Zend_Controller_Action_Helper_MultiPageForm - Jurriën Stutterheim

<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_Helper_MultiPageForm Component Proposal

Proposed Component Name	Zend_Controller_Action_Helper_MultiPageForm
Developer Notes	http://framework.zend.com/wiki/display/ZFDEV/Zend_Controller_Action_Helper_MultiPageForm
Proposers	Jurriën Stutterheim Matthew Weier O'Phinney (Zend Liaison)
Revision	0.1 - 1 January 2008: Initial revision. (wiki revision: 6)

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

Zend_Controller_Action_Helper_MultiPageForm provides support for multi-page forms and Post-Redirect-Get with Zend_Form.

2. References

This proposal heavily based on Simon Mundy's Multiform action helper. Credit to this idea should go to Simon. I merely made it play nice with Zend_Form

- Simon's original proposal
- My take on the helper

3. Component Requirements, Constraints, and Acceptance Criteria

- This component **will** provide PRG support.
- This component **will** only support Zend_Form.
- This component **will** provide multi-page form support.
- This component **could** use additional back-ends to store the temporary form data

4. Dependencies on Other Framework Components

- Zend_Exception
- Zend_Session
- Zend_Controller_Action_Helper_Abstract

5. Theory of Operation

The helper makes use of Zend_Form's subform support to provide multi-page forms. Each subform is essentially a form page.
At the end of the ride all the filtered/validated data is collected and made available to the developer to process it.
A session is used to store the data between requests.

Each subform needs a corresponding controller action. The subform and action have the same name.
The action serves two purposes. It allows custom logic for the data handling and it renders the view.

There is some more functionality that could be added to the helper, but I would like to discuss the options of that first.
One of those possibilities could be the branches idea of Mitchell's form proposal.
Another is the ability to allow unique instances of the same form to be open and filled in simultaneously. However, this would heavily increase the helper's complexity. This probably would be an option for a second iteration.

6. Milestones / Tasks

- Milestone 1: [DONE] Rewrite Simon's helper to specialize it for Zend_Form
- Milestone 2: Discuss points mentioned above.
- Milestone 3: Update proposal to reflect comments and results of discussions.
- Milestone 4: Helper checked in to incubator.
- Milestone 5: Documentation & unit test.
- Milestone 6: Helper accepted into core.

7. Class Index

- Zend_Controller_Action_Helper_MultiPageForm

8. Use Cases

UC-01: Basic usage

```
class MyformController extends Zend_Controller_Action
{
    public function init()
```

```
    {
        $this->_helper->multiPageForm->setForm(new MyForm());
        $this->_helper->multiPageForm->setActions(array('order', 'user', 'survey'));
    }

    public function indexAction()
    {
        $this->_helper->multiPageForm->clear();
        $this->getHelper('Redirector')->gotoAndExit('order');
    }

    public function orderAction()
    {
        // custom logic for order action
    }

    public function userAction()
    {
        // custom logic for user action
    }

    public function surveyAction()
    {
        // custom logic for survey action
    }

    public function processAction()
    {
        $data = $this->_helper->multiPageForm->getFormData();

        /**
         * After fetching the data you'd probably want to clear the form
         * so it can go for another round.
         */
        $this->_helper->multiPageForm->clear();

        /**
         * Display a success screen for this demo.
         * In real-life use you'd probably write the data to the DB here.
         * Note that all data in the $data array is filtered and validated,
         * so you don't have to do that here anymore.
         */
        $this->view->formData = $data;
    }
}

class MyForm extends Zend_Form
{
    public function __construct($options = null)
    {
        parent::__construct($options);

        $this->_initOrderForm();
        $this->_initUserForm();
        $this->_initSurveyForm();
    }

    protected function _initOrderForm()
    {
        // setup a new Zend_Form as subform

        $this->addSubForm($orderForm);
    }

    protected function _initUserForm()
    {
        // setup a new Zend_Form as subform

        $this->addSubForm($userDataForm);
    }
```

```
    protected function _initSurveyForm()
    {
        // setup a new Zend_Form as subform
```

```
        $this->addSubForm($surveyForm);
    }
}
```

9. Class Skeletons

```
class Zend_Controller_Action_Helper_MultiPageForm extends Zend_Controller_Action_Helper_Abstract
{
    /**
     * Default keys for controlling the multiform
     *
     */
    const ACTION_PREFIX     = '_';
    const ACTION_KEY_NEXT   = '_next';
    const ACTION_KEY_BACK   = '_back';
    const ACTION_KEY_SUBMIT = '_submit';
    const ACTION_KEY_CANCEL = '_cancel';

    /**
     * Zend_Session storage object
     *
     * @var Zend_Session
     */
    protected $_session = null;

    /**
     * The complete Zend_Form instance
     *
     * @var Zend_Form
     */
    protected $_form = null;

    /**
     * The current subform instance
     *
     * @var Zend_Form
     */
    protected $_currentSubForm = null;

    /**
     * The form data
     *
     * @var array
     */
    protected $_tempFormData = array();

    /**
     * The form action in order of apearance
     *
     * @var array
     */
    protected $_subFormActions = array();

    /**
     * The action that will be used for processing the form
     *
     * @var string
     */
    protected $_processAction = 'process';

    /**
     * The action for canceling the form
```

```
     *
     * @var string
     */
    protected $_cancelAction = null;

    /**
     * The last valid action
     *
     * @var string
     */
    protected $_lastValidAction = null;

    /**
     * Construct and set default session object
     */
    public function __construct()
    {
    }

    /**
     * Executed at preDispatch
     *
     */
    public function preDispatch()
    {
    }

    /**
     * Get the subform data
     *
     * @return array
     */
    public function getSubformData()
    {
    }

    /**
     * Determine if an action has been validated
     *
     * @param string $current
     * @return string
     */
    public function isValidAction($current)
    {
    }

    /**
     * Determine if an action has been submitted
     *
     * @param string $action
     * @return mixed
     */
    public function isCompleteAction($action)
    {
    }

    /**
     * Set the action used for processing the complete form
     *
     * @param string $action
     * @return Zend_Controller_Action_Helper_MultiForm
     */
    public function setProcessAction($action)
    {
    }

    /**
     * Get the processing action
     *
     * @return unknown
```

```
     */
    public function getProcessAction()
    {
    }

    /**
     * Set a custom cancel action
     *
     * @param string $action
     * @return Zend_Controller_Action_Helper_MultiForm
     */
    public function setCancelAction($action)
    {
    }

    /**
     * Get the custom cancel action
     *
     * @return string
     */
    public function getCancelAction()
    {
    }

    /**
     * Set sequence of actions
     *
     * @param array $actions
     * @return Zend_Controller_Action_Helper_MultiForm
     */
    public function setActions(array $actions)
    {
    }

    /**
     * Set values for an action
     *
     * @param Zend_Form $values
     * @param boolean $valid
     * @return Zend_Controller_Action_Helper_MultiForm
     */
    public function setValues(Zend_Form $form, $valid = false)
    {
    }

    /**
     * Retrieve action values
     *
     * @param string $action
     * @return mixed
     */
    public function getValues($action = null)
    {
    }

    /**
     * Retrieve current lsat valid action
     *
     * @return string
     */
    public function getLastValidAction()
    {
    }

    /**
     * Set a form instance
     *
     * @param Zend_Form $form
     * @return Zend_Controller_Action_Helper_MultiForm
     */
```

```
    public function setForm(Zend_Form $form)
    {
    }

    /**
     * Get the form instance
     *
     * @return Zend_Form
     */
    public function getForm()
    {
    }

    /**
     * Get the current subform
     *
     * @return Zend_Form
     */
    public function getCurrentSubForm()
    {
    }

    /**
     * Get a subform by name.
     *
     * @param string $formName
     * @return Zend_Form
     */
    public function getSubForm($formName)
    {
    }

    /**
     * Check if the entire form is valid
     *
     * @return boolean
     */
    public function isValid()
    {
    }

    /**
     * Get all data from the subforms
     *
     * @return array
     */
    public function getFormData($flatten = false)
    {
    }

    /**
     * Get the form data. If it's empty and a submitted form is available,
     * populate it first from POST.
     *
     * @return array
     */
    public function getPostData()
    {
    }

    /**
     * Use the redirector helper to navigate the controller
     *
     * @param string $action
     * @param string $controller
     * @param string $module
     */
    public function redirect($action, $controller = null, $module = null)
    {
    }
```

```
    /**
     * Get the action used to submit the form
     *
     * @return string|false
     */
    public function getSubmitAction()
    {
    }

    /**
     * Handle the form
     *
     * @return boolean
     */
    public function handle()
    {
    }

    /**
     * Check if the action is an action for this form.
     *
     * @param string $action
     * @return boolean
     */
    public function isSubformAction($action)
    {
    }

    /**
     * Reset all session data
     */
    public function clear()
    {
    }

    /**
     * Throw an exception
     *
     * @param string $message
     * @throws Zend_Controller_Exception
     */
    protected function _throwException($message)
```

```
        {
        }
    }
```

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