Zend_Validate error messages - Bill Karwin

<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_Validate error messages Component Proposal

Proposed Component Name	Zend_Validate error messages
Developer Notes	http://framework.zend.com/wiki/display/ZFDEV/Zend_Validate error messages
Proposers	Bill Karwin
Revision	1.1 - 13 May 2007: Initial write-up. (wiki revision: 11)

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

This is a proposal to add to Zend_Validate classes the ability to specify error messages.

2. References

- TBD

3. Component Requirements, Constraints, and Acceptance Criteria

- This component **will** provide a method in Zend_Validate classes to allow developers to specify the text for error messages
- This component **will** allow messages to contain placeholder text, to be replaced by specific properties of the validator class.
- This component **will not** allow arbitrary access to any protected properties of the validator class, only the ones specifically identified as legal to interpolate into the message templates.
- This component **will** support multiple messages per validator class; the messages are identified by mnemonic keys; developers **can** specify multiple messages in one call.
- This component **will** have default messages in each validator; it is not required to specify messages.
- This component **will** accept a message template without a message key; there is a default message.

4. Dependencies on Other Framework Components

- Zend_Exception

5. Theory of Operation

This solution requires a change to Zend_Validate classes, to store error messages in a class member instead of in hard-coded text as the error conditions are met. Since the messages are stored, their value can be changed.

Since a given validator class can generate more than one message, the interface to change a value of a messages requires that the messages be identifiable. We use a string key to identify the error message. Then the developer can change the value using a method `setMessage($messageText, $key)`. The $key argument is optional, and defaults the first message stored in the class. This makes usage easy if the validator is simple enough to have only one message.

There is also a method `setMessages(array)` so you can pass an associative array to set all messages by key.

When the validator's `isValid()` method returns false, you can call `getMessages()` in the familiar way to get one or more messages related to the failure. This returns either message text that the developer specified in the `setMessage()` method.

In addition, this proposal provides a new method `getErrors()` which returns an array of strings which are taken from the key values of the associative array. Thus the validator can return both developer-specified messages, and standard key strings.

6. Milestones / Tasks

- Milestone 1: design notes will be published here
- Milestone 2: Working prototype checked into the incubator supporting use cases #1, #2, ...
- Milestone 3: Working prototype checked into the incubator supporting use cases #3 and #4.
- Milestone 4: Unit tests exist, work, and are checked into SVN.
- Milestone 5: Initial documentation exists.

7. Class Index

- Zend_Validate_Abstract
- Zend_Validate_<All>

8. Use Cases

UC-01

Using a single validate class, overriding the message, and re-running the validation.

```
<?php

require_once 'Zend/Validate/LessThan.php';

$lt = new Zend_Validate_LessThan(100);
if (!$lt->isValid(101)) {
    print_r($lt->getMessages());
    print_r($lt->getErrors());
}

$lt->setMessage('You are supposed to give a value less than %max%, but you gave %value%');
if (!$lt->isValid(101)) {
    print_r($lt->getMessages());
    print_r($lt->getErrors());
}
```

UC-02

Usage from proposed Zend_Filter_Input.

```
$validators = array(
    'field1' => array(
        new Zend_Validate_LessThan(100),
        'errors' => 'You are supposed to give a value less than %max%, but you gave %value%'
    )
);
$input = new Zend_Filter_Input(null, $validators, $data);
if ($input->hasInvalid()) {
    print_r($input->getMessages());
    print_r($input->getErrors());
}
```

Zend_Validate_Abstract

```
<?php

require_once 'Zend/Validate/Interface.php';

abstract class Zend_Validate_Abstract implements Zend_Validate_Interface
{
    protected $_messageVariables = array();

    protected $_messageTemplates = array();

    protected $_messages = array();

    protected $_errors = array();

    protected function _createMessage($value, $messageKey = null)
    {
        if ($messageKey === null) {
            $messageKey = current(array_keys($this->_messageTemplates));
        }
        if (!isset($this->_messageTemplates[$messageKey])) {
            return null;
        }
        $message = $this->_messageTemplates[$messageKey];
        $message = str_replace('%value%', (string) $value, $message);
        foreach ($this->_messageVariables as $ident => $property) {
```

```
        $message = str_replace("%$ident%", $this->$property, $message);
    }
    return $message;
}

protected function _error($value, $messageKey)
{
    $this->_errors[Zend_Db_Adapter_Odbtp_Mssql] = $messageKey;
    $this->_messages[Zend_Db_Adapter_Odbtp_Mssql] = $this->_createMessage($value, $messageKey);
}

protected function _reset()
{
    $this->_errors = array();
    $this->_messages = array();
}

public function getErrors()
{
    return $this->_errors;
}

public function getMessages()
{
    return $this->_messages;
}

public function setMessage($messageString, $messageKey = null)
{
    if ($messageKey === null) {
        $messageKey = current(array_keys($this->_messageTemplates));
    }
    if (!isset($this->_messageTemplates[$messageKey])) {
        require_once 'Zend/Validate/Exception.php';
        throw new Zend_Validate_Exception("No message template exists for key '$messageKey'");
    }
    $this->_messageTemplates[$messageKey] = $messageString;
    return $this;
}

public function setMessages(array $messages)
{
    foreach ($messages as $key => $message) {
        $this->setMessage($message, $key);
    }
    return $this;
```

```
    }

}
```

Zend_Validate_LessThan

This is a sample of utilizing Zend_Validate_Abstract.

```
<?php

require_once 'Zend/Validate/Abstract.php';

class Zend_Validate_LessThan extends Zend_Validate_Abstract
{
    protected $_max;

    protected $_messageVariables = array(
        'max' => '_max'
    );

    protected $_messageTemplates = array(
        'notLessThan' => "'%value%' is not less than %max%"
    );

    public function __construct($max)
    {
        $this->setMax($max);
    }

    public function getMax()
    {
        return $this->_max;
    }

    public function setMax($max)
    {
        $this->_max = $max;
        return $this;
    }

    public function isValid($value)
    {
        $this->_reset();
        if ($this->_max <= $value) {
            $this->_error($value, 'notLessThan');
            return false;
        }
        return true;
    }

}
```

9. Class Skeletons

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