Writing a Simple Module

Last Updated Thursday, 09 March 2006

As an example, we will create a module that will list any content items that have keywords that match those in the one that is being viewed. Where do we start?

First decide on a module name. We are going to call this a Related Items module. The module name will be mod_relcontent. All module names must be prefixed with "mod_" and the "relcontent" just stands for "related content" in our case.

In a scratch area on your file system, create a directory called mod_relcontent. In this directory, just create two empty files for the moment; one called mod_relcontent.php and the other called mod_relcontent.xml. Let's build the XML file first.Â This is a definition file that tells the Joomla installer, most importantly, what files are required and other metadata about the module. [Please ignore this dummy sentence which is intended to deflect the glossary bot from incorrectly autocapitalising the word "{glossarbot=disable}author" in the XML sample below] Copy and paste the following code into the XML file:

```
<?xml version="1.0" ?>
<mosinstall type="module">
   <name>Related Items</name>
   <creationDate>19/Aug/2003</creationDate>
   <author>Andrew Eddie</author>
   <copyright>This template is released under the GNU/GPL License</copyright>
   <authorEmail>eddieajau(at)users.sourceforge.net</authorEmail>
   <authorUrl></authorUrl>
   <version>1.0</version>
   <description>Shows related content items based on
      keywords in the meta key field</description>
    <files>
       <filename module="mod_relcontent">mod_relcontent.php</filename>
    </files>
</mosinstall>
```

The important tags here are:

name
The name used in menus.
files
There is only one file required for a module.

Â

Save the XML file and move to the php file. Modules used to store their output in a $content variable. This is still supported, but you can now either use echo statements or escape in and out of php to provide the output. The complete code is shown at the end of this article. Let's step through it.

The first line after the comment block is extremely important. This prevents direct and potentially malicious execution of the script.

Next, a couple of URL variables are collected. When you are writing your

modules, never assume that variables you need from the URL are already available. This is not good programming practice and they simply may not be within the scope of the code. For example, this module code is actually called from within a function, so many global variables simply are not visible. However, several code variables are made available, like $database.

The script then checks to see if you are viewing a content item. If you are, it selects the value of the `metakey` field from the item.The Database Connector

First you need to "initialise" the query. The main reason for this is that the query string you supply is parsed for a hash-underscore-underscore. This is replaced by the database prefix stored in the system configuration variables.

```
$database->setQuery( "SELECT metakey FROM #__content WHERE id='$id'" );
```

OK, we've set up our query. This query returns only a single value. This is a really common exercise so we've provided a method called loadResult() just to grab that single value. Having got the value and checked that it contains something, we explode the string on commas. We then use arrays to help build a new database query that, in rough pseudo-code says "get all the id's of the content items where their metakey field is like *this* or *that*".

Now you'll see we have another common operation, that is, getting a list of results from a query. Here we use the database class method called loadObjectList() to return an array of rows where each row is stored as an object. The method returns null if the query fails to facilitate error checking.

Having received your list of matched records, it's now a trivial exercise to loop through the array and print out a list of links.Finishing Up

Well, now we've got some code, how do we get it into Joomla. The module installer now requires a zip file of the php and XML file under it's parent directory (that is, in this case, mod_relcontent). Zip the two files up. Then, in the Joomla Administrator, select Modules > Install from the menubar. At the bottom of the list you'll see an upload area. Browse to the zip file and upload it. Viola, all going well, your new module is now installed and ready to use.

Go to Modules > Manage Modules to publish and select the 'side' and pages you want the module to appear on.

Now, in a couple of content items, put in a couple of different matching keywords in the Metadata tabs. From the front-end, view the items. You should get a list of other records that have matching keywords.

```
<?php
//Related Content//
/**
* Related Content Module
* @package Joomla
* @copyright Copyright (C) 2005 Open Source Matters. All rights reserved.
* @license http://www.gnu.org/copyleft/gpl.html GNU/GPL, see LICENSE.php
* Joomla! is free software and parts of it may contain or be derived from the
* GNU General Public License or other free or open source software licenses.
* See COPYRIGHT.php for copyright notices and details.
*/
```

```
defined( '_VALID_MOS' ) or die( 'Direct Access to this location is not allowed.' );

$option = trim( mosGetParam( $_REQUEST, 'option', null ) );
$task = trim( mosGetParam( $_REQUEST, 'task', null ) );
$id = intval( mosGetParam( $_REQUEST, 'id', null ) );

if ($option == 'content' && $task == 'view' && $id) {

 // select the meta keywords from the item
 $query = "SELECT metakey FROM #__content WHERE id='$id'";
 $database->setQuery( $query );

 if ($metakey = trim( $database->loadResult() )) {
  // explode the meta keys on a comma
  $keys = explode( ',', $metakey );
  $likes = array();

  // assemble any non-blank word(s)
  foreach ($keys as $key) {
   $key = trim( $key );
   if ($key) {
    $likes[] = $key;
   }
  }

  if (count( $likes )) {
   // select other items based on the metakey field 'like' the keys found
   $query = "SELECT id, title"
    . "nFROM #__content"
    . "nWHERE id<>$id AND state=1 AND access <=$my->gid AND (metakey LIKE '%";
   $query .= implode( "%' OR metakey LIKE '%", $likes );
   $query .= "%')";

   $database->setQuery( $query );
   if ($related = $database->loadObjectList()) {
    foreach ($related as $item) {
     echo "<a href="index.php?option=content&task=view&id=$item->id">"
        ."$item->title</a><br />";
    }
   }
   echo $database->getErrorMsg();
  }
 }
}
?>
```