Guide pour la création d'un module Xoops 2.x

Version	Date	Modifications
1	03/01/04	
1a	01/02/04	Remplacement des ex. avec $HTTP_POST_VARS & $HTTP_POST_VARS par $_POST & $_GET qui n'ont pas besoin d'être déclarées globales. Précisions sur $xoopsDB->queryF() dans § base de données § formulaire complété et ajout d'une annexe éléments de formulaire
2	08/04/04	Modification § structure fichier – Ajout $ bdd – Modification $ requêtes complété avec exemples et information sur queryF – Ajout de $xoopsModule->getVar('dirname') – Ajout § smarty avec utilisation {$smarty.const. _ DEFINE} – Ajout § commentaires - Diverses corrections de fautes de frappe La majorité des corrections apportées est issue de commentaires de la team

1 Avertissement

Ce guide n'avait pas à l'origine vocation à être publié : c'était une compilation de notes personnelles, et n'étant pas issu de traductions de documentations officielles, pouvait donc comporter diverses erreurs ou inexactitudes.

La première version de ce guide ayant été utilisé par la team chargée de la documentation officielle de xoops, a fait l'objet d'un certain nombre de commentaires de leur part.

Cette nouvelle version a donc fait l'objet de corrections importantes, issues pour la plupart de ces commentaires. Si vous détectez des erreurs, ou disposez d'informations intéressantes, n'hésitez pas à m'en faire part afin que je puisse les répercuter et en faire bénéficier la communauté.

C. Félix, alias theCat, le 08/04/2004

---

2 La préparation

Mettre en place les éléments de base du module, ce qui nous permettra entre autres, de l'installer, le désinstaller et le mettre à jour.

- Création des répertoires
  - Répertoire du module (en minuscules) portant le nom celui-ci.
  - sous répertoires du module :
    - admin facultatif ; si vous créez une adminitration du module
    - language obligatoire, avec le sous-répertoire english (voir § langages)
    - images facultatif, pour vos images éventuelles
    - blocks facultatif **sauf** si vous créez des blocs pour votre module
    - sql facultatif **sauf** si votre module utilise une base de données
    - templates facultatif **sauf** si votre module utilise des templates
    - include facultatif, pour vos fichiers functions éventuels
    - class facultatif, pour vos classes éventuelles
- Création du logo du module
  - Nommé mymodule_slogo.png
  - Au format png si possible (gif ou jpeg sinon)
  - Placé à la racine du module, ou dans le répertoire images s'il existe : paramétrer en conséquence la variable $modversion['image'] dans xoops_version.php.
- Création des fichiers de base :
  - xoops_version.php
    - Placé a la racine du module
    - Voir en annexe le fichier xoops_version.php
    - Avec pour commencer les informations du module, le reste pouvant être complété au fur et à mesure
- Création des fichiers langage :
  - modinfo.php
  - main.php
- Création du fichier mysql.sql
  - Placé dans le répertoire sql
  - Fichier dump des tables de votre module, structure seule en principe, mais **sans** le préfixe xoops_

---

3 L'écriture des scripts

On commence par le fichier index.php qui sera appelé par le lien dans le menu.
Dans un premier temps, il vaut mieux ne pas se préoccuper de l'affichage via un template ou des blocs, pour se concentrer sur le fonctionnement correct du script.

NB utiliser impérativement <?php et non <?

3.1 La structure d'un fichier

```
<?php
Ne pas omettre la licence
// ------------------------------------------------------------------------ //
// XOOPS - PHP Content Management System //
// <http://www.xoops.org/> //
//
// etc

// Appel obligatoire du mainfile (voir remarques)
```

```
include ('../../mainfile.php');

// si applicable, ici code pour déterminer le nom du template à utiliser
// suivi de l'assignation
$xoopsOption['template_main'] = "templatename";

// puis appel obligatoire de l'entête
include(XOOPS_ROOT_PATH.'/header.php');

// ici votre code principal
……

// Ne pas oublier le pied de page, avec cette ligne obligatoire.
include_once XOOPS_ROOT_PATH.'/footer.php';
?>
```

3.2 Remarques

- Respectez l'orde ci-dessus(mainfile, template, header) pour que votre module puisse utiliser correctement le système de cache de xoops.

- Une alternative à cette structure consiste à créer un fichier header.php à la racine de votre module qui contiendra

```
<?php
include("../../mainfile.php");
// ici vos autres include éventuels et systématiques par ex.
include XOOPS_ROOT_PATH."/modules/mymodule/include/functions.php";
?>
```

  Dans ce cas vous remplacerez en début de votre fichier :

```
include ('../../mainfile.php');
```

  par

```
include "header.php";
```

- Respectez les conventions d'écriture :
  - n'utilisez les majuscules que pour les variables Xoops et les instructions Sql
  - écrivez les variables en minuscules sans caractères spéciaux, hormis le 'underscore' pour séparer les mots

- Commentez largement votre code (en anglais ?), ça pourra servir aux autres, mais aussi à vous même.

```
// If notify checkbox is set, add subscription for approve
```

- Ne codez pas en dur le répertoire du module, mais utilisez :

```
$xoopsModule->getVar('dirname')
```

- Quelques instructions utiles :

  Pour créer un lien sur un texte :

```
$link = '<a href="'.XOOPS_URL.'/modules/ ' . $xoopsModule->getVar('dirname') . '/index.php.">texte lien</a>';
```

  Pour créer un lien sur une image :

```
$link = '<a href="'.XOOPS_URL.'/modules/ ' . $xoopsModule->getVar('dirname') . '/index.php.">';
$link .= '<img src="'.XOOPS_URL.'/images/image.gif" /> ';
$link.= '</a>';
```

  Pour inclure un fichier de fonctions:

```
include_once XOOPS_ROOT_PATH.'/modules/ ' . $xoopsModule->getVar('dirname') . '/include/functions.php';
```

---

4 Base de données :

* Réflechissez à la répartition de vos données dans les tables : créez dans la mesure du possible une table par type d'objet.

* Si votre application s'y prête, créez une table de catégories. Ex :
  - gestion de DVD une table pour les films, une table (catégorie) pour les genres
  - gestion de petites annonces : une table pour les annonces et une table (catégorie) pour les rubriques

  Cette table de catégorie aura au minimum 3 champs : (ces champs peuvent être nommés différemment)
  - cat_id identifiant unique en auto incrément
  - cat_pid identifiant de la catégorie parent, pour la gestion de sous-catégorie. A 0 défaut.
  - cat_title titre de la catégorie

Cela vous permettra ensuite d'utiliser dans votre module de nombreuses fonctionnalités de xoops.

* Vos tables devront être préfixées avec le nom de votre module en minuscules : mymodule_table1 p.ex.

* Le premier champ de chaque table sera un identifiant unique en auto incrément. (int(8) à adapter)
  Nommez le sous la forme item_id, cat_id, etc.. : item_id int(8) unsigned NOT NULL auto_increment

5 Requêtes :

Généralités

Utilisez impérativement les fonctions xoops de base de données ($xoopsDB->prefix(), $xoopsDB->query(), etc.)

* Pour bénéficier du mode debug sql.
* Pour une éventuelle future compatibilité de xoops avec une autre base de données.

De préférence, écrivez votre requête dans une chaine de caractère ($sql) :

```
$sql = 'SELECT * FROM '.$xoopsDB->prefix(mymodule_table1).' WHERE champ1 = '.$variable.'  AND
champ2 = '.$variable2';
```

Puis exécutez la requête. Cela peut être sous cette forme :

```
$result =  $xoopsDB->query($sql)
```

Ou mieux avec un test et redirection en cas d'erreur

```
if ( !$result = $xoopsDB->query($sql) ) {
        redirect_header('index.php',2,_MD_ERROROCCURED);
        exit();
}
```

Mise au point

Pendant la phase de mise au point (et uniquement), vous pouvez utiliser le test ci-dessous à la place du précédent :

```
if ( $result === false ) { die( 'Erreur ligne 125 details : '.$sql .'<br>'); }
```

L'astuce ci-dessus présente plusieurs avantages si votre requête ne s'exécute pas :

- elle vous indiquera à quel endroit du code le problème se présente
- votre requête s'affichera et vous permettra de voir s'il manque un élément (ex WHERE champ1= AND…)
- vous pourrez faire un copier /coller de cette requête et l'exécuter avec phpmyadmin pour voir le résultat.

query ou queryF ?

Il existe 2 méthodes :

$xoopsDB->query() et $xoopsDB->queryF()

Utilisez en priorité $xoopsDB->query() , mais si votre requête d'update ou d'insert ne fonctionne pas, alors vous pouvez utilisez $xoopsDB->queryF().

Ci-dessous, commentaire de Mithrandir

queryF permet les requêtes d'insertion et de mise à jour basées sur des variables GET ,et ne doit pas être prise à la légère. Employez TOUJOURS query à moins que vous ne deviez absolument employer la méthode GET - et seulement si vous êtes absolument certain que vous avez la maitrise complète de la variable fournie par l'intermédiaire de GET.

La méthode $xoopsDB->queryF($sql, $limit, $start) retourne :

- o Le résultat de la requête si exécution et au moins un résultat
- o True si exécution, mais aucun résultat
- o False si erreur d'exécution

Nota : les paramètres $limit, $start sont optionnels.

Exploitation de requêtes SELECT

Il vous reste ensuite à exploiter le résultat de la requête.
Quelques exemples :

fetchRow, avec list()

* lorsque la requête ne retourne qu'une ligne (requête SELECT COUNT(*) , requête avec clause WHERE sur l'id,....)

```
list($count) = $xoopsDB->fetchRow($result);
```

ou encore

```
list($id, $name, $phone) = $xoopsDB->fetchRow($result);
```

Les variables $count, ou $id, $name, $phone contiendront les résultats de votre requête.

* lorsque les résultats peuvent être exploités directement dans une boucle

```
while (list($id, $name) = $xoopsDB->fetchRow($result)) {
        echo 'Le nom est '.$name.' et le numero id '.$id;
}
```

fetchArray() :

* lorsque la requête ne retourne qu'une seule ligne

```
$myrow = $xoopsDB->fetchArray($result))
$variable1 = $myrow['nom_champ1'];
$variable2 = $myrow['nom_champ2'];
```

* lorsque la requête retourne un tableau

```
$i = 0;
while ( $myrow = $xoopsDB->fetchArray($result) ) {
        $var_array[$i]['elt1'] = $myrow['nom_champ1'];
        $var_array[$i]['elt2'] = $myrow['nom_champ2'];
        $i++;
}
```

* ou encore

```
while ($myrow = $xoopsDB->fetchArray($result)) {
    $var_array['element'][] = array('elt' => $myrow['nom_champ1'], 'elt2' => ($myrow['nom_champ2']));
}
```

Quelques autres fonctions disponibles :

* $xoopsDB->genId($sequence) ne sert à rien avec mysql (id en auto incrément)
* $xoopsDB->getInsertId() retourne l'id du dernier INSERT réalisé
* $xoopsDB->getRowsNum($result)
* $xoopsDB->getAffectedRows()

Attention : si vous les utilisez dans une fonction, n'oubliez pas la déclaration : global $xoopsDB

---

6 Les templates

Il faut bien faire la distinction entre :

- les scripts php qui traitent les données (saisie, insertion/extraction dans la bdd, calculs,…)
- les templates qui permettent d'afficher ces données (ordre, position, affichage contitionnel,…)
- les thèmes qui permettent de modifier la présentation (couleurs de police, de fond, …)

L'utilisation des templates permet de séparer le traitement des données de leur affichage ; on peut ainsi modifier la présentation des données, sans toucher au code, en modifiant simplement le template.

Bien qu'il soit souvent plus facile de créer une variable unique en php (par ex. nom du poster + date + heure), il est préférable de transmettre les variables **séparément** et de les mettre en forme dans le template : cela permettra une plus grande souplesse pour des modifications ultérieures, par vous-même ou vos futurs utilisateurs.

6.1 Coté php

Pour afficher les données d'un script via un template, 2 opérations sont nécessaires dans le code:

- indiquer au script le nom du template à utiliser .
- transmettre au template les données à afficher .

Structure du fichier :

```
include ('../../mainfile.php');

// si vous avez des fonctions :
include("include/functions.php");

$xoopsOption['template_main'] = 'mymodule_template_1.html'       // nom du template

include XOOPS_ROOT_PATH.'/header.php'

// si vous utilisez la navigation par page :
include(XOOPS_ROOT_PATH."/class/pagenav.php");

// votre code …
// exemple assignation de valeurs uniques
$xoopsTpl->assign("counter", $total);

// votre code  …
// exemple d'assignation de tableau
```

```
$array_msg = array();
while ($sqlfetch=$xoopsDB->fetchArray($result)) {
   $ array _msg['title'] = $myts->sanitizeForDisplay($sqlfetch["title"], 0, 0, 0);
   $ array _msg['msg'] = $myts->sanitizeForDisplay($sqlfetch["message"],0,1,1);
   $xoopsTpl->append('posts', $array_msg);
}

// sans oublier de finir par :
include(XOOPS_ROOT_PATH."/footer.php");
```

6.2 Remarque

Dans la boucle qui génère le tableau, on peut avoir des instructions conditionnelles genre

```
If ($variable1>variable2) {
        $array _msg['title'] = $montexte
}
```

Si la condition n'est pas remplie, l'élément `$ array _msg['title']` n'existera pas dans le tableau transmis au template et génèrera (en mode debug php) un :

Notice [PHP]: Undefined index: title in file .....

Bien que celà n'empêche pas votre module de fonctionner, il est préférable de déclarer dans ce cas l'élément du tableau avec une chaine nulle et donc d'utiliser une boucle if … else

```
If ($varaible1>$variable2) {
        $array _msg['title'] = $montexte;
} else {
        $array _msg['title'] = '';
}
```

ou sa version condensée

```
$array _msg['title'] = ($variable1>$variable2) ? $montexte : '';
```

6.3 Coté html

* Affichage de valeurs uniques

```
<td  style="text-align:center;"> <{$smarty.const._MD_MYMODULE_NBPOSTS}> : <{$counter}></td>
```

* Affichage des valeurs du tablau par une boucle

```
<{section name=i loop=$posts}>
 <tr>
 <{cycle values=odd,even assign=class}>          // facultatif: pour alterner la couleur de fond des lignes
        <td class="<{$class}>"><{$ posts [i].title}></td>
        <td class="<{$class}>"><{$ posts [i].msg}></td>
   </tr>
   <{/section}>
```

Idem, mais avec inclusion d'un second template

```
<{section name=i loop=$posts}>
        <{include file="db:mymodule_template2.html" post2=$posts[i]}>
        <br />
<{/section}>
```

Dans ce second template, les valeurs seront affichées par

```
<{$post2.title}>
<{$post2.msg}>
```

* Autre méthode d'affichage des valeurs d'un tableau (pour tableau multidimensionnel notamment)

```
<{foreach item=category from=$categories}>
        .....
<{/foreach}>
```

6.4 Smarty

Xoops vous permet d'utiliser des variables smarty dans vos templates. Quelques exemples de variables disponibles :

{$xoops_sitename}	le nom du site
{$xoops_uname}	le pseudo de l'utilisateur
{$xoops_isadmin}	true si administrateur

Si par exemple, si vous voulez afficher le nom de votre site, insérez dans votre code html :

```
<{$xoops_sitename}>
```

Si vous ne voulez afficher certains éléments qu' à l'administrateur :

```
<{if $xoops_isadmin == true}>
    <td >
    <!--partie visible à l'admistrateur seulement -->
    </td>
<{else}>
    <td > </td>
<{/if}>
```

IMPORTANT :

Les 'define' de langage peuvent (et doivent) désormais être utilisées dans les templates avec la variable smarty

```
{$smarty.const. _MD_MYMODULE_DEFINE}
```

(_MD_MYMODULE_DEFINE étant bien entendu une définition de langage de : language/…./main.php)

N'utilisez plus la précédente méthode consistant à assigner la valeur coté Php

```
$xoopsTpl->assign('lang_total_post', _MD_MYMODULE_NBPOSTS);
```

Et à l'afficher dans le template par

```
<{$lang_total_post}>
```

Exemple d'utilisation (affichage d'un message 'Bienvenue sur Monsite')

```
<{$smarty.const._MD_MYMODULE_WELCOME}>  <{$xoops_sitename}>
```

---

7 Blocs

- Créer un fichier pour gérer l'affichage (et éventuellement l'édition des options) du (ou des) block(s) : mymodule/block/mymodule_block1.php
- Créer un fichier de définition des variables de langage des blocs : mymodule/language/english/blocks.php
- Créer le template du block : mymodule/templates/blocks/ mymodule _block1.html
- Insérer les paramètres du block dans le fichier xoops_version.php
- Mettre à jour le module pour la prise en compte des modifications

7.1 Fichier principal : mymodule_block1.php

Ce fichier contient :

- Une fonction pour la gestion de l'affichage du block : b_ mymodule _show p.ex.
- Eventuellement une fonction de gestion des options du block : b_ mymodule _edit p.ex.
  Exemple ci-dessous avec choix du nomtre d'items affichés dans le block.
  Ces options seront accessibles dans la fonction d'édition du bloc du menu admin system

```
// fonction pour l'affichage
function b_ mymodule _show ($options) {
    global $xoopsDB;
    $block = array();
    $myts =& MyTextSanitizer::getInstance();

// requête sql
$sql = "SELECT id,  title, post_time, from ".$xoopsDB->prefix("mymodule")." WHERE condition=0 ORDER BY
post_time DESC LIMIT 0, $options[0]";

$result=$xoopsDB->query($sql);

// construction du tableau pour le passage des données au template
    while($myrow=$xoopsDB->fetchArray($result)){
        $message = array();
        $message['id'] = $myrow[id'];
        $title = $myts->makeTboxData4Show($myrow["title"]);
        $message['title'] = $title;
        $message['date'] = formatTimestamp($myrow['post_time'],"s");
        $block[mymodule][] = $message;
    }
    return $block;
}

// fonction édition option
function b_ mymodule _edit($options) {
    $form = ""._MB_MYMODULE_DISP." ";
```

```
            $form .= "<input type=\"text\" name=\"options[]\" value=\"".$options[0]."\"
            /> "._MB_MYMODULE_NBITEM."";
return $form;
  }
```

7.2 Xoops_version :

Insérer les paramètres du bloc

```
  // Blocks
  $modversion['blocks'][1]['file'] = "mymodule_block1.php";
  $modversion['blocks'][1]['name'] = _MI_MYMODULE_BNAME1;
  $modversion['blocks'][1]['description'] = "Shows recently added ... ";
  $modversion['blocks'][1]['show_func'] = "b_ mymodule _show";          // fonction affichage du bloc
  $modversion['blocks'][1]['edit_func'] = "b_ mymodule _edit";          // fonction édition options du bloc
  $modversion['blocks'][1]['options'] = "5";                            // options (séparation par | si plusieurs)
  $modversion['blocks'][1]['template'] = mymodule _block1.html';
```

7.3 Template du bloc

```
<ul>
        <{foreach item=message from=$block.mymodule}>
         <li>
         <{$message.date}><br>
        <a href ="<{$xoops_url}>/modules/mymodule/index.php?id=<{$message.id}>”><{$message.title}></a>
        </li>
         <{/foreach}>
</ul>
```

7.4 Remarque

Il est souvent utile de prévoir une option permettant de gérer la longueur de texte affichée dans le bloc, afin de permettre à l’utilisateur de la modifier selon que le bloc sera mis au centre ou sur un coté.

```
// dans la fonction d’affichage
        ….
if ( !XOOPS_USE_MULTIBYTES ) {
        if (strlen($myrow['texte']) >= $options[1]) {
                $title = $myts->makeTboxData4Show(substr($myrow['texte'],0,($options[1] -1))).".....";
        }
}
…..
        // dans la fonction édition option
  $form = "" ….
  $form .= " <br>"._MB_MYMODULE_CHARS." <input type='text' name='options[]'
  value='".$options[1]."' /> "._MB_ MYMODULE _LENGTH."";
```

7.5 Fichiers langage

Ne pas oublier d’insérer les variables de langage

- Des blocks (_MB_ MYMODULE_DISP, …) dans mymodule/language/english/blocks.php
- De xoops_version (_MI_MYMODULE_BNAME1,…) dans mymodule/language/ english /modinfo.php

Et toujours penser à faire la mise à jour du module …

8 Formulaires

Xoops dispose de tous les éléments pour créer des formulaires:
Il est préférable de créer 2 fichiers :

- un fichier pour gérer ce formulaire : affichage, prévisualisation, récupération des données : p.ex. myform.php placé à la racine du module
- un fichier 'formulaire' qui contiendra la liste des champs à compléter : myform.inc.php placé dans le répertoire include du module.

8.1 Fichier principal : myform.php

```
<?php
// on inclut la licence ici

include("header.php");

// par défaut on affichera le formulaire
$op = 'form';

// pour récupérer toutes les variables du formulaire avec leurs valeurs (évite les $_POST)
foreach ( $_POST as $k => $v ) {
        ${$k} = $v;
}

// le formulaire a été posté : soit prévisualisation, soit enregistrement
if ( isset($preview)) {
        $op = 'preview';
} elseif ( isset($post) ) {
        $op = 'post';
}

// on traite les différents cas
switch ($op) {

case "preview":
    $myts =& MyTextSanitizer::getInstance();          // MyTextSanitizer object
    include  XOOPS_ROOT_PATH.'/header.php';

    $p_title = $myts->makeTboxData4Preview($title);
    $title   = $myts->makeTboxData4PreviewInForm($title);

    $p_comment = $myts->makeTareaData4Preview($comment, 0, 1, 1);
    $comment =  $myts->makeTareaData4PreviewInForm($comment);

    themecenterposts($p_title, $p_comment);          // preview box titre et message
    include "include/myform.inc.php";                // puis inclusion du formulaire
    include XOOPS_ROOT_PATH."/footer.php";
    break;

case "post":
    $myts =& MyTextSanitizer::getInstance();
    // récupération et mise en forme des données
    $title    =     $myts->oopsAddSlashes($title);
    $message =  $myts->oopsAddSlashes($comment);
    $email =      $myts->oopsAddSlashes($email]);
    $datetime =   time();
    $poster_ip =  $GLOBALS['REMOTE_ADDR'];

    // requête d'insertion dans la base
    $sql =  "INSERT INTO ".$xoopsDB->prefix("mymodule")."
        (title, message, post_time, email, poster_ip)
        VALUES ('".$title."','".$comment."','".$datetime."','".$email."','".$poster_ip."')";

    // insertion dans la base, sinon message d'erreur
    if ( !$result = $ xoopsDB ->query($sql) )  {
        $messagesent = _MI_ERRORINSERT;
    }

    // redirection après post du formulaire (à adapter)
    // l'exemple ci-dessous redirige sur le fichier index du module après affichage 'message envoyé'
    redirect_header("index.php",2,$messagesent);
```

```
        break;

    case 'form':
        default:
        include XOOPS_ROOT_PATH."/header.php";

        // initialisation des variables titre, commentaire et email de l'utilisateur
        $title = "";
        $comment = "";
        $email = !empty($xoopsUser) ? $xoopsUser->getVar("email", "E") : "";

        include "include/myform.inc.php";                    // inclusion du formulaire

        include XOOPS_ROOT_PATH."/footer.php";
        break;
    }
    ?>
```

8.2 **Fichier formulaire** : myform.inc.php

```
// include du formloader
include XOOPS_ROOT_PATH."/class/xoopsformloader.php";

// création du formulaire (simple ou avec thème)
$my_form = new XoopsThemeForm(_MI_MYMODULE, "myform", "myform.php");

// création text box pour le titre
$my_form->addElement(new XoopsFormText(_MI_TITLE, "title", 50, 100, $title), true);

// création text box pour l'email
$my_form->addElement(new XoopsFormText(_MI_EMAIL, "email", 50, 100, $email), true);

// création text area (simple) pour le message
$my_form->addElement(new XoopsFormTextArea (_MI_MESSAGE, "comment", $comment),true);

// variante de text area message avec toutes les options de mise en forme (insertion lien, police, smiley,…)
$my_form->addElement(new XoopsFormDhtmlTextArea(_MI_MESSAGE, 'comment', $comment, 15, 60), true);

// création boutons preview et submit
$button_tray = new XoopsFormElementTray(" ,");
$button_tray->addElement(new XoopsFormButton('', 'preview', _MI_PREVIEW, 'submit'));
$button_tray->addElement(new XoopsFormButton('', 'post', _MI_SEND, 'submit'));
$my_form->addElement($button_tray);

// affichage du formulaire, sauf si affichage par template
$my_form->display();
?>
```

8.3 **Affichage du formulaire** :

2 possiblités :

* Affichage diret du formulaire par
  `$my_form->display();`
* Affichage par template ; il faudra modifier le fichier principal myform.php en rajoutant juste avant le include XOOPS_ROOT_PATH."/header.php";

```
$xoopsOption['template_main'] = 'my_template_form.html';
```

et juste avant le include XOOPS_ROOT_PATH."/footer.php";

```
$my_form->assign($xoopsTpl);
```

Dans le template on récupérera les éléments du formulaire par les variables :

```
<{$my_form.javascript}>
<{$ my_form.name}>,  <{$ my_form.action}>, <{$ my_form.method}>, <{$ my_form.title}>
<{$my_form.elements.caption}>, <{$my_form.elements.body}>
```

Plutôt que de récupérer les élément du formulaire séparément, on pourra le faire par une boucle

```
<{foreach item=element from=$ my_form.elements}>

<{/foreach}>
```

---

9 Fichiers de langues

A placer dans les répertoires language/french et language/english de votre module.

A moins de pratiquer correctement l'anglais et de mettre à jour en parallèle les différents fichiers de langage, la méthode peut être la suivante, sachant que l'anglais est obligatoire et sera pris par défaut :

- ne pas créer de répertoire language/french .
- ne créer que les fichiers du répertoire language/english.
- Une fois votre module terminé faites en une copie dans le répertoire language/french de votre module
- traduisez (ou faites traduire) les fichiers du répertoire language/english.
- Si vous ne savez pas traduire, laissez la définition française et commentez votre ligne :
  define("_MD_ERROROCCURED","Une Erreur est Survenue"); // translate

Seuls les fichier main.php et modinfo.php sont nécessaires au départ.

9.1 main.php

Fichier principal des variables de langage utilisées par vos différents fichiers.
Mieux vaut les commenter et les classer, sans oublier d'éliminer celles que vous n'utilisez pas, ou plus.
Il ne semble pas y avoir de règle clairement établie pour le préfixage : on peut préfixer avec _MD ou le nom du module (en entier ou abrégé).

```
// functions.php        (si vous avez un fichier include de fonctions)
define("_MD_ERROR","Erreur");
define("_MD_GO","Ok");

// index.php
define("_MD_TITLE","Mon module");
define("_MD_SUBJECT","Sujet : ");

// myfile.php
define("_MD_TITLE","Titre");
define("_MD_TOP","Top");

// messages d'erreur
define("_MD_ERROROCCURED","Une Erreur est Survenue");

// myform.inc.php      (si vous avez un fichier include de formulaire)
define("_MD_YOURNAME","Votre Nom :");
define("_MD_SUBJECTC","Sujet :");
```

9.2 modinfo.php

Fichier d'information du module utilisé par xoops pour la gestion de votre module.
Ne pas oublier de faire une mise à jour du module après chaque modification de ce fichier.
Préfixe _MI suivi du nom de votre module

```
// * Le nom du module
define("_MI_MYMODULE_NAME","Mon module");

// * Brève description du module
define("_MI_MYMODULE_DESC","Module ??? pour XOOPS");

// Noms des blocs de ce module (si besoin)
define("_MI_MYMODULE_BNAME1","Mon bloc 1");

// Noms des items de la popup admin du module (si besoin)
define("_MI_MYMODULE_ADMENU1","Ajouter qqchose");
define("_MI_MYMODULE_ADMENU2","Editer qqchose");

// Notification: description des évenements et texte pour les mail templates (si besoin)
define ('_MI_MYMODULE_NEWPOST_NOTIFY', 'Nouvel Envoi');
define ('_MI_MYMODULE_NEWPOST_NOTIFYCAP', 'Notifiez-moi des nouveaux envois .');
```

9.3 admin.php

Si vous avez une administration de votre module, variables de langage utilisées par celle-ci.
Préfixe _AM
Nota : en réalité ce fichier n'est nécessaire que pour les messages de redirection ; vos variables de langage peuvent être mises dans le main.php, mais procéder ainsi donne une meilleure lisibilité.

```
define("_AM_DBUPDATED","Base de Donn&eacute;es mise &agrave; jour avec succ&eacute;s !");
```

```
define("_AM_CONFIG","Configuration du module");
define("_AM_DELETE","Effacer");
```

9.4 blocks.php

Si vous avez des blocs de votre module, variables de langage utilisées par ceux-ci.
Préfixe _MB suivi du nom de votre module

```
define("_MB_ MYMODULE _TMRSI","Aujourd'hui l'article le plus lu est :");
define("_MB_ MYMODULE _HITS","Nombre de Lectures");
```

9.5 mail_template

Si votre module permet l'envoi de mails (notification p.ex.) créez un sous répertoire mail_template dans language/english et mettez y les templates nécessaires.

Ex : newpost_notify.tpl
Bonjour {X_UNAME}, Un nouveau message a été ajouté sur {X_SITENAME}. Vous pouvez voir ce message ici : {COMMENT_URL} ---------- Vous recevez ce message parce que vous avez choisi d'être notifié quand des nouveaux articles sont ajoutés à notre site. Si c'est une erreur ou si vous ne voulez plus recevoir de tels avis, mettez s'il vous plaît à jour vos souscription en visitant le lien ci-dessous : {X_UNSUBSCRIBE_URL} Merci de ne pas répondre à ce message. ---------- {X_SITENAME} ({X_SITEURL}) Le Webmestre {X_ADMINMAIL}

9.6 Récapitulatif

Fonctionnalités	Fichiers concernés	Fichier langage	Préfixe
Les fichiers de votre module (à la racine, répertoire include, etc.)	mymodule/index.php mymodule/nclude/functions.php	main.php	_MD
Install et màj module par xoops Préférences du module Popup admin du module	mymodule/admin/menu.php	modinfo.php	_MI
Fichiers d'administration Messages de redirection	mymodule/admin/index.php	admin.php	_AM
Blocs	mymodule/block/myblock.php	block.php	_MB

---

10 Administration

Vous pouvez avoir besoin de créer une administration de votre module.
L'accès à cette administration peut se faire de 2 façons :
- en cliquant sur le logo du module qui ouvrira le fichier index de l'administration
- en cliquant sur un lien dans la popup associée à votre module

Ce sont deux choses différentes, même si elles peuvent utiliser le même fichier admin/index.php

10.1 index.php

Attention : les variables de langage devront être définies dans le main.php

```
// header de l'admin
include '../../../include/cp_header.php';

// si pas le language du site, anglais par défaut
if ( file_exists("../language/".$xoopsConfig['language']."/main.php") ) {
        include "../language/".$xoopsConfig['language']."/main.php";
} else {
        include "../language/english/main.php";
}

// différents include
include '../include/functions.php';
include_once XOOPS_ROOT_PATH."/class/xoopslists.php";
include_once XOOPS_ROOT_PATH."/include/xoopscodes.php";
include_once XOOPS_ROOT_PATH.'/class/module.errorhandler.php';
$myts =& MyTextSanitizer::getInstance();
```

```
$eh = new ErrorHandler;

// fonction ayant obligatoirement le nom de votre module
// c'est elle qui sera affichée par défaut lors du click sur l'icône du module
function mymodule() {
   global $xoopsDB, $xoopsModule;
        xoops_cp_header();

// Affichage du titre, du style 'configuration générale de mon module'
        echo "<h4>"._MD_MM_CONFIG."</h4>";
                echo"<table width='100%' border='0' cellspacing='1' class='outer'>"
                ."<tr class=\"odd\"><td>";

// premier item : donne accès au paramétrage général (préférences) du module
        echo " - <a href = '".XOOPS_URL."/modules/system/admin.php?fct = preferences&amp;op =
        showmod&amp;mod = ".$xoopsModule->getVar('mid')."'>"._MD_MM_GENERALSET."</a>";
        echo "<br />";

// second item de l'administration
        echo "<br>";
        echo " - <a href=index.php?op=Messageshow>"._MD_MM_EDIT."</a>";
        echo "<br /><br />";
        echo"</td></tr></table>";
        xoops_cp_footer();
}

// seconde fonction de votre administration
function Messageshow() {
        global $xoopsDB;
        $myts =&amp; MyTextSanitizer::getInstance();
   xoops_cp_header();
        // ici votre code .....
         xoops_cp_footer();
}

// on termine par le selecteur de fonction

if(!isset($_POST['op'])) {
        $op = isset($_GET['op']) ? $_GET['op'] : 'main';
} else {
        $op = $_POST['op'];
}
switch ($op) {
        case &quot;Messageshow&quot;:
                Messageshow();
                break;
        case 'main':
                default:
                mymodule();
                break;
}
```

10.2 popup

Attention : les variables de langage devront être définies dans le modinfo.php
La popup a besoin:
1- Du paramétrage du fichier modinfo.php

```
// Names of admin menu items
define('_MI_ MYMODULE _ADMENU1', ' Gestion de qqchose ');
define('_MI_ MYMODULE _ADMENU2', ' Edition de qqchose ');
```

2- D'un fichier menu.php placé dans le répertoire admin de votre module

```
$adminmenu[1]['title'] = _MI_MYMODULE_ADMENU1;          // nom du premier item
$adminmenu[1]['link'] = "admin/index.php?op= Messageshow ";        // lien du premier item
$adminmenu[2]['title'] = _MI_MYMODULE_ADMENU2;
$adminmenu[2]['link'] = "admin/index.php?op=option";
```

Toujours penser à faire une mise à jour du module pour la prise en compte des modifications.

10.3 Préférences

Xoops vous permet de gérer les paramétres généraux de votre module par l'intermédiaire d'un formulaire. Les données seront stockées dans la table xoops_config.
Il vous suffit de déterminer pour chaque paramètre un nom de variable, le type de variable, le type de champ, le titre du champ, éventuellement une description complémentaire, et la valeur par défaut.

Ces paramètres seront définis dans le fichier xoops_version.php
Les variables de langage seront définies dans modinfo.php

```
// nom de la variable
$modversion['config'][1]['name'] = 'variable1';

// titre du champ affiché dans le formulaire, à gauche du champ de saisie
$modversion['config'][1]['title'] = '_MI_MYMODULE_TITRE1;

// description complémentaire, affichée en dessous du titre du champ
$modversion['config'][1]['description'] = '_MI_ MYMODULE _DESCR1;

// Type du champ (textbox, textarea, select, select_multi, yesno, group, group_multi)
$modversion['config'][1]['formtype'] = 'textbox';

// type de variable (int, text, float, array, …)
$modversion['config'][1]['valuetype'] = 'int';

// valeur par défaut
$modversion['config'][1]['default'] = 5;

// On continue pour l'option 2
$modversion['config'][2]['name'] = 'variable2';
$modversion['config'][2]['title'] = '_MI_MYMODULE_TITRE2;
// etc
```

Remarque
Certains modules utilisent une méthode différente :
Les variables de paramétrage ne sont pas stockées dans la bdd mais dans un fichier cache/config.php directement sous la forme (ex.) : $var_prefer1 = 10 ;
Cette méthode est obsolète.

10.4 Effacement d'un enregistrement

Xoops dispose d'une fonction xoops_confirm utilisable en particulier pour demander confirmation, avant la suppression d'un enregistrement p.ex.

```
function Messagedel($idmsg) {
        global $xoopsDB;
           $idmsg =  isset($_POST['idmsg']) ? intval($_POST['idmsg']) : intval($_GET['idmsg']);
        $ok =  isset($_POST['ok']) ? intval($_POST['ok']) : 0;
        // on a confirmé, donc on supprime
          if ( $ok == 1 ) {
                    $result=$xoopsDB->query("DELETE FROM ".$xoopsDB->prefix("mymodule")." WHERE
                    xtguestbook_id=$idmsg");
                    // on retourne p.ex. à la liste des messages après avoir affiché la confirmation d'effacement
                redirect_header("index.php?op=Messageshow",1,_AM_MYMODULE_MSGDEL);

        // la demande de confirmation : si oui on revient à Messagedel avec $ok==1, sinon retour à index de l'admin
        } else {
                xoops_cp_header();
                    xoops_confirm(array('op' => 'Messagedel', 'idmsg' => $idmsg, 'ok' => 1), 'index.php',
                    _AM_MYMODULE_DEL);
                xoops_cp_footer();
        }
}
```

---

11 Notification, commentaires et recherche

11.1 Recherche

C'est assez facile à réaliser, les explications seront donc succintes.
Créer un fichier search.inc.php dans votre répertoire include : il contiendra la fonction de recherche. Celle-ci comprend 2 parties :

- la création de la requête sql
- la création du tableau contenant les résultats de la requête, dont le lien sur chaque résultat trouvé.

Le plus simple est de copier le search.inc.php d'un module (news, forum,…) et de l'adapter.

Ne pas oublier :

- d'insérer les lignes nécessaires dans votre fichier xoops_version .php

```
// Search
$modversion['hasSearch'] = 1;
$modversion['search']['file'] = "include/search.inc.php";          // votre fichier search.inc.php
$modversion['search']['func'] = "news_search";                     // le nom de votre fonction search
```

- de mettre à jour votre module pour que la fonction de recherche soit prise en compte.

11.2 Commentaires

Le but est d'ajouter à un "article" la possibilité de lire et poster des commentaires.
NB: le terme "d'article" ne doit pas être pris dans le sens d'un article de journal, mais au sens large (item, élément). Il pourra s'agir de n'importe quel type d'élément géré par votre module, tel qu'une personne parmi une liste de contacts, d'un objet d'une collection, de la définition d'un nom dans un dictionnaire,...

Conditions
Chaque "article" doit être repéré dans la table par un identifiant unique et doit pouvoir être affiché individuellement par un template.
Votre table pourra également comporter un champ "comments , int(11), 0 par défaut, destiné à comptabliser le nombre de commentaires par "article".

Fichier xoops_version.php
Ajouter ces 3 lignes, en adaptant les parties en rouges à votre module.

```
// Comments
$modversion['hasComments'] = 1;
$modversion['comments']['itemName'] = 'item_id'; // nom de l'identifiant unique d'un "article" dans la base
$modversion['comments']['pageName'] = 'article.php'; // le fichier qui affichera "l'article" individuellement
```

(le fichier article.php devra pouvoir afficher un article unique en étant appelé par article.php?item_id=15 p.ex.)

Fichiers comment_delete, comment_edit, ...
Faites une copie des fichiers suivants du module news (p.ex), dans le répertoire de votre module (rien à modifier dans ces fichiers)

- comment_delete.php
- comment_edit.php
- comment_new.php
- comment_post.php
- comment_reply.php

Fichier article.php
Editez le fichier indiqué dans $modversion['comments']['pageName'], (article.php dans notre exemple) et ajoutez immédiatement avant le footer.php

```
include XOOPS_ROOT_PATH.'/include/comment_view.php';
```

Fichier template
Dans le fichier template qui affiche individuellement votre "article", ajoutez les lignes suivantes à la fin

```
<div style="text-align: center; padding: 3px; margin:3px;">
<{$commentsnav}>
<{$lang_notice}>
</div>
<div style="margin:3px; padding: 3px;">
<!-- start comments loop -->
<{if $comment_mode == "flat"}>
        <{include file="db:system_comments_flat.html"}>
<{elseif $comment_mode == "thread"}>
        <{include file="db:system_comments_thread.html"}>
<{elseif $comment_mode == "nest"}>
```

```
        <{include file="db:system_comments_nest.html"}>
    <{/if}>
```

Administration
Votre administration du module prévoit sans doute la possibilité d'effacer un "article" de la base.
Cette action devra également effacer les commentaires associés à cet "article".
Il vous faudra insérer (et adapter) la ligne suivante dans la partie de votre script qui efface un "article".

```
xoops_comment_delete( $xoopsModule -> getVar( 'mid' ), $item_id );
```

Comptage des commentaires
Si vous voulez compter le nombre de commentaires associés à un "article", insérez dans votre script à l'endroit désiré l'instruction suivante:

```
$count = xoops_comment_count($xoopsModule->getVar('mid'), $item_id);
```

Si vous omettez le second paramètre ($item_id), la variable $count vous donnera le nombre total des commentaires associés aux "articles" de votre module.

Options
Vous pouvez inclure 2 fonctions supplémentaires pour:
- actualiser le nombre de commentaires par "article" dans le champ comments de votre table (conseillé si c'est le cas)
- exécuter une action lorsqu'un commentaire est approuvé (dans le cas où les commentaires sont soumis à approbation évidemment !), tel que l'envoi d'un mail au 'poster' (en principe non implémenté: voir le fichier include/comment_functions.php).

Fichier xoops_version : rajouter

```
// Comment callback functions
$modversion['comments']['callbackFile'] = 'include/comment_functions.php'; // nom du fichier contenant les fonctions
$modversion['comments']['callback']['approve'] = 'mymodule_comments_approve'; // fonction approbation
$modversion['comments']['callback']['update'] = 'mymodule_comments_update'; // fonction de mise à jour
```

Attention de bien insérer le nom exact (dirname) de votre module.

11.3 Notification

A faire

11.4 Tables Commentaires et notification

Commentaires	Notification	
com_id	Not_id	id de l'enregistrement (auto_increment)
com_modid	Not_modid	id du module
com_itemid	Not_itemid	id de l'item
com_uid	Not_uid	Id de l'user (qui a posté, qui veut être notifié)
com_pid		Id parent du commentaire
com_rootid		Id 'racine' du commentaire

12 Tests et mise au point

Servez vous abondamment et selon les cas, des 3 modes de mise au point de xoops :
- Php
- Mysql/blocks
- Templates Smarty

En laissant de préférence le mode Php lorsque vous n'utilisez pas les autres.
Nettoyez votre code : suppressions commentaires inutiles, portions commentées pour test, genre // echo $ma variable,
Installez, désinstallez, mettez à jour, testez vos options, etc et modifiez votre code pour supprimer tout ce qui vous engendre des Warning (obligation) et des Notice (souhaitable) en mode debug php (voir § 5.3)

Maintenant pensez que vos utilisateurs n'auront pas forcément la même configuration de Php : testez votre module avec
- register_globals à Off
- safe_mode à Off
- asp_tags à Off

xoops_version.php

// Informations principales sur le module	
$modversion['name'] = _MI_MYMODULE_NAME;	Titre du module
$modversion['version'] = 1.0;	Version
$modversion['description'] = _MI_MYMODULE_DESC;	Description
$modversion['author'] = "Auteur";	Auteur
$modversion['credits'] = "Credits";	Crédits
$modversion['help'] = " mymodule.html";	Fichier d'aide
$modversion['license'] = "GPL see LICENSE";	License
$modversion['official'] = 1;	Version officielle (1) sinon 0
$modversion['image'] = "images/mymodule_slogo.png";	Logo du module
$modversion['dirname'] = " mymodule ";	Nom du module
// Les tables du module	
$modversion['sqlfile']['mysql'] = "sql/mysql.sql";	
// Tables créées par le fichier sql (sans le prefixe !)	
$modversion['tables'][0] = " mymodule _table1";	Nom première table (sans préfixe)
$modversion['tables'][1] = " mymodule _table2";	Nom deuxième table (sans préfixe)
//etc	
// Administration	
$modversion['hasAdmin'] = 1;	1 si administration du module
$modversion['adminindex'] = "admin/index.php";	index administration
$modversion['adminmenu'] = "admin/menu.php";	menu d'administration
// Blocks	
$modversion['blocks'][1]['file'] = " mymodule _block1.php";	Fichier du bloc 1
$modversion['blocks'][1]['name'] = _MI_ MYMODULE _BNAME1;	Titre du bloc
$modversion['blocks'][1]['description'] = "Description du block";	Description
$modversion['blocks'][1]['show_func'] = "b_ mymodule _show";	Nom de la fonction d'affichage
$modversion['blocks'][1]['edit_func'] = "b_ mymodule _edit";	Nom de la fonction d'édition
$modversion['blocks'][1]['options'] = "date|10|19";	Options à l'appel du bloc
$modversion['blocks'][1]['template'] = mymodule _block1.html';	Template du bloc
$modversion['blocks'][2]['file'] = " mymodule _block2.php";	Fichier du bloc 2
// etc ….	
// Menu	
$modversion['hasMain'] = 1;	1 si lien dans menu principal
$modversion['sub'][1]['name'] = _MI_MYMODULE_SMNAME1;	Titre du sous-menu 1
$modversion['sub'][1]['url'] = "sousmenu1.php";	Fichier appelé par sous menu 1
$modversion['sub'][2]['name'] = _MI_MYMODULE_SMNAME2;	Titre du sous-menu 2
$modversion['sub'][2]['url'] = "sousmenu2.php?param=1";	
//etc	
// Templates	
$modversion['templates'][1]['file'] = mymodule _templ1.html';	Fichier template 1
$modversion['templates'][1]['description'] = '';	Description
$modversion['templates'][2]['file'] = mymodule _ templ2.html';	Fichier template 2
//etc	
// Config settings	Configuration des préférences
$modversion['config'][1]['name'] = 'notifysubmit';	Nom de la variable
$modversion['config'][1]['title'] = '_MI_NOTIFYSUBMIT';	Titre affiché à gauche du champ de saisie
$modversion['config'][1]['description'] = '_MI_NOTIFYSUBMITDSC';	Texte facultatif affiché sous le titre
$modversion['config'][1]['formtype'] = 'yesno';	
$modversion['config'][1]['valuetype'] = 'int';	Type de l'élément de formulaire
$modversion['config'][1]['default'] = 1;	Format Valeur par défaut
$modversion['config'][2]['name'] = 'notifysubmit';	
$modversion['config'][2]['title'] = '_MI_NOTIFYSUBMIT';	
$modversion['config'][2]['description'] = '_MI_NOTIFYSUBMITDSC';	
$modversion['config'][2]['formtype'] = 'yesno';	
$modversion['config'][2]['valuetype'] = 'int';	
$modversion['config'][2]['default'] = 1;	
$modversion['config'][2]['options'] = 1;	
// Recherche	
$modversion['hasSearch'] = 1;	1 si utilisation fonction recherche
$modversion['search']['file'] = "include/search.inc.php";	Fichier search.inc.php
$modversion['search']['func'] = "news_search";	Nom de la fonction search