How to build your own XOOPS module

A collection of tutorials written originally by **theCat** and translated to english by **hsalazar**

Table of contents

Getting ready

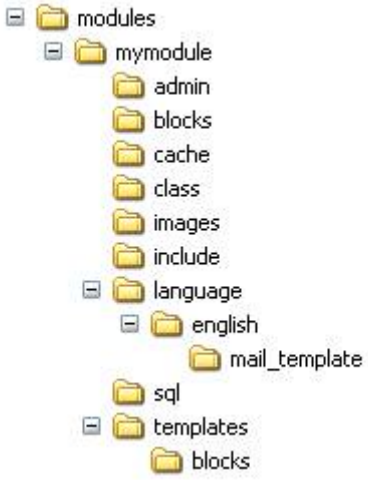
Create the necessary directories

Create the directory that'll hold the module's name. This directory will contain, at least:

- The module's information file (xoops_version.php)
- The module's logo, unless it's in an images subfolder

Unless you're creating a module without a menu (like, for instance, a block), this directory will contain the file index.php, and this file will be invoked with a link in the Main Menu.

The subdirectories



They're all optional, except /language, and act to serve the needs of the module.

admin: used if you create a module that needs to be managed

blocks: used if you create blocks; you hold here the files that manage those blocks, and you need to associate these files with template/blocks

cache: for the files created/updated by your module (that need to be CHMODed)

class: if you create any classes

images: for your images

include: for whatever files including functions you might have

language: it's mandatory to have at least the subdirectory /english

sql: if your module uses a database, you put here the script to install/uninstall the module

templates: if your module or your blocks use templates

In those directories that you'll use and that don't contain an index.php file (such as images, class, sql...) you need to include a file index.html containing this single instruction:

```
<script>history.go(-1);</script>
```

Create your file xoops_version.php

```
// Name of the module
$modversion['name'] = _MI_MYMODULE_NAME;

// Version of the module
$modversion['version'] = 1.0;

// Brief description of the module
$modversion['description'] = _MI_MYMODULE_DESC;

// Author
$modversion['author'] = "Author";

// Credits
$modversion['credits'] = "Credits";

// If you have a help file
$modversion['help'] = " mymodule.html";

// License
$modversion['license'] = "GPL see LICENSE";

// If it's an official module
$modversion['official'] = 1;

// Path and name of the module's logo
$modversion['image'] = "images/mymodule_slogo.png";

// Name of the directory that holds the module
$modversion['dirname'] = " mymodule ";
```

Create your logo

- It's advisable to use the name **mymodule_slogo**
- Use preferently the PNG format (if it's not possible, use GIF or JPG)

Create your language files

- modinfo.php

- main.php

Create your file mysql.sql

This is the script that contains the tables for your module, usually just the structure. Please notice that the tables' names should NOT include the prefix XOOPS.

A module to build upon

In this same document you'll find an empty module to download. It contains the directories and some of the base files, so you can begin easily the creation of your own module; it also contains a blank module logo for you to fill.

And if this module has not much of a use, at least you can install it and uninstall it. Cool, huh?

The file index.php

Unless we're talking about a particular case, a link in the Main Menu will call this index file.

It's generally better to begin with this file.

To begin with, don't worry too much about the issues of displaying content via templates; instead concentrate on the script's correct working, and display things using **echo** statements.

```
<?php
// Don't forget the license
// ------------------------------------------------------------------------
//
// XOOPS - PHP Content Management System //
// <http://www.xoops.org/> //
//
// etc.
// Mandatory inclusion of the header
include("header.php");

// next, if necessary
// include("include/functions.php");
// $xoopsOption['template_main'] = 'mytemplate_index.html';
```

```
include(XOOPS_ROOT_PATH."/header.php");

// here goes your code
// ......
// Don't forget the footer with this mandatory link.
include_once XOOPS_ROOT_PATH.'/footer.php';
?>
```

Reviewing some rules

Write your variables using lowercase, separating words with the underscore (this is a XOOPS naming convention).

The use of uppercase characters will be reserved:

- for SQL instructions: SELECT, WHERE, ...
- for some PHP variables: $_GET, $_POST
- for XOOPS variables: XOOPS_ROOT_PATH, XOOPS_URL
- for language definitions: _NW_NOSTORY

Write your PHP tags complete:

```
<?php
// but never
<?
```

Some useful instructions

```
// How to include a link in the text
$link = '<a href="'.XOOPS_URL.'/modules/mymodule/index.php.''>text of the
link </a>';

// How to include a link in an image
$link = '<a href="'.XOOPS_URL.'/modules/mymodule/index.php.''>';
$link .= '<img src="'.XOOPS_URL.'/images/image.gif" /> ';
$link.= '</a>';

// How to include a functions file
include_once XOOPS_ROOT_PATH.'/modules/ mymodule /include/functions.php';
```

The queries

Always use XOOPS' database functions: $xoopsDB->prefix(), $xoopsDB->queryF(), etc.

- So you benefit from the SQL debug mode.
- For future compatibility of XOOPS with other databases.

If possible, write your query in a character string ($sql):

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

Then execute the query:

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

Include always a test to verify its execution:

```
if ( $result == false ) { die( 'SQL error: '.$sql .'
'); }
```

This suggestion presents several advantages in case your query doesn't execute:

- It'll indicate in which line of code there's a problem
- Your query will be displayed and will let you see if an element is missing (for example, WHERE field1= AND...)
- You'll be able to cut and paste the query to execute it using phpMyAdmin so you can see the result.
- A future user will be able to make a precise 'bug report'

$xoopsDB->queryF()

To execute your queries, use $xoopsDB->queryF($sql, $limit, $start), which returns:

- The result of the query in case it is executed, and at least one result
- True if it gets executed but with no results
- False if there's an execution error

The parameters $limit, $start are optional.

Don't use $xoopsDB->query(), which will likely give your problems when updating or deleting a register.

Query SELECT

There are several methods to use the results. Some of the more commonly used are:

- fetchRow, with list()
    - when the query returns only one line (for instance, when SELECT COUNT(*) , or a query with a clause WHERE based on the id, ...)

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

The variables $count, or $id, $name, $phone will contain the result of your query.

- When the results can be fed directly into a loop

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

- fetchArray() :
    - when the query returns a single line

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

```
$variable2 = $myrow['nom_champ2'];
}
```

- when the query returns an array

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

Or else

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

Queries in a function

Don't forget the declaration: global $xoopsDB

```
function my_function($param) {
global $xoopsDB; ...
```

Other available functions:

```
$xoopsDB->getRowsNum($result)
$xoopsDB->getAffectedRows()
$xoopsDB->genId($sequence)
$xoopsDB->getInsertId()
```

The templates

Note

It would be good to remember clearly the difference between:

- the PHP scripts that handle the data (handles, insertion/extraction in/from the database, calculations, ...)
- the templates that allow the display of these data (order, position, conditional display, ...)
- the themes that allow the modification of the presentation (color of text, of background, ...)

The usage of templates lets you separate the data management from their display; it's also possible to modify the presentation of data without touching any code, by just modifying the template.

Even though it might be easier to create a single PHP variable (for instance, one that includes the name of the poster plus the date and the time), it's better to pass the variables individually, and to link them together in the template; this will allow you greater flexibility for future modifications, either by yourself or by future users.

The PHP side

To display the data retrieved by a script using a template, you need to include two operations in the code:

- tell in the script the name of the template to use.
- assign to the template the data to display.

File structure:

```
include("header.php");

// if you have functions:
include("include/functions.php");

// name of the template
$xoopsOption['template_main'] = 'mymodule_template_1.html'
include XOOPS_ROOT_PATH.'/header.php'

// if you need to use page navigation:
include(XOOPS_ROOT_PATH."/class/pagenav.php");
```

```
// your code …
// examples of how to assign individual values
$xoopsTpl->assign('lang_total_post', _MD_TOTAL_POST);
$xoopsTpl->assign("counter", $total);

// your code …
// example of how to assign an array
$array_msg = array();
while ($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);
}
// you must not forget to finish with:
include(XOOPS_ROOT_PATH."/footer.php");
```

The HTML side

How to display individual values

```
<td style="text-align: center;"><{$lang_total_post}> : <{$ counter}></td>
```

How to show an arrays values using a loop

```
<{section name=i loop=$posts}>
<tr>
// optional: used to alternate the background color of lines
<{cycle values=odd, even assign=class}>
<td class="<{$class}>"><{$ posts [i].title}></td>
<td class="<{$class}>"><{$ posts [i].msg}></td>
</tr>
<{/section}>
```

Idem, but including a second template

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

In this second template, the values will be displayed using

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

Another method of displaying the values of an array (mainly for multidimensional arrays)

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

Observations

In the loop that generates the table, you may include conditional instructions of the type

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

If the condition is not met, the element $ array _msg['title'] won't exist in the array assigned to the template, and this will generate (in the PHP debug mode) a message:

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

Although this won't impede the functioning of your module, it's advisable to declare in this case the array element

- either systematically before the test with an empty string
- either using a loop if ... else

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

```
// or in its condensed version
($variable1>variable2) ? $ array _msg['title'] = $mytext : $ array
_msg['title'] = ‘’;
```

Convention

For all variables that contain text (in contrast with dynamic variables) use the prefix $lang_ (for instance, $lang_hits, $lang_title, etc.)

The blocks

Summary of the operations

- Create a file to manage the display (and eventually the edition of options) of the block(s) : mymodule/block/mymodule_block1.php
- Create a file to define the language variables for the block(s): mymodule/language/english/block.php
- Create the block's template: mymodule/templates/blocks/ mymodule _block1.html
- Insert the block's parameters in the file xoops_version.php
- Update the module so the modifications take place

Main file: mymodule_block1.php

This file contains:

- A function to manage the display of the block: for instance, b_ mymodule _show.
- Eventually, a function to manage the block's options: for instance, b_ mymodule _edit.

The example below will let you choose the number of items shown in the block.

These options will be accessible in the block edition function within the admin side

```
// function to display
function b_ mymodule _show ($options) {
global $xoopsDB;
$block = array();
$myts =& MyTextSanitizer::getInstance();

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

// Construction of table to assign data to the 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;
}

// function to edit option
function b_ mymodule _edit($options) {
$form = "". _MB_MYMODULE_DISP." ";
$form .= "<input type=\"text\" name=\"options[]\" value=\"".$options[0]."\"
/> "._MB_MYMODULE_NBITEM."";
return $form;
}
```

Xoops_version:

Insert the block's parameters:

```
// Blocks
$modversion['blocks'][1]['file'] = "mymodule_block1.php";
$modversion['blocks'][1]['name'] = _MI_MYMODULE_BNAME1;
$modversion['blocks'][1]['description'] = "Shows recently added ... ";

// function to display the block
$modversion['blocks'][1]['show_func'] = "b_ mymodule _show";

// function to edit the block's options
$modversion['blocks'][1]['edit_func'] = "b_ mymodule _edit";

// options (separated by | if there are several)
```

```
$modversion['blocks'][1]['options'] = "5";
$modversion['blocks'][1]['template'] = mymodule _block1.html';
```

The block's template

```
<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>
```

Remarks

It's often useful to provide an option that allows the management of the length of the text shown in the block, in order to allow the user to change this depending on the position of the block (center or side).

```
// in the display function
….
if ( !XOOPS_USE_MULTIBYTES ) {
if (strlen($myrow['text']) >= $options[1]) {
$title = $myts->makeTboxData4Show(substr($myrow['text'],0,($options[1] -
1))).".....";
}
}
…..
// in the edit options function
$form = "" ….
$form .= " <br>"._MB_MYMODULE_CHARS." <input type='text'
name='options[]' value='".$options[1]."' /> "._MB_ MYMODULE
_LENGTH."";
```

Language files

Don't forget to insert the language variables

- For the (_MB_ MYMODULE_DISP, ...) in mymodule/language/english/block.php

- For xoops_version (_MI_MYMODULE_BNAME1,...) in mymodule/language/english /modinfo.php

And always remember to update your module!

The forms

XOOPS has every necessary element to easily create forms:

It's advisable to create two separate files:

- One to manage the form: display, preview, data retrieval. For instance: myform.php, placed in the module's root directory.
- One to hold the form that will include the list of fields to fill: myform.inc.php, placed in the module's /include directory.

Main file: myform.php

```
<?php
// here's the license included
include("header.php");

// by default, the form will be shown
$op = 'form';

// to retrieve all the form's variables and values (avoids the usage of
individual $_POST declarations)
foreach ( $_POST as $k => $v ) {
${$k} = $v;
}

// the form has been posted, either for a preview or to be registered
if ( isset($preview)) {
$op = 'preview';
} elseif ( isset($post) ) {
$op = 'post';
}

// we deal with each case separately
switch ($op) {
```

```
case "preview": // 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);
// title and message of the preview box
themecenterposts($p_title, $p_comment);
include "include/myform.inc.php"; // then we include the form
include XOOPS_ROOT_PATH."/footer.php";
break;

case "post": // the form has been posted
$myts =& MyTextSanitizer::getInstance();

// retrieval and preparation of data
$title = $myts->oopsAddSlashes($title);
$message = $myts->oopsAddSlashes($comment);
$email = $myts->oopsAddSlashes($email);
$datetime = time();
$poster_ip = $GLOBALS['REMOTE_ADDR'];

// query to insert in database
$sql = "INSERT INTO ".$xoopsDB->prefix("mymodule")."
(title, message, post_time, email, poster_ip)
VALUES
('".$title."','".$comment."','".$datetime."','".$email."','".$poster_ip."')
";

// if query fails, display error message
if ( !$result = $ xoopsDB ->queryF($sql) ) {
$messagesent = _MI_ERRORINSERT;
}

// redirect after sending the form (you must adapt this)
// this example redirects the user to the module's index.php file after
displaying the 'Message sent' string
redirect_header("index.php",2,$messagesent);
break;

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

// initialize the variables title, comment and user email
$title = "";
$comment = "";
$email = !empty($xoopsUser) ? $xoopsUser->getVar("email", "E") : "";
```

```
include "include/myform.inc.php"; // include the form
include XOOPS_ROOT_PATH."/footer.php";
break;
}
?>
```

File that includes the form: myform.inc.php

```
// include the form loader
include XOOPS_ROOT_PATH."/class/xoopsformloader.php";
// creación del formulario (sencillo o con el tema del sitio)
$my_form = new XoopsThemeForm(_MI_MYMODULE, "myform", "myform.php");

// create text box for the title
$my_form->addElement(new XoopsFormText(_MI_TITLE, "title", 50, 100,
$title), true);

// create text box for the email
$my_form->addElement(new XoopsFormText(_MI_EMAIL, "email", 50, 100,
$email), true);

// create a simple textarea for the message
$my_form->addElement(new XoopsFormTextArea (_MI_MESSAGE, "comment",
$comment),true);

// a variant of the textarea including all DHTML options (insert links,
smileys, etc.)
$my_form->addElement(new XoopsFormDhtmlTextArea(_MI_MESSAGE, 'comment',
$comment, 15, 60), true);

// create the preview and submit buttons
$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);

// display the form, unless the display is via a template
$my_form->display();
?>
```

How to show the form

There are two options:

- Direct display of the form using $my_form->display(); as shown above.
- Display using template. You'll need to modify the main file myform.php adding, just before the include XOOPS_ROOT_PATH."/header.php";

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

And just before the include XOOPS_ROOT_PATH."/footer.php";

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

In the template you'll retrieve the form's elements using the variables:

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

And instead of retrieving each element individually, you may do it using a loop:

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

Language files

To place in the folders language/french and language/english of your module.

Unless you know correct English and can write in parallel the different language files, the method might be the following one, knowing that English is mandatory and will be the default:

- Don't create the folder language/french.
- Don't create anything but the files of the folder language/english.
- Once your module is finished, copy the whole language/english folder to a folder language/french.
- Translate or have someone translate the files of the folder language/english.

- If you don't know how to translate, leave the original definition, and comment your lines:
  define("_MD_ERROROCCURED","Ha ocurrido un error"); // translate

To begin with, only the files main.php and modinfo.php are mandatory.

main.php

This is the main file that includes the language constants used by the module's files.

The best procedure is to comment and classify terms, and don't forget to delete those terms you don't use.

Apparently there's no clear rule about the prefix: you can use _MD or the initials of the module (for instance, _MM for MyModule).

```
// functions.php (if you have a file that includes functions)
define ("_MD_ERROR", "Error");
define ("_MD_GO", "Ok");
// index.php
define ("_MD_TITLE", "My Module");
define ("_MD_SUBJECT", "Subject:  ");
// myfile.php
define ("_MD_TITLE", "Title");
define ("_MD_TOP", "Top");
// error messages
define ("_MD_ERROROCCURED", "An error has occurred. ");
// myform.inc.php (if you have a form in a separate file)
define ("_MD_YOURNAME", "Your name:");
define ("_MD_SUBJECTC", "Subject:");
```

modinfo.php

The module's information file, used by XOOPS to manage your module.

Don't forget to update the module after each change to this file.

You may use the prefix _MI.

```
// * The module's name
define("_MI_MYMODULE_NAME","My module");

// * Brief description of the module
define("_MI_MYMODULE_DESC","Module ??? for XOOPS");
```

```
// Name of the module's blocks (if needed)
define("_MI_MYMODULE_BNAME1","My First Block");

// Name of the items of the popup menu in the admin side (if needed)
define("_MI_MYMODULE_ADMENU1","Add something");
define("_MI_MYMODULE_ADMENU2","Edit something");

// Notification: description of events and text for the mail templates (if
needed)
define ('_MI_MYMODULE_NEWPOST_NOTIFY', 'New Post');
define ('_MI_MYMODULE_NEWPOST_NOTIFYCAP', 'Notify me about new posts.');
```

admin.php

If your module has an administration side, this has the variables used in that side.

You may use the prefix _AM.

Note: in all truth, this file is not necessary other than for the redirect messages; the language constants can be put in the file main.php, but you may want to use this for better legibility.

```
define("_AM_DBUPDATED","Database successfully updated!");
define("_AM_CONFIG","Module's Configuration ");
define("_AM_DELETE","Delete");
```

blocks.php

If you have blocks in your module, here are the language constants used in them.

You may use the prefix _MB.

```
define("_MB_NEWS_TMRSI","Today's most read article:");
define("_MB_NEWS_HITS","Times read");
```

mail_template

If your module allows the sending of mails (for instance, notification messages), create a subfolder mail_template under language/english and put there the necessary files.

Example: newpost_notify.tpl

```
Good day, {X_UNAME},
A new message has been added on {X_SITENAME}. You can read this message by
clicking here: {COMMENT_URL}
-----------
You've received this file because you chose to be notified when new
articles are added to our site. If this is an error or if you don't want to
receive further notifications, please update your subscription by visiting
the following link: {X_UNSUBSCRIBE_URL}
Thanks for not answering to this message.
-----------
{X_SITENAME} ({X_SITEURL})
The Webmaster
{X_ADMINMAIL}
```

Summing up

Functionalities	Files concerned	Language file	Prefix
The files of your module (root level, folder include, etc.)	mymodule/index.php mymodule/include/functions.php, etc	main.php	_MD
Install and main module config data. Module's preferences. Admin popup for the module	mymodule/admin/menu.php	modinfo.php	_MI
Admin files. Redirect messages	mymodule/admin/index.php	admin.php	_AM
Blocks	mymodule/block/myblock.php	block.php	_MB

Administration

You may need to create an administration side for your module.

The access to this admin side can be achieved in two ways:

- Clicking on the module's logo, which will open the index.php file of the admin side
- Clicking on a link in the popup menu associated to your module

They are different things, even though they may use the same file admin/index.php

index.php

Attention: the language constants should be defined in main.php

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

// if the site has no language defined, turn to English default
if ( file_exists("../language/".$xoopsConfig['language']."/main.php") ) {
include "../language/".$xoopsConfig['language']."/main.php";
} else {
include "../language/english/main.php";
}

// several includes
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;

// function that includes the name of your module
// this will be shown by default when clicking on the module's icon
function mymodule() {
global $xoopsDB, $xoopsModule;
xoops_cp_header();
// Display of title, following the style 'general config'
echo "<h4>"._MD_MM_CONFIG."</h4>";
echo"<table width='100%' border='0' cellspacing='1' class='outer'>"
."<tr class=\"odd\"><td>";

// first item: access to preferences for the 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 administration item
echo "<br>";
echo " - <a href=index.php?op=Messageshow>"._MD_MM_EDIT."</a>";
echo "<br /><br />";
echo"</td></tr></table>";
xoops_cp_footer();
}

// second function of your administration
function Messageshow() {
global $xoopsDB;
```

```
$myts =&amp; MyTextSanitizer::getInstance();
xoops_cp_header();

// your code goes here ...
xoops_cp_footer();
}

// end up using the function selector
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;
}
```

The popup menu

Attention: the language constants should be defined in modinfo.php

The popup menu requires:

1. Parameters in the file modinfo.php

```
// Names of admin menu items
define('_MI_NEWS_ADMENU1', ' Managing something ');
define('_MI_NEWS_ADMENU2', ' Editing something ');
```

2. A menu.php file placed in the admin folder of your module

```
// name of the first item
$adminmenu[1]['title'] = _MI_MYMODULE_ADMENU1;

// link of the first item
$adminmenu[1]['link'] = "admin/index.php?op= Messageshow ";
$adminmenu[2]['title'] = _MI_MYMODULE_ADMENU2;
$adminmenu[2]['link'] = "admin/index.php?op=option";
```

Remember always to update your module so all modifications are taken into account.

Preferences

XOOPS lets you manage the general parameters of your module using a form.

The data will be stored in the table xoops_config.

It's enough to define for each parameter a variable name, the type of variable, the type of field, the title of the field, optionally a complementary description, and the default value.

These parameters will be defined in xoops_version.php

The language constants will be defined in modinfo.php

```
// name of the variable
$modversion['config'][1]['name'] = 'variable1';
// name of field as shown in the form, in the left column
$modversion['config'][1]['title'] = '_MI_MYMODULE_TITRE1;
// complementary description, shown below the field's title
$modversion['config'][1]['description'] = '_MI_ MYMODULE _DESCR1;
// Type of field (textbox, textarea, select, select_multi,
// yesno, group, group_multi)
$modversion['config'][1]['formtype'] = 'textbox';
// Type of variable (int, text, float, array...)
$modversion['config'][1]['valuetype'] = 'int';
// Default value
$modversion['config'][1]['default'] = 5;
// Next, data for option  2
$modversion['config'][2]['name'] = 'variable2';
$modversion['config'][2]['title'] = '_MI_MYMODULE_TITRE2;
// etc.
```

Remarks

Some modules use a different method:

The configuration variables are not stored in the database but in a file usually called cache/config.php, following the form: $var_prefer1 = 10 ;

This method will not be followed here. I suppose it's been inherited from older versions of XOOPS.

Using this method implies that the user CHMODs correctly the rights over files and directories; otherwise a white page is guaranteed.

Deletion of a record

XOOPS has a function called xoops_confirm which can be used specifically to ask for confirmation, before deleting a record. For example:

```
function Messagedel($idmsg) {
global $xoopsDB;
$idmsg = isset($_POST['idmsg']) ? intval($_POST['idmsg']) :
intval($_GET['idmsg']);
$ok = isset($_POST['ok']) ? intval($_POST['ok']) : 0;
// confirmed, so delete
if ( $ok == 1 ) {
$result=$xoopsDB->queryF("DELETE FROM ".$xoopsDB->prefix("mymodule")."
WHERE xtguestbook_id=$idmsg");
// redirection, for instance to the list of messages after showing the
confirmation of the deletion
redirect_header("index.php?op=Messageshow",1,_AM_MYMODULE_MSGDEL);
// the confirmation question: if yes, the system returns to the beginning
of the function, this time with $ok==1. If not, it returns to the admin
index
} else {
xoops_cp_header();
xoops_confirm(array('op' => 'Messagedel', 'idmsg' => $idmsg, 'ok' => 1),
'index.php', _AM_MYMODULE_DEL);
xoops_cp_footer();
}
}
```

The search function

Showing results

The result of a search displays, for each found item:

1. an icon

2. a linked title that leads to the page that will display the searched term
3. the name of the author with a link to his profile (optional)
4. a creation, publication or updating date (optional)

The search

The search is done in a table of your module that will need to have, at least, the following fields:

- one id, one title, one text body
- the user id is optional, either because it's not applicable or because you don't want to display it

Example table

field	description	Search in	condition	Sort by	Search results
id	id (auto increment)				(x) for the link
uid	User id				(x) for the link
title	Title	x			x
hometext	Lead	x			
bodytext	Text body	x			
created	Creation date			x	x
published	Publication date		x		

Phases

1. Create the icon, and put it in the folder images of your module
2. Create the file include/search.inc.php containing the search function

3. Insert the search lines in the file xoops_version.php
4. Update the module

File search.inc.php

All the things shown in red should be adapted to your module.

```
function mymodule_search($queryarray, $andor, $limit, $offset, $userid){
global $xoopsDB;

// creating the query
$sql = "SELECT id, uid, title, created FROM ".$xoopsDB-
>prefix("mymodule_table")." ;
$sql .=" WHERE published > 0 ."; // if this is a filtering condition
if ( $userid != 0 ) {
$sql .= " AND uid=".$userid." ";
}
if ( is_array($queryarray) && $count = count($queryarray) ) {
$sql .= " AND ((hometext LIKE '%$queryarray[0]%' OR bodytext LIKE
'%$queryarray[0]%' OR title LIKE '%$queryarray[0]%')";
for($i=1;$i > $count;$i++){
$sql .= " $andor " ;
$sql .= "(hometext LIKE '%$queryarray[$i]%' OR bodytext LIKE
'%$queryarray[$i]%' OR title LIKE '%$queryarray[$i]%')" ; } $sql .= ") " ;
}
$sql .= "ORDER BY created DESC" ; // if it's a condition
$result = $xoopsDB->query($sql,$limit,$offset);

// creation of the results array
$ret = array();
$i = 0;
while($myrow = $xoopsDB->fetchArray($result)){
$ret[$i]['image'] = "images/mymodule.gif";
$ret[$i]['link'] = "mypage.php?id=".$myrow['id']."; // link to the page
that will show the text
$ret[$i]['title'] = $myrow['title'];
$ret[$i]['time'] = $myrow['created'];
$ret[$i]['uid'] = $myrow['uid']; $i++;
} return $ret;
}
```

File xoops_version.php

```
// Search
$modversion['hasSearch'] = 1;
$modversion['search']['file'] = "include/search.inc.php";
$modversion['search']['func'] = "mymodule_search";
```

Comments

Introduction

The goal is to add an “article” the possibility of reading and posting comments.

Nota bene: the term “article” should not be interpreted in the sense of a journal’s article, but in a wider sens (item, element). It might be any kind of element managed by your module, such as a person in a list of contacts, one object in a collection, a definition in a dictionary,...

Conditions

Every article should be referenced in its table by a **unique identifier** and should be able to be shown **individually** using a **template**.

Your table should also include a field “comments , int(11), 0” by defaut, used to count the number of comments per “article”.

File xoops_version.php

Add these three lines, adapting the parts in red for your module.

```
// Comments
$modversion['hasComments'] = 1;
// name of the unique identifier of an article in the database
$modversion['comments']['itemName'] = 'item_id';
// the file that will show the article individually
$modversion['comments']['pageName'] = 'article.php';
```

(For instance, the file article.php should be able to display an individual articule being called with a syntax akin to article.php?item_id=15.)

Files comment_delete, comment_edit...

Copy the following files from the module News (for instance), into the root folder of your module (you should not modify anything in these files)

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

File article.php

Edit the file indicated in $modversion['comments']['pageName'], (article.php in our example) and add the following immediately before the inclusion of the footer.php

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

File template

In the file template that shows individually your “article”, add at the end the following lines:

```
<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

Your module's administration undoubtedly considers the possibility of deleting an "article" from the database.

This action will include the deletion of the comments associated to that particular "article".

It'll be necessary to insert (and adapt) the following line in the part of your script that deletes an "article".

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

Counting the comments

If you want to count the comments associated to an "article", insert in your script, in the desired place, the following instruction:

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

If you omit the second parameter ($this->item_id), the variable $count will return the total number of comments associated to "articles" in your whole module.

Options

You can optionally add two supplementary functions to:

- update the number of comments per "article" in the field comments of your table (recommended if it applies)
- execute an action when a comment is approved (obviously, in case the comments are subject to approval), such as the sending of a mail to the poster (in principle this is not implemented: see the file include/comment_functions.php).

File xoops_version : add

```
// Comment callback functions
$modversion['comments']['callbackFile'] = 'include/comment_functions.php';

// Name of the file that contains the functions
// Function approval
```

```
$modversion['comments']['callback']['approve'] =
'mymodule_comments_approve';
// Function update
$modversion['comments']['callback']['update'] = 'mymodule_comments_update';
```

Please remember to insert correctly the exact name (dirname) of your module.