MVC View Integration

<ac:macro ac:name="unmigrated-inline-wiki-markup"><ac:plain-text-body><![CDATA[

<ac:macro ac:name="unmigrated-inline-wiki-markup"><ac:plain-text-body><![CDATA[

Zend Framework: MVC View Integration Component Proposal

Proposed Component Name	MVC View Integration
Developer Notes	http://framework.zend.com/wiki/display/ZFDEV/MVC View Integration
Proposers	Matthew Weier O'Phinney
Revision	1.0 - 7 March 2007: Original proposal creation (wiki revision: 5)

Table of Contents

1. Overview
2. References
3. Component Requirements, Constraints, and Acceptance Criteria
4. Dependencies on Other Framework Components
5. Theory of Operation
6. Milestones / Tasks
7. Class Index
8. Use Cases
9. Class Skeletons

1. Overview

The MVC components consist of the Zend_Controller and Zend_View families (and, nominally, Zend_Db_Table). However, there is no current integration of these components; a simple patch would enable this.

2. References

- Ruby on Rails

3. Component Requirements, Constraints, and Acceptance Criteria

- View instantation **will** be lazy-loaded and optional
- View instantation **will** be overrideable to allow for custom view objects or loading view objects from alternate sources (e.g. registry)
- View script suffixes **will** be configurable via a controller property
- View rendering **will not** require an argument, and **will** use the controller and action to determine the view script by default
- View rendering **will** accept an optional view script name

4. Dependencies on Other Framework Components

- Zend_Exception
- Zend_View_Interface
- Zend_Controller_Action

5. Theory of Operation

Two new methods will be added to Zend_Controller_Action, initView() and render(), and two new properties, $view and $viewSuffix. When called the first time, render() will check to see if $view has been populated, and if not, call initView() to initialize the property.

render() will by default render the script <controller>/<action>.$viewSuffix in the view object's script path. If an argument is passed in, it will render $action.$viewSuffix. All rendered views are appended to the response object.

The default implementation of initView() will instantiate a Zend_View object and set the script, filter, and helper paths based on the following directory structure:

```
application/
    controllers/
        IndexController.php
        ErrorController.php
    views/
        scripts/
            index/
            error/
        filters/
        helpers/
    modules/
        app/
            controllers/
                IndexController.php
                ListController.php
                FormController.php
            views/
                scripts/
                    index/
                    list/
                    form/
                filters/
                helpers/
```

Thus, the script path will be set to ../views/scripts, helpers to ../views/helpers, etc, relative to the controller class file.

6. Milestones / Tasks

- Milestone 1: initial proposal accepted
- Milestone 2: working and tested code checked into core
- Milestone 3: documentation exists

7. Class Index

- Zend_Controller_Action

8. Use Cases

UC-01

This example illustrates the default usage.

```
class IndexController extends Zend_Controller_Action
{
    public function indexAction()
    {
        // Renders views/scripts/index/index.phtml
        $this->render();
    }

    public function listAction()
    {
        // Initialize view so we can assign data to it
        $this->initView();
        $results = $this->model->fetchAll();
        $this->view->results = $results;

        // Renders views/scripts/index/list.phtml
        $this->render();
    }
}
```

UC-02

The following example illustrates two concepts:

- Initializing the view to allow assigning data to it
- Rendering an alternate view script (i.e., something other than controller/action.phtml)

```
class CrudController extends Zend_Controller_Action
{
    public function createAction()
    {
        if ('post' != strtolower($_SERVER['REQUEST_METHOD'])) {
            // Renders views/scripts/crud/form.phtml
            return $this->render('form');
        }

        // process form...
    }

    public function updateAction()
    {
        if ('post' != strtolower($_SERVER['REQUEST_METHOD'])) {
            // Initialize view and assign user details to it
            $this->initView();
            $this->view->user = $this->model->find($this->_getParam('id', 1));

            // Renders views/scripts/crud/form.phtml
            return $this->render('form');
        }

        // process form...
    }

    public function readAction()
    {
        // Initialize view and populate with user
        $this->initView();
        $this->view->user = $this->model->find($this->_getParam('id', 1));

        // Renders views/scripts/crud/read.phtml
        return $this->render();
    }
}
```

UC-03

This example illustrates using a different suffix

```
class IndexController extends Zend_Controller_Action
{
    public function init()
    {
        // set a different suffix
        $this->viewSuffix = 'php';
    }

    public function indexAction()
    {
        // Renders views/scripts/index/index.php
        $this->render();
    }
}
```

UC-04

This example illustrates overriding initView() and render().

```
class IndexController extends Zend_Controller_Action
{
    public function initView()
    {
        if (null !== $this->view) {
            return;
        }

        require_once 'My/View/Adapter.php';
        $this->view = new My_View_Adapter();
    }

    public function render($action = null)
    {
        if ( null === $action) {
            throw new Exception('No template specified');
        }
        if (!Zend::isReadable($action, true)) {
            throw new Exception('Template does not exist or is not readable');
        }
        $this->_response->appendBody($this->view->fetch($action));
    }
}
```

9. Class Skeletons

```
abstract class Zend_Controller_Action
{
    /**
```

```
     * @var Zend_View_Interface
     */
    public $view;

    /**
     * @var string
     */
    public $viewSuffix = 'phtml';

    /**
     * Initialize view object
     *
     * @return void
     */
    public function initView()
    {
        if (null !== $this->view) {
            return;
        }

        require_once 'Zend/View.php';
        $basePath = dirname(__FILE__) . DIRECTORY_SEPARATOR . '..'
                  . DIRECTORY_SEPARATOR . 'views' . DIRECTORY_SEPARATOR;

        $this->view = new Zend_View(array(
            'scriptPath' => $basePath . 'scripts',
            'helperPath' => $basePath . 'helpers',
            'filterPath' => $basePath . 'filters'
        ));
    }

    /**
     * Render a view
     *
     * By default, renders controller/action.phtml in the current script path.
     * If $action is provided, controller/$action.phtml will be rendered.
     *
     * The view script suffix may be changed by modifying the value of
     * {@link $viewSuffix}.
     *
     * Rendered views are appended to the response object.
     *
     * @param null|string $action
     * @return void
     */
    public function render($action = null)
    {
        if (null === $this->view) {
            $this->initView();
        }

        $controller = $this->_request->getControllerName();

        if (null === $action) {
            $action = $this->_request->getActionName();
        }

        $script = $controller . DIRECTORY_SEPARATOR . $action . '.' .
$this->viewSuffix;
```

```
        $this->_response->appendBody($this->view->render($script));
```

```
        }
    }
```

]]></ac:plain-text-body></ac:macro>
]]></ac:plain-text-body></ac:macro>