

Zend_Stdlib_Configurator - Vincent de Lau

<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\Stdlib\Configurator Component Proposal

Proposed Component Name	Zend\Stdlib\Configurator
Developer Notes	http://framework.zend.com/wiki/display/ZFDEV/Zend\Stdlib\Configurator
Proposers	Renan de Lima Vincent de Lau (original)
Zend Liaison	TBD
Revision	1.0 - 5 July 2010: Initial Draft. (wiki revision: 7)

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\Stdlib\Configurator is intended to provide classes with an easy way to process options. It will accept a Zend\Config object or array and call the appropriate setters with the provided value. The class will accept names as both underscored lowercase and camelcaps.

This component fills the gap that comes with lack of multiple inheritance or mixins in the PHP language. Without this component, each class that accepts a Zend\Config object or array has to process the options manually, which is a repetitive task. Extracting the code should lead to less code duplication and more stable code.

2. References

3. Component Requirements, Constraints, and Acceptance Criteria

- This component **will** call the setter for each provided option.
- The component **will** align with coding standards and practices.
- The component **will not** have any knowledge of the target object.
- The component **will** allow option names to be passed as underscored words.
- The component **will** treat option names case insensitive.

- The component **will** require the target object to have setters for each configurable option.
- The component **will not** convert or test any configuration data passed.

4. Dependencies on Other Framework Components

- Zend_Config (optional)
- Zend_Exception

5. Theory of Operation

Zend\Stdlib\Configurator will accept a target object and an object implementing Traversable. This includes Zend\Config objects and plain arrays.

For each option passed, the Configurator will call the appropriate setter with the value provided. Option names are stripped of underscores before resolving the setter. Since call_user_method(), is_callback() and method_exists() are case sensitive, option names are practically case insensitive.

Since no state has to be maintained, the Configurator will be implemented as a class with only static methods.

6. Milestones / Tasks

- Milestone 1: Wait for input based on coding standards and the need for configurable objects
- Milestone 2: Finish the proposal
- Milestone 3: Unit tests exist, work, and are checked into SVN.
- Milestone 4: Initial documentation exists.
- Milestone 5: The component is applied to all appropriate framework components

7. Class Index

- Zend\Stdlib\Configurator
- Zend\Stdlib\ConfiguratorException

8. Use Cases

UC-01

```
class MyClass {
    protected $_fooBar;
    public function __construct($options) {
        Zend\Stdlib\Config::configure($this,$options);
    }

    public setFooBar($value) {
        $this->_fooBar = $value;
    }
}

$options = array('foo_bar' => 'baz');
$object = new MyClass($options);

// $object->_fooBar === 'baz'
```

9. Class Skeletons

```
namespace Zend\Stdlib;

class Configurator {
    /** Configure a target object with the provided options.
     *
     * The options passed in must be a Traversable object with option names for keys.
     * Option names are case insensitive and will be stripped of underscores. By convention,
     * option names are in lowercase and with underscores separated words.
     *
     * The target object is expected to have a setter method for each option passed. By convention,
     * setters are named using camelcase with a 'set' prefix.
     *
     * Example: option_name -> setOptionName()
     *
     * @param object $target The object that needs to be configured.
     * @param \Traversable $options The configuration to apply. Traversable is amongst
     *                               others implemented by Zend\Config and arrays
     * @param boolean $tryCall When true and $target has a __call() function, try call if no setter
     *                           is available.
     * @return void Nothing
     */
    public static function configure($target, $options, $tryCall=false)
    {
        if ( !is_object($target) )
        {
            throw new ConfiguratorException('Target should be an object');
        }
        if ( !($options instanceof Traversable) && !is_array($options) )
        {
            throw new ConfiguratorException('$options should implement Traversable');
        }

        $tryCall = (bool) $tryCall && method_exists($target, '__call');

        foreach ($options as $name => &$value)
        {
            $setter = 'set' . str_replace(' ', '', ucwords(str_replace('_', ' ', $name)));

            if( $tryCall || method_exists($target,$setter) )
            {
                call_user_func(array($target,$setter),$value);
            }
            else
            {
                continue; // instead of throwing an exception
            }
        }
    }
}

class ConfiguratorException extends Exception {}
```

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

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