internally by CMSMS and ADODB
      // to increment our id value each time a new record is inserted into the table.
      $db->CreateSequence(cms_db_prefix().'module_catlist_products_seq');
}
```

Maintenant, ajouter à la fin de la fonction, le code qui permettra de créer notre permission rôle:

```
function Install()
{

/* ... all the database code from above... */

//Create a permission
//The first argument is the name for the permission that will be used by the system.
//The second argument is a more detailed explanation of the permission.
$this->CreatePermission('Catlist Admin', 'Manage Catlist');
}
```

Ajoutez également cette fonction, qui sera affiché, à l'administrateur, à la réussite de l'installation

```
function InstallPostMessage()
{
   return $this->Lang('postinstall');
}
```

Et bien sûr nous avons besoin d'ajouter la texte dans la version de la langue

```
$lang['postinstall'] = 'Catlist successfully installed!';
```

Pour la désinstallation du module nous allons créer la fonction suivante:

```
function Uninstall()
{
  //Get a reference to the database
  $db = $this->cms->db;

  //Remove the database table
  $dict = NewDataDictionary( $db );
  $sqlarray = $dict->DropTableSQL( cms_db_prefix().'module_catlist_products' );
  $dict->ExecuteSQLArray($sqlarray);

  //Remove the sequence
  $db->DropSequence( cms_db_prefix().'module_catlist_products_seq' );

  //Remove the permission
  $this->RemovePermission('Catlist Admin');
}
```

De la même manière nous allons afficher un message indiquant que la désinstallation a réussi mais avant nous afficherons une alerte pour demander une validation car les tables de la base de données seront supprimées. Comme d'habitude on ajoute ces fonctions dans le fichier Catlist.module.php:

```
function UninstallPreMessage()
{
  return $this->Lang('uninstall_confirm');
}

function UninstallPostMessage()
{
  return $this->Lang('postuninstall');
}
```

Et bien sûr, nous ajoutons la traduction:

```
$lang['uninstall_confirm'] = 'All product data in the catalog will be deleted!'
                           . 'Are you sure you want to uninstall the Catlist module?';
$lang['postuninstall'] = 'Catlist successfully un-installed.';
```

5.7.Présentation des Produits sur le site

Comme indiqué précédemment, nous allons créer une fonction dans notre fichier Catlist.module.php appelé "DoAction". Cette fonction est appelée quand un tag {cms_module module = 'Catlist'} est rencontrée dans une page, et il répondra par la sortie html (dans ce cas, une liste de tous les produits dans notre base de données):

```
function DoAction($action, $id, $params, $returnid=-1)
 {
  if ($action == 'default')
  {
    $db =& $this->GetDb();
    $sql = 'SELECT * FROM ' . cms_db_prefix().'module_catlist_products;';
    $dbresult =& $db->Execute($sql);

    $list = "<br /><br />\n";
    $list .= "<ul>\n";
    while ($dbresult && $row = $dbresult->FetchRow())
    {
       $list .= "<li><b>" . $row['name'] . '</b><br />';
       $list .= $row['description'] . '<br />';
       $list .= money_format('%n', $row['price']) . '<br />';
       $list .= "</li>\n";
    }
    $list .= "</ul>\n";
    $list .= "<br /><br />\n";
  }
  // assign to Smarty
  global $gCms;
  $this->smarty->assign('list', $list);
  /**
   *Insert: {cms_module module='Catlist'}{$list}
   *in your page template
   *But there has to be a way for this to work without the {$list} tag...
   **/
  return;
 }
```

Ce code se connecte à la base de données et lit la table module_catlist_products pour afficher les enregistrements. Comme nous n'avons pas encore implémenter les fonctions d'administration vous devez ajouter des enregistrements directement dans la base de données à l'aide de phpmyadmin.

Il vous suffit alors d'ajouter le tag {cms_module module='Catlist'} dans une page, d'afficher cette page et vous verrez alors s'afficher la liste des produits.

5.8.Utiliser le moteur de template Smarty

Tout d'abord, nous devons créer un nouveau répertoire "templates". Dans ce nouveau répertoire, nous allons mettre tous les modèles dédiés qui seront nécessaires à notre module. Nous allons créer un nouveau fichier appelé "display.tpl" Dans ce nouveau fichier, placez le code suivant:

```
<br />
<br />
<ul>
  {section name=element loop=$list}
    <li><b>{$list[element].name}</b><br />{$list[element].description}<br />{$list[element].price}
€ <br /></li>
    {/section}
</ul>
<br />
<br />
```

La ligne * {$list[element].name}
 {$list[element].description}
 {$list[element].price} €
* affiche le nom, la description et le prix de l'élément courant.

Nous utilisons *{section name=element loop=$list}* pour appliquer le même mécanisme à tous les articles sous forme $list.

Maintenant, nous avons notre modèle. Nous avons besoin de donner tous les éléments du modèle, afin de les afficher. Par conséquent, nous modifions la méthode précédente DoAction Comme ceci:

```
function DoAction($action, $id, $params, $returnid=-1)
{
  if ($action == 'default')
  {
    $db =& $this->GetDb();
    $sql = 'SELECT * FROM ' . cms_db_prefix().'module_catlist_products;';
    $dbresult =& $db->Execute($sql);

    // Creating a new array for all products
    $list = array();
    //For each product of the database
    while ($dbresult && $row = $dbresult->FetchRow())
    {
        // Get the number of current items in the list
        $i = count($list);
        // Set the different params
        $list[$i]['name'] = $row['name'];
        $list[$i]['description'] = $row['description'];
        $list[$i]['price'] = $row['price'];
    }
    // assign to Smarty
    $this->smarty->assign('list', $list);
    // Display the populated template
    echo $this->ProcessTemplate('display.tpl');
  }
  return;
}
```

5.9.Creer la page d'administration

Pour la partie frontend du site l'action default etait appelé. Pour la partie backend du site l'action appelée est 'defaultadmin'.

Nous ajoutons donc cette fonction comme ci-dessous:

```
function DoAction($action, $id, $params, $returnid=-1)
{
  if ($action == 'default')
  {
    // see the code above
  }
  if ($action == 'defaultadmin')
  {
    // we'll put our admin panel here
  }
  return;
}
```

Le contenu de notre panneau d'administration sera les fonctions importantes:

- Ajout, modification et suppression de produits de la liste
- Modifier le modèle, avec lequel les produits seront affichés dans l'interface.
- Pour fournir un accès aisé à ces fonctionnalités, nous allons diviser la page en deux onglets: un pour la liste de produits, et l'autre pour le modèle.

Les onglets peuvent être créés dans le module avec ce code:

```
// choose the tab to display. If no tab is set, select 'shows' as the default
// tab to display, because - in my opinion - this is the mostly needed tab.
if (!empty($params['active_tab']))
  $tab = $params['active_tab'];
else
  $tab = 'shows';

// and finally, display all those tabs. First, setup the tabs, and than include
// the function.{tab}.php file, in which the tab's code is stored to keep this
// file a bit tidier.
echo $this->StartTabHeaders();
echo $this->SetTabHeader('shows', 'Shows', 'shows' == $tab ? true : false);
echo $this->SetTabHeader('template', 'Template', 'template' == $tab ? true : false);
echo $this->EndTabHeaders();

// Display each tab's content
echo $this->StartTabContent();

echo $this->StartTab('shows');
include 'function.shows.php';
echo $this->EndTab();

echo $this->StartTab('template');
include 'function.template.php';
echo $this->EndTab();
echo $this->EndTabContent();
```

À suivre ...

6.Trucs et astuces

6.1.Squelette pour bien commencer

Skeleton est un design pattern. Les Design Patterns, en français patrons de conception, modèles de conception ou encore motifs de conception sont un recueil de bonnes pratiques de conception pour un certain nombre de problèmes récurrents en programmation orientée objet. http://fr.wikipedia.org/wiki/Patron_de_conception

Il ne sert qu'a fournir un bon point de départ pour créer vos propres modules.

6.2.ModuleMaker pour créer vos modules

1. Téléchargez le module ModuleMaker
2. Installez-le comme n'importe quel autre module
3. Aller à la ModuleMaker dans le menu Extensions et suivez les instructions.
4. Pour Unix / Linux, le changement de propriété module.
5. Ajouter des fonctionnalités nécessaires à votre goût.

ModuleMaker est limitée à CMSMS Version 1.0 (14 novembre 2006)

6.3.Étendre la classe CMSModule

Un module CMS Made Simple étend la classe CMSModule comme ci-dessous:

```
class MyModule extends CMSModule {
  function MyModule() {
    parent::CMSModule(); // Call the parent constructor parent:: CMSModule ();
    ...your own constructor code...
  }
}
```

6.4.{debug} des modèles Smarty

Voir : http://www.smarty.net/manual/fr/language.function.debug.php

6.5.Accès page d'id ou alias

```
$gCms->variables['page_name'] (alias)
$gCms->variables['content_id'] (id)
```

6.6.Obtenir l'URL d'une page donnée (id)

```
$hm =& $gCms->GetHierarchyManager();
$curnode =& $hm->getNodeById($pageid);
$curcontent =& $curnode->GetContent();
echo $curcontent->GetURL();
```

6.7.Préparer la mise à jour du module

Si vous avez suivi le module skeleton, vous avez trouvé le fichier mymodule.upgrade.php. Ce fichier permet de faire le changement. Tout d'abord, vous devez avoir installé le module (appelé ici MyModule).

1) mise à jour de votre mymodule.module.php et de changer le numéro de version dans la fonction GetVersion ();

2) mettre à jour votre module. Régler le numéro de version comme en 1.

```
switch($current_version)
{       case "0.1.0":
                ... your coding here ...
                $current_version = "0.1.1";
        case "0.1.1":
            break;
}
```

3) mettre à jour le fichier mymodule.install.php avec les derniers paramètres de votre table SQL.

4) mettre à jour le fichier de désinstallation mymodule.uninstall.php

5) changer la version dans les fichiers de langues.

6) transférer vos modification par FTP

6.8.Sécurité

Mettre en haut de vos fichiers:

```
if (!isset($gCms)) exit;
```

On peut aussi au début de chaque fichier, tester les permissions:

```
if (!$this->CheckPermission('Use Album')) exit;
```

7.Autres ressources

PHP
http://www.phpfrance.com, communauté francophone
http://fr.php.net/manual/fr/, documentation en français
http://www.phpdebutant.org, initiation au langage
http://www.nexen.net, actualités et portail
http://www.evolt.org/article/PHP_coding_guidelines/18/60247/,

MySQL
http://dev.mysql.com/doc/refman/5.0/fr/index.html, documentation
http://mysql.ifrance.com/, forum francophone
http://www.nexen.net, actualités et portail

Template Smarty
http://www.smarty.net, communauté
http://www.smarty.net/manual/fr/, documentation en français

8.Conclusion

Merci d'avance à tous les contributeurs. N'hésitez pas à nous contacter pour vos remarques et vos corrections.

Rejoignez l'équipe francophone

On recherche des modérateurs, des rédacteurs, des traducteurs, des graphistes et des développeurs.

http://forum.cmsmadesimple.fr/viewtopic.php?id=1031

Cordialement

L'équipe francophone