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 (topic markup language) translator script. For WYSIWYG editing in Foswiki and Foswiki, you will also need to install a specific editor package such as Foswiki:Extensions.TinyMCEPlugin.

This plugin provides a generic framework that supports editing of topics using any browser-based HTML editor. It works by transforming TML 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 (topic markup language) to HTML translator
- HTML to TML translator (with stand-alone script)
- Generic 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 Foswiki 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 Foswiki 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. CPAN: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 specialised 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 `Foswiki::Plugins::WysiwygPlugin::TranslateTML2HTML`.
3. User edits and saves
4. Editor-specific plugin `afterEditHandler` converts HTML back to TML by calling `Foswiki::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 specialised 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 vetos 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 macros 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 specialised edit template*
2. As *Generating content directly in a specialised edit template*
3. As *Generating content directly in a specialised 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 Foswiki 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 macros, %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 specialised edit template*

Other techniques

Asynchronous saves

Editors can use XmlHttpRequest to perform saves, by POSTing to the Foswiki 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 Foswiki upload script. The upload script normally redirects to the containing topic; a behaviour 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

You do not need to install anything in the browser to use this extension. The following instructions are for the administrator who installs the extension on the server.

Open configure, and open the "Extensions" section. Use "Find More Extensions" to get a list of available extensions. Select "Install".

If you have any problems, or if the extension isn't available in `configure`, then you can still install manually from the command-line. See http://foswiki.org/Support/ManuallyInstallingExtensions for more help.

Plugin Configuration Settings

Translator control

WYSIWYG_EXCLUDE - Prevent WYSIWYG editing

⚠ This is not supported currently by the TinyMCE Editor. This setting is not used.

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. You can set it up to veto an edit if the topic contains:

- `html` - HTML tags (e.g. `<div>`, not including
), or
- `macros` - simple macros (e.g. `%VAR%`) or
- `calls` - macros with parameters e.g. `%MACRO{...}%`
- `pre` blocks (`<pre>`)
- HTML `comments` (`<!-- ... -->`)

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 macros that do **not** block edits. this is done in the **global** preference setting `WYSIWYG_EDITABLE_CALLS`, which should be a list of

macro 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 SitePreferences, or in WebPreferences for each web.

WYSIWYGPLUGIN_STICKYBITS - Protext 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 Foswiki `| table | column |` markup.

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 = .*=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,wid
```

```
TFOOT=align,char,charoff,valign;
TH=abbr,align,axis,bgcolor,char,charoff,height,nowrap,rowspan,scope,valign,width,heade
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 `Foswiki::Plugins::WysiwygPlugin::isWysiwygEditable()` to check these controls.

Known issues

Incompatible with "non-standard" syntax

WysiwygPlugin is incompatible with plugins that expand non-standard syntax e.g. Foswiki:Extensions.MathModePlugin (WysiwygPlugin)

Plugins that extend the syntax using macros, such as `%MYMACRO%`, should work fine.

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

Overlapping styles

Because Foswiki 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 formating in the original Foswiki 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 Foswiki. This problem is unfortunately unavoidable due to the way TML works.

Flattening of `<div>` tags

The "Normal" format command in the current release of TinyMCEPlugin depends on WysiwygPlugin flattening `<div>` tags. If you have a customised `WYSIWYGPLUGIN_STICKYBITS` set, check that plain `<div>` tags will be flattened if you expect this feature to work. It is expected that this requirement will be removed in a future release of TinyMCEPlugin.

Plugin Info

This plugin is brought to you by a WikiRing partner - working together to improve your wiki experience!

Many thanks to the following sponsors for supporting this work:

- ILOG
- Carrier Corporation
- TWIKI.NET

Author:	Crawford Currie
Copyright	© ILOG 2005 http://www.ilog.fr
License	GPL (Gnu General Public License)
Version:	6068 (2010-01-17)
Release:	17 Jan 2010
Change History:	
17 Jan 2010	Foswikitask:Item2337: ATTACHFILESIZELIMIT check fails confusingly if value is "0 "
08 Dec 2009	Foswikitask:Item2447: Add TinyMCEPlugin's flattening requirement to Known Issues Foswikitask:Item2352: Use Foswiki's regex instead of a hard-coded regex, so that WysiwygPlugin will always "see" macros in the same way as Foswiki's renderer Foswikitask:Item2286: Cleaning up missing calls to Foswiki::finish()
22 Oct 2009	Foswikitask:Item2183: Protect div style= by default
18 Sep 2009	Foswikitask:Item1980: Prevent dataloss when saving a topic in Wysiwyg where there are a pair of sticky tags inside verbatim tags
28 Jun 2009	Foswikitask:Item1770: Protect XML tags registered by plugins, and not just the content between them (Michael Tempest)
06 Jun 2009	Foswikitask:Item1013: Correct dependency on HTML::Parser (Will Norris) Foswikitask:Item1397: Foswikitask:Item1535: Foswikitask:Item1666: Correct processing of colour and typewriter-text in several situations, include application to bold text and table cells (Michael Tempest) Foswikitask:Item1667: Remove unwanted extra <sticky> tags (Michael Tempest) Foswikitask:Item1674: Let plugins register XML tags that should be protected like macros
10 Apr 2009	Foswikitask:Item1394: fixed colour handling
03 Dec 2008	Foswikitask:Item6041: fixed empty bullet list problem. Foswiki version
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 TWikibug:Item4969: autoclose BR and HR tags TWikibug:Item5132: fixed IMG tags TWikibug:Item5076: fixed line-sensitive TML embedded in tables
8 Nov 2007	TWikibug:Item4923: fixed blocking of table conversion due to empty attributes 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
2 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 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. TWikibug:Item4847: don't call non-existent function with older HTML::Parser releases TWikibug:Item4844: Saving a table from IE didn't convert it back to TML TWikibug:Item4855: table rows generated from TWiki variables were being eaten
6 Oct 2007	TWikibug:Item4700: fixed colspans TWikibug:Item4701: removed extra line between %TABLE and the table TWikibug:Item4705: fixed spacing around literal and verbatim blocks TWikibug:Item4706: merge adjacent verbatim blocks separated only by whitespace TWikibug:Item4712: fixed eating of noautolink and literal TWikibug:Item4763: list items spanning multiple lines fixed TWikibug:Item4867: released tml2html
17 Sep 2007	TWikibug:Item4647: TWikibug:Item4652: problems related to DIV fixed. 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
7 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
4 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://foswiki.org/Extensions/WysiwygPlugin
Support:	http://foswiki.org/Support/WysiwygPlugin

- System

- Log In

- **Toolbox**
- Users
- Groups
- Index
- Search
- Changes
- Notifications
- RSS Feed
- Statistics
- Preferences

- **User Reference**
- BeginnersStartHere
- TextFormattingRules
- Macros
- FormattedSearch
- QuerySearch
- DocumentGraphics
- SkinBrowser
- InstalledPlugins

- **Admin Maintenance**
- Reference Manual
- AdminToolsCategory
- InterWikis
- ManagingWebs
- SiteTools
- DefaultPreferences
- WebPreferences

- **Categories**
- Admin Documentation
- Admin Tools
- Developer Doc
- User Documentation
- User Tools

- **Webs**
- Public
- System