



Templates and custom formats

Templates and custom formats

amusewiki.org

Edit site amw

Restricted configuration variables -

Restricted format options -

Directory with templates in the site repository, e.g. "templates"
(ASCII only). Do not set it unless you know exactly what you are
doing

 Use LuaLaTeX instead of XeLaTeX (slower)

 The logo has the site name on itself

PDF logo

The PDF logo value can be either an absolute path or just a filename if you choose to install it in the TeXlive tree. You can do so with the following commands. The last one checks if the file can be found.

```
mkdir -p `kpsewhich -var-value TEXFRONT`/tex/generic/amusewiki/data
cp mylogo.pdf `kpsewhich -var-value TEXFRONT`/tex/generic/amusewiki/data
texhash `kpsewhich -var-value TEXFRONT`
kpsewhich mylogo.pdf
```

I.e., you can test of the existence of `safe_options.format_id.CODE` and branch the template to do different things depending on the format.

Now, if you have a lot of formats, such approach can go out of hand very easily.

Alternatively, you can use a template for a single format only, naming it: `code-hyphen-template name`. E.g. `c11-latex.tt` (under the same directory) which will be used only with the `c11` format, avoiding the use of conditionals in the master template. I'd suggest this approach, hardcoding all the LaTeX settings but binding to a single format, instead of hacking the master template (which naturally will be used in the BookBuilder and if you are not careful the results could be very unexpected).

Templates

Beware, here you're entering an advanced topic, and assumes you're comfortable both with LaTeX and with a template markup.

You should use the templates when you want to control the LaTeX delegating to amusewiki only the text body.

In the archive root (e.g. /var/lib/amusewiki/repo/MYSITE-ID/) you would do

```
$ muse-compile.pl --ttdir templates --output-templates
Using Text::Amuse 1.29, Text::Amuse::Compiler 1.33, PDF::Im
Creating ./templates/html.tt
Creating ./templates/minimal_html.tt
Creating ./templates/bare_html.tt
Creating ./templates/css.tt
Creating ./templates/latex.tt
Creating ./templates/bare_latex.tt
```

This will populate the templates directory with the default templates. Remove the ones you don't need and start editing the one you need (usually latex.tt)

Set this directory in super-user admin console: /admin/sites/edit/MYSI

This is a subset of the Perl Template::Toolkit templating system, to be exact Template::Tiny, so don't expect to do crazy stuff there.

However, you can use conditionals like:

```
[% IF safe_options.format_id.c122 %]
Here I'm in format c122
[% ELSE %]
[% IF safe_options.format_id.c123 %]
Here I'm in format c123
[% ELSE %]
My Text
[% END %]
```

Contents

Custom formats	6
Leveraging the custom formats to correct the final PDFs . . .	7
Templates	10

- `\strut`
- `\newpage`
- `\enlargethispage{DIMENSION}`
- `\flushbottom`
- `\raggedbottom`
- `\markright{TEXT}`
- `\markboth{TEXT}{TEXT}`
- `\sloppy`
- `\fussy`
- `\vfill`
- `\clearpage`
- `\cleardoublepage`
- `\looseness=INTEGER`

Possible arguments:

STYLE One of `plain`, `empty`, `headings`, `myheadings`, `scrheadings`.

TEXT Plain text, without any of the reserved LaTeX characters.

DIMENSION regular TeX size in `mm`, `cm`, `pt`, `em`, e.g. `3mm`.

INTEGER An integer positive or negative

This assumes that you are comfortable with LaTeX and you know what these commands do, but they should be enough to give you full control on the page and to do some corrections. If you think that something else should be added, please open a bug report against <https://github.com/melmothx/text-amuse-compile>

For maximum flexibility, you can define commands starting with `\amusewiki` in your template (without any argument) and use them in the magic comments. This allows you to inject arbitrary (but still) TeX commands in the body in a relatively safe fashion.

going to need more control on the output. For that we are still going to use the custom formats code (as a typesetting correction would apply only on given format, not to all).

For that we have the magic comments.¹ It allows the injection of a limited set of LaTeX commands at an arbitrary position in the text.

The formal definition is:

- start a line with a semicolon (normal Muse comment)
- colon - format code - colon (without spaces)
- LaTeX command

This is an example:

My text starts...

```
; :c9: \sloppy
; :c9: \vskip 10mm
; :DEFAULT: \fussy
; :c111: \sloppy
; :ALL: \newpage
; :*: \newpage
```

My text continues...

The DEFAULT code is used when no custom format is used.

The ALL code (and its alias, an asterisk), *always* trigger the command, regardless of the custom format passed.²

As you can see, commands for different formats can coexist without problems.

The list of permitted commands is:

- `\thispagestyle{STYLE}`
- `\pagestyle{STYLE}`
- `\vskip DIMENSION`

¹ This is an Amusewiki extension, it was not present in the original Emacs Muse implementation. List updated to Text::Amuse::Compile version 1.52, released on January 14, 2021

² The ALL trigger was introduced with Text::Amuse::Compile 1.61, released on 2021-04-11.

Even if Amusewiki can be used as a blog or wiki engine, its prominent feature is the production of camera-ready PDFs. There are various ways you can affect the output. This article explains what to do if you need more control on the output. Before you continue, please be sure to have an up-to-date Amusewiki instance (2.411 at the time of writing)

Let's first clarify the concepts:

template this refers to an optional file with extension `.tt`. Normally this file exists only in memory but you can use it if you set the path in the admin (super-user only).

custom format this is the definition to produce a PDF/EPUB output file. You can have as many custom formats as you want with a recent Amusewiki (in the past there were only 3 of them, plain, A4 imposed and Letter imposed).

Custom formats

Normally you may want to use the Amusewiki facilities to format your texts. A stock installation offers a couple of them but you can add/disable them at will. Please note that they have the same options as the BookBuilder, the only difference is that the custom formats are site-wide, while the bookbuilder is on request.

The list of custom formats is accessible in the main site configuration panel, `/user/site`, and they can be managed at `/settings/formats`.

The screenshot shows the 'Custom formats for staging.amusewiki.org' configuration page. It features a table with columns for Code, Name, Description, Active, Clone, Priority, and File extension. Below the table is a 'Format name' input field and a 'Create a new format' button.

Code	Name	Description	Active	Clone	Priority	File extension
c2	EPUB with embedded fonts	EPUB with Iwona fonts	<input checked="" type="checkbox"/>		1	.c2.epub
c1	PDF with cropmarks	A5 pdf on A4	<input checked="" type="checkbox"/>		1	.c1.pdf
c5	plain PDF	Standard format. Changes to these fields will be ignored: format:pdf imposed:0	<input checked="" type="checkbox"/>		1	.pdf
c3	A4 Imposed PDF	Standard format. Changes to these fields will be ignored: cover:1 format:pdf imposed:1 papersize:a5 schema:2up	<input checked="" type="checkbox"/>		2	.a4.pdf
c4	Letter imposed PDF	Standard format. Changes to these fields will be ignored: cover:1 format:pdf imposed:1 papersize:5.5in:8.5in schema:2up	<input checked="" type="checkbox"/>		3	.l1.pdf
c6	Slides (PDF)	Standard format. Changes to these fields will be ignored: format:slides	<input type="checkbox"/>		4	.s1.pdf

By default, Amusewiki tries to build every active format for each text in the archive.

On this page you can turn make them inactive and modify them. Please note the "Code" field (c1, c2, etc), because we'll need that.

You can have as many custom formats you want. They will then appear in the text infobox once they are ready (the building happens in the background).

The screenshot shows the 'File Gallery' interface with a toolbar containing icons for file operations like copy, paste, delete, and refresh.

However, you may want to use only some of them in a text. To do so, add to the muse header in the text:

```
#formats c1 c2
```

Where c1 and c2 are the wanted formats, by code, as shown at `/settings/formats`. (Usually the code also appears in the file suffix).

The site formats can also be used as predefined settings in the bookbuilder (right-hand top menu).

The screenshot shows the 'Use one of the site's formats' configuration page. It features a table with columns for URI, Partial selection, and Action. Below the table is a 'Format name' input field and a 'Create a new format' button.

URI	Partial selection	Action
test-custom-formats-all-en		Move up Move down Delete
Total pages: 2		Clear the book list Reset all

To suppress their generations for a given text, you can add to the header:

```
#formats none
```

(as none is just a non-existing format).

Leveraging the custom formats to correct the final PDFs

Muse is an abstract markup. It tries to hide the complexities of type-setting. Sooner or later, if you are going to print something, you are