Views

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 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.

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

Implementation

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

```
<?php defined('_HZEXEC_') or die(); // no direct access ?>
<?php echo Lang::txt('MOD_LISTNAMES_RANDOM_USERS'); ?>
<ul>
 <?php foreach ($this->items as $item) : ?>
 <li>
  <?php echo Lang::txt('MOD_LISTNAMES_USER_LABEL', $item->name); ?>
 </li>
 <?php endforeach; ?>
</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 display() method. What this means is that the variable $items can be defined in the helper.php file, assigned to $this 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('_HZEXEC_') or die();

class modHelloWorld extends HubzeroModuleModule
{
    /**
     * Retrieves the hello message
     *
     * @param array $params An object containing the module parameters
     * @access public
     */
    public function display()
    {
        $this->greeting = 'Hello, World!';

        parent::display();
    }
}
```

Here we can see that display() method calls its parent class' display() method which, in turn loads the module's view. This will load default.php and stores the output in an output buffer which is then rendered onto the page output.