Module Hello World 3

Last Updated Thursday, 07 September 2006

IntroductionThis tutorial aims to further build on the helloworld module we expanded in part three.Â You will learn how to use parameters to fine tune some of the information that is displayed.Â We will also look at looking at ways to 'skin' your module output..
RequirementsYou need for this tutorial:

- Joomla 1.0 or greater

Let's RollYou will recall that we finished part 2 with your module displaying a list of the most hit times.Â Unfortunately we were stuck with displaying only 5 items and could not change the ordering.Â What we will do in this example is learn how to add parameters to the module so that you can vary these conditions.
Setting up the Parameters
We need to revisit the XML file that we created in part 1.Â We define all our parameters in this file.Â First let's consider what variables we want to allow the user to change:

- We want a variable to change the number of items displayed.
- We want to be able to select the ordering of the items from a list.
- We want to be able to select a template to use to change the display of the items (sort of like skinning).

To do this we add a number of param tags to the params tag.Â Open the mod_helloworld.xmlÂ file.Â Delete its contents and replace it with the following code:

```
  <?xml version="1.0" encoding="iso-8859-1"?>
<mosinstall type="module" version="4.5.2">
  <name>Hello World</name>
  <author>Andrew Eddie</author>
  <creationDate>March 2005</creationDate>
  <copyright>(C) 2005 Andrew Eddie</copyright>
  <license>http://www.gnu.org/copyleft/gpl.html GNU/GPL</license>
  <version>2.0</version>
  <description>A module that says hello and displays a list of the
            Â most hit content items</description>
  <files>
    <filename module="mod_helloworld">mod_helloworld.php</filename>
    <filename>mod_helloworld/default.html</filename>
  </files>
  <params>
    <param name="moduleclass_sfx" type="text" default=""
          label="Module Class Suffix"
          description="A suffix to be applied to the css class of the module
                    Â (table.moduletable), this allows individual module styling" />

    <param name="@spacer" type="spacer" default="" label="" description="" />
    <param name="count" type="text" size="20" default=""
          label="Number of items" description="The number of items to display" />
    <param name="ordering" type="list" default="hits" label="Back Button"
          description="Show/Hide a Back Button, that returns you to the previously
                     Â view page">
      <option value="hits">Hits</option>
      <option value="title">Title</option>
    </param>
    <param name="@spacer" type="spacer" default="" label="" description="" />
    <param name="skin" type="list" default="default" label="Module Skin"
          description="The skin for the module display">
      <option value="default">Default</option>
      <option value="bullets">Bullets</option>
    </param>
  </params>
</mosinstall>
```

Each paramÂ tag has a number of common attributes:
nameThe name of the html form field and also the name of the paremeter that you will access in your module code.typeThis is the type of field.Â The standard types are:

- text - a normal text field

- textarea - a normal textarea field.Â You can also add attributes for the rows and cols.

- list - a normal select list form field.Â Lists can have any number of option child tags.

- radio - a radio group.Â The radio type can have any number of option child tags

- spacer - shows a html horizontal rule.

- imagelist - shows a select list of images.Â You also provide a required directory attibute (the default is /images/stories), an optional hide_none attribute (either 0 or 1, where 1 will show a "Use no image" option) and a hide_default attribute (either 0 or 1, where 1 will show a "Use default image" option).

- mos_category - shows a select list of Joomla content categories

- mos_section - shows a select list of Joomla content sections.

- mos_menu - shows a list of Joomla menu items.

defaultA default value for the form field if no value is provided.
labelA label for the form field.descriptionA description (or tooltip) for the form field
There are some additonal attributes based on the type of parameter as described above.Â For those parameters that allow for options, a standard html option tag is used, usually with a value attribute.

Let's have a look at each paramÂ tag in the params tag (each parameter will be identified by it's name attribute):
moduleclass_sfx This is a standard parameter to include in all modules.Â Most modules are placed in a wrapper (unless you are using the loadModules template macro with a style of less than zero) and this wrapper has a css class of moduleclass.Â You can include your own variant of this class and suffix it with a unique identifier, for example, moduleclass_hello.Â To use your custom class you would enter _hello in this field@spacerThis param simply displays as a horizontal rule.Â It is useful for visually separating groups of associated parameters.countThis will define the number of items we will showorderingThis will show a list of possible options for ordering the list of items@spacerAnother spacer.skinThis will show a list of the available 'skins' for the output of the module. When you updated the xml file, navigate to Modules -> Site Modules from the menu in the Administrator.Â Click on the linked name for the Hello World module (it might be on another page of the list depending on it's position).

When the edit screen appears you should see that the parameters show at the bottom of the first column of the form.

You can see how the spacer works, separating the parameters into groups.Â You can also see that the lists are populated with the options defined in the xml file.Â For now you can leave the count blank, but when we have finished altering all the code, come back to this edit form and experiment with different values.Â Note that the Page Class Suffix will only be relevant if you have a custom style in the style sheet for your site template.
Coding for Module ParametersOur next step is to retrieve the parameters in our module and apply them.Â Open the module php file, mod_helloworld.php, in your editor, delete the code and replace it with the following:

```
<?php
/**
* @version 1.0 $
* @package HelloWorld
* @copyright (C) 2005 Andrew Eddie
* @license http://www.gnu.org/copyleft/gpl.html GNU/GPL
*/

/** ensure this file is being included by a parent file */
defined( '_VALID_MOS' ) or
  Â die( 'Direct Access to this location is not allowed.' );

// COLLECT DATA

// assemble query
global $mosConfig_offset;

$now = date( 'Y-m-d H:i:s', time() + $mosConfig_offset * 3600 );

// Retreive parameters

$count = intval( $params->get( 'count', 10 ) );
$ordering = $params->get( 'ordering', 'hits' );
$skin = $params->get( 'skin', 'default' );

// Assign ordering
```

```
switch ($ordering) {
 case 'title':
  $orderBy = 'title ASC';
  break;
 case 'hits':
 default:
  $orderBy = 'hits DESC';
  break;
}

$query = '
 SELECT id, title, hits
 FROM #__content
 WHERE ( state = \'1\' AND checked_out = \'0\' )
  AND ( publish_up = \'0000-00-00 00:00:00\' OR publish_up <= \''
              . $now . '\' )
  AND ( publish_down = \'0000-00-00 00:00:00\' OR publish_down >= \''
              . $now . '\' )
    AND access <= \'' . $my->gid .'\'
 ORDER BY ' . $orderBy . '
 LIMIT ' . $count
;

// prepare the query in the database connector
$database->setQuery( $query );

// retrieve the rows as objects
$rows = $database->loadObjectList();

// DISPLAY DATA

// load the patTemplate library
require_once( $mosConfig_absolute_path . '/includes/patTemplate/patTemplate.php' );

// create the template
$tmpl =& patFactory::createTemplate( '', false, false );

// set the path to look for html files
$tmpl->setRoot( dirname( __FILE__ ) . '/mod_helloworld' );

// load the template based on the selected skin
$tmpl->readTemplatesFromInput( $skin . '.html' );

// add the 'rows' to the rows template with a prefix
$tmpl->addObject( 'rows', $rows, 'row_' );

// output the template
$tmpl->displayParsedTemplate( 'helloworld' );
?>
```

You will see that after the "retrieve parameters" comment we've added a few lines.Â An object variable called $params is already made available to the module.Â It has a method called get.Â This method takes two arguments; the first is the name of the parameter, the second is the default value to assign it if the user has not given it a value.

Once we have the parameters in variable form, you'll see we use them to modify the ordering of the query and also the number of items returned by the query.Â Further down you can see we use the $skinÂ variable to load the selected skin for the output.

Adding Another SkinLast of all we need to add our alternative skin.Â We are going to display the resulted in a bullet list instead of a table.Â Recall that our template current template file is called default.html in the mod_helloworld directory.Â In that same directory create a new file called bullets.html and copy the following code into it:

```
<mos:comment>
@version 1.0
@package HelloWorld
@copyright (C) 2005 Andrew Eddie
@license http://www.gnu.org/copyleft/gpl.html GNU/GPL
</mos:comment>
```

```
<mos:tmpl name="helloworld">
Â Â Â  <h1>Hello World</h1>
Â Â Â  How is this for a change.
Â Â Â  <ul>
Â Â Â  Â Â Â  <mos:tmpl name="rows">
Â Â Â  Â Â Â  <li>
Â Â Â  Â Â Â  Â Â Â  {ROW_TITLE} <em>( {ROW_HITS} )</em></li>
Â Â Â  Â Â Â  </mos:tmpl>
Â Â Â  </ul>
</mos:tmpl>
```

When you are finished, go back to the module edit screen and change some of the values and see what happens.

What we are demonstrating here is the ability to present the same data in different ways without having to re-engineer the core php code.Â The change to the presentation layer is done in familiar HTML files, making Joomla even easier to tune to your specific needs.

You can download the files for this tutorialÂ mod_helloworld_3.zip.