Zend_Form_Decorator_JavascriptValidator - Ricardo Buquet

<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\Form\Decorator\JavascriptValidator Component Proposal

Proposed Component Name	Zend\Form\Decorator\JavascriptValidator
Developer Notes	http://framework.zend.com/wiki/display/ZFDEV/Zend\Form\Decorator\JavascriptValidator
Proposers	Ricardo Buquet
Zend Liaison	TBD
Revision	1.0 - 1 March 2012: Initial Draft. (wiki revision: 15)

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\Form\Decorator\JsAutoValidator is a unobtrusive JavaScript validator that takes into account every (javascript) possible validator rule, and writes down a json with all the information, making available all you need to know to make your own javascript validation.
It also has a built in javascript validator, which can be replaced by your own implementation.

2. References

3. Component Requirements, Constraints, and Acceptance Criteria

Zend\Form\Decorator\JsAutoValidator will generate an agnostic json for element validation, filtering and error decoration. This will allow you to use the built in javascript validator or your own implementation with a simple adapter pattern.

- This component **will** correctly read the elements validators, filters and error decorators to generate a json with any of these, if present: validator names, parameters, messages variables, messages, errors, filters, error decorators and break chain on failure.

- This component **will** optionally read correctly the errors and formErrors decorators in the form and elements, and expose their desired tags.
- This component **will** optionally read the filtering parameters for the elements and expose their filter names.
- This component **will** allow the user to specify on which event validation should occur (submitting or clicking a submit button)

4. Dependencies on Other Framework Components

- Zend_Form_Decorator_Abstract
- Zend_View
- Zend_View_Helper_HeadScript by default, could be changed by own implementation
- Zend_View_Helper_InlineScript by default, could be changed by own implementation
- Zend_Exception
- Jquery (only the draft will be changed to native javascript)

5. Theory of Operation

The component is added to any Zend_Form

6. Milestones / Tasks

- Milestone 1: [DONE] design notes will be published here
- Milestone 2: [DONE] Working prototype checked attached to the proposal (I don't know how to publish into the incubator) supporting use cases #1, #2
- Milestone 3: [DONE] Unit tests exist, covering 100% of the code.
- Milestone 4: [DONE] Initial documentation exists.

7. Class Index

- Zend_Form_Decorator_JsAutoValidation

8. Use Cases

UC-01	**Assign decorator through ini file**

```
[login]
login.class = "login"
login.id = "loginFormForm"
login.decorators.formElements.decorator = "FormElements"
login.decorators.form.decorator = "Form"
login.decorators.JsAutoValidation.decorator = "jsAutoValidation"
login.decorators.JsAutoValidation.options.validatorTriggerSelector = "loginSubmit"

;;;;;;;;;;;;;;;;;;;;;
;; usernameOrEMail ;;
;;;;;;;;;;;;;;;;;;;;;
login.elements.usernameOrEmail.type = "text"
login.elements.usernameOrEmail.id = "usernameOrEmail"
login.elements.usernameOrEmail.name = "usernameOrEmail"
login.elements.usernameOrEmail.options.required = true
login.elements.usernameOrEmail.options.validators.notEmpty.validator = "NotEmpty"
login.elements.usernameOrEmail.options.validators.notEmpty.breakChainOnFailure = true
login.elements.usernameOrEmail.options.validators.notEmpty.options.messages.notEmptyInvalid = "Not
valid"
login.elements.usernameOrEmail.options.validators.notEmpty.options.messages.isEmpty = "Not Valid"
login.elements.usernameOrEmail.options.validators.stringLength.validator = "stringLength"
login.elements.usernameOrEmail.options.validators.stringLength.options.breakChainOnFailure = true
login.elements.usernameOrEmail.options.validators.stringLength.options.min = 5
login.elements.usernameOrEmail.options.validators.stringLength.options.max = 30
login.elements.usernameOrEmail.options.validators.stringLength.options.messages.stringLengthInvalid =
"You must provide an user"
login.elements.usernameOrEmail.options.validators.stringLength.options.messages.stringLengthTooShort =
"The user must have at least %min% characters"
login.elements.usernameOrEmail.options.validators.stringLength.options.messages.stringLengthTooLong =
"The user must have at most %max% characters"

;;;;;;;;;;;;
;; submit ;;
;;;;;;;;;;;;
login.elements.submit.type = "submit"
login.elements.submit.id = "loginSubmit"
login.elements.submit.name = "loginSubmit"
login.elements.submit.options.label = "LOGIN"
```

UC-02	**Or using a php file.**

```
<?php
$view = new Zend_View();
$form = new Zend_Form();
$form->setAttrib('id', 'formId');
$form->setView($view);

$decorator = new \Zend\Form\Decorator\JavascriptValidator();
$decorator->setJavascriptValidatorPath(
    'http://localhost/zftrunk/demos/Zend/Form/Decorator/JavascriptValidator.js'
);
$form->addDecorator($decorator);

$form->addElement(
    'text',
    'email',
    array(
        'required' => true,
        'validators' => array(
            'NotEmpty',
            array(
                'StringLength',
                false,
                array(
                    'min' => 5,
                    'max' => 20,
                    'messages' => array(
                        Zend_Validate_StringLength::TOO_SHORT => '%value% has less than %min%
characters',
                        Zend_Validate_StringLength::TOO_LONG => '%value% has more than %max%
characters',
                        Zend_Validate_StringLength::INVALID => '%value% is invalid'
                    )
                )
            )
        )
    )
);

$form->addElement(
    'submit',
    'submitForm'
);
```

And in both UC something like this will be printed in json through the inlineScript view helper.

```
var Zend = Zend || {};
Zend.Forms = Zend.Forms || {};
Zend.Forms.formName = {};
Zend.Forms.formName.elements = {
    "usernameOrEmail": {
        "validators": {
            "NotEmpty": {
                "messages": [],
                "messageVariables": [],
                "messageTemplates": {
                    "isEmpty": "You must enter a value"
                },
                "errors": [],
                "breakChainOnFailure": "1"
            },
            "StringLength": {
                "min": 2,
                "max": 255,
                "encoding": null,
                "messages": [],
                "messageVariables": ["min", "max"],
                "messageTemplates": {
                    "stringLengthInvalid": "You must enter a value",
                    "stringLengthTooShort": "The length must be %min% min",
                    "stringLengthTooLong": "The length must be %max% max"
                },
                "errors": [],
                "messageLength": - 1,
                "breakChainOnFailure": ""
            }
        },
        "decorators": {
            "errorDecorator": {
                "separator": "\n",
                "placement": "APPEND"
            },
            "formErrorsHelper": {
                "elementStart": "<ul><li>",
                "elementSeparator": "</li><li>",
                "elementEnd": "</li></ul>"
            },
            "options": {
                "class": "errors"
            }
        }
    },
};
```

9. Class Skeletons

This is the class skeletons

```
<?php
namespace Zend\Form\Decorator;
class JavascriptValidator extends \Zend_Form_Decorator_Abstract
{
    /**
     * Namespace holding the javascript
     * @var String Namespace
     */
```

```
    protected $_javascriptNamespace = 'Zend';

    /**
     * The javascript file validator path
     * @var String javascript validator path
     */
    protected $_javascriptValidatorPath = '/js/library/Zend/Form/Decorator/JavascriptValidator.js';

    /**
     * Holds the reference to the element that must trigger the validation
     * @var string DOM's selector.
     */
    protected $_validationTriggerSelector = null;

    /**
     * Holds the form id
     * @var string Form id
     */
    protected $_formId = null;
    /**
     * Returns the javascript namespace
     * @return string Namespace
     */
    public function getJsNamespace();

    /**
     * Sets the javascript namespace
     * @param string $namespace namespace
     */
    public function setJsNamespace($namespace);

    /**
     * Retrieves element decorators for javascript
     * @param \Zend_Form_Element|\Zend_Form_DisplayGroup|\Zend_Form $element
     * @return array
     */
    protected function _getElementErrorDecorator($element);

    /**
     * Set the javascript validator path
     * @param string $path validator path
     * @return \Zend\Form\Decorator\JavascriptValidator
     */
    public function setJavascriptValidatorPath($path);

    /**
     * Returns the javascript validator path
     * @return string validator path
     */
    public function getJavascriptValidatorPath();

    /**
     * Retrieves element filters for javascript
     * @param \Zend_Form_Element|\Zend_Form_DisplayGroup|\Zend_Form $element
     * @return array
     */
    protected function _getElementFilters(\Zend_Form_Element $element);

    /**
     * Return the javascript equivalent for the element validators
     * @param \Zend_Form_Element | \Zend_Form_DisplayGroup | \Zend_Form $element
     * @return string the javascript validators
     */
    protected function _getElementValidators($element);

    /**
     * Retrieve the validators and prepare a json object to share validator params with javascript
     * @return void
     */
    protected function _buildJsVars();
```

```
/**
 * Generate the javascript code for the validation rules
 * @param \Zend_Form_Element $element
 * @return string
 */
protected function _buildValidationRules(\Zend_Form_Element $element);

/**
 * Generate parameters for a validator rule
 * @param \Zend_Validate_Interface $validator the validator
 * @return array
 */
protected function _buildValidatorParameters(\Zend_Validate_Interface $validator);

/**
 * This method is intented to be overrided and to extend the functionality.
 * as \Zend does with init() methods.
 */
protected function getValidatorOptions();

/**
 * Returns the form id or generates a random one
 * @return string
 */
public function getFormId();

/**
 * Injects the validation event in the inlineScript helper, and appends the javascript validator's
 * file indicated in JavascriptValidator::$_javascriptValidatorPath to the Zend's headLink Helper.
 * @return void
 */
public function render($content);

/**
 * Sets the element that triggers the validation
 * @param string $validationTriggerSelector a valid element in the form
 * @throws \Zend_Exception
 * @return My_Form_Decorator_JsAutoValidation
 */
public function setValidationTriggerSelector($validationTriggerSelector);

/**
 * Retrieves form decoratos for javascript
 * @return array
```

```
     */
    protected function _getFormErrorsDecorator();
}
```

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