Molo Documentation

Release 5.22.4

Praekelt.org

Nov 23, 2017

Contents

Molo is a set of tools for publishing mobi sites with a community focus. It is built on top of Django and Wagtail.

CHAPTER 1

Contents

1.1 Installation

Molo requires Python (version 2.6 or 2.7) to be installed. This installation method also requires pip as well as virtualenv. All three of these must be installed before following the installation steps below.

1.1.1 Installing Molo

Molo can be then installed using:

```
$ virtualenv ve
$ source ve/bin/activate
(ve)$ pip install molo.core
```

1.2 Getting Started

Molo scaffolds a Django application for you with sensible defaults, packages and configuration to help you get going as soon as possible.

1.2.1 Scaffold a site using Molo

The goal of Molo is to provide a solid base of proven, stable packages that help Praekelt.org and partners to deliver on project scope:

```
$ molo scaffold myapp
$ cd myapp/
$ ./manage.py migrate
$ ./manage.py createsuperuser
$ ./manage.py runserver
```

Open the sample site in your browser at http://localhost:8000/ and the CMS at http://localhost:8000/admin/.

1.2.2 Scaffolding a site in an existing repository

It's not always desirable to create a new directory for an application, especially when scaffolding an application for a repository that's already been created. Specifically for that Molo allows a second argument for the directory.

To scaffold an application called `myapp` in the current directory do:

```
$ molo scaffold myapp .
```

1.2.3 Specifying extra requires

Molo in itself is not expected to be enough to deliver on a client request. During scaffolding use the `--require` commandline parameter to include more libraries that are required for installation:

```
$ molo scaffold myapp --require=django-contrib-comments
```

Adds the *django-contrib-comments* to the generated requirements file which is read by the generated package's `setup.py` file.

Multiple requires can be specified on the command line:

```
$ molo scaffold myapp --require=django-contrib-comments \
>   --require=molo.profiles
```

1.2.4 Automatically adding installed apps

If you're including a Django app chances are you're going to want to add it to your `INSTALLED_APPS` settings as well as adding an entry to the generated `urls.py` file:

```
$ molo scaffold myapp --include=django_comments ^comments/
```

This results in the following `urls.py` entry:

```
url(r'^comments/',
    include('django_comments.urls',
            namespace='django_comments',
            app_name='django_comments')),
```

Note: multiple includes can be specified on the command line, the format is `--include=<app_name> <regex-for-urls>`

For convenience, here's the full scaffold command for the current plugins:

```
$ molo scaffold myapp \
    --require=molo.profiles --include=molo.profiles ^profiles/ \
    --require=django-contrib-comments --include=django_comments ^comments/ \
    --require=molo.commenting --include=molo.commenting ^commenting/ \
    --require=molo.yourwords --include=molo.yourwords ^yourwords/
```

Note: `molo.profiles` is a requirement of molo core and is therefore automatically installed when molo is installed.

Molo, Django & settings files

You now have a standard Django application set up for normal development. The only difference is that your settings are Python modules found in the `settings/dev.py` and `settings/production.py` files in your applications folder. Both of these inherit settings from `settings/base.py`.

To create your own custom settings add a `local.py` file in the `settings` folder. The `settings/dev.py` will automatically include those settings for your local development environment.

Unpacking Templates from Packages

Sometimes a package's existing templates simply are not enough and need some amount of customization. Use the `unpack-templates` command in the scaffolded application to unpack a package's templates in your application's templates directory:

```
$ molo scaffold testapp \
>   --require=molo.profiles \
>   --include=molo.profiles ^profiles/
$ pip install -e testapp
...
```

You'll see the default templates that `molo.core` ships with available in the `templates` directory:

```
$ ls testapp/testapp/templates
404.html  500.html  base.html core
```

Now we unpack the `profiles` templates directory from the `molo.profiles` package into the `testapp` package template directory:

```
$ molo unpack-templates molo.profiles testapp
$ ls testapp/testapp/templates
404.html  500.html  base.html core profiles
```

The format is:

```
$ molo unpack-templates <source package> <target package>
```

Writing tests

Now develop your application and write tests for the features you add. Running your tests for Django works as you would expect:

```
$ ./manage.py test
```

1.2.5 Testing the Molo scaffolding tool

If you're interested in working on or contributing to the code that does the scaffolding then clone this repository from the GitHub repository at http://github.com/praekelt/molo.

Install the requirement development & testing dependencies:

```
$ pip install -r requirements-dev.txt
```

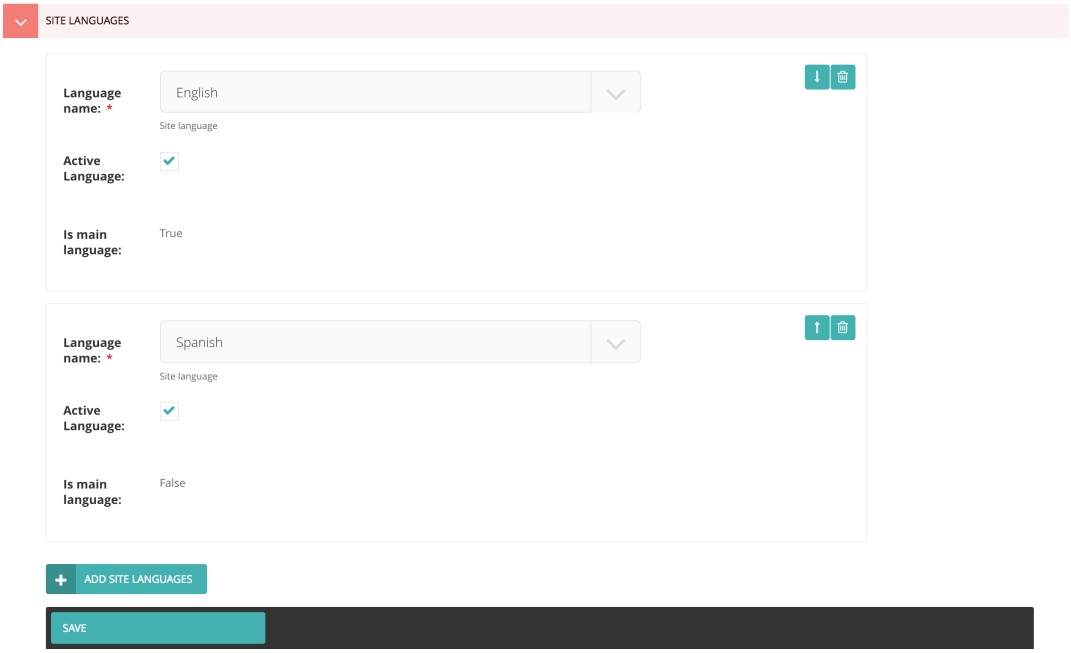
And then run the full test suite with:

```
$ py.test
```

Pull requests are expected to follow Praekelt's Ways Of Working.

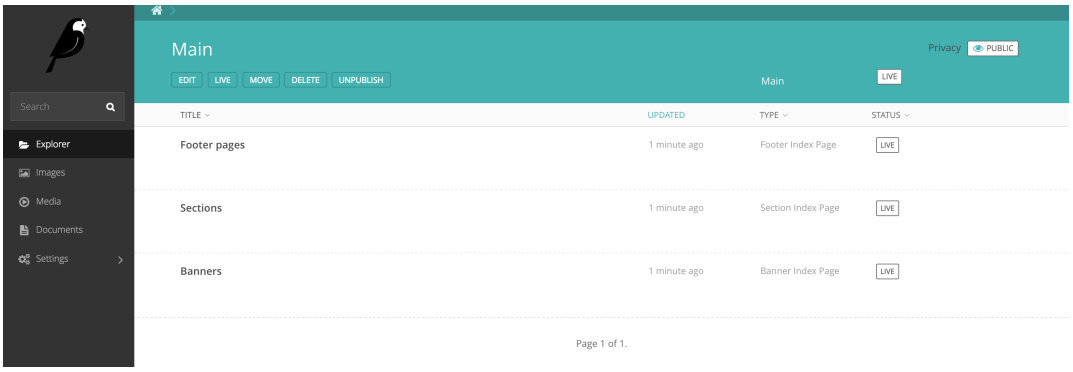
1.3 What is bundled with Molo?

1. Wagtail CMS
2. Molo Profiles
3. Basic feature phone template set.
4. Basic models for the following tree structure:
   (a) **A site has a main language, and the option of one or more additional languages.**
      - All content has to initially be created in the main language. Thereafter translations can be made for that content.
      - Translations for content cannot exist for additional languages if it does not first exist for the main language.
      - The first language added will be the main language, any other languages added after will be additional languages.



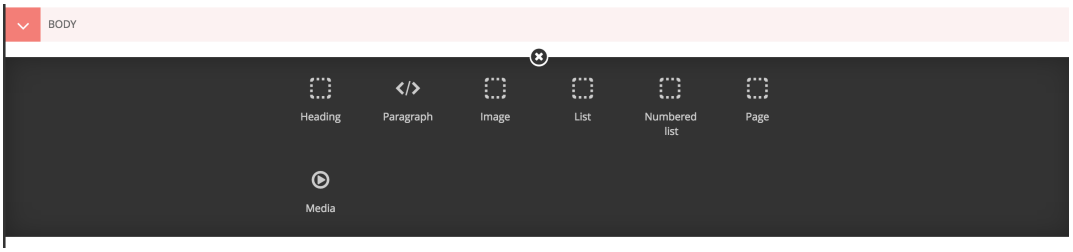
   (a) **Once a main language has been created, a main page will be created as well. A main page consists of index pages.**
      - Index pages exist for each content type.

- All section pages are grouped into the ‘Sections’ index page.
- All banners are grouped into the ‘Banners’ index page.

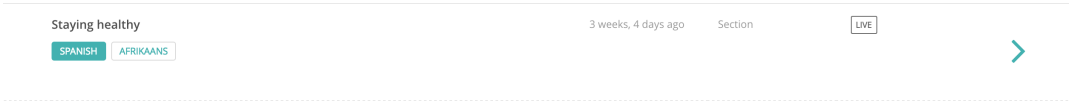


(a) **Once a section is made, articles can then be added to that section.**

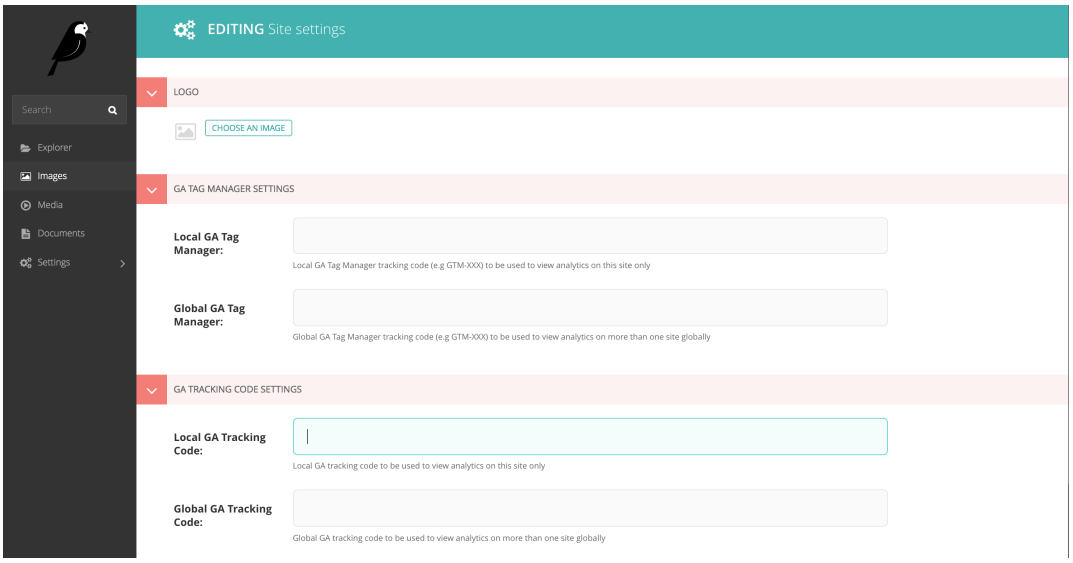
- Articles only exist as a child of a section page.
- Articles are composed from one or more blocks.
- Blocks can be headings, paragraphs, images, lists or links to other pages.



(a) Content such as sections or articles are displayed in their main language. Their translation in any additional language added is shown below the content. If one would like to edit the Spanish version of ‘Staying Healthy’, one would click on ‘SPANISH’, and then edit.



(a) A Settings tab that includes Site Settings. Site Settings is where the logo, google analytics and various other settings are set.



1.4 Features

Molo consists of a core structure onto which new feature plugins can be added. This core is the foundation that allows you to create a site in Wagtail.

1.4.1 Core Features

- **Multiple Languages**
  - Molo allows you to offer you content in multiple languages using the TranslatablePageMixin
- **Sections (and subsections)**
  - Content sections that allows structuring of content on the site
- **Articles**
  - The main content element of molo.
  - It allows you to create rich articles containing multiple images, lists (bulleted/numbered) and links to other pages
- **Footer pages**
  - Content pages mostly used for About, Terms and Contact information
- **Tags**
  - Tags are words or hashtags that can be added to articles and can be used to navigate through a site.
- **Reaction Questions**
  - Questions that have set responses that a user can choose from. These are added to articles to get a response for a specific article.
- **Banners**
  - Image banners on the home page that can be linked to any page on the site
- **Search**

- The ability to search for any content on the site
- The ability to show a highlighted term in the results
- Support for both Elastichsearch 1.x & 2.x
- Search term highlighting using Elasticsearch Backend

Note: Search highlighting is only supported by the Elasticsearch backend.

You can use Elasticsearch 1 with the following settings:

```
WAGTAILSEARCH_BACKENDS = {
    'default': {
        'BACKEND': 'molo.core.wagtailsearch.backends.elasticsearch',
        'INDEX': 'base',
    },
}
```

Or Elasticsearch 2:

```
WAGTAILSEARCH_BACKENDS = {
    'default': {
        'BACKEND': 'molo.core.wagtailsearch.backends.elasticsearch2',
        'INDEX': 'base',
    },
}
```

The example below shows how to show the highlighted word in the search results page with the following rules:

1. Title field is always displayed: if the term appears in this field, it will be highlighted.
2. Display highlighted term in subtitle or body. If the term appears in the title only, display the original content of the subtitle field.

You need to update the `search_results.html` page with the following code:

```
{% for page in search_results %}
  {% with parent_section=page.get_parent_section ancestor=page.get_parent_section.get_
↪ancestors.last %}
    <a href="{% pageurl page %}">
      <div class="nav">
        {% if ancestor.sectionpage.image %}
            <h6>{{ancestor.title}}</h6>
        {% else %}
            <h6>{{parent_section.title}}</h6>
        {% endif %}
        {% if page.title_highlight %}
            <h3>{{page.title_highlight|safe}}</h3>
        {% else %}
            <h3>{{page.title}}</h3>
        {% endif %}
        {% if page.subtitle_highlight or page.body_highlight %}
            {% if page.subtitle_highlight %}
                <p>{{page.subtitle_highlight|safe}}</p>
            {% elif page.body_highlight %}
                <p>{{page.body_highlight|safe}}</p>
            {% endif %}
        {% else %}
            <p>{{page.subtitle}}</p>
```

```
        {% endif %}
      </div>
    </a>
  {% endwith %}
{% endfor %}
```

1.5 Multiple Languages

Molo features the ability to create translatable pages. This means that pages such as a section or article is translatable. This is done via adding the TranslatablePageMixin in the Page's definition.

1.5.1 Creating A Translatable Page Model

In your models.py import the TranslatablePageMixin:

```
from molo.core.models import TranslatablePageMixin
```

Add it to the definition of your model:

```
class Competition(TranslatablePageMixin, Page):
  description = models.TextField(null=True, blank=True)
```

1.5.2 Getting Translations

In order to get the translations for a page model we use the following helper functions from the TranslatablePageMixin. Given a locale and a site, this will return the translation of the page:

```
competition.get_translation_for(locale, site)
```

We use template tags to get the locale.

The following will return the main language that the content was created in, if the content is currently in any additional language. For example, if the content is currently in French, and the main language is English, this function will English page:

```
competition.main_language_page
```

1.6 Merged CMS

Molo allows you to manage more than one site in a single CMS using wagtail's multi-site functionality.

1.6.1 Sharing a database

The database is shared between the number of sites you have. It is important to know this when querying data. One should always make sure the page/s you are querying for are descendants of the correct Main page.

See wagtail documentation for more on Multi-Site CMS

1.6.2 Copying between sites with multi-language content

Molo allows you to create content in multiple languages, as well as have sites in multiple languages. When copying content in language x and y to a site that has only language x, the content will be copied over as well as language y. However, language y will be set as inactive as it never existed on the desination site before the copy.

See help centre docs for more info on copying content and sites (insert link to helpcentre docs)

1.6.3 Copying content with relations to different sites

An *ArticlePage* has a relation to a *ReactionQuestion*. When copying an article from site one to site two with a link to a reaction from site one, the article will be copied over to site two. However, the linked reaction question will still point to the reaction question question in site one. This is avoided by creating new article reaction question relations after the article has been copied:

```
# replace old reaction question with new reaction question
question_relations = \
    ArticlePageReactionQuestions.objects.filter(page=article)
for relation in question_relations:
    if relation.reaction_question:
        new_question = ReactionQuestion.objects.descendant_of(
            copied_main).filter(
                slug=relation.reaction_question.slug).first()
        relation.reaction_question = new_question
        relation.save()
```

For any new content that has a relation to other content, the same will need to be done.

1.7 Template Tags

1.7.1 Using template tags to get translations

In order to get the translation for a page model, we need to have the locale which is found in the context of a template. This can be accessed from a template tag.

get_translation

Returns the translation for the page model. It gets the locale from the context and returns the translation for that locale:

```
get_translation(context, page)
```

get_pages

Takes in context, a queryset and a locale. It returns all the pages in their correct locale:

```
get_pages(context, qs, locale)
```

render_translations

Takes in context and a page. It renders the translated pages in wagtail CMS:

```
render_translations(context, page)
```

1.7.2 Using template tags to render content

Content for Home Page

The `section_listing_homepage` tag will return all the sections with articles in them that are featured on the homepage:

```
section_listing_homepage(context)
```

The `latest_listing_homepage` tag will return all the articles that are featured in latest section on the homepage:

```
latest_listing_homepage(context)
```

The `bannerpages` tag will return all the banners that are live:

```
bannerpages(context)
```

The `footer_page` tag will return all the footer pages that are live:

```
footer_page(context)
```

social_media_footer

Returns the social media footer:

```
social_media_footer(context)
```

The `tag_menu_homepage` tag will return all the tags that are live:

```
tag_menu_homepage(context)
```

The `topic_of_the_day` tag will return the article that has been promoted as the topic of the day:

```
topic_of_the_day(context)
```

The `get_tag_articles` tag is a bit more complex. It returns all the content you can have on your homepage. It takes in a `section_count`. This is the amount of section you want appearing on your homepage. It takes in a `tag_count`. This is the amount of articles you want appearing under each tag that has been promoted to the homepage. It takes in a `sec_articles_count`. This is the amount of articles you want appearing under each section, as well as a latest article count. This is the amount of articles featured in latest that will have preference in placement. This will return all the content rendered on the homepage, excluding banners and footers. It is done in such a way that no articles on the homepage are repeated:

```
get_tag_articles(
        context, section_count=1, tag_count=4, sec_articles_count=4,
        latest_article_count=3)
```

Content for Section Pages

get_tags_for_section

Returns the tags that are shown in a section (amount specified by `tag_count`) with the articles for that tag (amount specific by `tag_article_count`):

```
get_tags_for_section(context, section, tag_count=2, tag_article_count=4)
```

load_sections

Returns all the section that are live for a specific Main page:

```
load_sections(context)
```

breadcrumbs

Returns the breadcrumb for the current page if the current page is not the homepage:

```
breadcrumbs(context)
```

load_child_articles_for_section

Returns articles that are children of a specific section page (amount specified by `count`):

```
load_child_articles_for_section(context, section, count=5)
```

load_descendant_articles_for_section

Returns articles that are descendants of a specific section page (amount specified by `count`). It is possible to specify whether these articles have to be featured in the homepage, section or latest:

```
load_descendant_articles_for_section(
        context, section, featured_in_homepage=None, featured_in_section=None,
        featured_in_latest=None, count=5)
```

load_child_sections_for_section

Returns all the child sections for a specific question. The amount returned can be limited by count:

```
load_child_sections_for_section(context, section, count=None)
```

Content for Articles Pages

get_parent

Returns the parent page of an article page:

```
get_parent(context, page)
```

get_next_article

Returns the next article in a list of articles:

```
get_next_article(context, article)
```

get_recommended_articles

Returns a list of all articles that have been set as recommended for this article:

```
get_recommended_articles(context, article)
```

load_reaction_question

Returns all reaction questions that have been linked to this article:

```
load_reaction_question(context, article)
```

load_user_can_vote_on_reaction_question

Returns True or False based on whether a user has already voted on this question or this article or not:

```
load_user_can_vote_on_reaction_question(context, question, article_pk)
```

load_choices_for_reaction_question

Returns all the choice that are live for a reaction question:

```
load_choices_for_reaction_question(context, question)
```

load_tags_for_article

Returns all the tags that have been attached to this article in the CMS:

```
load_tags_for_article(context, article)
```

social_media_article

Returns the social media article:

```
social_media_article(context)
```

Content for Tag Pages

`get_articles_for_tag`

Returns all the articles that have been linked to a specific tag:

```
get_articles_for_tag(context, tag)
```

The `get_next_tag` tag returns the next tag in the list of live tags. If the current tag is the last in the list, it will return the first tag:

```
get_next_tag(context, tag)
```

Content for CMS

The `should_hide_delete_button` tag returns True or False based on whether a page should be deletable or not:

```
should_hide_delete_button(context, page)
```

1.8 Template Patterns

1.8.1 Praekelt Molo Design System

Molo design system a living document of visual design components and element markups, which provides a set of reusable patterns that can be combined to make a cohesive website. It documents the visual language, such as header styles and color palettes, used to create the site. It's purpose is as a one-stop place for the entire team to reference when discussing new site designs and iterations.

The design system will make it easy to build/scaffold custom Molo mobi-sites from our Molo Framework with a consistent look and feel using predefined Molo Core patterns and features - without reverse engineering our styles. The Design System and Molo Core serve as a single source of truth for our Frontend Templates stack, to help us establish cohesive user experience, a common taxonomy, enforce a modular approach and good quality code on mobile site applications.

Mote is a container that renders pattern libraries being built by Praekelt.com: http://www.praekelt.com/

Website Components

Mote: http://white-frog-248.seed.p16n.org

1.8.2 Website Glossary

These are the Molo sites:

- TuneMe: https://tuneme.org/
- Springster: http://sa.heyspringster.com/
- FreeBasics: http://amabhungane.molo.site/
- Babycenter: http://southafrica.babycenter.io/
- IoGT (Internet of Good Things): http://za.goodinternet.org/

1.8.3 Front-end Project Setup

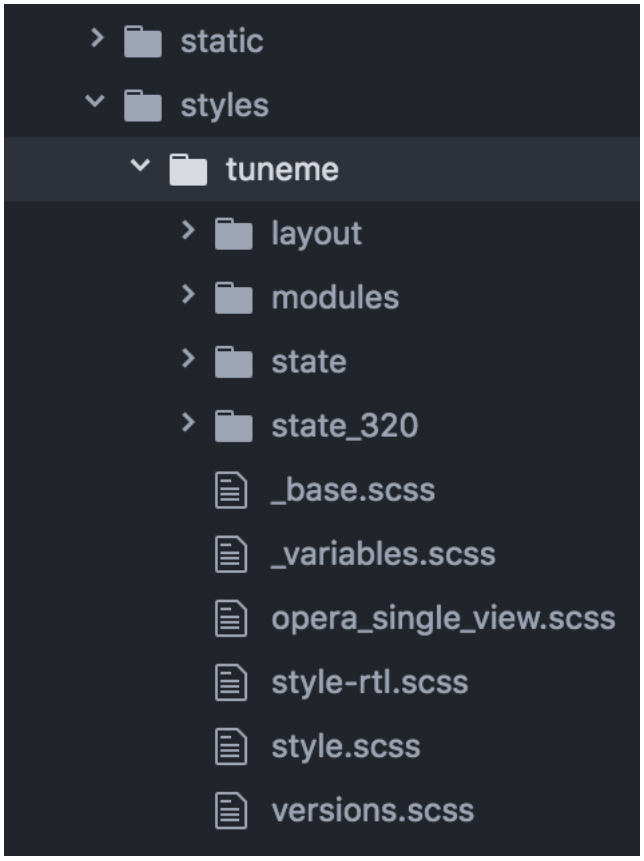
Backend developers tie a collection of reusable apps with a set of guidelines - the guidelines allow developers to share, reuse, maintain and improve the source code quality which does not extend to Frontend source code resulting in unstructured, inconsistent markup with poor quality and maintainability.

MARKUP & CSS - SMACSS,BEM for single responsibility principle

BEM is a naming convention of markup and CSS classes sticking to single responsibility principle and organising e.g.

```
<ul class="article-list article-list{{self.get_effective_extra_style_hints}}">
  {% load_child_articles_for_section self count=None as articles %}
  {% for article in articles %}
  <li class="article-list__item">
   <a href="{% pageurl article %}" class="article-list__item--anchor">
    {% if article.image %}
      {% image article.image width-90 height-90 jpegquality-70 as article_image %}
      <img src="{{ article_image.url }}" alt="{{ article_image.title }}" class="article-list__image
      article-list__image--section"/>
    {% endif %}
    <h1 class="article-list__title">{{article.title}}</h1>
    <h4 class="article-list__subtitle">{{article.subtitle}}</h4>
    </a>
  </li>
```

SMACSS is the categorization of CSS in a logical and layered fashion broken down into 5 categories: Base, layout, module, state and theme. Each category can either be a folder, or file which contains CSS. We use CSS preprocessors (SASS) that removes performance penalties of using separate files by compressing and combining them into a single file. e.g.



We use gulp scripts to compress and combine SASS, JS, Icon Sprites into a minified file assets.

Gulp is a Node plugin, you need to have Node.js installed on your computer https://nodejs.org/en/ All the Node.js npm packages for automation are on the application's package.js to install run the following commands:

```
npm install
npm install --global gulp-cli npm install
gulp
```

Asset processing & bundling, concatenating and minification script and package modules on project root folder:

```
gulpfile.js
package.json
```

base.py gulpfile.js

```
'use strict';

var gulp             =   require('gulp'),
    sass             =   require('gulp-sass'),
    watch            =   require('gulp-watch'),
    cleanCSSMinify   =   require('gulp-clean-css'),
    rename           =   require('gulp-rename'),
    gzip             =   require('gulp-gzip'),
    notify           =   require('gulp-notify'),
    sourcemaps       =   require('gulp-sourcemaps'),
    livereload       =   require('gulp-livereload');

var sassPaths = [
    'tuneme/styles/tuneme/opera_single_view.scss',
    'tuneme/styles/tuneme/style.scss',
    'tuneme/styles/tuneme/state_320/state_320.scss',
    'tuneme/styles/tuneme/style-rtl.scss',
];

var sassDest = {
    prd: 'tuneme/static/new/css/prd',
    dev: 'tuneme/static/new/css/dev'
};
```

Target Audience

Molo is a set of tools for publishing mobile sites primarily for feature phone and slow end device users - often e.g. slow bandwidth 2G network connections therefor speed performance is important.

Website speed performance is done using software service tool - Dareboost: https://www.dareboost.com/en/home to measure speed, analyze and monitor websites.

Having a robust established ways of working we have effectively created a sustainable system that produces good quality and maintainable code

1.9 Plugins

1.9.1 Installing Plugins

Molo plugins are normal python modules and can be installed using pip.

If you didn't include the app during the scaffolding step using `molo scaffold myapp --include=molo.profiles ^profiles/` and you wish to add it manually, complete the following steps:

Install the plugin using pip:

```
$ pip install molo.profiles
```

Add the new plugin to your `INSTALLED_APPS` in your `myapp/settings/base.py`:

```
INSTALLED_APPS = (
    ...
    'molo.profiles',
)
```

Then add the new plugin urls to your `myapp/urls.py`:

```
url(r'^polls/', include('polls.urls', namespace='molo.polls')),
```

The final step is to run migrations as the plugins usually have their own migrations:

```
$ ./manage.py migrate
```

1.9.2 Existing Plugins

The following plugins are available to extend the core features of Molo.

molo.profiles

`Github`: https://github.com/praekelt/molo.profiles

Profiles provides user profiles which adds registration, login and user data functionality.

This library does not provide a Django user model, it provides a profile model that can be attached to a user. Our experience is that custom User models in Django add all sorts of unpleasantries when using migrations.

Main features:

- Loging/Registration
- User profile to store user data

molo.commenting

`Github`: https://github.com/praekelt/molo.commenting

Commenting builds on the *molo.profiles* plugin. It allows users to comment on articles and these comments to be moderated.

It is built using Django's Comments Framework.

Main features:

- Commenting on article pages
- Moderation of comments using django-admin
- Comment reporting by users to allow for community moderation
- `COMMENTS_FLAG_THRESHHOLD` allows for comments to be automatically removed if they have been reported by multiple users

molo.surveys

`Github`: https://github.com/praekelt/molo.surveys/

Surveys allows for user feedback on content.

Main Features:

- Multiple Questions of various types
- Multi-page surveys
- Direct and Linked surveys

molo.yourwords

`Github`: https://github.com/praekelt/molo.yourwords

YourWords (User generated content) allows users to submit content that can be converted into an article by an admin.

Main features:

- Setting up a Your Words competition
- Downloading competition entries as a CSV
- Ability to shortlist entries
- Converting winning entries to Articles

molo.polls

`Github`: https://github.com/praekelt/molo.polls

A poll is a short set of questions (or typically only one question) with predetermined answers that a user can choose from.

Main features:

- Creating and publishing a Question to the home page, section page and article page
- Multiple Question types (Single choice, Multiple Choice, Free Text, Numeric)
- Exporting polls results as a CSV (currently in dev)

molo.usermetadata

`Github`: https://github.com/praekelt/molo.usermetadata

User meta data allows one to create persona pages so that when a user visits a site for the first time, they are able to choose a persona, or choose to skip this. This does not require the user to log in.

Main features:

- Creating and publishing persona pages to be displayed when the user visits the site for the first time

1.10 Release Notes

1.10.1 5.22.4

- Admin View vertical scrolling touchpad bug fixed
- Scroller added on other Admin Views
- overlapping edit/delete controls fix on Admin View lists

1.10.2 5.22.3

- Exclude ArticlePageLanguageProxy from being indexed
- Use strings for paths
- Run part of the test suite on Python 3

1.10.3 5.22.2

- Fix Admin View scroller styles

1.10.4 5.22.1

- Admin View FED bug fixes updates

1.10.5 5.22.0

- Remove UC content import

1.10.6 5.21.4

- Wagtail style reverts and cleanup

1.10.7 5.21.3

- Admin View FED updates

1.10.8 5.21.2

- Bug fix: exclude pages that are submitted for moderation from MultiSiteRedirect

1.10.9 5.21.1

- Continued update to front end setup. See PR#465 for more details

1.10.10 5.21.0

- Update the project setup. See PR#477 for more details
- Fix Image Hashing update bug
- Fix errant ? in URLs

1.10.11 5.20.0

- only allow access to sites if the user has permissions for that site

Note: - once upgrading to this version, superusers need to give non-superusers users permissions to access their relevant sites - This release would need molo.profile 5.4.1

1.10.12 5.19.0

- Add Facebook Analytics in Site Settings

1.10.13 5.18.1

- Fix duplicate ImageInfo creation when image is saved

1.10.14 5.18.0

- Update image hashing function
- Update log settings to accomodate api logs

1.10.15 5.17.2

- Bug fix: remove update from social_media template tag

1.10.16 5.17.1

- Allow passing obj to social_media template tag

1.10.17 5.17.0

- Allow adding service directory api settings in CMS
- Used logging for the api import process

1.10.18 5.16.1

- Add more caching to improve performance

1.10.19 5.16.0

- add CSV mapping foreing page IDs to local IDs, to success email when site has been imported

1.10.20 5.15.0

- add management command to add tag to article
- add management command to set promotion date on article
- add caching to improve performance

1.10.21 5.14.0

- updated documentation for multi-site functionality
- add utilities to convert embedded page stream blocks in Recommended Articles
- exposed utilities via command `move_page_links_to_recomended_articles`

1.10.22 5.13.1

- fix image import bug which did not handle absolute URLs (i.e. storage on S3)

1.10.23 5.13.0

- refactored importing of site content via api
- created ImageInfo model to store image hashes
- bug fixes in api endpoints
- bug fixes in site importing

1.10.24 5.12.0

- added Migration for converting Media to MoloMedia (FIXED)

1.10.25 5.11.0

- DO NOT ADD THIS RELEASE (Migration Faulty)
- added Migration for converting Media to MoloMedia
- added feature in homepage for MoloMedia
- fixed admin layout

1.10.26 5.10.0

- add support for youtube links in MoloMedia

1.10.27 5.9.5

- fix admin layout styling bugs
- fix api locale field in translation when language has been deleted

1.10.28 5.9.4

- Bug Fix: Ensure load_tags_for_article only returns tags for article Pages
- Remove content_import tests

1.10.29 5.9.3

- Temporarily removed API import from sidebar

1.10.30 5.9.2

- Mote Update: Mote files updated to flexible accept applications style directory

1.10.31 5.9.1

- Bug Fix: Revert accidental travis setup change

1.10.32 5.9.0

- New Feature: API that exposes content via the */api/v2/* url
- New Feature: Import some site content to a new site via the newly created API. Imports the following content: - Site languages - Images - Sections - Articles - Tags - Banners Pages - Footer Pages

1.10.33 5.8.2

- Fix the responsive styling for Admin dashboard

1.10.34 5.8.1

- Fix the styling for Admin dashboard

1.10.35 5.8.0

- Add Admin View menu with the Article View to the CMS

1.10.36 5.7.0

- Deprecate use of search backends in Molo. Use wagatailsearch instead.

1.10.37 5.6.0

- New Feature: Add Article Publish action to shortcuts

1.10.38 5.5.2

- Bug fix: ensure that the old article exist in create_new_article_relations
- Bug fix: use full path for GA tracking

1.10.39 5.5.1

- Add get_effective_banner
- Run node tests in node_js Travis environment
- Fix npm module caching
- Run against latest Node LTS release
- Allow first priority of articles on homepage to go to latest articles when tag navigation is enabled
- Bug fix: make sure the delete button is not shown in drop down menus on cms
- Bug fix: only allow voting to shown for main language page for reaction questions in cms

1.10.40 5.5.0

- Remove PyPy Travis builds
- Clean up Travis file
- Travis: push wheels (bdist_wheel) to PyPI
- Remove unused dependencies
- Move some test dependencies out of main dependencies
- Don't pin the required setuptools version
- Update LICENSE file
- Move requirements to setup.py
- Remove django-modelcluster from scaffolded app dependencies, molo.core depends on newer version already
- Allow minor updates to wagtail package (e.g. 1.9.1, not just 1.9)
- Update .gitignore to newer standard (more Python 3 friendly)
- Fix and cleanup MANIFEST.in

1.10.41 5.4.7

- Update static files to fix missing/incorrect references

1.10.42 5.4.6

- Increase character limit on reaction question success message

1.10.43 5.4.5

- Add reaction question success_messages

1.10.44 5.4.4

- Add *get_effective_image* to reaction question choices

1.10.45 5.4.3

- Fix a bug for *get_next_tag* template tag

1.10.46 5.4.2

- show correct articles for language in load more and next tag on tag page

1.10.47 5.4.1

- Add get_next_tag Template Tag
- Add admin views for Reaction Questions
- Add util for creating new article relations when copying

1.10.48 5.4.0

- Add load more for Search Page
- Add load more for Tag Page
- Add reaction questions basic functionality

1.10.49 5.3.1

- Use get_effective_image instead of image in templates

1.10.50 5.3.0

- Add load more functionality to section page

1.10.51 5.2.5

- Bug Fix: Only index tag list if list not empty for sections and tags

1.10.52 5.2.4

- Bug Fix: Only show articles in search results
- Bug Fix: Only index tag list if list not empty

1.10.53 5.2.3

- Bug Fix: Show translation for Section Page on Home Page
- Bug Fix: Only show articles relevant to site under a tag
- Bug Fix: Ensure new article tag relations are made when copying sites

1.10.54 5.2.2

- Added Positional Banner Pages functionality
- Bug Fix: Return Main language pages for latest articles

1.10.55 5.2.1

- Added Tags to SectionPage
- Added Load More functionality for ArticlePages on the homepage

1.10.56 5.2.0

- Add gef_effective_image for ArticlePage (returns the image of article's main language page if article has no image, else returns article's image)
- Add get_parent template tag (returns the parent of a page)
- Bug fix: Filter tags via descendant of main
- Bug fix: Use 'to' id directly for copying in celery

1.10.57 5.1.1

- Bug fix: Call correct template for tag navigation
- Bug fix: Only call translation hook for translatable pages

1.10.58 5.1.0

- Add basics and components for Springster
- Add tag navigation
- Add better error handling for copying section index contents

1.10.59 5.0.4

- Use celery for copying section index contents

1.10.60 5.0.3

- Add parent_page_types to SectionPage

1.10.61 5.0.2

- Fix test for admin url redirect

1.10.62 5.0.1

- Version bump for molo profiles to resolve pin dependencies

1.10.63 5.0.0

- Pin molo.profiles to latest version
- Move templates out from cookiecutter
- Implement pattern library components to templates
- Add Mote to cookiecutter
- Fix of previous release
- Added index creation signals
- Added non routable mixin for Surveys
- Added profiles urls
- Added multi-site cms functionality (Merged CMS)
- Added authentication backend for linking users to sites
- Added middleware for site redirect

1.10.64 4.x

Main Features:

```
- Upgraded to Wagtail 1.8
- Added upload/download functionality for zipped media files
- Next and Recommended articles in articles
```

Backwards incompatible changes:

```
- Deprecatad use of ``wagtailmodeladmin``: ``wagtailmodeladmin`` package has been␣
↪replaced by ``wagtail.contrib.modeladmin``
- ``wagtailmodeladmin_register`` function is replaced by ``modeladmin_register``
- ``{% load wagtailmodeladmin_tags %}`` has been replaced by ``{% load modeladmin_
↪tags %}``
- ``search_fields`` now uses a list instead of a tuple
```

1.10.65 4.4.13

- Insure content demotion happens for each section individually

1.10.66 4.4.12

- Remove promotion settings from footer pages

1.10.67 4.4.11

- Fixed content import to return all data and not just default 10

1.10.68 4.4.10

- Fixed recommended article ordering in templatetag logic

1.10.69 4.4.9

- Added Non routable page mixin

1.10.70 4.4.8

- Pulled in changes from previous versions that were accidentally excluded
- Consolidated celery tasks in base settings file

1.10.71 4.4.7

- Fixed random test failures in content rotation test

1.10.72 4.4.6

- consolidate minute tasks into 1 call

1.10.73 4.4.5

- consolidate minute tasks into 1 call

1.10.74 4.4.4

- Fixed bug for previewing pages

1.10.75 4.4.3

- Excluded metrics URL from Google Analytics
- Fixed access to Explorer bug for superuser's with non-superuser roles

1.10.76 4.4.2

- Allows content rotation to pick from descendant articles not only child articles

1.10.77 4.4.1

- Updated template overrides to fix missing Page admin buttons

1.10.78 4.4.0

- Content rotation enhancement:
- Only promote pages that are exact type of ArticlePage
- Only demote an article if there is more than two promoted articles

1.10.79 4.3.3

- Add django clearsessions to celery tasks

1.10.80 4.3.2

- Added missing classes in custom admin template

1.10.81 4.3.1

- Fixed template error

1.10.82 4.3.0

- Removed the ability to delete index pages using the admin UI

1.10.83 4.2.0

- added multi-language next and recommended article feature

1.10.84 4.1.0

- Add sitemap - include translations

1.10.85 4.0.2

- Fixed template overrides for django-admin templates

1.10.86 4.0.1

- Added upload/download functionality for zipped media files

1.10.87 4.0.0

- upgraded wagtial to 1.8
- removed external dependency on wagtailmodeladmin to use internal wagtailadmin feature
- added bulk-delete permission feature for the Moderator group
- added edit permission for Main page to moderator and editor groups

1.10.88 3.x

Major revamp to the way we handle Multi Language on Molo and a bunch of new features

Main features:

```
- Revamped Multi Language support
- We added content automated content rotation and a way to schedule when content␣
↪should be cycled
- We now offer specifying Google Analytics from the CMS for both GA and GTM (this␣
↪uses celery for GA)
- Renamed HomePage module to BannerPage
- Changed content structure to introduce index pages
- Upgraded wagtail to 1.4.3
- We've added the option to allow un-translated pages to be hidden
- We now show a translated page on the front end when it's main language page is␣
↪unpublished
- Add Topic of the Day functionality
- Add Support for both Elastichsearch 1.x & 2.x
- Add ability to show a highlighted term in the results
- Implement custom error page for CSRF error
```

Backwards incompatible changes:

```
- Deprecated use of ``LanguagePage``: use ``SiteLanguage`` for multi-language support
- Deprecated use of ``Main`` : all pages are now children of their index page (e.g.␣
↪Section Pages are now children of Section Index Page)
- Deprecated use of ``Section.featured_articles``: use the template tag ``{% load_
↪descendant_articles_for_section section featured_in_section=True %}``
- Deprecated use of ``Section.featured_articles_in_homepage``: use the template tag ``
↪{% load_descendant_articles_for_section section featured_in_homepage=True %}``
- Deprecated use of ``Section.latest_articles_in_homepage``: use the template tag ``{
↪% load_descendant_articles_for_section section featured_in_latest=True %}``
- Deprecated use of ``Section.articles``: use the template tag ``{% load_child_
↪articles_for_section page %}``
```

1.10.89 3.17.4

- Fix the bug with draft article publishing when content rotation is on

1.10.90 3.17.3

- Ensure email address is set when using SSO

1.10.91 3.17.2

- Put ForceDefaultLanguageMiddleware before django.middleware.locale.LocaleMiddleware

1.10.92 3.17.1

- (bug) use datetime instead of UTC timezone for rotation

1.10.93 3.17.0

- Add celery task for publishing pages

1.10.94 3.16.2

- (bug) content rotation on homepage

1.10.95 3.16.1

- (bug) only show published articles on front end

1.10.96 3.16.0

- Add promote and demote dates to article promotion setting
- Remove boolean promotion options
- Data migration to set all articles with feature ticks to have a promotion start date
- Order articles by promotion date

1.10.97 3.15.0

- Enable the sharing of articles to Facebook and Twitter from the article page.

1.10.98 3.14.1

- Change create to get_or_create in migration 47

1.10.99 3.14.0

- Redefine core permissions for groups

1.10.100 3.13.0

- Add clickable front-end tags to articles

1.10.101 3.12.3

- Add migrations for external link

1.10.102 3.12.2

- Signal on page moving and Allow adding external link to banner page

1.10.103 3.12.1

- (bug) search URL was defined using the wrong regex (it broke Service Directory plugin)

1.10.104 3.12.0

- Implement custom error page for CSRF error

1.10.105 3.11.2

- Remove automatic opening of comments when an article is promoted to Topic of the Day

1.10.106 3.11.1

- Exclude future-dated Topic of the Day articles from Latest articles list

1.10.107 3.11.0

- Add Support for both Elastichsearch 1.x & 2.x
- Add ability to show a highlighted term in the results

Note: Search highlighting is only supported by the Elasticsearch backend.

1.10.108 3.10.0

- Add Topic of the Day functionality

1.10.109 3.9.2

- Set GOOGLE_ANALYTICS to None in settings

1.10.110 3.9.1

- Fix the issue with switching between child languages
- Fix the issue with allowing articles to exist in multiple sections

1.10.111 3.9.0

- Update user permissions

1.10.112 3.8.3

- Ensure title is encoded properly for GA

1.10.113 3.8.2

- Ensure title is filled in for GA middleware

1.10.114 3.8.0

- Add custom GA celery middleware
- Use celery for GA instead of gif pixel

1.10.115 3.7.5

- Add middleware to ignore accept language header

1.10.116 3.7.4

- Return the language code for languages that are not supported

1.10.117 3.7.3

- Make sure Locales are not restricted to 2 char codes and we can use the country code

1.10.118 3.7.2

- Return the language code for languages that babel is not supporting

1.10.119 3.7.1

- Make sure unpublished translated pages are not appearing on front end

1.10.120 3.7.0

- Show the translated page on front end when it's main language page is unpublished

1.10.121 3.6.0

- Add the option that untranslated pages will not be visible to the front end user when they viewing a child language of the site

1.10.122 3.5.0

- Add date and time options to content rotation

1.10.123 3.4.2

- Fixed Migration Bug

1.10.124 3.4.1

- Add GA urls to Molo Urls
- Pinned Flake8 to 2.6.2

1.10.125 3.4.0

- Add local and global GA tracking codes

1.10.126 3.3.0

- Add random content rotation for articles featured on homepage

1.10.127 3.2.8

- Add global GA Tag model

1.10.128 3.2.7

- Add get_translation template tag

1.10.129 3.2.6

- Delete the translated page when a page is deleted

1.10.130 3.2.5

- Return Marathon app & version information in the health checks.

1.10.131 3.2.4

- Default count for sections set to 0

1.10.132 3.2.3

- Add session key middleware for each user to use with GTM when javascript is disabled

1.10.133 3.2.2

- Handling import * error with noqa

1.10.134 3.2.1

- Delete translated page when a page is deleted
- Added extra lang info for languages that django doesn't support

1.10.135 3.2.0

- Added wagtail multimedia support
- Allow articles to exist in multiple sections

1.10.136 3.1.11

- Fixed bugs with UC content importing, Arabic slugs and path issue

1.10.137 3.1.10

- Fixed another small bug with UC content validation

1.10.138 3.1.9

- Fixed a bug with UC content validation

1.10.139 3.1.8

- Limit import content to users belonging to *Universal Core Importers* group

1.10.140 3.1.7

- Content validation now happens in a celery task

1.10.141 3.1.6

- Added pagination for articles in section
- Show the active language and display the local name
- Added load_sections template tag

1.10.142 3.1.5

- Importing validation errors to be shown in the UI for celery task

1.10.143 3.1.4

- Upgraded wagtail to 1.4.5
- Effective style hint to support multi-language

1.10.144 3.1.3

- Content import now happens in a celery task

1.10.145 3.1.2

- Added templates for forgot password

1.10.146 3.1.1

- Pined django-cas-ng to 3.5.4

1.10.147 3.1.0

- Upgraded to Django 1.9 and Wagtail 1.4.4

1.10.148 3.0.3

- Improved performance of UC content import

1.10.149 3.0.2

- Changed molo.core version number in get_pypi_version test

1.10.150 3.0.1

- Changed molo.core version number in versions_comparison test

1.10.151 3.0.0

- Added multi-language support
- Added content import from Universal Core content repos (using REACT)
- Renamed `HomePage` module to `BannerPage`
- Updated language switcher url to include `?next={{request.path}}`
- `section_page.html` now uses new template tags (see below)
- `section_listing_homepage.html` now uses new template tags (see below)
- Changed content structure to introduce index pages
- Added GA tag manager field to site settings
- Upgraded wagtail to 1.4.3

2.x

This is the initial release of Molo (1.x was considered beta)

Main features:

```
- Scafolding a Wagtail site with basic models
- Core features including Banners, Sections, Articles, Footer Pages, Search
- Out the box support for plugins (molo.profiles, molo.commenting, molo.yourwords,␣
↪molo.polls)
- Upgraded Wagtail to 1.0
```

1.10.152 2.6.17

- Moved tasks.py to core

1.10.153 2.6.16

- Moved content rotation from cookiecutter to core

1.10.154 2.6.15

- Added automatic content rotation

1.10.155 2.6.14

- Added plugins version comparison
- Added logo as wagtail setting

1.10.156 2.6.13

- Re-release of version 2.6.12 because we forgot to increment the version number.

1.10.157 2.6.12

- Added metadata tag field

1.10.158 2.6.11

- Added social media fields

1.10.159 2.6.10

- Ensure CAS only applies to admin views

1.10.160 2.6.9

- Fixed the issue with CAS not being compatible with normal login

1.10.161 2.6.8

- Updated plugins instructions
- Updated the polls plugin in the documentation

1.10.162 2.6.7

- core urls are not defined correctly

1.10.163 2.6.6

- Bug fixes

1.10.164 2.6.5

- Added search functionality
- Updated core templates

1.10.165 2.6.4

- Added support for Central Authentication Service (CAS)(CAS)

1.10.166 2.6.3

- Updated documentation

1.10.167 2.6.2

- Added missing files in the scaffold (pypi package) 2nd attempt

1.10.168 2.6.1

- Added missing files in the scaffold (pypi package)

1.10.169 2.6.0

- updated documentation
- adding tags to ArticlePage model
- upgraded wagtail to v1.3.1
- better testing base for Molo

1.10.170 2.5.2

- Promoted articles 'featured in latest' will be ordered by most recently updated in the latest section.

1.10.171 2.5.1

- pinned cookiecutter to version 1.0.0

1.10.172 2.4.2

- ordering of articles within a section uses the Wagtail ordering

1.10.173 2.3.7

- bump to official wagtail v1.0
- add health check

1.10.174 2.3.6

- remove first_published_at from models (casuing migration issues)

1.10.175 2.3.3

- added *extra styling hints* field to section page

1.10.176 2.3.2

- allow articles to be featured on the homepage

1.10.177 2.3.1

- *first published at* is not a required field

1.10.178 2.3.0

- add homepage models
- ensure articles ordered by published date
- allow articles to be featured

1.10.179 2.2.1

- Add images to sections
- Add support for sub sections

1.10.180 2.2.0

- Add multi language support

1.10.181 2.1.1

- ensure libffi-dev in sideloader build file

1.10.182 2.1.0

- ensure libffi-dev in sideloader build file

1.10.183 2.1.0

- Add basic models
- Add basic templates
- upgraded to v1.0b2

1.10.184 2.0.5

- Add sideloader scripts

1.10.185 2.0.4

- Fix cookie cutter path

1.10.186 2.0.3

- pypi fix - include cookie cutter json

1.10.187 2.0.2

- Use cookie cutter for a project template

1.10.188 2.0.1

- Fix pypi package manifest

1.10.189 2.0.0

- Initial release