Layouts

Overview

While technically not necessary for a module to function, it is considered best practices to have a more MVC structure to your module and put all HTML and display code into view files. This allows for separation of the logic from presentation. There is a second advantage to this, however, which is that it will allow the presentation to be overridden easily by any Joomla! 1.5 template for optimal integration into any site.

Overriding module and component presentation in templates is further explained in the Templates: Overrides section.

Directory Structure & Files

The directory structure used for MVC oriented modules includes a tmpl directory for storing view files. While more views may be possible, modules should include at least one view names default.php.

```
/hubzero
  /modules
    /mod_{ModuleName}
      /tmpl
        default.php
```

Implementation

A simple view (default.php) for a module named mod_listnames:

```
<?php defined('_JEXEC') or die('Restricted access'); // no direct acce
ss ?>
<?php echo JText::_('MOD_LISTNAMES_RANDOM_USERS'); ?>
<ul>
 <?php foreach ($items as $item) { ?>
 <li>
  <?php echo JText::sprintf('MOD_LISTNAMES_USER_LABEL', $item->name);
?>
 </li>
 <?php } ?>
</ul>
```

Here we simply create an unordered HTML list and then iterate through the items returned by our helper (in mod_listnames.php), printing out a message with each user's name.

An important point to note is that the template file has the same scope as the mod_listnames.php file. What this means is that the variable $items can be defined in the mod_listnames.php file and then used in the default.php file without any extra declarations or function calls.

Now that we have a view to display our data, we need to tell the module to load it. This is done in the module's controller file and typically occurs last.

```
<?php
// No direct access
defined('_JEXEC') or die('Restricted access');

// Include the helper file
require_once(dirname(__FILE__).DS.'helper.php');

// Get a parameter from the module's configuration
$userCount = $params->get('usercount');

// Get the items to display from the helper
$items = modListNamesHelper::getItems($userCount);

// Include the template for display
require(JModuleHelper::getLayoutPath('mod_listnames'));
```

Here we can see that the name of the module must be passed to the getLayoutPath method of JModuleHelper. This will load default.php and stores the output in an output buffer which is then rendered onto the page output.