# Table of Contents

# Wysiwyg Plugin

Translator framework for WYSIWYG editors

Support for the integration of WYSIWYG (What-You-See-Is-What-You-Get) editors. On its own, the only thing this plugin gives you is a stand-alone HTML to TML (TWiki Markup Language) translator script. For WYSIWYG editing you will also need to install a specific editor package such as TinyMCEPlugin.

This plugin provides a generic framework that supports editing of topics using any browser-based HTML editor. It works by transforming TML (TWiki Markup Language) into HTML for the editor, and then transforming HTML back into TML on save.

## Features

- Supports the input of malformed HTML
- Full round-trip (TML -> XHTML -> TML)
- Framework is editor-agnostic

## Details

### What's in the package

The package includes the following pieces:

- TML (TWiki Markup Language) to HTML translator
- HTML to TML translator (with stand-alone script)
- Generic TWiki plugin for automating the translation during editing

### How it works

The plugin works by translating the topic text into HTML when someone edits a topic. The HTML is then fed to the WYSIWYG editor. On save, the edited HTML is run through the reverse translation before saving to the topic. TML is used in preference to HTML in the stored topic wherever possible, though HTML may be used if the translator can't find a suitable TML equivalent.

The default rendering that TWiki uses to generate HTML for display in browsers is 'lossy' - information in the TML is lost in the HTML output, and a round-trip (recovering the original TML from the HTML) is impossible. To solve this problem the plugin instead uses its own translation of TML to XHTML. The generated XHTML is annotated with CSS classes that support the accurate recovery of the original TML.

*Before you ask the obvious question, yes, the translator **could** be used to replace the TWiki rendering pipeline for generating HTML pages. In fact, the translator is taken almost directly from the implementation of the rendering pipeline for the TWiki-4 release*

Translation of the HTML back to TML uses the CPAN:HTML::Parser. This parser is used in preference to a more modern XML parser, because the WYSIWYG editor may not generate fully compliant XHTML. A strict parser would risk losing content. HTML::Parser is better at handling malformed HTML.

There is also the advantage that the translator can be used to **import** HTML from other sources - for example, existing web pages. Due to the simple nature of TML and the potential complexity of web pages, this translation is often lossy - i.e. there will be HTML features that can be entered by editors that will be lost in this translation step. This is especially noticeable with HTML tables.

## Using the translators from Perl scripts

Both translators can be used directly from Perl scripts, for example to build your own stand-alone translators.

A stand-alone convertor script for HTML to TML is included in the installation. It can be found in `tools/html2tml.pl`.

## Integrating a HTML Editor

The plugin can be used to integrate an HTML editor in a number of different ways.

1. The HTML for the content-to-be-edited can be generated directly in the standard edit template.
2. The HTML for the content-to-be-edited can be generated directly in a specialized edit template.
3. A URL can be used to fetch the content-to-be-edited from the server, for use in an IFRAME.
4. REST handlers can be called from Javascript to convert content.

### Generating content directly in the standard edit template

This is the technique used by WYSIWYG editors that can sit on top of HTML textareas, such as TinyMCE. The topic content is pre-converted to HTML before inclusion in the standard edit template. These editors use plugins that have a `beforeEditHandler` and an `afterEditHandler`. These handlers are responsible for the conversion of topic text to HTML, and post-conversion of HTML back to TML.

1. User hits "edit".
2. Editor-specific plugin `beforeEditHandler` converts topic content to HTML by calling `TWiki::Plugins::WysiwygPlugin::TranslateTML2HTML`.
3. User edits and saves
4. Editor-specific plugin `afterEditHandler` converts HTML back to TML by calling `TWiki::Plugins::WysiwygPlugin::TranslateHTML2TML`.

- WysiwygPlugin should **not** be enabled in `configure`.
- `WYSIWYGPLUGIN_WYSIWYGSKIN` should **not** be set.
- Your plugin should set the `textareas_hijacked` context id, to signal to skins to suppress their textarea manipulation functions.

This is the recommended integration technique, if your editor can support it.

### Generating content directly in a specialized edit template

This technique is useful when the editor requires the topic content in a variety of different formats at the same time. In this scenario the editor uses a custom edit template. The WYSIWYG content is made available for instantiation in that template in a number of different formats. `WYSIWYGPLUGIN_WYSIWYGSKIN` **must** be set for this to work.

The flow of control is as follows:

1. User hits "edit" with the skin (or cover) set the same as `WYSIWYGPLUGIN_WYSIWYGSKIN`.
2. The WysiwygPlugin `beforeEditHandler` determines if the topic is WYSIWYG editable, and vetoes the edit if not by redirecting to the standard edit skin. the edit
3. The `edit` template containing the JS editor is instantiated.
4. The following variables are available for expansion in the template:
    - `%WYSIWYG_TEXT%` expands to the HTML of the content-to-be-edited. This is suitable for use in a `textarea`.

    - ♦ %JAVASCRIPT_TEXT% expands to the HTML of the content-to-be-edited in a javascript constant.
5. User edits and saves
6. The afterEditHandler in the WyswiygPlugin sees that wysiwyg_edit is set, which triggers the conversion back to TML.

- The HTML form in the edit template **must** include an <input called wysiwyg_edit and set it to 1, to trigger the conversion from HTML back to TML.
- WYSIWYGPLUGIN_WYSIWYGSKIN must be set to the name of the skin used for WYSIWYG editing. This is often the name of the editor e.g. xinha.

### Fetching content from a URL

In this scenario, the edit template is generated **without** the content-to-be-edited. The content is retrieved from the server using a URL e.g. from an IFRAME.

The flow of control is as follows:

1. As *Generating content directly in a specialized edit template*
2. As *Generating content directly in a specialized edit template*
3. As *Generating content directly in a specialized edit template*
4. When the document loads in the browser, the JS editor invokes a content URL (using an IFRAME or a XmlHttpRequest) to obtain the HTML document to be edited
    - ♦ The content URL is just a TWiki view URL with the wysiwyg_edit parameter set.
    - ♦ The WysiwygPlugin recognises the wysiwyg_edit parameter and uses the TML2HTML translator to prepare the text, which is then returned as text/plain to the browser.
    - ♦ Two TWiki variables, %OWEB% and %OTOPIC%, can be used in the content URL in the edit template to refer to the source topic for the content.
5. After edit handling is as for *Generating content directly in a specialized edit template*

## Other techniques

### Asynchronous saves

Editors can use XmlHttpRequest to perform saves, by POSTing to the TWiki save script with the wysiwyg_edit parameter set to 1. This parameter tells the beforeSaveHandler in the WysiwygPlugin to convert the content back to TML. See CommandAndCGIScripts for details of the other parameters to the save script.

Once the save script has completed it responds with a redirect, either to an Oops page if the save failed, or to the appropriate post-save URL (usually a view). The editor must be ready to handle this redirect.

### Handling Attachments

Attachment uploads can be handled by URL requests from the editor template to the TWiki upload script. The upload script normally redirects to the containing topic; a behavior that you usually don't want in an editor! There are two ways to handle this:

- If the uploads are done in an IFRAME or via XmlHttpRequest, then the 302 redirect at the end of the upload can simply be ignored.
- You can pass noredirect to the upload script to suppress the redirect. In this case you will get a text/plain response of OK followed by a message if everything went well, or an error message if it did not.

### REST handlers

If you are confident in Javascript you can use REST handlers with `XmlHttpRequest` to convert content from TML to HTML and back again.

The plugin defines the following REST handlers:

```
.../rest/WysiwygPlugin/html2tml?topic=Web.Topic;text=htmltexttotranslate
```

Converts the HTML text to TML. `topic` **must** be specified.

```
.../rest/WysiwygPlugin/tml2html?topic=Web.Topic;text=tmltexttotranslate
```

Converts the TML text to HTML. `topic` **must** be specified. The response is a `text/plain` page of converted content.

# Plugin Installation Instructions

This plugin is pre-installed. TWiki administrators can upgrade the plugin as needed on the TWiki server.

Show details Hide details

- For an ***automated installation***, run the configure script and follow "Find More Extensions" in the in the ***Extensions*** section.
  - ♦ See the installation supplement on TWiki.org for more information.

- Or, follow these ***manual installation*** steps:
  - ♦ Download the ZIP file from the extension home on twiki.org (see below).
  - ♦ Unzip **`WysiwygPlugin.zip`** in your twiki installation directory.
  - ♦ Set the ownership of the extracted directories and files to the webserver user.
  - ♦ Install the dependencies (if any).

- Plugin ***configuration and testing***:
  - ♦ Run the configure script and enable the plugin in the ***Plugins*** section.
  - ♦ Configure additional plugin settings in the ***Extensions*** section if needed.
  - ♦ Test if the installation was successful using the examples provided.

# Plugin Configuration Settings

## Translator control

### WYSIWYG_EXCLUDE - Prevent WYSIWYG editing

The **global** preference setting `WYSIWYG_EXCLUDE` can be set to make the plugin sensitive to what is in a topic, before allowing it to be edited. The comma separated list to fall back to text edit can include:

- `html` - HTML tags (e.g. `<div>`, not including `<br>`), or
- `variables` - simple variables (e.g. `%SOMEVAR%`) or
- `calls` - variables with parameters e.g. `%SOMECALL{...}%`
- `pre` - pre-formatted blocks (`<pre>`)
- `comments` - HTML comments (`<!-- ... -->`)
- `script` - inline HTML Script tags - *default*
- `style` - inline CSS style tags - *default*

- `table` - inline HTML tables (`<table ..>`. TML tables are not excluded)

If the plugin detects an excluded construct in the topic, it will refuse to allow the edit and will redirect to the default editor.

#### WYSIWYG_EDITABLE_CALLS - Exceptions to WYSIWYG_EXCLUDE

If you excluded `calls` in `WYSIWYG_EXCLUDE`, you can still define a subset of variables that do **not** block edits. this is done in the **global** preference setting `WYSIWYG_EDITABLE_CALLS`, which should be a list of variable names separated by vertical bars, with no spaces, e.g: `* Set WYSIWYG_EDITABLE_CALLS = COMMENT|CALENDAR|INCLUDE`

You should set `WYSIWYG_EXCLUDE` and `WYSIWYG_EDITABLE_CALLS` in TWikiPreferences, or in WebPreferences for each web.

### WYSIWYGPLUGIN_PROTECT_EXISTING_TAGS - Protect specific tags originally in the topic text

The `WYSIWYGPLUGIN_PROTECT_EXISTING_TAGS` preference tells the translator that certain HTML tags which were originally in the topic text should *remain* as HTML tags; the translator will not try to convert them to TML. This protects the tags themselves, and not the contents enclosed between the `<tag>` and `</tag>`

The default setting for this preference is defined within the plugin. It corresponds to `div, span`.

This feature may be disabled by setting the preference to a single comma. This does *not* guarantee that HTML markup will be removed; the conversion of HTML tags to TML markup remains subject to the other controls provided by the WysiwygPlugin, including the `WYSIWYGPLUGIN_STICKYBITS` preference, `<sticky>` blocks, `<literal>` blocks and the rules applied to tables and lists.

### WYSIWYGPLUGIN_PROTECT_TAG_BLOCKS - Protect specific tag blocks originally in the topic text

The `WYSIWYGPLUGIN_PROTECT_TAG_BLOCKS` preference tells the translator that certain HTML tag blocks which were originally in the topic text should *remain* as HTML blocks; the translator will not try to convert them to TML.

The default setting for this preference is defined within the plugin. It corresponds to `script, style`.

As an example, individual html tables can be protected by surrounding them with `<sticky> .. </sticky>` block. However,if you want to have all =<table> markup preserved as entered into topics by default, rather than subject to WYSIWYG editing, add =table to this list, and =<table> markup will become automatically sticky.

This feature may be disabled by setting the preference to a single comma.

### WYSIWYGPLUGIN_STICKYBITS - Protect tags based upon their arguments

You can define the global preference `WYSIWYGPLUGIN_STICKYBITS` to stop the plugin from ever trying to convert specific HTML tags into TML when certain specific attributes are present on the tag. This is most useful when you have styling or alignment information in tags that must be preserved.

This preference setting is used to tell the translator which attributes, when present on a tag, make it "stick" i.e. block conversion back to TML.

For example, setting it to `table=background,lang;tr=valign` will stop the translator from trying to convert any `table` tag that has `background` or `lang` attributes, and any `tr` tag that has a `valign` attribute back to TWiki `| table | column |` markup (regardless of where that `table` tag comes from).

This setting is used only after the page has been processed by the editor. If the editor does not support a particular tag or attribute and the editor corrupts the tag, this setting will not be helpful. It is only used to prevent an HTML tag from being converted back to TML.

Format of the setting is `tag1=attrib,attrib;tag2=attrib`. Attributes delimited by comma, and tags delimited by semicolon.

- The left side of the equal sign is the tag.
- The right side of the equal sign is a comma delimited list of attributes to be matched.

If a matching tag is found, that matches any of the attributes listed, the tag will not be converted back to TML. You can use perl regular expressions to match tag and attribute names, so `.*=id,on.*` will ensure that any tag with an id or `on*` event handler is kept as HTML.

The default setting for this preference are hard coded in the plugin. If you wish to change the settings, the following list is the default setting coded in the plugin:

```
* Set WYSIWYGPLUGIN_STICKYBITS =
     (?!IMG).*=id,lang,title,dir,on.*;
     A=accesskey,coords,shape,target;
     BDO=dir;
     BR=clear;
     COL=char,charoff,span,valign,width;
     COLGROUP=align,char,charoff,span,valign,width;
     DIR=compact;
     DIV=align,style;
     DL=compact;
     FONT=size,face;
     H[0-9]=align;
     HR=align,noshade,size,width;
     LEGEND=accesskey,align;
     LI=value;
     OL=compact,start,type;
     P=align;
     PARAM=name,type,value,valuetype;
     PRE=width;
     Q=cite;
     TABLE=align,bgcolor,frame,rules,summary,width;
     TBODY=align,char,charoff,valign;
     TD=abbr,align,axis,bgcolor,char,charoff,headers,height,nowrap,rowspan,scope,valign
     TFOOT=align,char,charoff,valign;
     TH=abbr,align,axis,bgcolor,char,charoff,height,nowrap,rowspan,scope,valign,width,h
     THEAD=align,char,charoff,valign;
     TR=bgcolor,char,charoff,valign;
     UL=compact,type
```

If you edit using the plain-text editor, you can use the <sticky>..</sticky> tags to delimit HTML (or TML) that you do **not** want to be WYSIWYG edited.

**Implementors note:** If you are using your own before/after edit handlers, you can call `TWiki::Plugins::WysiwygPlugin::isWysiwygEditable()` to check these controls.

# Known issues

## Incompatible with "non-standard" syntax

The WysiwygPlugin is incompatible with plugins that expand non-standard syntax e.g. TWiki:Plugins.MathModePlugin (WysiwygPlugin)

Plugins that extend the syntax using TWiki variables, such as `%MYVARIABLE%`, should work fine.

**Implementors note:** Plugins that use XML-like tags may call `TWiki::Plugins::WysiwygPlugin::addXMLTag()` from their `initPlugin` handlers to make the WysiwygPlugin protect the content between XML-like tags, just like it does for TWiki variables.

## Overlapping styles

Because TWiki uses a "best guess" approach to some formatting, it allows overlapping of tags in a way forbidden by HTML, and it is impossible to guarantee 100% that formatting in the original TWiki document will still be there when the same document is loaded and then saved through the WysiwygPlugin. The most obvious case of this is to do with styles. For example, the sentence

```
*bold _bold-italic* italic_
```

is legal in TML, but in HTML is represented by

```
<strong>bold <em>bold-italic</em></strong> <em>italic</em>
```

which gets translated back to TML as

```
*bold _bold-italic_* _italic_
```

which is correct by construction, but does not render correctly in TWiki. This problem is unfortunately unavoidable due to the way TML works.

## Rowspan processing needs TablePlugin

The WysiwygPlugin is able to convert tables with cells that span rows into TML. This requires syntax provided by the TablePlugin (that is, the `|  ^  |` markup). the WysiwygPlugin will therefore only perform row-span related conversion if the TablePlugin is enabled. The TablePlugin is enabled by default and hence the WysiwygPlugin converts tables with cells that span rows between TML and HTML by default.

If the TablePlugin is **not** enabled, then TML table cells containing only `^` are not converted to rowspans, and HTML tables containing rowspans are not converted to TML.

## TinyMCEPlugin integration

- Anchors are not handled by WysiwygPlugin
- WysiwygPlugin fails to roundtrip tables with align="center", border attributes, etc.
    - Description: Sometimes tables will fail to be converted to TML syntax (will stay as HTML) because there are attributes on the table (such as alignment or border decorations) that the WysiwygPlugin does not know how to preserve. If such attributes are necessary, please use VarTABLE instead.
    - Work-around:
        - Click inside the problematic table

◊ Click the table toolbar button (usually used to create a new table)
◊ With the exception of `Cols` and `Rows`, delete/reset all content from the fields on the 'General' and 'Advanced' tabs.
◊ Write a VarTABLE variable above the offending table that adds the desired attributes safely

# Plugin Info

| | |
|---|---|
| Author: | Crawford Currie, TWiki Contributors |
| Copyright | © 2005 ILOG http://www.ilog.fr<br>© 2007 Twiki, Inc.<br>© 2008-2012 Foswiki Contributors<br>© 2005-2018 Peter Thoeny and TWiki:TWiki.TWikiContributor |
| License | GPL (Gnu General Public License) |
| Sponsors: | ILOG, Carrier Corporation, Twiki Inc, Wave Systems Corp |
| Plugin Version: | 2018-07-06 |

Show Change History Hide Change History

| | | | |
|---|---|---|---|
| 2018-07-06: | TWikibug:Item7841: Copyright update to 2018 | | |
| 2016-01-09: | TWikibug:Item7708: Copyright update to 2016 | | |
| 2015-02-16: | TWikibug:Item7604: Switch from GPL v2 to v3 | | |
| 2014-12-10: | TWikibug:Item7594: Disable WYSIWYG editor if | >> ... << | table syntax is present -- TWiki:Main.PeterThoeny |
| 2014-02-19: | TWikibug:Item7436: WYSIWYG editor prepends Â to spaces inside verbatim and sticky tags -- TWiki:Main.HideyoImazu | | |
| 2013-09-18: | TWikibug:Item7338: Import WYSIWYG plugin fixes 2008-2013 -- TWiki:Main.PeterThoeny | | |
| 1.1.16 (16 May 2013) | Fix problem with entities being expanded to unrepresentable characters | | |
| 1.1.15 (16 Dec 2012) | Minor perlcritic coding change | | |
| (21 Dec 2012) | Changing a wikiword should not require visiting the TinyMCE link dialog. | | |
| 1.1.14 (28 Nov 2012) | Clean up hex markers left behind by TinyMCEPlugin<br>Convert to perl version strings<br>Preserve Square bracket links | | |
| 1.1.13 (5 Jun 2012) | Backslash line continuation incorrectly requires a space dlimitier.<br>Extraneous hex 03 characters replace % in nested tags | | |
| 1.1.12 (30 May 2012) | Fix for Item10089 caused link corruption in certain cases. | | |
| 1.1.11 (22 May 2012) | Compile errors with perl 5.8.8 due to use of new regular expression features. | | |
| 1.1.10 (21 May 2012) | Better fix for <div tags, also cover <blockquote tags.<br>Unable to position cursor above initial verbatim, pre and blockquote blocks<br>Syntax for indent was added earlier, but missed from release notes. | | |
| 1.1.9 (test release) | | | |

| | |
|---|---|
| |  tags are wrapped in <p> tags. TMCE auto closes them. |
| 1.1.8 (test release) | Fix for Item11814 breaks %ATTACHURL variable in link. |
| 1.1.7 (test release) | Wysiwyg removes <br /> tags at end of lines. |
| 1.1.6 (test release) | Process TML links as HTML links<br>TMCE should honor NOAUTOLINK preference and noautolink blocks.<br>Allow TMCE to recognize TML links as HTML links.<br>Protect Glue format markup in variables.<br>Don't merge verbatim blocks if they have different classes.<br>Handle colors implemented using CSS classes.<br>Preserve URI Encoding in links.<br>WikiWords escaped with ! are show as linking.<br>TMCE is failing to protect newlines. |
| 1.1.5 (06 Apr 2012) | protect inline script and style tags from wysiwyg.<br>protect tags inside pre.<br>Protect TML tables from corrupting embedded html markup.<br>Prevent #Anchors from being wrapped to the previous line.<br>Prevent corruption of HTML tables containin blank lines. |
| 1.1.4 | support pass-through of DEL and INS tags |
| 1.1.3 (08 Nov 2011) | Fix WysiwygPlugin eating newlines inside `%MACRO{...}` expressions (Michael Tempest) |
| 1.1.2 (11 Apr 2011) | Version released with TWiki 1.1.3. Only a minor change related to how the plugin is being upgraded |
| 1.1.1 (19 Jan 2011) | Switch to x.y.z release numbering<br>Try to use Variables in the src URLs of images with titles<br>Fix attachments REST handler to deal with topics named with international characters<br>Protect div and span tags with `style` attributes |
| 28 Jun 2010 | Fix conversion between character encodings. Any characters may be entered in the WYSIWYG editor, regardless of the site's encoding. Where possible, they are converted to the site encoding, otherwise they become entities.<br>Fix cursor-movement problems on Mozilla browsers (introduced by )<br>Can now place cursor into empty list-item<br>Can now move cursor above a table at start of a topic and below a table at the end of the topic<br>Protect <br /> tags at the end of a protected line (e.g. in a variable parameter)<br>Protect newlines within a <pre> block<br>Keep the content of <big> and <var> tags<br>Fix stand-alone (command-line) HTML-to-TML conversion |
| 21 May 2010 | Use Wysiwyg transition to remove usually unwanted paragraph html tags in table cells, which |

| | |
|---|---|
| | are otherwise impossible to remove in TinyMCE up to at least 3.3.6<br>Fix problem where Wysiwyg transition merges two consecutive lists (a result of work on ) |
| 2010-05-17: | TWikibug:Item6433 - more doc fixes; replacing TWIKIWEB with SYSTEMWEB |
| 2010-04-24: | TWikibug:Item6433 - doc improvements, prepare for TWiki 5.0 release |
| 17 Jan 2010 | ATTACHFILESIZELIMIT check fails confusingly if value is "0 " |
| 18 Dec 2009 | move code out of the plugin module to accelerate loading |
| 18 Nov 2009 | Convert tables with cells that span rows |
| 22 Oct 2009 | Protect div style= by default |
| 18 Sep 2009 | Prevent dataloss when saving a topic in Wysiwyg where there are a pair of sticky tags inside verbatim tags |
| 28 Jun 2009 | Protect XML tags registered by plugins, and not just the content between them (Michael Tempest) |
| 06 Jun 2009 | Correct dependency on HTML::Parser (Will Norris). Correct processing of colour and typewriter-text in several situations, include application to bold text and table cells (Michael Tempest). Remove unwanted extra <sticky> tags (Michael Tempest). Let plugins register XML tags that should be protected like variables |
| 10 Apr 2009 | fixed colour handling |
| 03 Dec 2008 | fixed empty bullet list problem. |
| 22 Oct 2008 | Fixed TWikibug:Item5961 (emphasis), TWikibug:Item6089 (backslash in verbatim) |
| 07 Aug 2008 | Fixed TWikibug:Item5707 (mod_perl) |
| 03 Aug 2008 | TWiki 4.2.1 release version |
| 25 May 2008 | TWikibug:Item5457: TWikibug:Item5528: TWikibug:Item5626: using a debug simulation, I believe I have finally fixed all the complexities of using international character sets with the translator. |
| 13 Apr 2008 | TWikibug:Item4946: TWikibug:Item5530: I think I have finally fixed non-iso-8859-1 character sets. Painful. TWikibug:Item5393: removed spurious DIV generated by IE inside LI tags |
| 31 Mar 2008 | TWikibug:Item5314: TWikibug:Item5457: Fixed pickaxe mode for editing UTF-8. Characters above 255 are converted to entitites, which is a bit of a PITA, but at least it no longer corrupts topics. |
| 28 Mar 2008 | TWikibug:Item5294: fixed angle brackets in plain text and promoted sticky to be higher priority than any other tag, solving several problems in one go |
| 24 Jan 2008 | TWikibug:Item5257: remove extra spaces at end of Set lines |
| 20 Dec 2007 | TWikibug:Item5022: made TT font size same as verbatim. Had to add a new style to do it, as TMCE didn't want to play with TT or CODE tags. TWikibug:Item5138: post-conversion of 8-bit |

| | |
|---|---|
| | entities to characters to aid searching etc. |
| 19 Dec 2007 | TWikibug:Item4836: make the parser tolerant of META, so pasting OO docs works<br>TWikibug:Item4969: autoclose BR and HR tags<br>TWikibug:Item5132: fixed IMG tags<br>TWikibug:Item5076: fixed line-sensitive TML embedded in tables |
| 08 Nov 2007 | TWikibug:Item4923: fixed blocking of table conversion due to empty attributes<br>TWikibug:Item4936: An em embedded in an em was getting eaten TWikibug:Item4817: added typewriter text button TWikibug:Item4850: added font colour controls TWikibug:Item4645: added REST handlers for upload and fetching lists of attachments |
| 02 Nov 2007 | TWikibug:Item4903: corrected over-enthusiastive interpretation of ! as an escape |
| 21 Oct 2007 | TWikibug:Item4788: fixed unbalanced protect, which could cause loss of protected status<br>TWikibug:Item4811: noautolink **looks** like an HTML construct but in fact is not; the tag is infact an "on-off" switch and does not imply any HTML structure, so cannot be converted to a DIV or a span, so has to be removed. TWikibug:Item4747: added <sticky> to try to overcome limitations in translation TWikibug:Item4831: added increased flexibility in deciding what HTML get converted to TML, and what does not. Analysed all the HTML4 tags to establish initial settings.<br>TWikibug:Item4847: don't call non-existent function with older HTML::Parser releases<br>TWikibug:Item4844: Saving a table from IE didn't convert it back to TML<br>TWikibug:Item4855: table rows generated from TWiki variables were being eaten |
| 06 Oct 2007 | TWikibug:Item4700: fixed colspans<br>TWikibug:Item4701: removed extra line between %TABLE and the table TWikibug:Item4705: fixed spacing around literal and verbatim blocks<br>TWikibug:Item4706: merge adjacent verbatim blocks separated only by whitespace<br>TWikibug:Item4712: fixed eating of noautolink and literal TWikibug:Item4763: list items spanning multiple lines fixed<br>TWikibug:Item4867: released tml2html |
| 17 Sep 2007 | TWikibug:Item4647: TWikibug:Item4652: problems related to DIV fixed.<br>TWikibug:Item4653: fixed multi-line twiki variables |
| 16 Sep 2007 | TWikibug:Item4630: polished up the way the secret string is done, to ensure synch between perl and JS. Item4622: added UTF-8 handling steps that fixup malformed UTF8 strings before presenting them to the editor (saves Moz) and stops the editor |

| | |
|---|---|
| | passing them back to TWiki (saves IE). Removed extra entity decoding steps that were causing problems. TWikibug:Item4629: fixed issues with verbatim, highlighted by previous mangling of this topic |
| 13 Sep 2007 | TWikibug:Item4613 cleaned up spurious message when navigating away TWikibug:Item4615 fixed incorrect rendering of emphasis next to br |
| 12 Sep 2007 | TWikibug:Item4604 Fixes to REST handler, and add ability to trigger HTML2TML conversion from a content comment alone (required for some editors) TWikibug:Item4588 fixes to conversion of double-character emphases |
| 07 Sep 2007 | TWikibug:Item4503 excess empty lines TWikibug:Item4486 no toc headers with unofficial syntax TWikibug:Item4560: empty lines lost TWikibug:Item4566: corrupted table on save TWikibug:Item4550 section tags being eaten |
| 04 Sep 2007 | TWikibug:Item4534 TWikibug:Item4535 fixed |
| | See Subversion logs for earlier revisions |

Dependencies:

| Name | Version | Description |
|---|---|---|
| HTML::Parser | >=3.28 | Required. Available from CPAN. |
| HTML::Entities | >=1.25 | Required. Available from CPAN. |
| Encode | >=2.01 | Required. |

| | |
|---|---|
| Plugin Home: | http://twiki.org/cgi-bin/view/Plugins/WysiwygPlugin |
| Feedback: | http://twiki.org/cgi-bin/view/Plugins/WysiwygPluginDev |
| Appraisal: | http://twiki.org/cgi-bin/view/Plugins/WysiwygPluginAppraisal |

***Related Topics:*** WysiwygPluginSettings, WysiwygPluginTopicLister, TinyMCEPlugin, TWikiPreferences, TWikiPlugins

---

This topic: TWiki > WysiwygPlugin
Topic revision: r1 - 2018-07-06 - TWikiContributor


or Ideas, requests, problems regarding TWiki? use Discourse or Send feedback
***Note:*** Please contribute updates to this topic on TWiki.org at TWiki:TWiki.WysiwygPlugin