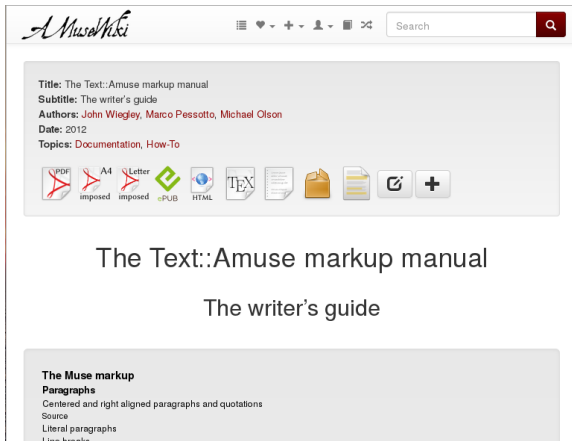


# AmuseWiki: a library oriented wiki engine (talk)

Marco Pessotto (melmothX)

September 3, 2015, Granada

# How does it look like?



The screenshot shows a web browser displaying a page from 'A Muse Wiki'. The page title is 'The Text::Amuse markup manual' with the subtitle 'The writer's guide'. The authors listed are John Wiegley, Marco Pessotto, and Michael Olson, and the date is 2012. The page is categorized under 'Documentation' and 'How-To'. Below the text, there are several icons representing different file formats: PDF, A4 imposed, Letter imposed, e-PUB, HTML, TEX, a document icon, a folder icon, a document with a pencil icon, and a plus sign icon. The main content area features the title 'The Text::Amuse markup manual' and the subtitle 'The writer's guide' in a large, centered font. At the bottom, there is a section titled 'The Muse markup' with a sub-section 'Paragraphs' containing a list of topics: 'Centered and right aligned paragraphs and quotations', 'Source', 'Literal paragraphs', and 'Line breaks'.

*A Muse Wiki*    Search

**Title:** The Text::Amuse markup manual  
**Subtitle:** The writer's guide  
**Authors:** John Wiegley, Marco Pessotto, Michael Olson  
**Date:** 2012  
**Topics:** Documentation, How-To

PDF   A4 imposed   Letter imposed   e-PUB   HTML   TEX   [document icon]   [folder icon]   [document with pencil icon]   [plus icon]

## The Text::Amuse markup manual

### The writer's guide

**The Muse markup**  
**Paragraphs**  
Centered and right aligned paragraphs and quotations  
Source  
Literal paragraphs  
Line breaks

# Scenario

Digital library with more than 2000 texts, including full-length books

Long term archiving (not fire and forget texts), control revision

Quality output required (read: LaTeX output)

Imposing of PDF for home-printing

EPUB output for mobile devices

Preference for a flat file storage (like ikiwiki or MoinMoin)

Creation of collections (like on mediawiki)

One-man project

# The lightweight markup

No standard, even if Markdown seems to be the winner (but with dialects)

Emacs Muse: project kind of dead, but the markup is compact and expressive, documented, and has a reference implementation.

`https://www.gnu.org/software/emacs-muse/`

Some incompatibilities have been introduced, but they are documented (to address corner cases where the syntax can be confusing).

Bottom line: all these markups are easy to use and it takes 5 minutes to learn one of them, as long as it is documented.

# Our own dialect of Emacs Muse

Manual:

<http://www.amusewiki.org/library/manual>

Module: `Text::Amuse` (produces LaTeX and HTML)

Ill-suited for technical papers, though. No math support, no syntax highlight, but well-suited for general prose and even poetry.

It has every feature one could expect from a lightweight markup: images, sectioning, footnotes, simple tables, bold, italics, subscript, superscript, lists, verbatim, quotations.

So far proved itself good and expressive.

# Importing

Legacy library had the texts in filtered HTML

People usually have the texts in Word format or copy and paste from HTML pages

The javascript HTML editor CKEditor has a “paste from Word” feature <http://ckeditor.com/>

Need to convert the HTML to Muse, preserving as much as possible the logical structure of the document (and discarding the noise).

Need some common search-and-replace patterns (like typographical quotes, text cleaning).

Text::Amuse::Preprocessor

# Compiling

Templating for output: `Template::Tiny`

PDF generation: `XeTeX` or `LuaTeX` (Unicode aware, system fonts)

`EBook::EPUB::Lite` (this is a port of `EBook::EPUB` without XS dependencies) using `Text::Amuse`'s splat HTML output

`PDF::Imposition` (written for this project but it's a general purpose module): put logical pages into a physical page according to a schema (for booklets and home printing)

All the above glued together by `Text::Amuse::Compile`  
`muse-compile.pl` script is shipped with `Text::Amuse::Compile`, so you can generate the formats from the command line.

# Data storage

Texts themselves are self-contained. All the information describing the text (like author, title, categories) is stored in the header of the text. 1 text (even a whole book), 1 file.

Texts are stored in a Git archive

Git integration on the site with cgit:

<http://www.amusewiki.org/git/amw/>

Full text search: Xapian (light, fast, fairly simple to setup, well integrated in Perl with `Search::Xapian`).

Database integration: `DBIx::Class`



## Web backend

A daemon takes care of all the operations which are slow or somehow delicate where concurrent access could be a problem (text compilation, publication, indexing, Git interaction).

Formats are pregenerated, including the HTML. The frontend just serves them.

The backend and the frontend communicate via a job queue in the database.

Some message queue systems were examined, but resorted to use the database because it was the most straightforward and other solutions looked like over-engineering.

# Web Frontend

Catalyst application: chaining, method-to-uri mapping, actively developed, great community, back-compatibility approach.

Plack-able application (currently deployed via nginx + FCGI)

Template: Template Toolkit

Localization via Catalyst::Plugin::I18N (plus local overriding via local JSON file).

Localized for English, Italian, Croatian, Macedonian, Russian, Finnish, Swedish, German, Spanish.

Multisite: on one instance you can run as many sites as you want (this was the most compelling argument to write AmuseWiki).

# User management

Kept at minimum reusing existing solutions.

```
Catalyst::Plugin::Authentication
```

```
Catalyst::Plugin::Authorization::Roles
```

```
DBIx::Class::PassphraseColumn
```

No hierarchical structure: each librarian can create other peer librarians (plus root for site management) with the same level of privileges.

Modes:

- private site

- blog site (only logged-in can edit)

- moderated wiki (approval required)

- open wiki (undertested)

# The Bookbuilder

The basic idea is like the Wikimedia's book creator, but with goodies. Features:

- LaTeX output

- Font selection

- Paper size selection

- Imposition schema selection

- Cover images upload

Custom files are compiled by the backend, even if the users sees the live logs and the process is pretty fast.

Epub output if required, with embedded fonts.

A basic question to keep robots away (probably will not scale, but so far works well)

# Some time left?

If we have some more time and no questions...

# The past

Drupal + filtered HTML, texts kept in sync on a local Git repo with scripts. Obviously it wasn't a brilliant idea, to be generous.

Same filtered HTML inherited from Drupal, plus home-brewed CGI scripts. It kind of worked.

Dancer application and Emacs Muse markup, no database. Worked, but didn't scale with multisite.

# The future

Slides (upcoming release)

A better installer

Teasers

Decorative images