

Скелет синтаксического плагина

[syntax.php](#)

```
<?php
/**
 * Plugin Skeleton: Displays "Hello World!"
 *
 * Syntax: <TEST> - will be replaced with "Hello World!"
 *
 * @license    GPL 2 (http://www.gnu.org/licenses/gpl.html)
 * @author     Christopher Smith <chris@jalakai.co.uk>
 */

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_test extends DokuWiki_Syntax_Plugin {

    /**
     * Get an associative array with plugin info.
     *
     * <p>
     * The returned array holds the following fields:
     * <dl>
     * <dt>author</dt><dd>Author of the plugin</dd>
     * <dt>email</dt><dd>Email address to contact the author</dd>
     * <dt>date</dt><dd>Last modified date of the plugin in
     * <tt>YYYY-MM-DD</tt> format</dd>
     * <dt>name</dt><dd>Name of the plugin</dd>
     * <dt>desc</dt><dd>Short description of the plugin (Text only)</dd>
     * <dt>url</dt><dd>Website with more information on the plugin
     * (eg. syntax description)</dd>
     * </dl>
     * @param none
     * @return Array Information about this plugin class.
     * @public
     * @static
     */
    function getInfo(){
        return array(
            'author' => 'me',
```

```
        'email'    => 'me@somewhere.com',
        'date'     => '20yy-mm-dd',
        'name'     => 'Test Plugin',
        'desc'     => 'Testing 1, 2, 3...',
        'url'      => 'http://www.dokuwiki.org/plugin:test',
    );
}

/**
 * Get the type of syntax this plugin defines.
 *
 * @param none
 * @return String <tt>'substitution'</tt> (i.e. 'substitution').
 * @public
 * @static
 */
function getType(){
    return 'substitution';
}

/**
 * What kind of syntax do we allow (optional)
 */
// function getAllowedTypes() {
//     return array();
// }

/**
 * Define how this plugin is handled regarding paragraphs.
 *
 * * <p>
 * This method is important for correct XHTML nesting. It returns
 * one of the following values:
 * </p>
 * <dl>
 * <dt>normal</dt><dd>The plugin can be used inside paragraphs.</dd>
 * <dt>block</dt><dd>Open paragraphs need to be closed before
 * plugin output.</dd>
 * <dt>stack</dt><dd>Special case: Plugin wraps other
 paragraphs.</dd>
 * </dl>
 * @param none
 * @return String <tt>'block'</tt>.
 * @public
 * @static
 */
// function getPType(){
//     return 'normal';
// }
```

```
/**
 * Where to sort in?
 *
 * @param none
 * @return Integer <tt>6</tt>.
 * @public
 * @static
 */
function getSort(){
    return 999;
}

/**
 * Connect lookup pattern to lexer.
 *
 * @param $aMode String The desired rendermode.
 * @return none
 * @public
 * @see render()
 */
function connectTo($mode) {
    $this->Lexer->addSpecialPattern('<TEST>', $mode, 'plugin_test');
//    $this->Lexer->addEntryPattern('<TEST>', $mode, 'plugin_test');
}

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

/**
 * Handler to prepare matched data for the rendering process.
 *
 * <p>
 * The <tt>$aState</tt> parameter gives the type of pattern
 * which triggered the call to this method:
 * </p>
 * <dl>
 * <dt>DOKU_LEXER_ENTER</dt>
 * <dd>a pattern set by <tt>addEntryPattern()</tt></dd>
 * <dt>DOKU_LEXER_MATCHED</dt>
 * <dd>a pattern set by <tt>addPattern()</tt></dd>
 * <dt>DOKU_LEXER_EXIT</dt>
 * <dd>a pattern set by <tt>addExitPattern()</tt></dd>
 * <dt>DOKU_LEXER_SPECIAL</dt>
 * <dd>a pattern set by <tt>addSpecialPattern()</tt></dd>
 * <dt>DOKU_LEXER_UNMATCHED</dt>
 * <dd>ordinary text encountered within the plugin's syntax mode
 * which doesn't match any pattern.</dd>
 * </dl>
```

```
* @param $aMatch String The text matched by the patterns.
* @param $aState Integer The lexer state for the match.
* @param $aPos Integer The character position of the matched text.
* @param $aHandler Object Reference to the Doku_Handler object.
* @return Integer The current lexer state for the match.
* @public
* @see render()
* @static
*/
function handle($match, $state, $pos, &$amp;handler){
    switch ($state) {
        case DOKU_LEXER_ENTER :
            break;
        case DOKU_LEXER_MATCHED :
            break;
        case DOKU_LEXER_UNMATCHED :
            break;
        case DOKU_LEXER_EXIT :
            break;
        case DOKU_LEXER_SPECIAL :
            break;
    }
    return array();
}

/**
 * Handle the actual output creation.
 *
 * <p>
 * The method checks for the given <tt>$aFormat</tt> and returns
 * <tt>FALSE</tt> when a format isn't supported. <tt>$aRenderer</tt>
 * contains a reference to the renderer object which is currently
 * handling the rendering. The contents of <tt>$aData</tt> is the
 * return value of the <tt>handle()</tt> method.
 * </p>
 * @param $aFormat String The output format to generate.
 * @param $aRenderer Object A reference to the renderer object.
 * @param $aData Array The data created by the <tt>handle()</tt>
 * method.
 * @return Boolean <tt>TRUE</tt> if rendered successfully, or
 * <tt>FALSE</tt> otherwise.
 * @public
 * @see handle()
 */
function render($mode, &$amp;renderer, $data) {
    if($mode == 'xhtml'){
        $renderer->doc .= "Hello World!";           // ptype =
'normal'
//          $renderer->doc .= "<p>Hello World!</p>";       // ptype =
'block'
```

```
        return true;
    }
    return false;
}

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

Замечание

Имя этого плагина — «test». You can see that by the name of the class: `syntax_plugin_test` — everything following `syntax_plugin_` is taken as the name of the plugin.

Т. к. имя плагина — «test», то и храниться он должен в папке `lib/plugins/test` для автоматического определения движком «ДокуВики». Имя файла должно быть «`syntax.php`».

From:

<https://wwoss.ru/> - worldwide open-source software

Permanent link:

https://wwoss.ru/doku.php?id=wiki:devel:syntax_plugin_skeleton&rev=1724651690

Last update: **2024/08/26 08:54**

