Zend_Controller_Action with Model Proposal .. HAS TO BE REMOVED

<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: Zend_Controller_Action Component Proposal

Proposed Component Name	Zend_Controller_Action
Developer Notes	http://framework.zend.com/wiki/display/ZFDEV/Zend_Controller_Action
Proposers	Aleksey V. Zapparov
Revision	1.1 - 9 January 2008: Initial propose (wiki revision: 13)

Table of Contents

1. Overview
2. References
Wikipedia resources
Similar proposal
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
File: /path/to/mvc/controllers/BookshelfController.php
File: /path/to/mvc/models/BookshelfModel.php
File: /path/to/mvc/views/scripts/bookshelf/index.phtml
9. Class Skeletons

1. Overview

The aim of this proposal is to give a simple way to load models according to Zend Framework code style standards.

2. References

- MVC Pattern Version 2

Wikipedia resources

MVC Pattern
Business logic

Similar proposal

Zend_Controller_Action_Helper_ModelLoader
Zend_Db_Model - Borja Fernandez

3. Component Requirements, Constraints, and Acceptance Criteria

4. Dependencies on Other Framework Components

- Zend_Loader

5. Theory of Operation

The component is instantiated with a mind-link that it will make work parts of MVC pattern alltogether with unified style.

6. Milestones / Tasks

7. Class Index

- Zend_Controller_Action

8. Use Cases

UC-01 (The simplest)

9. Class Skeletons

```
abstract class Zend_Controller_Action
{
    /**
     * Directory where MVCs are stored.
     *
     * @var string
     */
    protected $_mvcPath;


    /**
     * Model object
     *
     * @var object
     */
    protected $model;
```

```
/**
 * Retrieve MVCs path, where models, views and controllers are stored.
 * If optional $appendPath is specified then
 *
 * @param string $appendPath (optional)
 * @return string
 * @throws Zend_Controller_Action_Exception if appendix was given and is not a string
 */
protected function _getMvcPath($appendPath = null)
{
    if (!isset($this->_mvcPath)) {
        $front   = $this->getFrontController();
        $request = $this->getRequest();
        $module  = $request->getModuleName();
        $dirs    = $front->getControllerDirectory();

        if (empty($module) || !isset($dirs[$module])) {
            $module = $front->getDispatcher()->getDefaultModule();
        }

        $this->_mvcPath = dirname($dirs[$module]);
    }

    if (null !== $appendPath) {
        if (!is_string($appendPath)) {
            throw new Zend_Controller_Action_Exception();
        }
        return $this->_mvcPath . DIRECTORY_SEPARATOR . $appendPath;
    }

    return $this->_mvcPath;
}


/**
 * Retrieve Model object
 *
 * If {@link $model} is not set, instantiates a new Model object, using
 * the 'modules' subdirectory at the same level as the controller directory
 * for the current module as the base directory {@link _getMvcPath()}.
 *
 * By default (with $modelNameWithAction = false) the name of model
 * will be composed from Controller name and suffix Model. If optional
 * param $modelNameWithAction = true then name of model will be composed
 * from name of Controller followed by underscope, Action name and suffix
 * Model.
 *
 * Assume we calling $this->initModel() within FooController::DooAction().
 * $this->initModel() or $this->initModel(false) will retrieve you:
 *   FooModel stored in models/FooModel.php
 * $this->initModel(true) will retrieve you:
 *   Foo_DooModel stored in models/Foo/DooModel.php
 *
 * @param $modelNameWithAction
 * @return object
 */
public function initModel($modelNameWithAction = false)
{
    if (!isset($this->model)) {
        $request = $this->getRequest();
        $model   = ucfirst($request->getControllerName());
        if ($modelNameWithAction) {
            $model .= '_' . ucfirst($request->getActionName());
        }
        $model  .= 'Model';
        Zend_Loader::loadClass($model, $this->_getMvcPath('models'));
        $this->model = new $model();
    }
```

```
        return $this->model;
    }

    /**
     * Initialize View object
     *
     * Initializes {@link $view} if not otherwise a Zend_View_Interface.
     *
     * If {@link $view} is not otherwise set, instantiates a new Zend_View
     * object, using the 'views' subdirectory at the same level as the
     * controller directory for the current module as the base directory.
     * It uses this to set the following:
     * - script path = views/scripts/
     * - helper path = views/helpers/
     * - filter path = views/filters/
     *
     * @return Zend_View_Interface
     * @throws Zend_Controller_Exception if base view directory does not exist
     */
    public function initView()
    {
        if (!$this->getInvokeArg('noViewRenderer') && $this->_helper->hasHelper('viewRenderer')) {
            return $this->view;
        }

        require_once 'Zend/View/Interface.php';
        if (isset($this->view) && ($this->view instanceof Zend_View_Interface)) {
            return $this->view;
        }

        $baseDir = $this->_getMvcPath('views');
        if (!file_exists($baseDir) || !is_dir($baseDir)) {
            throw new Zend_Controller_Exception('Missing base view directory ("' . $baseDir . '")');
        }

        require_once 'Zend/View.php';
        $this->view = new Zend_View(array('basePath' => $baseDir));
```

```
        return $this->view;
    }
}
```

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