Zend_Filter_StringLength - Thomas Weidner

<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_Filter_StringLength Component Proposal

Proposed Component Name	Zend_Filter_StringLength
Developer Notes	http://framework.zend.com/wiki/display/ZFDEV/Zend_Filter_StringLength
Proposers	Thomas Weidner
Zend Liaison	TBD
Revision	1.0 - 12 April 2009: Initial Draft 1.1 - 13 April 2009: added View Helper (wiki revision: 8)

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_Filter_StringLength is a filter which allows to pad/extend or truncate a string to a given length. It will also have a view helper which allows to use this filter easily within your view.

2. References

- Zend_Filter

3. Component Requirements, Constraints, and Acceptance Criteria

- This component **will** pad a string to a given length
- This component **will** allow to pad/extend a string on both sides
- This component **will** allow to use any character to pad the string
- This component **will** truncate a string at a given length
- This component **will** only work on strings

4. Dependencies on Other Framework Components

- Zend_Filter
- Zend_View_Helper

5. Theory of Operation

This component allows to pad/extend a string to a given length the same way as str_pad would do. And to truncate a string to a given length as substr allows to.

It will additionally allow to seperate text based on words by using eighter whitespaces, hypens or any other character.

And it adds a view helper which allows to use this filter within your view. The View Helper will also allow to set the filter within a registry using the name 'Zend_Filter_StringLength'.

6. Milestones / Tasks

- Milestone 1: [DONE] Proposal finished
- Milestone 2: Proposal accepted
- Milestone 3: Working implementation
- Milestone 4: Unit tests
- Milestone 5: Documentation
- Milestone 6: Moved to core

7. Class Index

- Zend_Filter_StringLength
- Zend_View_Helper_StringLength

8. Use Cases

UC-01

Pad a string to 15 chars

```
$filter = new Zend_Filter_StringLength(array('min' => 15);
$filter->filter('My string');
// returns 'My string      '
```

UC-02

Pad a string to 15 chars, using another pad char

```
$filter = new Zend_Filter_StringLength(array('min' => 15, 'end' => '.'));
$filter->filter('My string');
// returns 'My string......'
```

UC-03

Pad a string to 15 chars, using another pad char and another direction

```
$filter = new Zend_Filter_StringLength(array('min' => 15, 'end' => '.', 'type' =>
Zend_Filter_StringLength::LEFT));
$filter->filter('My string');
// returns '......My string'
```

UC-04

Truncate a string at 6 chars

```
$filter = new Zend_Filter_StringLength(array('max' => 6));
$filter->filter('My string');
// returns 'My str'
```

UC-05

Truncate a string at 6 chars, using ending chars

```
$filter = new Zend_Filter_StringLength(array('max' => 6, 'end' => '..'));
$filter->filter('My string');
// returns 'My s..'
```

UC-06

Define padding and truncating at the same time

```
$filter = new Zend_Filter_StringLength(array('min' => 12, 'max' => 15, 'end' => '..'));
$filter->filter('My string');
// returns 'My string   ' as the string is padded from 9 to 12 chars

$filter->filter('My string with more content');
// returns 'My string with..' as the string is truncated at 15 chars
```

UC-07

Define truncating based on words

```
$filter = new Zend_Filter_StringLength(array('max' => 15, 'end' => '', 'word' => true));
$filter->filter('My string with more content');
// returns 'My string with' as the string is truncated at 15 chars but the last word ends at char
14... truncating ONLY at whitespaces
```

UC-08

Using the viewhelper

```
$this->view->stringlength('my long text', array('max' => 8, 'end' => '...'));
// would return 'my long ...'
```

UC-09

Define truncating based on different trunc-characters

```
$this->view->stringlength('my-long-text', array('max' => 8, 'end' => '...', 'word' => '-'));
// would return 'my-long-' as the text is word-seperated using the '-' char
```

UC-10

Using the viewhelper with registry

```
// bootstrap or module
$filter = new Zend_Filter_StringLength(array('max' => 8, 'end' => '...'));
Zend_Registry::set('Zend_Filter_StringLength', $filter);

// view, note that no options have to be set as the registry will be used in this case
$this->view->stringlength('my long text');
// would return 'my long ...'
```

9. Class Skeletons

```
class Zend_Filter_StringLength implements Zend_Filter_Interface {
    /**
     * @param string|integer|array $options (Optional) Options to set
     */
    public function __construct($options = array())

    /**
     * @param integer $min Minimum length to pad to
     * @return Zend_Filter_StringLength
     */
    public function setMin($min)

    /**
     * @return integer
     */
    public function getMin()

    /**
     * @param integer $max Maximum length to truncate to
     * @return Zend_Filter_StringLength
     */
    public function setMax($max)

    /**
     * @return integer
     */
    public function getMax()

    /**
     * @param string $character Character to use for padding/truncating
     * @return Zend_Filter_StringLength
     */
    public function setChar($character)

    /**
     * @return string
     */
    public function getChar()

    /**
     * @param string $type Type to use for padding/truncating
     * @return Zend_Filter_StringLength
     */
    public function setType($type)

    /**
     * @return string
     */
    public function getType()

    /**
     * @param string @value Value to filter
     */
    public function filter($value)
}

class Zend_View_Helper_StringLength extends Zend_View_Helper_Acstract
{
}
```

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