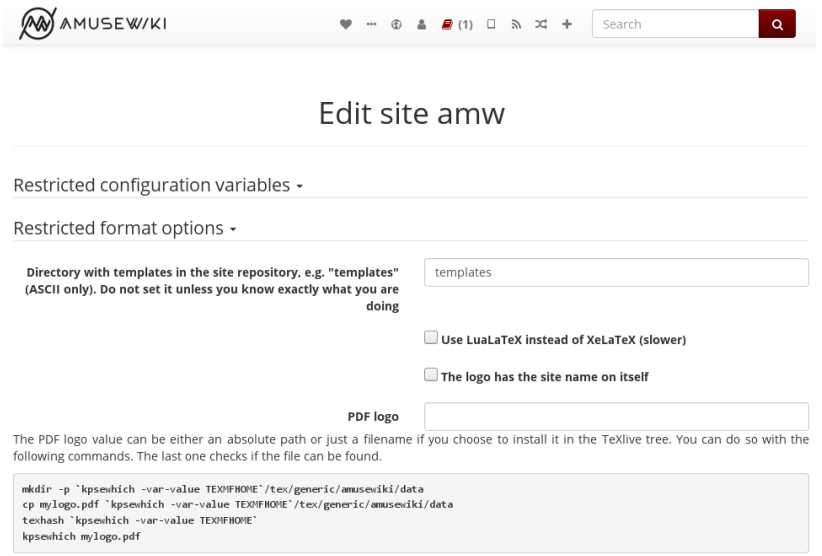
Templates and custom formats

Templates and custom formats

amusewiki.org



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

- \newpage
- \enlargethispage{DIMENSION}
- \flushbottom
- \raggedbottom
- \markrigth{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.

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`
- `\strut`

[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:

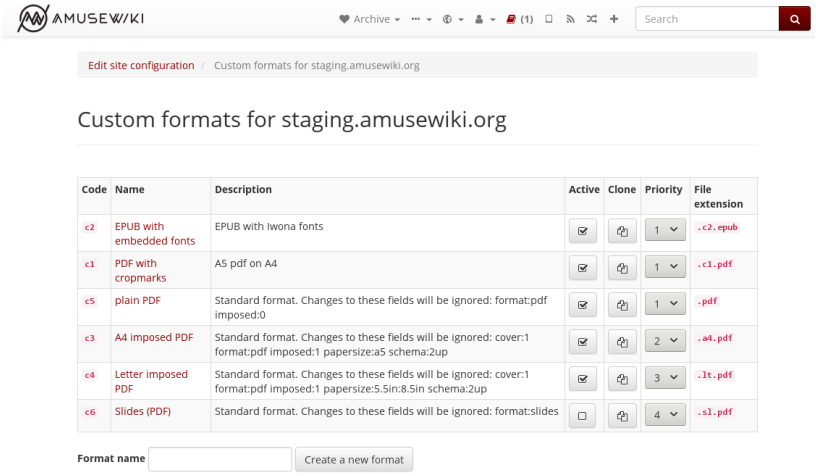
template this refers to an optional file with extension `.tt`. Normally this file exists only in memory but you can use it if you set the path in the admin (super-user only).

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.

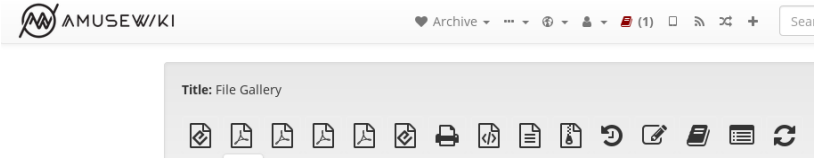
The list of custom formats is accessible in the main site configuration panel, `/user/site`, and they can be managed at `/settings/formats`.



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).

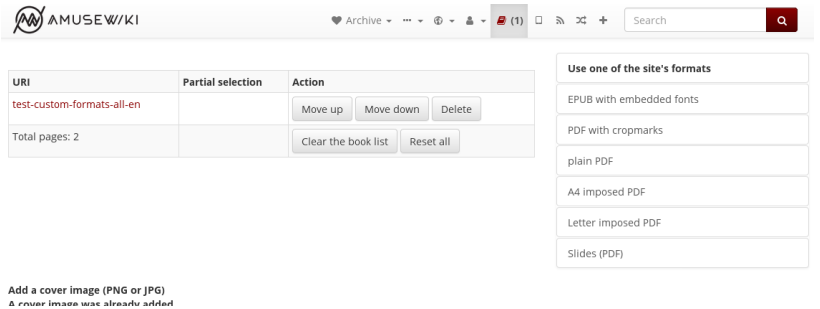


However, you may want to use only some of them in a text. To do so, add to the muse header in the text:

```
#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 also be used as predefined settings in the bookbuilder (right-hand top menu).



Leveraging the custom formats to correct the final PDFs

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