

Sourcetext des Plugins exttab1

```
<?php
/**
 * exttab1-Plugin: Parses extended tables (like MediaWiki)
 *
 * @license    GPL 2 (http://www.gnu.org/licenses/gpl.html)
 * @author     Werner Flamme <w.flamme@web.de>
 * @date       2006-10-19
 */

if(!defined('DOKU_INC'))
    define('DOKU_INC', realpath(dirname(__FILE__) . '/../..') . '/');
if(!defined('DOKU_PLUGIN'))
    define('DOKU_PLUGIN', DOKU_INC . 'lib/plugins/');
require_once(DOKU_PLUGIN . 'syntax.php');

/**
 * All DokuWiki plugins to extend the parser/rendering mechanism
 * need to inherit from this class
 */
class syntax_plugin_exttab1 extends DokuWiki_Syntax_Plugin {
    /** settings
     *
     */
    var $lineHasStartTag = false;
    var $lineHasEndTag   = false;
    var $bodyHasStartTag = false;
    var $tableHasHeader  = false;

    /**
     * return some info
     */
    function getInfo()
    {
        return array(
            'author' => 'Werner Flamme',
            'email'  => 'w.flamme@web.de',
            'date'   => '2006-10-19',
            'name'   => 'Extended Table Plugin',
            'desc'   => 'parses MediaWiki-like tables',
            'url'    =>
'http://wiki.splitbrain.org/doku.php?id=users:wflamme:exttab1'
        );
    } // function getInfo

    /**
     * What kind of syntax are we?
     */
}
```

```
function getType()
{
    return 'protected';
} // function getType

/**
 * What kind of plugin are we?
 */
function getPType()
{
    return 'block';
} // function getPType

/**
 * Where to sort in?
 */
function getSort()
{
    return 100;
} // function getSort

/**
 * Connect pattern to lexer
 */
function connectTo($mode)
{
    $this->Lexer->addEntryPattern('<exttab1(?.*\x3C/exttab1\x3E)',
    $mode, 'plugin_exttab1');
} // function connectTo

function postConnect()
{
    $this->Lexer->addExitPattern('</exttab1>', 'plugin_exttab1');
} // function postConnect

/**
 * Handle the match
 */
function handle($match, $state, $pos)
{
    if ( $state == DOKU_LEXER_UNMATCHED ) {
        $matches = preg_split('/>/u', $match, 2);
        $matches[0] = trim($matches[0]);
        if ( trim($matches[0]) == '' ) {
            $matches[0] = NULL;
        } // if ( trim($matches[0]) == '' )
        return array($matches[1], $matches[0]);
    } // if ( $state == DOKU_LEXER_UNMATCHED )
    return true;
} // function handle
```

```

/**
 * Create output
 * @param $mode      current mode of DokuWiki (see
http://wiki.splitbrain.org/plugin:tutorial)
 * @param $renderer  DokuWiki's rendering object
 * @param $data      the data between <exttab1> and </exttab1> tags
 * @var $rawdata     $data split into lines
 * @var $rawline     any line of $rawdata
 * @var $mytable     the code that will be inserted into the document
 * @return true, if rendering happens, false in all other cases
 */
function render($mode, &$renderer, $data)
{
    if ($mode == 'xhtml' && strlen($data[0]) > 1) {
        $rawdata = explode("\n", $data[0]);
        $mytable = '';
        $this->lineHasStartTag = false;
        $this->lineHasEndTag   = false;
        foreach ($rawdata as $rawline) {
            $rawline = trim($rawline);
            if (substr($rawline, 0, 2) == '| -') {
                // new table line
                $mytable .= $this->_handleNewTableLine($rawline);
            } // new table line
            elseif (substr($rawline, 0, 2) == '{|') {
                // start table
                $mytable .= $this->_handleTableStart($rawline);
            } // start table
            elseif (substr($rawline, 0, 2) == '|}') {
                // end table
                $mytable .= "    </tr>\n </tbody>\n</table>\n";
                $this->bodyHasStartTag = false;
            } // end table
            elseif (substr($rawline, 0, 2) == '|+') {
                // table caption
                $mytable .= $this->_handleTableCaption($rawline);
            } // table caption
            elseif (substr($rawline, 0, 1) == '|') {
                // regular line
                $mytable .= $this->_handleRegularLine($rawline);
            } // regular line
            elseif (substr($rawline, 0, 1) == '!') {
                // header line
                $mytable .= $this->_handleHeaderLine($rawline);
            } // header line
        } // foreach ($rawdata as $rawline)
        $renderer->doc .= $mytable;
        return true;
    } // if ($mode == 'xhtml' && strlen($data[0]) > 1)
    return false;
} // function render

```

```

/**
    handle tags for a new table line
    @param $linedata    the line to be interpreted
    @return HTML code
**/
function _handleNewTableLine($linedata)
{
    $my2c = '';
    if (!$this->lineHasEndTag) {
        $my2c .= "    </tr>\n";
        $this->lineHasEndTag = true;
        $this->lineHasStartTag = false;
    } //if (!$this->lineHasEndTag)
    if ($this->tableHasHeader) {
        $my2c .= "    </thead>\n";
        $this->tableHasHeader = false;
    } // if ($this->tableHasHeader)
    if (!$this->bodyHasStartTag) {
        $my2c .= "    <tbody>\n";
        $this->bodyHasStartTag = true;
    } // if (!$this->bodyHasStartTag)
    if (!$this->lineHasStartTag) {
        $my2c .= "    <tr " . substr($linedata, 2) . ">\n";
        $this->lineHasStartTag = true;
        $this->lineHasEndTag = false;
    } // if (!$this->lineHasStartTag)
    return $my2c;
} // function _handleNewTableLine

/**
    handle start of table
    @param $linedata    the line to be interpreted
    @return HTML code
**/
function _handleTableStart($linedata)
{
    if (strlen(trim($linedata)) > 2)
        $fillInData = 'class="exttab1" ' . substr($linedata, 2);
    else
        $fillInData = 'class="exttab1"';
    $my2c = "\n<table $fillInData>\n";
    $this->lineHasStartTag = false;
    $this->bodyHasStartTag = false;
    $this->lineHasEndTag = true;
    return $my2c;
} // function _handleTableStart

/**
    handle table caption
    @param $linedata    the line to be interpreted

```

```

        @return HTML code
    **/
    function _handleTableCaption($linedata)
    {
        $my2c = '';
        if ($this->lineHasStartTag)
            $my2c .= "    </tr>\n";
        if ( ($pos = strpos($linedata, '|', 2)) !== false ) {
            $parsedData = $this->_parseDisplayData(trim(substr($linedata,
$pos + 1)));
            $my2c .= '    <caption ' . trim(substr($linedata, 2, $pos - 2)) .
'>' .
                "$parsedData</caption>\n";
        } // formatting data plus content
        else {
            $parsedData = $this->_parseDisplayData(trim(substr($linedata,
2)));
            $my2c .= "    <caption>$parsedData</caption>\n";
        } // content only
        $this->lineHasStartTag = false;
        $this->lineHasEndTag   = true;
        $my2c .= "    <tbody>\n";
        $this->bodyHasStartTag = true;
        return $my2c;
    } // function _handleTableCaption

    /**
        handles regular table line
        @param $linedata    the line to be interpreted
        @return HTML code
    **/
    function _handleRegularLine($linedata)
    {
        $my2c = '';
        if (!$this->lineHasStartTag) {
            if (!$this->bodyHasStartTag) {
                $my2c .= "    <tbody>\n";
                $this->bodyHasStartTag = true;
            } // if (!$this->bodyHasStartTag)
            $my2c .= "    <tr>\n";
            $this->lineHasStartTag = true;
            $this->lineHasEndTag   = false;
        } // if (!$lineHasStartTag)
        $subitems = explode('||', $linedata);
        foreach($subitems as $pick) {
            if ( ($pos = strpos($pick, '|', 1)) !== false ) {
                $parsedData = $this->_parseDisplayData(trim(substr($pick,
$pos + 1)));
                $my2c .= '        <td ' . trim(substr($pick, 1, $pos - 1)) .
'>' .
                    "$parsedData</td>\n";
            }
        }
    }

```

```

        } // formatting data plus content
    else {
        $parsedData = $this->_parseDisplayData(trim(substr($pick,
1)));
        $my2c .= '        <td>' . "$parsedData</td>\n";
    } // content only
} // foreach($subitems as $pick)
return $my2c;
} // function _handleRegularLine

/**
    handles table header entries
    @param $linedata    the line to be interpreted
    @return HTML code
**/
function _handleHeaderLine($linedata)
{
    $my2c = '';
    if (!$this->lineHasStartTag) {
        $my2c .= "    <thead>\n    <tr>\n";
        $this->lineHasStartTag = true;
        $this->lineHasEndTag    = false;
        $this->tableHasHeader  = true;
    } // if (!$lineHasStartTag)
    $subitems = explode('!!!', $linedata);
    foreach ($subitems as $pick) {
        if ( ($pos = strpos($pick, '|', 1)) !== false ) {
            $parsedData = $this->_parseDisplayData(trim(substr($pick,
$pos + 1)));
            // formatting data plus content
            $my2c .= '        <th>' . trim(substr($pick, 1, $pos - 1)) .
'>' .
                "$parsedData</th>\n";
        } // formatting data plus content
    else {
        // content only
        $parsedData = $this->_parseDisplayData(trim(substr($pick,
1)));
        $my2c .= '        <th>' . "$parsedData</th>\n";
    } // content only
} // foreach ($subitems as $pick)
return $my2c;
} // function _handleHeaderLine

/**
    this function parses data "internally"
**/
function _parseDisplayData($mydata)
{
    $retval = $mydata;
    $really = false;

```

```
$allowedTags = array('}}', ']]', "''");
foreach ($allowedTags as $tag) {
    $really = ($really or (strpos($mydata, $tag) > 0));
} // foreach ($allowedTags as $tag)
if ($really) {
    if (!function_exists('p_get_instructions'))
        require_once(DOKU_INC . 'inc/parserutils.php');
    $instruc = p_get_instructions($mydata);
    $myinfo = array();
    $retval = p_render('xhtml', $instruc, &$myinfo);
    $retval = str_replace('\n', '', $retval);
    $retval = substr($retval, 4, -5);
} // if ($really)
return $retval;
} // function _parseDisplayData

} // class syntax_plugin_exttab1

//Setup VIM: ex: et ts=4 enc=utf-8 :
```

From:

<http://wernerflamme.net/> - **Werners Wiki**

Permanent link:

<http://wernerflamme.net/doku.php?id=users:werner:exttab1source>

Last update: **2006-12-02 1524**

