Zend_Filter_CharacterEntityEncode&Decode - Marc Bennewitz

<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_CharacterEntityEncode & Zend_Filter_CharacterEntityDecode Component Proposal

Proposed Component Name	Zend_Filter_CharacterEntityEncode & Zend_Filter_CharacterEntityDecode
Developer Notes	http://framework.zend.com/wiki/display/ZFDEV/Zend_Filter_CharacterEntityEncode & Zend_Filter_CharacterEntityDecode
Proposers	Marc Bennewitz
Zend Liaison	TBD
Revision	0.1 - 24. Oct 2009: Initial Draft. 1.0 - 03. December 2010: Archived (wiki revision: 18)

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

The encoder is a simple and full configurable filter to encode characters to its entities.
The decoder is the opposite filter to decode entities to its characters.

> **Moved to GitHub**
> This proposal was moved to GitHub
> -> http://github.com/marc-mabe/EntityCoder

2. References

- ZF-3013
- List of XML and HTML character entity references
- Numeric character reference

- Zend_Filter_HtmlEntities

3. Component Requirements, Constraints, and Acceptance Criteria

- This component **will** provide a complete and good configurable interface to encode and decode texts with entities.
- This component **will not** replace Zend_Filter_HtmlEntities.
- This component **will use** iconv to convert character-sets
- This component **will not** handle special CDATA content (like the content of <script></script>)

4. Dependencies on Other Framework Components

- Zend_Filter
- Zend_Exception

5. Theory of Operation

The encoder converts the complete text from intput character set to UTF-8 and replaces only characters which aren't available by given output character set with a named entity given by user or by numeric or hex entity. After this the text will reconverted to output character set.

The decoder converts all entities (named by user entity reference, numeric and hex) to its equipollent by given character sets. If an entity can't convert to the charset the configured action will be used (exception, translit, ignore, entity, substitute). Furthermore it is configurable if the special chars (&,<,>,",') must keep.

6. Milestones / Tasks

- Milestone 1: [DONE] Finish proposal
- Milestone 2: [DONE] Working prototype
- Milestone 3: Prototype checked into the incubator
- Milestone 4: Unit tests exist finished and component is working
- Milestone 5: Initial documentation exists
- Milestone 6: Changed related components
- Milestone 7: Moved to core.

7. Class Index

- Zend_Filter_CharacterEntityEncode
- Zend_Filter_CharacterEntityDecode
  or
- Zend_Filter_EntityEncode
- Zend_Filter_EntityDecode

8. Use Cases

UC-01 - Rich-Text-Editor

You get a html formated text from a rich text editor in ISO-8859-1 and need to convert it to UTF-8.
-> This example converts the text to UTF-8 and converts all entities to UTF-8 but special chars ",',<,>,&.
-> You will get a valid html formated string with a minimum of entities.

```
<?php

ini_set('include_path', '/home/mabe/workspace/zf-proposals/zend_filter_entity/library'
                        . PATH_SEPARATOR . '/home/mabe/workspace/zf-incubator/library'
                        . PATH_SEPARATOR . '/home/mabe/workspace/zf-trunk/library');

header('Content-Type: text/html; charset=ISO-8849-1');
?>
<html>
<head>
<title>Example of Zend_Filter_Encode_Entity</title>
<meta http-equiv="content-type" content="text/html; charset=ISO-8859-15">
<script type="text/javascript">
function sendForm(form) {
    document.getElementById("messageInput").value =
document.getElementById("messageDiv").innerHTML;
    form.submit();
}
</script>
</head>
<body>
<form action="test.php" method="post" onsubmit="sendForm(this);">
  <input type="hidden" name="message" id="messageInput"><div id="messageDiv"
style="border: 1px solid #000;" contentEditable>html <b>formated</b> text with chars
&gt;, &OElig;, &frac14;, &euro;, &#8465; and &auml;</div>
  <input type="submit">
</form>
<br>
<pre>
<?php
if (isset($_POST['message'])) {
    echo "<b>Please display source!</b>\n\n";
    require_once 'Zend/Filter/Encode/Entity.php';
    $entityFilter = new Zend_Filter_Encode_Entity();
    $entityFilter->setInputCharset('ISO-8859-1');
    $entityFilter->setOutputCharset('UTF-8');
    $entityFilter->setKeepSpecial(true);
    echo $entityFilter->decode($_POST['message']) . "\n";
}
?>
</pre>
</body>
</html>
```

UC-02 - Convert HTML to plain

```
// get the html formated text
$htmlTxt = file_get_contents('text.html');

$entityFilter = new Zend_Filter_Encode_Entity();
$entityFilter->setInputCharset('ISO-8859-1');
$entityFilter->setOutputCharset('ASCII');
$entityFilter->setOnInvalidChar(Zend_Filter_Encode_Entity::INVALID_CHAR_TRANSLIT);
$entityFilter->setKeepSpecial(false);

// remove all tags and convert all entities to native characters
// -> if an character can't convert to ASCII the translit feature of iconv will be
used (e.g. € -> EUR)
// -> if translit failed an substitution character will be displayed (default: "?")
echo $entityFilter->decode(strip_tags($htmlTxt));
```

UC-03

```
$text = "This is a text with the umlaut ä, the char € and the special chars & and '";
$filter = new Zend_Filter_Encode_Entity(array(
    'inputCharset' => 'UTF-8',
    'outputCharset' => 'ISO-8859-1'
));

// convert only UTF-8 characters not available on ISO-8859-1 to entities and all other
to native ISO-8859-1
echo $filter->encode($text); // ISO-8859-1: This is a text with the umlaut ä, the char
&#8364; and the special chars &#38; and &#39;

// same but use hax entities
$filter->setHex(true);
echo $filter->filter($text); // ISO-8859-1: This is a text with the umlaut ä, the char
&#x20AC; and the special chars &#x26; and &#x27;

// same but use named entity reference of xml (' -> &apos;)
$filter->setEntityReference('xml');
echo $filter->filter($text); // ISO-8859-1: This is a text with the umlaut ä, the char
&#x20AC; and the special chars &amp; and &apos;

// same but use normal html entity reference (& -> &amp; but ' -> &#x27;)
$filter->setEntityReference('html');
echo $filter->filter($text); // ISO-8859-1: This is a text with the umlaut ä, the char
&#x20AC; and the special chars &amp; and &#x27;

// same but convert to ASCII (ä -> &auml;)
$filter->setOutputCharset('ASCII');
echo $filter->filter($text); // ASCII: This is a text with the umlaut &auml; and the
special chars &amp; and &x27;
```

UC-04

```
$text = '<p>a text with ä, &auml;, &#x20AC;, &lt;, &amp;</p>';
$filter = new Zend_Filter_EntityDecode(array(
    'keepSpecial' => true,
    'output_charset' => 'ISO-8859-1',
    'on_invalid_char' => Zend_Filter_Encode_Entity::INVALID_CHAR_TRANSLIT,
    'substitute' => '?',
));
echo $filter->filter($text); // <p>a text with ä, ä, EUR, &lt;, &amp;</p>

// decode all entities and if not possible translit (&auml; -> ä, &#x20AC; -> EUR,
...)
$filter->setKeepSpecial(false);
echo $filter->filter($text); // <p>a text with ä, ä, EUR, <, &</p>

// same but ignore not possible characters (remove &#x20AC;)
$filter->setOnIllegalChar('ignore');
echo $filter->filter($text); // <p>a text with ä, ä, , <, &</p>

// same but leave not possible characters (&#x20AC; -> &#x20AC;)
$filter->setOnIllegalChar(Zend_Filter_EntityDecode::ONILLEGALCHAR_ENTITY);
echo $filter->filter($text); // <p>a text with ä, ä, &#x20AC;, <, &</p>

// same but use a substitution character for not possible characters (&#x20AC; -> ?)
$filter->setOnIllegalChar(Zend_Filter_EntityDecode::ONILLEGALCHAR_SUBSTITUTE);
echo $filter->filter($text); // <p>a text with ä, ä, ?, <, &</p>
```

9. Class Skeletons

```
class Zend_Filter_EntityEncode implements Zend_Filter_Interface
{

    /**
     * Predefined entity references.
     *
     * @var array
     */
    public static $_entityReferences = array(
        /* special entities */
        'special' => array(
            "&amp;"  => '&',
            "&lt;"   => '<',
            "&gt;"   => '>',
            "&quot;" => '"',
        ),

        /* available on xml without any definition */
        'xml' => array(
            "&amp;"  => '&',
            "&lt;"   => '<',
            "&gt;"   => '>',
            "&quot;" => '"',
            '&apos;' => "'", // not available in html
```

```
        ),

        /* All HTML 4.0 entities */
        'html' => array(
            /* special entities */
            "&amp;"  => '&',
            "&lt;"   => '<',
            "&gt;"   => '>',
            "&quot;" => '"',

// ...

        ),
    );

    /**
     * Entity reference.
     *
     * @var array
     */
    protected $_entityReference = array();

    /**
     * Character set of input value.
     *
     * @var string
     */
    protected $_inputCharSet = 'ISO-8859-1';

    /**
     * Character set of output value.
     *
     * @var string
     */
    protected $_outputCharSet = 'ISO-8859-1';

    /**
     * Use hexadecimal or numeric entities for characters not in character reference
     * and not valit for output char set or special characters.
     *
     * @var boolean
     */
    protected $_hex = false;

    /**
     * Sets filter options
     *
     * @param  integer|array $quoteStyle
     * @param  string  $charSet
     * @return void
     */
    public function __construct($options = array());

    /**
     * Returns input character set.
     *
     * @return string
     */
    public function getInputCharSet();
```

```
    /**
     * Set input character set.
     *
     * @param  string $enc
     * @return Zend_Filter_EntityEncode Provides a fluent interface
     */
    public function setInputCharSet($enc);

    /**
     * Returns output character set.
     *
     * @return string
     */
    public function getOutputCharSet();

    /**
     * Set output character set.
     *
     * @param  string $enc
     * @return Zend_Filter_EntityEncode Provides a fluent interface
     */
    public function setOutputCharSet($enc);

    /**
     * Returns entity reference.
     * Format: array("&<string entity>;" => <utf8 replace>[, ...])
     *
     * @return array
     */
    public function getEntityReference();

    /**
     * Set entity reference.
     * Format: array("&<string entity>;" => <utf8 replace>[, ...])
     *    or:  name of a predefined entity reference
     *
     * @param array|string $entityReference Entity reference.
     * @return Zend_Filter_EntityEncode Provides a fluent interface
     */
    public function setEntityReference($entityReference);

    /**
     * Get the hex option
     *
     * @return boolean
     */
    public function getHex();

    /**
     * Sets the hex option.
     *
     * @param bool $flag
     * @return Zend_Filter_EntityEncode Provides a fluent interface
     */
    public function setHex($flag);

    /**
     * Defined by Zend_Filter_Interface
```

```
     *
     * Returns the string $value, converting characters to their corresponding HTML
entity
     * equivalents where they exist
     *
     * @param  string $value
     * @return string
     */
    public function filter($value);

}

class Zend_Filter_EntityDecode implements Zend_Filter_Interface
{

    const ONILLEGALCHAR_EXCEPTION  = 'exception';
    const ONILLEGALCHAR_TRANSLIT   = 'translit';
    const ONILLEGALCHAR_IGNORE     = 'ignore';
    const ONILLEGALCHAR_ENTITY     = 'entity';
    const ONILLEGALCHAR_SUBSTITUTE = 'substitute';

    /**
     * The Action if an entity can't convert to the given charset
     * (Value of Zend_Filter_EntityDecode::ONILLEGALCHAR_*)
     *
     * @var string
     */
    protected $_onIllegalChar = self::ONILLEGALCHAR_IGNORE;

    /**
     * Output character encoding
     *
     * @var string
     */
    protected $_charSet = 'ISO-8859-1';

    /**
     * entity reference.
     *
     * @var array
     */
    protected $_entityReference = null;

    /**
     * Don't decode entities of special chars.
     * (", &, <, >)
     *
     * @var bool
     */
    protected $_keepSpecial = false;

    /**
     * The substituting character used with constant ONILLEGALCHAR_SUBSTITUTE
     *
     * @var string
     */
    protected $_substitute = '?';

    /**
```

```
     * Sets filter options
     *
     * @param  integer|array $quoteStyle
     * @param  string  $charSet
     * @return void
     */
    public function __construct($options = array());

    /**
     * Get entity reference.
     *
     * @return array
     */
    public function getEntityReference();

    /**
     * Set entity reference.
     *
     * @param array $entityReference
     * @return Zend_Filter_EntityDecode
     */
    public function setEntityReference(array $entityReference);

    /**
     * Get the action which is done if an illegal character was detected.
     *
     * @return string The current action string
     */
    public function getOnIllegalChar();

    /**
     * Set the action which is done if an illegal character was detected.
     *
     * @param string $action The action string to set or empty to get the current
action.
     * @return Zend_Filter_EntityDecode Provides a fluent interface
     */
    public function setOnIllegalChar($action);

    /**
     * Returns the charSet option
     *
     * @return string
     */
    public function getCharSet();
    /**
     * Sets the charSet option
     *
     * @param  string $charSet
     * @return Zend_Filter_EntityDecode Provides a fluent interface
     */
    public function setCharSet($charSet);

    /**
     * Get keep special option.
     *
     * @return bool
     */
    public function getKeepSpecial();
```

```
    /**
     * Sets keep special option
     *
     * @param bool $flag
     * @return Zend_Filter_EntityDecode Provides a fluent interface
     */
    public function setKeepSpecial($flag);

    /**
     * Set the substituting character.
     *
     * @param string $substitute
     */
    public function setSubstitute($substitute);
    /**
     * Get the substituting character.
     *
     * @return string
     */
    public function getSubstitute();

    /**
     * Defined by Zend_Filter_Interface
     *
     * Returns the string $value, converting characters to their corresponding HTML
entity
     * equivalents where they exist
     *
     * @param  string $value
     * @return string
     */
    public function filter($text);
```

```
}
```

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