Module Hello World 2

Last Updated Thursday, 07 September 2006

IntroductionThis tutorial aims to build on the helloworld module started in part one.Â You will learn how to retrieve information from the database and how to present this data is a table using patTemplate.
RequirementsYou need for this tutorial:

- Joomla 1.0 or greater

Let's RollYou will recall that we finished part 1 with your module displaying a message and the current time.Â This is all very interesting but far more interesting is the information held in the tables of the Joomla database.Â What we will do in this example is develop a type of Current News module.
The Presentation LayerLet's start with the output of the module.Â Because the code (or the logic) of the module is now separated from the presentation layer (the HTML), it gives the graphic designer an opportunity to design the output separate from the module.

The module code designer has informed us that a list of content items will be provided to the template.Â The content item data will include the item id, the title of the item, and the number of hits the item has received.

Now, open the default.htmlÂ file (remember we saved this in the /mod_helloworld directory).Â Delete what is there and replace it with the following code:

```
<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">
  <h1>Hello World</h1>
  This is the latest and the greatest from <strong>{SITENAME}</strong>
  <table>
    <tr>
      <th>
        Title
      </th>
      <th>
        Hits
      </th>
      </tr>
      <mos:tmpl name="rows">
      <tr>
        <td>
          {ROW_TITLE}
        </td>
        <td>
          {ROW_HITS}
        </td>
      </tr>
      </mos:tmpl>
    </table>
</mos:tmpl>
```

As in part 1, we wrap our whole module output in a template that we've named helloworld.Â It all looks like standard HTML except for a few things.

You'll see we display a message with the {SITENAME}Â variable.Â This variable has already been defined for you in the template.Â It's equivalent to printing the $mosConfig_sitenameÂ variable.Â Other predefined variables include {SITEURL}

and {ADMINURL}.Â These map to the URL of the site and administrator repsectively.

The other thing you'll notice is an embedded template that we have named rows.Â It enclosed a single row of the HTML table.Â The module designer has told us that he has prefixed all variables in the rows template with "row_", and that he has provided at least the Title, which therefore maps to {ROW_TITLE}, and the Hits, which maps to {ROW_HITS}.

Well, that's the template finished (is it that easy I hear you say?).
The Data Layer
Let's move back to the module file, mod_helloworld.php, where we will assemble the data to pass to the template.Â Open the file 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
$query = '
    SELECT id, title, hits
    FROM #__content
    ORDER BY hits DESC
    LIMIT 5
';

// 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
 Â   . '/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
$tmpl->readTemplatesFromInput( 'default.html' );

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

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

Let's examine what's happening here:

- We start with the standard comment and security block.
- We then prepare our query.Â The data we want is in the mos_content table.Â Because we don't always know if the

"mos_" prefix has been used, we use "#__" (hash, underscore, underscore) instead.Â This is automatically replaced with the correct database table prefix.Â You'll notice we are retrieving three fields, the id, the title and the hits.Â We are ordering them by the number of hits in descending order and limiting the number of results to a maximum of five (that is, the top five).
 - Next we initialise the query in the database connector ($database, which is always available to module files) using the setQuery method.
 - Following this we call the loadObjectList method to retrieve the results from the database as an array of PHP objects.Â This is stored in the variable $rows.
That's it for the first half of the module, that is, the collecting of the data.Â The last half of the module is much that same as our original module from part 1.Â The only difference is that we use the addObject method to add the array to the rows template.

Note: the addObject method can take either a single object or an array of objects.

The thing to note here is that for each array member in $rows, the rows template in the HTML file will display a copy of itself.Â In other words, if there are five elements in the $rows array (that is, the database was able to retrive five rows of records), then the rows template will cycle, or iterate, five times.Â The rows template acts much like a foreach loop in PHP.

Save all files and refresh your browser.Â You should see that the module now displays an opening message that includes the name of your site, and also a table of the most hit content items.

What's wrong with this picture.Â Well, you'll see that the query takes no account for publishing dates, whether they are indeed published at all and the security level of the items.Â You have to apply that logic yourself.Â To do this, modifying the query variable in the following way:

```
// assemble query
global $mosConfig_offset;

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

$query = '
Â Â Â  SELECT id, title, hits
Â Â Â  FROM #__content
Â Â Â  WHERE ( state = \'1\' AND checked_out = \'0\' )
Â Â Â  Â Â Â  AND ( publish_up = \'0000-00-00 00:00:00\'
Â Â Â Â Â Â Â Â Â Â Â  OR publish_up <= \'' . $now . '\' )
Â Â Â  Â Â Â  AND ( publish_down = \'0000-00-00 00:00:00\'
Â Â Â Â Â Â Â Â Â Â Â  OR publish_down >= \'' . $now . '\' )
Â Â Â  Â Â Â  AND access <= \'' . $my->gid .'\'
Â Â Â  ORDER BY hits DESC
Â Â Â  LIMIT 5
';
```

In the next part of this series, we will look at adding parameters to the module.

You can download the files for the final part of the tutorial here (mod_helloworld_2.zip).