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.

Creating the Front-end Controller

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

class HelloControllerOne extends HubzeroComponentSiteController
{
 public function displayTask()
 {
  // Pass the view any data it may need
  $this->view->greeting = 'Hello, World!';

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

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

The first, and most important part to note is that we're extending HubzeroComponentSiteController which brings several tools and some auto-setup for us.

Note: HubzeroComponentSiteController extends HubzeroBaseObject, so all its methods and properties are available.

In the execute() method, the list of available tasks is built from only methods that are 1) public and 2) end in "Task". When calling a task, the "Task" suffix should be left off. For example:

```
// This route
JRoute::_('index.php?option=com_example&task=other');

// Refers to
....
public function otherTask()
{
 ...
}
....
```

If no task is supplied, the controller will default to a task of "display". The default task can be set in the controller:

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

ximport('Hubzero_Controller');

class HelloControllerOne extends  HubzeroComponentSiteController
{
 public function execute()
 {
  // Set the default task
  $this->registerTask('__default', 'mydefault');

  // Set the method to execute for other tasks
  // The following can be called by task=delete and will execute the r
emoveTask method
  $this->registerTask('delete', 'remove');  // (task, method name);

  parent::execute();
 }
 ...
}
```

Each controller extending HubzeroComponentSiteController will have the following properties available:

- _option - String, component name (e.g., com_example)
- _controller - String, controller name
- view - Object (JView)
- config - Object (JRegistry), component config
- database - Object (JDatabase)
- juser - Object (JUser)

```
class HelloControllerOne extends HubzeroComponentSiteController
{
 public function displayTask()
 {
  $this->view->userName = $this->juser->get('name');
  $this->view->display();
 }
}
```

Auto-generation of views

The HubzeroComponentSiteController automatically instantiates a new HubzeroComponentView object for each task and assigns the component ($option) and controller ($controller) names as properties for use in your view. Controller names map to view directory and task names directly map to view names.

```
  /{component}
    /views
      /one  (controller name)
        /tmpl
          /display.php
          /remove.php
```

Example usage within a view:

```
<p>This is component <?php echo $this->option; ?> using controller: <?
php echo $this->controller; ?></p>
```

Changing view layout

As mentioned above, the view object is auto-generated with the same layout as the current $task. There are times, however, when you may want to use a different layout or are executing a task after directing through from a previous task (example: saveTask encountering an error and falling through to the editTask to display the edit form with error message). The layout can easily be switched with the setLayout method.

```
  /{component}
    /views
      /one  (controller name)
        /tmpl
          /display.php
          /world.php

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

class HelloControllerOne extends HubzeroComponentSiteController
{
 public function displayTask()
 {
  // Set the layout to 'world.php'
  $this->view->setLayout('world');

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

Any assigned data or vars to the view will not be effected.

Creating the Admin Controller

Delete Edit Move up Move down

Administrator component controls are built the same and function the same as the Front-end (site) controllers with one key difference: they extends HubzeroComponentAdminController.

CONTROLLERS

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

class ExampleControllerOne extends HubzeroComponentAdminController
{
    ...
}
```