Controllers

Overview

The controller is responsible for responding to user actions. In the case of a web application, a user action is (generally) a page request. The controller will determine what request is being made by the user and respond appropriately by triggering the model to manipulate the data appropriately and passing the model into the view. The controller does not display the data in the model, it only triggers methods in the model which modify the data, and then pass the model into the view which displays the data.

Most components have two controllers: one for the front-end and one for the back-end.

Creating the Front-end Controller

Joomla! 1.5 Method

Our component only has one task - greet the world. Therefore, the controller will be very simple. No data manipulation is required. All that needs to be done is the appropriate view loaded. We will have only one method in our controller: display(). Most of the required functionality is built into the JController class, so all that we need to do is invoke the JController::display() method.

The code for the base controller com_hello/controller.php is:

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

jimport('joomla.application.component.controller');

/**
 * Hello World Component Controller
 *
 * @package    Joomla.Tutorials
 * @subpackage Components
 */
class HelloController extends JController
{
    /**
     * Method to display the view
     *
     * @access    public
     */
    public function display()
    {
```

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

The JController constructor will always register a display() task and unless otherwise specified (using the registerDefaultTask() method), it will set it as the default task.

This barebones display() method isn't really even necessary since all it does is invoke the parent constructor. However, it is a good visual clue to indicate what is happening in the controller.

The JController::display() method will determine the name of the view and layout from the request and load that view and set the layout. When you create a menu item for your component, the menu manager will allow the administrator to select the view that they would like the menu link to display and to specify the layout. A view usually refers to a view of a certain set of data (i.e. a list of cars, a list of events, a single car, a single event). A layout is a way that that view is organized.

In our component, we will have a single view called hello, and a single layout (default).

HUBzero Method

Most HUBzero component controllers will differ from Joomla! 1.5 in some important ways. This is, in part, due to legacy issues. Some changes are made to aid in development while others may simply be a difference in philosophy. Note, however, that no differences require hacking or altering Joomla! in any way and HUBzero methodologies will run on any stock Joomla! install.

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

class HelloController extends JObject
{
 private $_name  = NULL;
 private $_data  = array();
 private $_task  = NULL;

 //-----------

 public function __construct( $config=array() )
 {
  $this->_redirect = NULL;
  $this->_message = NULL;
```

```
  $this->_messageType = 'message';

  // Set the controller name
  if (empty( $this->_name )) {
   if (isset($config['name'])) {
    $this->_name = $config['name'];
   } else {
    $r = null;
    if (!preg_match('/(.*)Controller/i', get_class($this), $r)) {
     echo "Controller::__construct() : Can't get or parse class name."
;
    }
    $this->_name = strtolower( $r[1] );
   }
  }

  // Set the component name
  $this->_option = 'com_'.$this->_name;
 }

 //-----------

 public function __set($property, $value)
 {
  $this->_data[$property] = $value;
 }

 //-----------

 public function __get($property)
 {
  if (isset($this->_data[$property])) {
   return $this->_data[$property];
  }
 }

 //-----------

 public function execute()
 {
  $this->_task = JRequest::getString('task', '');

  switch ($this->_task)
  {
   default: $this->hello(); break;
  }
```

```
 }

 //-----------

 public function redirect()
 {
  if ($this->_redirect != NULL) {
   $app =& JFactory::getApplication();
   $app->redirect( $this->_redirect, $this->_message );
  }
 }

 //-----------

 protected function hello()
 {
  // Instantiate a new view
  $view = new JView( array('name'=>'hello') );

  // Pass the view any data it may need
  $view->greeting = 'Hello, World!';

  // Set any errors
  if ($this->getError()) {
   $view->setError( $this->getError() );
  }

  // Output the HTML
  $view->display();
 }
}
```

There appears to be a bit more going on here than in the Joomla! example but both code examples are doing essentially the same thing, as we'll explain.

The first, and most important, difference to note is that we're extending JObject rather than JController. Since we're not employing JController, we need to set up many of our methods manually. Much of it, such as the __constructor and redirect methods are very similar to the Joomla! method--they're simply being established here rather than in JController.

One key difference is how the execute() method is handled. In Joomla! 1.5, any public method is assumed to be an executable task. In the HUBzero method, we're explicitly declaring a list of available tasks and what those tasks execute via the switch statement.

Finally, in our display method, we're instantiating a new view, assigning it some data, and then displaying the output.