Molo Documentation

Release 5.9.3

Praekelt Foundation

Aug 17, 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

Installation

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

Installing Molo

Molo can be then installed using:

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

Getting Started

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

Scaffold a site using Molo

The goal of Molo is to provide a solid base of proven, stable packages that help Praekelt Foundation 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/.

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 .
```

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
```

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/
```

Molo, Django & settings files

What you have now is a standard Django application set up for normal development like outlined in the Django documentation. The only main 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

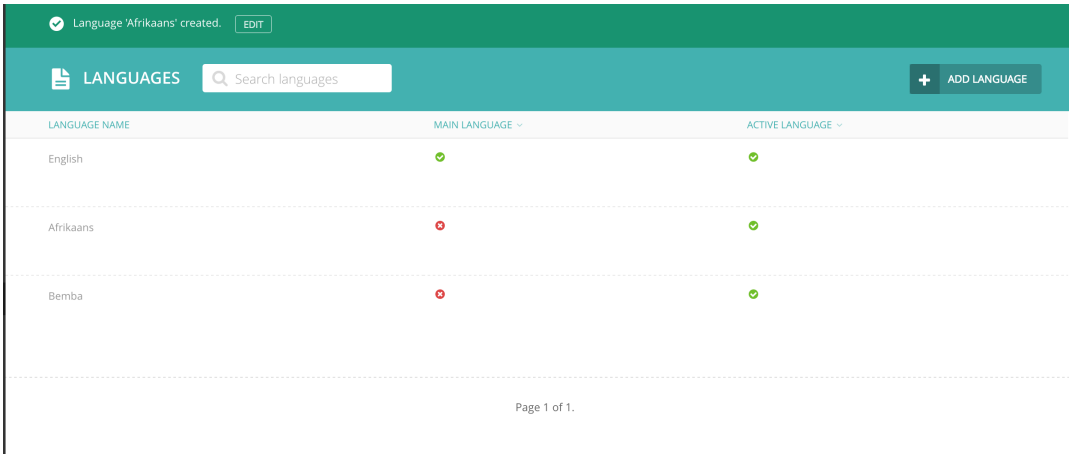
Now develop your application and write tests for the features you add. Running your tests for Django works as you would expect:

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

What is bundled with Molo?

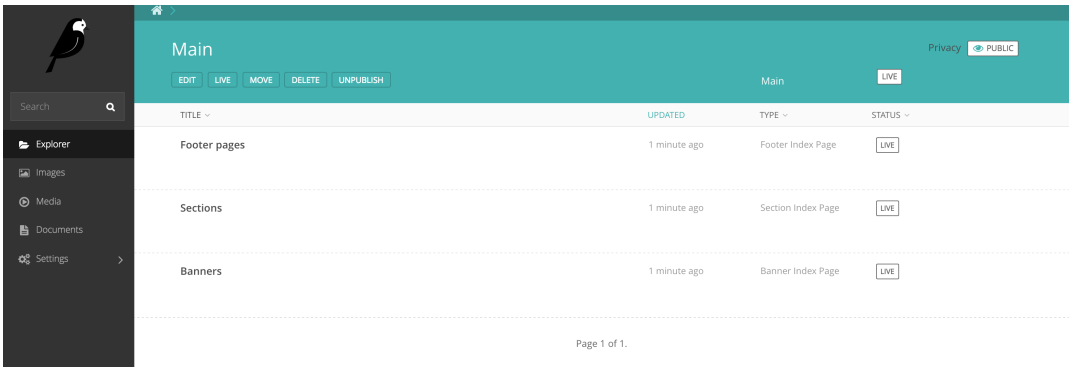
1. Wagtail CMS
2. Basic feature phone template set.
3. Basic models for the following tree structure:
   (a) A site has a main language, and the option of one or more additional languages.

- 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.



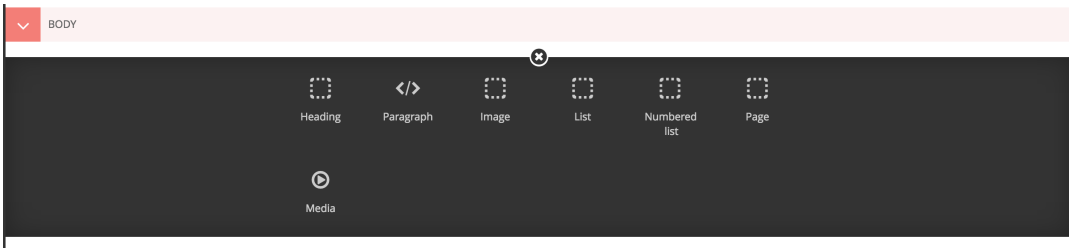
(a) Once a main language has been created, a main page will be created as well. A main page consists of index pages.

- 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.

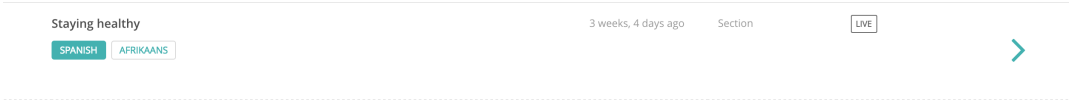


(a) Once a section is made, articles can then be added to that section.

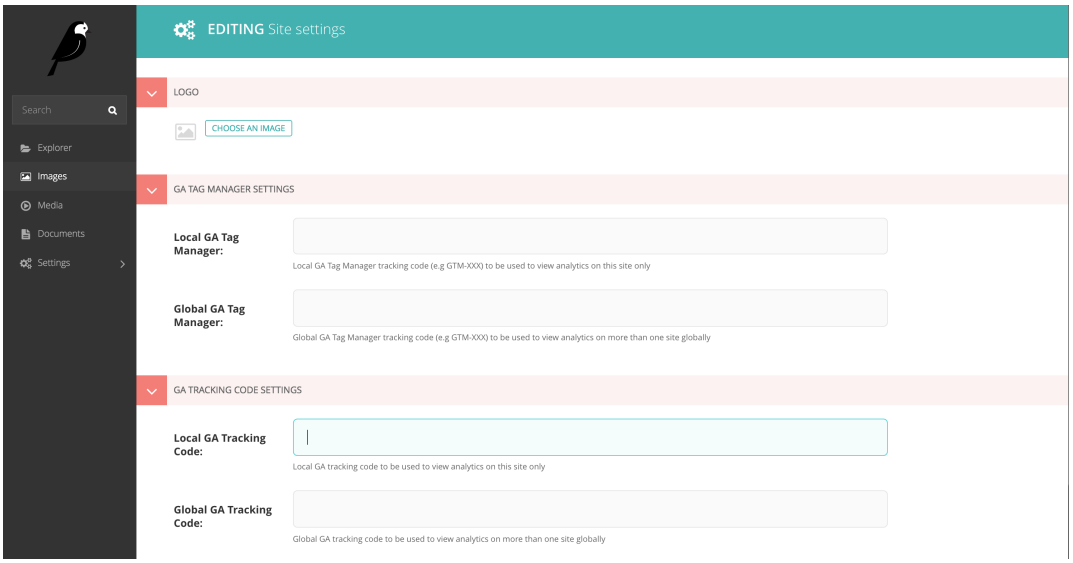
- Articles only exist as a child of a section page.
- Articles are composed from one or more blocks.
- Blocks can be headings, paragraphs, images, lists or links to other pages.



(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.



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.

Plugins

Installing Plugins

Molo plugings are normal python modules and can be installed using pip:

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

Next, you'll need to add the new plugin to your `INSTALLED_APPS` in your `myapp/settings/base.py` (if you did't include it during the scaffolding step using `molo scaffold myapp --include=molo.profiles ^profiles/`):

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

Next, you'll need to add the new plugin urls to your `myapp/urls.py` (if you didn't include it during the scaffolding step using `molo scaffold myapp --include=molo.profiles ^profiles/`):

```
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
```

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.

Core Features

- **Banners**
  - Image banners on the home page that can be linked to any page on the site
- **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
- **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

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 %}
```

- **Multiple Languages**
  - Molo allows you to offer you content in multiple languages using the TranslatablePageMixin

Existing Plugins

The following plugins are available to extend the core features of Molo. Please see Installing plugins for installation details.

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.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

Multiple Languages

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

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)
```

Getting Translations

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

```
competition.get_translation_for(locale)
```

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 return English as the main language:

```
competition.main_language_page
```

Template Tags

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.

The following template tag will return the translation for the page model. It gets the locale from the context and returns the translation for that locale:

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

Using template tags to render content

- The `load_sections` template tag returns all the sections for a specific page.
- The `section_listing_homepage` tag will return all the sections that are featured on the homepage.
- The `latest_listing_homepage` tag will return all the articles that have been promoted to feature on the homepage.
- The `bannerpages` tag will return all the bannerpages.
- The `footer_page` tag will return all the foooterpages.
- The `breadcrumbs` tag will return the current breadcrumbs.
- The `load_descendant_articles_for_section` tag will return all the articles that are children of a section or any of its child sections.
- The `load_child_articles_for_section` tag will return all the articled that are children of a section page.
- The `load_child_sections_for_section` tag will return all the sub-sections that are children of a section page.

Template Patterns

Praekelt Molo Design System

Molo design system a living document of visual design components, elements 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. This way, it's a one-stop place for the entire teams to reference when discussing new site designs and iterations.

The design system will make it easy to build/scaffold custom Molo mobisites from our Python Django 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, 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

Website Glossary

These are the Molo sites:

- TuneMe: https://tuneme.org/
- _ GEM (Girl Effect Mobile)
- Sprinster: http://sa.heyspringster.com/
- FreeBasics: http://amabhungane.molo.site/
- _ *Babycenter*: http://southafrica.babycenter.io/
- IoGT (Internet of Good Things): http://za.goodinternet.org/

Front-end Project Setup

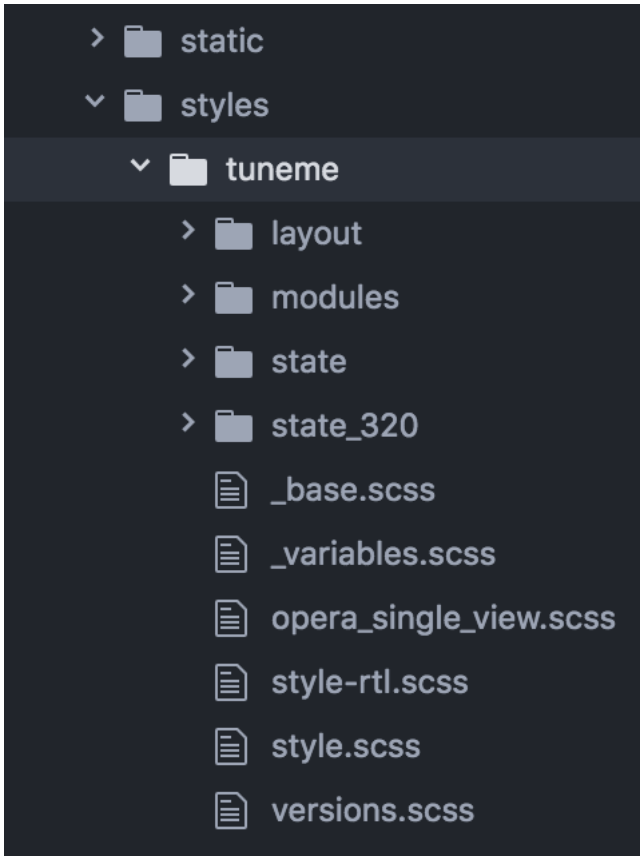
Backend developers tie a collection of reusable apps with a set of guidelines - the guidelines allow developers to share, reuse, maintain and improve the source code quality which does not extend to Frontend source code resulting in unstructured, inconsistent markup with poor quality and maintainability.

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

CHANGE LOG

5.9.3

- Temporarily removed API import from sidebar

5.9.2

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

5.9.1

- Bug Fix: Revert accidental travis setup change

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

5.8.2

- Fix the responsive styling for Admin dashboard

5.8.1

- Fix the styling for Admin dashboard

5.8.0

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

5.7.0

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

5.6.0

- New Feature: Add Article Publish action to shortcuts

5.5.2

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

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

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

5.4.7

- Update static files to fix missing/incorrect references

5.4.6

- Increase character limit on reaction question success message

5.4.5

- Add reaction question success_messages

5.4.4

- Add *get_effective_image* to reaction question choices

5.4.3

- Fix a bug for *get_next_tag* template tag

5.4.2

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

5.4.1

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

5.4.0

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

5.3.1

- Use get_effective_image instead of image in templates

5.3.0

- Add load more functionality to section page

5.2.5

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

5.2.4

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

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

5.2.2

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

5.2.1

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

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

5.1.1

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

5.1.0

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

5.0.4

- Use celery for copying section index contents

5.0.3

- Add parent_page_types to SectionPage

5.0.2

- Fix test for admin url redirect

5.0.1

- Version bump for molo profiles to resolve pin dependencies

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

4.4.13

- Insure content demotion happens for each section individually

4.4.12

- Remove promotion settings from footer pages

4.4.11

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

4.4.10

- Fixed recommended article ordering in templatetag logic

4.4.9

- Added Non routable page mixin

4.4.8

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

4.4.7

- Fixed random test failures in content rotation test

4.4.6

- consolidate minute tasks into 1 call

4.4.5

- consolidate minute tasks into 1 call

4.4.4

- Fixed bug for previewing pages

4.4.3

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

4.4.2

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

4.4.1

- Updated template overrides to fix missing Page admin buttons

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

4.3.3

- Add django clearsessions to celery tasks

4.3.2

- Added missing classes in custom admin template

4.3.1

- Fixed template error

4.3.0

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

4.2.0

- added multi-language next and recommended article feature

4.1.0

- Add sitemap - include translations

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

4.0.2

- Fixed template overrides for django-admin templates

4.0.1

- Added upload/download functionality for zipped media files

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

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 %}`

3.17.4

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

3.17.3

- Ensure email address is set when using SSO

3.17.2

- Put ForceDefaultLanguageMiddleware before django.middleware.locale.LocaleMiddleware

3.17.1

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

3.17.0

- Add celery task for publishing pages

3.16.2

- (bug) content rotation on homepage

3.16.1

- (bug) only show published articles on front end

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

3.15.0

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

3.14.1

- Change create to get_or_create in migration 47

3.14.0

- Redefine core permissions for groups

3.13.0

- Add clickable front-end tags to articles

3.12.3

- Add migrations for external link

3.12.2

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

3.12.1

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

3.12.0

- Implement custom error page for CSRF error

3.11.2

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

3.11.1

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

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.

3.10.0

- Add Topic of the Day functionality

3.9.2

- Set GOOGLE_ANALYTICS to None in settings

3.9.1

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

3.9.0

- Update user permissions

3.8.3

- Ensure title is encoded properly for GA

3.8.2

- Ensure title is filled in for GA middleware

3.8.0

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

3.7.5

- Add middleware to ignore accept language header

3.7.4

- Return the language code for languages that are not supported

3.7.3

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

3.7.2

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

3.7.1

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

3.7.0

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

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

3.5.0

- Add date and time options to content rotation

3.4.2

- Fixed Migration Bug

3.4.1

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

3.4.0

- Add local and global GA tracking codes

3.3.0

- Add random content rotation for articles featured on homepage

3.2.8

- Add global GA Tag model

3.2.7

- Add get_translation template tag

3.2.6

- Delete the translated page when a page is deleted

3.2.5

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

3.2.4

- Default count for sections set to 0

3.2.3

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

3.2.2

- Handling import * error with noqa

3.2.1

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

3.2.0

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

3.1.11

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

3.1.10

- Fixed another small bug with UC content validation

3.1.9

- Fixed a bug with UC content validation

3.1.8

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

3.1.7

- Content validation now happens in a celery task

3.1.6

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

3.1.5

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

3.1.4

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

3.1.3

- Content import now happens in a celery task

3.1.2

- Added templates for forgot password

3.1.1

- Pined django-cas-ng to 3.5.4

3.1.0

- Upgraded to Django 1.9 and Wagtail 1.4.4

3.0.3

- Improved performance of UC content import

3.0.2

- Changed molo.core version number in get_pypi_version test

3.0.1

- Changed molo.core version number in versions_comparison test

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
```

2.6.17

- Moved tasks.py to core

2.6.16

- Moved content rotation from cookiecutter to core

2.6.15

- Added automatic content rotation

2.6.14

- Added plugins version comparison
- Added logo as wagtail setting

2.6.13

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

2.6.12

- Added metadata tag field

2.6.11

- Added social media fields

2.6.10

- Ensure CAS only applies to admin views

2.6.9

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

2.6.8

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

2.6.7

- core urls are not defined correctly

2.6.6

- Bug fixes

2.6.5

- Added search functionality
- Updated core templates

2.6.4

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

2.6.3

- Updated documentation

2.6.2

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

2.6.1

- Added missing files in the scaffold (pypi package)

2.6.0

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

2.5.2

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

2.5.1

- pinned cookiecutter to version 1.0.0

2.4.2

- ordering of articles within a section uses the Wagtail ordering

2.3.7

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

2.3.6

- remove first_published_at from models (casuing migration issues)

2.3.3

- added *extra styling hints* field to section page

2.3.2

- allow articles to be featured on the homepage

2.3.1

- *first published at* is not a required field

2.3.0

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

2.2.1

- Add images to sections
- Add support for sub sections

2.2.0

- Add multi language support

2.1.1

- ensure libffi-dev in sideloader build file

2.1.0

- ensure libffi-dev in sideloader build file

2.1.0

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

2.0.5

- Add sideloader scripts

2.0.4

- Fix cookie cutter path

2.0.3

- pypi fix - include cookie cutter json

2.0.2

- Use cookie cutter for a project template

2.0.1

- Fix pypi package manifest

2.0.0

- Initial release