Zend_Filter_Bbcode proposal

<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_Bbcode Component Proposal

Proposed Component Name	Zend_Filter_Bbcode
Developer Notes	http://framework.zend.com/wiki/display/ZFDEV/Zend_Filter_Bbcode
Proposers	Van Belle Jonathan Ralph Schindler, Zend liaison
Revision	1.0 - 22 Augustus 2007 : first proposal (wiki revision: 9)

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_Bbcode is a component to parse bbcode tags to html syntax.

2. References

3. Component Requirements, Constraints, and Acceptance Criteria

4. Dependencies on Other Framework Components

- Zend_Filter_Interface

5. Theory of Operation

...

6. Milestones / Tasks

- Milestone 1: [DONE] basic code
- Milestone 2: add more comments on the code!
- Milestone 3: Optimisation
- Milestone 4: unit test

7. Class Index

- Zend_Filter_Bbcode

8. Use Cases

UC-01

```
$message = 'test [b]text bold[/b]';
$zfbbcode = new Zend_Filter_Bbcode();
echo $zfbbcode->filter($message);
```

9. Class Skeletons

```
<?php

/**
 * @see Zend_Filter_Interface
 */
require_once 'Zend/Filter/Interface.php';

/**
 * @category   Zend
 * @package    Zend_Filter
 */
class Zend_Filter_Bbcode implements Zend_Filter_Interface
{
 protected $_config = array(
  'url_limit' => 40);

 public function __construct($config = null)
 {
  if (!is_null($config) && is_array($config)) {
   foreach ($this->_config as $k => $v) {
    $this->_config[ $k ] = (isset($config[ $k ]))?$config[ $k ]:$this->_config[ $k ];
```

```
    }
   }
  }

  /**
   * Defined by Zend_Filter_Interface
   *
   * Returns the string $text, when bbcode are replace by html code
   *
   * @param  string $text
   * @return string
   */
  public function filter($text)
  {
   //removing /r because it's bad!
   $text = str_replace("\r", '', $text);

   // first [nobbcode][/nobbcode] -> don't interpret bbcode
   $this->_parseBbcodeNobbcode($text);

   // parse strange bbcode, before other bbcode
   $this->_parseBbcodeCode($text);
   $this->_parseBbcodeQuote($text);

   $this->_parseBbcodeList($text);

   // easy bbcode replacement
   //[br]
   $this->_parseBbcodeBr($text);
   // [i]txt[/i]
   $this->_parseSimpleBbcode('i', '<em>$1</em>', $text);
   // [u]txt[/u]
   $this->_parseSimpleBbcode('u', '<u>$1</u>', $text);
   // [b]txt[/b]
   $this->_parseSimpleBbcode('b', '<strong>$1</strong>', $text);
   // [del]txt[/del] & [strike]txt[/strike]
   $this->_parseSimpleBbcode('del', '<del>$1</del>', $text);
   $this->_parseSimpleBbcode('strike', '<del>$1</del>', $text);

   // [color=color]txt[/color]
   $this->_parseParamBbcode('color', '([a-zA-Z]*|\#?[0-9a-fA-F]{6})',
    '<span style="color: $1">$2</span>', $text);
   // [bgcolor=color]txt[/bgcolor]
   $this->_parseParamBbcode('bgcolor', '([a-zA-Z]*|\#?[0-9a-fA-F]{6})',
    '<span style="background-color: $1">$2</span>', $text);
   // [align=(center|left|right)][/align]
   $this->_parseParamBbcode('align', '(center|left|right|justify){1}',
    '<div style="text-alignement: $1">$2</div>', $text);
   // [size=$size][/size]
   $this->_parseParamBbcode('size', '([0-9].*)',
    '<span style="font-size: $1">$2</span>', $text);

   $this->_parseBbcodeEmail($text);
   $this->_parseBbcodeUrl($text);
   $this->_parseBbcodeImg($text);

   return $text;
  }

  /**
   * parse bbcode corespondig to $pattern inside $text and replace with $replace
   *
   * @param string $pattern
   * @param string $replace
   * @param string $text
   * @uses preg_replace
   */
  protected function _replaceLoop($pattern, $replace, &$text)
  {
   while (preg_match($pattern, $text))
```

```
   {
    $text = preg_replace($pattern, $replace, $text);
   }
  }


  protected function _parseSimpleBbcode($tag, $replace, &$text)
  {
   $this->_replaceLoop('#\[' . $tag . '\](.*?)\[/' . $tag . '\]#si', $replace, $text);
  }

  protected function _parseParamBbcode($tag, $param, $replace, &$text)
  {
   $this->_replaceLoop('#\[' . $tag . '=' . $param . '\](.*?)\[/' . $tag . '\]#si',
    $replace, $text);
  }

  //
  // Function to parse bbcode
  //
  // [br]
  protected function _parseBbcodeBr(&$text)
  {
   $text = preg_replace('#\[br\]#i', ' <br />', $text);
  }

  // [img]http://www.site.tld/image.png[/img]
  protected function _parseBbcodeImg(&$text)
  {
   $text = preg_replace('#\[img\](.*?)\[/img\]#sei',
    '\'<img src="$1" alt="\' . htmlspecialchars(\'$1\') . \'" />\'', $text);
  }

  /**
   * Execute *before* other bbcode parse!
   * [nobbcode]txt[/nobbcode]
   * @param unknown_type $text
   * @return unknown
   */
  protected function _parseBbcodeNobbcode(&$text)
  {
   $text = preg_replace('#\[nobbcode\](.+?)\[/nobbcode\]#sie', '\'<div class="bbcode_nobbcode">\' .
strtr(\'$1\', array(\'[\' => \'&#91;\', \']\' => \'&#93;\')) . \'</div>\'', $text);
  }

  // [url]url[/url] & [url=url]url txt[/url]
  protected function _parseBbcodeUrl(&$text)
  {
   $text = preg_replace('#\[url\]([Zend_Font - Karol Babioch^ \"\t\n\r<]*?)\[/url\]#ei',
    '$this->_encodeUrl(\'$1\', \'\', $this->_config[\'url_limit\'])', $text);
   $text = preg_replace('#\[url=([Zend_Font - Karol Babioch^ \"\t\n\r<]*?)\](.*?)\[/url\]#sei',
    '$this->_encodeUrl(\'$1\', \'$2\', $this->_config[\'url_limit\'])', $text);
  }

  // [email]email[/email] & [email=email]email txt[/email]
  protected function _parseBbcodeEmail(&$text)
  {
   $this->_replaceLoop('#\[email\]([^\[Zend_Font - Karol Babioch]*?)\[/email\]#sei',
    '$this->_encodeEmail(\'$1\')', $text);
   $this->_replaceLoop('#\[email=([^\[Zend_Font - Karol Babioch]*?)\](.*?)\[/email\]#sei',
    '$this->_encodeEmail(\'$1\', \'$2\')', $text);
  }

  // [code]txt[/code] & [code=language]txt[/code]
  protected function _parseBbcodeCode(&$text)
  {
   $text = preg_replace_callback('#\n?\[code(=[a-zA-Z0-9]*?)?\](.+?)\[/code\]\n?#is',
    array($this, '_cbParseBbcodeCode'), $text);
  }
```

```
 protected function _cbParseBbcodeCode($match)
 {
  $pattern = array("\n",  "\t",     '  ',   '[',  ']',  ')',  '(',  '<',  '>');
  $replace = array('<br />', '    ', '  ', '&#91;', '&#93;', '&#41;',
'&#40;', '&#60;', '&#62;');
  $code = str_replace($pattern, $replace, $match[2]);
  $code_style = 'bbcode_code';
  if ($match[1] != '') {
   $code_style .= '_' . substr($match[1], 1);
  }
  return '<div class="' . $code_style . '">' . $code . '</div>';
 }

 protected function _parseBbcodeList(&$text)
 {
  $pattern = '#\n?\[list=?(greek|square|circle|disc|I|i|A|a|1)?\](.+?)\[/list\]\n?#is';
  while (preg_match($pattern, $text))
  {
   $text = preg_replace_callback($pattern, array($this, '_cbParseBbcodeList') , $text);
  }
 }

 protected function _parseBbcodeQuote(&$text)
 {
  $this->_replaceLoop('#\n?\[quote\](.*)\[/quote\]\n?#mSi',
   '<blockquote>$1</blockquote>', $text);
  $this->_replaceLoop('#\n?\[quote=(.*)\](.*)\[/quote\]\n?#mSi',
   '<blockquote><h5>$1</h5>$2</blockquote>', $text);
 }


 protected function _encodeUrl($url, $txt = '', $limit = 40)
 {
  // @TODO test url and check "javascrip:..." and other bad stuf to stop hack, xss and other

  $url_txt = ($txt != '')?$txt:$url;
  if ($limit > 10 && strlen($url_txt) > $limit)
   $url_txt = substr($url_txt, 0, $limit - 10) . '&#8230;' . substr($url_txt, -10);
  return '<a href="' . $url . '">' . $url_txt . '</a>';
 }

 protected function _encodeEmail($email, $txt = '')
 {
  if (preg_match('#^\w([-_.]?\w)*@\w([-_.]?\w)*\.([a-z]{2,4})$#', $email))
  {
   //mini anti robots
   $new = '';
   $len = strlen($email);
   for ($i = 0; $i < $len; $i++)
   {
    $new .= '&#x' . bin2hex($email{$i}) . ';';
   }

   //formating email mailto.
   return '<a href="mailto:' . $new . '">' . ($txt!=''?$txt:$new) . '</a>';
  }
  return $email;
 }

 protected function _cbParseBbcodeList($match)
 {
  $text = '<ul>';
  $text_end = '</ul>';
  if ($match[1] != '') {
   switch ($match[1])
   {
    case 'disk':
    case 'circle':
    case 'square':
     $text = '<ul type="' . $match[1] . '">';
```

```
    break;
   case 'i':
   case 'I':
   case 'A':
   case 'a':
   case '1':
    $text = '<ol type="' . $match[1] . '">';
    $text_end = '</ol>';
    break;
  }
 }
 $this->_replaceLoop('#\n?(\[\*\](.*))\n?#is', '<li>$2</li>', $match[2]);
 $text .= $match[2];
```

```
    return $text . $text_end;
  }
}
```

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