wafer Documentation

Release 0.16.1

The wafer development team

Mar 11, 2024

CONTENTS

Contents:

CHAPTER

ONE

INSTALLATION

1.1 Supported versions

Wafer supports Django 3.2, 4.0-4.2, 5.0 and Python 3.8 to 3.12.

1.2 Requirements

In addition to Django, wafer has some requirements on external libraries. They're listed in `setup.py`.

1.3 Basic Dev install

1. Install all the dependencies `pip install -r requirements.txt`
2. Create the initial database schema `manage.py migrate`
3. If you don't have one yet, create a superuser with `manage.py createsuperuser`.
4. Wafer uses npm to manage front-end dependencies
   - Make sure you have a recent version of Node.js installed that includes `npm`.
   - Run `npm install` to install all dependencies, which also copies them to `wafer/static/vendor`.
5. Wafer uses the Django caching infrastructure in several places, so the cache table needs to be created using `manage.py createcachetable`.
6. Create the default 'Page Editors', 'Talk Mentors', and 'Talk Reviewers' groups using `manage.py wafer_add_default_groups`.
7. Log in and configure the Site:
   - The domain will be used as the base for emails sent during registration.
   - The name will be the conference's name.
   - By default, wafer assumes that the site will be accessible over SSL, so the registration emails will use an 'https' prefix. If this is not the case, override the `wafer/registration/activation_email.txt` template.
8. Ensure the permissions on the `MEDIA_ROOT` directory are correctly set so the webserver can create new files there. This location is used for files uploaded for pages and sponsor information.
9. Have a fun conference.

1.4 Recommended production setup

1. Create a new Django app, in your own VCS repository. Add wafer (probably pinned) as a requirement.
2. Include wafer's `wafer.settings` in your `settings.py`:

```
from wafer.settings import *

TIME_ZONE = 'Africa/Johannesburg'
...
```

3. You'll want to include wafer's default values for some settings, e.g. `INSTALLED_APPS`, rather than completely overriding them. See *Settings* for the wafer-specific settings.
4. Override templates as necessary, by putting your own templates directory early in `TEMPLATES`.
5. And then continue with the basic instructions above.

1.5 Example setup

For an example of a conference using wafer, see the 2017 PyCon ZA conference repository, available from github

CHAPTER

TWO

PAGES

2.1 Basic pages

Pages are used to describe static information for the conference.

The contents can be formatted using markdown syntax and images can be uploaded using the `files` field.

The `slug` defines the last part of the path.

The parent field is used to group the page under specific parts of the namespace. A page with the slug `announcements` and the parent `news` will have a URL of `/news/announcements`

2.2 Container pages

Container pages are created to act as parents for other pages. These should have minimal content, as they will typically not be displayed on the site, and should be excluded from the static site generation.

2.3 Files

Additional files, such as images, can be uploaded so they can be referenced in page. These files are placed in `MEDIA_ROOT/pages_files` by default. This location needs to be writeable by the webserver for uploads to work.

2.4 Maintaining pages in files

Pages live in the database, and can be edited through the web UI. However, it can be useful to store them in files (e.g. in a git repo, with the site source code).

There is a management command (`load_pages`) that will read pages from files into the database. It requires `PyYAML` to be installed.

Pages must be stored as markdown in a directory, in the same hierarchy as the desired URL structure. The `PAGE_DIR` Django setting should be an absolute path to root directory of this hierarchy, beginning and ending with `/`. e.g.:

```
/app/pages/                          ← PAGE_DIR
/app/pages/index.md                  ← Home Page: /
/app/pages/about.md                  ← Container Page: /about/
/app/pages/about
/app/pages/about/the-conference.md   ← /about/the-conference/
```

Each page starts with a YAML front-matter (similar to Jekyll), and is then followed by the Markdown page body. e.g. `pages/index.md`:

```
---
name: Index
---
Welcome to Foo Conf (not Foo Conf, that's another thing entirely)!

We invite you to [join us](/attend/) at [our venue](/venue/)
on the 31st of December for a day of fun conferencing.
```

The front matter can contain a couple of flags:

published
: If set to `false`, the page will not be loaded by the `load_pages` command.

include_in_menu
: If set to `true`, the page will be added to the menu structure.

menu_order
: If set to a number, controls the order in which pages are listed in the menu. Pages with lower values come first.

exclude_from_static
: If set to `true`, the page will not be archived to static HTML by *staticsitegen*.

CHAPTER

THREE

TALKS

3.1 Talk Properties

Talks have a title, an abstract / description and authors. These fields will be the publicly visible information about the talk once the talk is accepted.

In addition, talks have a `notes` field, which the submitters can use to provide additional private information about the talk, such as specialised equipment requirements.

The talks also have a `private notes` field that is only visible to organisers, which can be used to track any additional information on the talk, such as assigned reviewers and so forth.

3.2 Talk Types

Before opening up talk submissions, define the talk types available to talk submitters, such as Tutorial, Short Talks and so forth.

Each `Talk Type` can be opened or closed for submissions individually via the admin interface. Both the global `WAFER_TALKS_OPEN` setting and the individual `Talk Type` must be set to allow submissions for submissions of the given type to be accepted.

There is a Django view at `/talks/types` which displays the list of types and their descriptions. By default, this list isn't linked to the menu, since it's intended to be linked to by pages describing the talks and talk submission process.

3.3 Submitting Talks

Users can submit talks from their profile page using the `Submit Talk Proposal` option. The abstract can be formatted using Markdown.

The notes section is only visible to the talk author, talk mentors and admins. It is intended for providing extra information and recording developments that are useful to associate with the talk, but should not be public.

Talks can have multiple authors, but only one corresponding author. Only the corresponding author can edit the talk submission.

3.4 Talk Mentors

The "Talk Mentors" group has permission to do the following through the `/talks/` (public) talk interface, and the detail pages of each talk:

- View *all* talk submissions (not just accepted talks).
- Edit submitted talks.
- View and edit the notes submitted along with a talk, which are visible to the talk submitter.
- View and edit the private notes which are only visible to the "Talk Mentors", "Talk Reviewers", and administrators by default.

3.5 Talk Reviewers

The "Talk Reviewers" group has permission to do the following through the `/talks/` (public) talk interface, and the detail pages of each talk:

- View *all* talk submissions (not just accepted talks).
- View the notes submitted along with a talk, which are visible to the talk submitter.
- View the private notes which are only visible to the "Talk Mentors", "Talk Reviewers", and administrators by default.
- Leave reviews on talks. See *Talk Reviews*.

3.6 Managing talks from the admin interface

From the admin interface talks can be modified, and the status can be updated as required.

Talks can have following states:

- Submitted
- Under Consideration
- Withdrawn
- Provisionally Accepted
- Accepted
- Cancelled
- Not Accepted

When a talk is first submitted, the state is set to `Submitted`.

Once the talk has received its first review, the state of the talk will change to `Under Consideration`. It can also be changed manually, if not using wafer's review system.

While the talk is either `Submitted` or `Under Consideration`, the submitter can withdraw the talk from consideration, which sets the state of the talk to `Withdrawn`. The submitter can also edit and update the talk abstract while the talk is in these two states.

Once a decision one the talk has been made, the talk should be set to either `Accepted` or `Not Accepted`. For conferences where a submitter needs to confirm attendance before the decision is finalised, the status can be set to `Provisionally Accepted` for talks waiting for confirmation. Once a talk is in any of these states, it can no longer

be edited by the submitter. `Not Accepted` and `Provisionally Accepted` talks are not publicly visible, while `Accepted` talks are public.

If for some reason, an `Accepted` talk cannot be given, it can be marked as `Cancelled`. `Cancelled` talks are still public, so that cancelled a scheduled talk does not invalidate the schedule.

If the talk has a type that has `show_pending_submissions` enabled, then it will be public immediately after submission, in any of the `Submitted`, `Under Consideration`, and `Provisionally Accepted` states.

3.7 Talk tracks

Wafer optionally supports multiple talk tracks. Create the tracks in the admin interface. If there are multiple tracks, submitters will be asked to choose a track for each submission. If there are no tracks specified, the option will be hidden from the submitter.

Currently, tracks merely provide extra information for talk reviewers and attendees.

There is a Django view at `/talks/tracks` which displays the list of tracks and their descriptions. As with talk types, this list isn't linked to the menu, since it's intended to be linked to by pages describing the talks and talk submission process.

3.8 Talk Reviews

Talk Mentors can review talks by metrics chosen by the administrators. Create the desired metrics in the admin interface.

Reviewers ("Talk Reviewers" team by default) see a "Review" button on talk pages, and will be prompted to review each talk by each defined metric. The reviews are a score for each metric (in the range of -2 to 2, by default, configurable via `WAFER_TALK_REVIEW_SCORES`). The reviewer can also leave a textual review, in Markdown.

If a reviewer re-reviews a talk, it just updates the previous review.

The aggregate reviews are visible in the talk admin.

In the public talk listing, reviewers will see a symbol next to talks they have reviewed. It will change to a clock symbol, if their review is out of date (someone has changed the talk, since the review was last updated).

3.9 Talk URLs

URLs can be associated with talks using the admin interface. This is intended for adding links to slides and videos of the talk after the conference.

CHAPTER

FOUR

SPONSORS

4.1 Packages

Sponsor packages describe the details of the various sponsor packages available.

The `order` field controls the order in which packages are listed on the sponsor packages page.

4.2 Sponsors

This is used to add details of the sponsors.

The description can be formatted using markdown syntax.

Images can be uploaded and used in the description using the files field.

4.3 Files

Additional files, such as images, can be uploaded so they can be referenced. These files are placed in `MEDIA_ROOT/sponsors_files` by default. This location needs to be writeable by the webserver for uploads to work.

4.3.1 Using files in templates

Uploaded files can be associated with a sponsor and a name in the admin interface which can be used with the `sponsor_tagged_image` templatetag in the templates.

The default wafer sponsor templates expect each sponsor to have an image labelled `main_logo` for use in the sponsor list.

Wafer also provides an example template block for adding sponsors as a footer to pages called `sponsors_footer`. This expects images labelled `footer_logo`.

CHAPTER

FIVE

SCHEDULE

5.1 Permissions

Setting up the schedule blocks and using the schedule editor requires access to the admin site.

5.2 Specifying Block and Venues

The first things that need to be specified are the blocks for the schedule and the venues available.

Blocks are the parts that the schedule is divided into. Typically they correspond to days of the conference, but they can be longer or shorter depending on the needs. Each block can be rendered independently of the others, provided there are no errors in the schedule.

Each block has a start and end date and time. These can be on different days, to allow for events that go past midnight.

Each venue is associated with a number of blocks, and is assumed not to be available if it hasn't been assigned to the corresponding block in the schedule. At times when a venue is not available, it will not appear in the schedule.

5.3 Slots

The fundamental unit of the schedule is a schedule slot. Each slot is assigned to a given block, and has a start and end date and time. The start time may be specified as the end time of a different slot using the `previous_slot`.

The times of a slot must be within the times given for it's associated block.

Each slot can have a name to make it easier to distinguish.

Slots cannot overlap, but items can use multiple slots, so this can be emulated by breaking the slots down into small enough time intervals.

5.4 Assigning items to slots

Each item in the schedule has a number of slots, a venue and either a talk or a page. Each talk can only be assigned to a single schedule item, but pages can be assigned to multiple schedule items to make it easy to add items such as tea breaks to the schedule.

If the schedule item has been assigned to a page, the details field can be used to override the information from the page. For talks, details will be added to the information from the talk.

5.5 Schedule views

The schedule can be restricted to a single block by specifying the `day` parameter in the URL - e.g. `https://localhost/schedule/?day=2014-10-23`. If the specified day cannnot be matched to one of the blocks in the schedule, the full schedule is shown.

By passing using the `highlight-venue` parameter in the url, all items in a specific venue will have the `schedule-highlight-venue` class, which can be used to style these differently - e.g. `https://localhost/schedule/?highlight-venue=3` will annotate all items occuring in the venue with the id 3. Invalid ids will be ignored.

The `schedule/current` view can be used to show events around the current time. The `refresh` parameter can be used to add a refresh header to the view - e.g `https://localhost/schedule/current/?refresh=60` will refresh every 60 seconds.

Note that the current time is the time of the webserver. If this is in a different timezone from the conference, the correct `TIME_ZONE` value should be set in the `settings.py` file.

A specific time can be passed via the `time` parameter to the current view, specified as `HH:mm` e.g. `https://localhost/schedule/current/?time=08:30` will generate the current view for 8:30 am.

5.6 Styling notes

The entry for each talk gets a custom CSS class derived from the talk type. This constructed CSS class is shown in the Talk Type admin view.

Schedule items which are not talks have `talk-type-none` as the CSS class.

A per item CSS class can also be set using the `css_class` attribute on the schedule item.

5.7 Adding additional schedule validation

Wafer runs validation on the slots and the schedule items. This behaviour can be extended by providing custom validators.

Each slot validator is called with a list of all the slots, and each schedule item validator is called with a list of all schedule items. Validators are expected to return a list of invalid items or an empty list if the validator finds no error.

Use `register_schedule_item_validator` and `register_slot_validator` to add the validators to the list.

To display the errors in the admin form, you will also need to extend the `displayerrors` block in `scheduleitem_list.html` and `slot_list.html` templates.

5.8 Schedule fails to render

To avoid displaying misleading or incorrect information to attendees, the schedule will not be rendered if the schedule fails to validate, and it will display a "The final schedule has not been published" message instead.

Users with permissions to use the schedule editor will see a list of validation errors as well, to help diagnose the problem preventing the schedule from rendering correctly. These errors will also be displayed in the schedule editor and in the admin site.

5.9 Hiding the schedule while editing

The setting `WAFER_HIDE_SCHEDULE` will prevent the schedule from rendering for users without admin access. Users with admin access will see a note that the schedule is a draft and not public.

CHAPTER

SIX

MENUS

6.1 Overview

Wafer includes a simple system for generating either static or dynamic menus for the navigation bar at the top of each page.

A single level of sub-menus is supported.

6.2 Static menus

Static menus are configured using the `WAFER_MENUS` setting. `WAFER_MENUS` is a list of either sub-menus or menu items.

Menu items have the following keys:

`label`
: The text displayed for the menu item.

`url`
: The URL the item links to.

`sort_key`
: A value used to sort the list of items into a custom order (optional).

`image`
: An absolute or relative URL to an image to display instead of the label (optional).

Sub-menu have the keys:

`menu`
: The unique name of the sub-menu.

`label`
: The text to display for the sub-menu.

`items`
: A list of menu items in the sub-menu (sub-sub-menus are not supported).

`sort_key`
: A value used to sort the list of items into a custom order (optional).

Example snippet from `settings.py`:

```
from django.utils.translation import gettext_lazy as _
from django.core.urlresolvers import reverse_lazy
```

(continues on next page)

(continued from previous page)

```
WAFER_MENUS += (
    {"menu": "about", "label": _("About"), "items": []},
    {"name": "venue", "label": _("Venue"),
     "url": reverse_lazy("wafer_page", args=("venue",))},
    {"menu": "sponsors", "label": _("Sponsors"),
     "items": []},
    {"menu": "talks", "label": _("Talks"),
     "items": [
         {"name": "schedule", "label": _("Schedule"),
          "url": reverse_lazy("wafer_full_schedule")},
         {"name": "accepted-talks", "label": _("Accepted Talks"),
          "url": reverse_lazy("wafer_users_talks")},
         {"name": "speakers", "label": _("Speakers"),
          "url": reverse_lazy("wafer_talks_speakers")},
     ]},
}
```

The empty sub-menus are populate dynamically (see the next section).

6.3 Dynamic menus

Dynamic menus are configured using the `WAFER_DYNAMIC_MENUS` setting. `WAFER_DYNAMIC_MENUS` is a list of functions or names of functions to call to dynamically add sub-menus or menu items.

Dynamic sub-menus and menu items are added after static ones.

By default, two kinds of menu items are dynamically generated:

- Menus and menu items for pages.
- A menu and menu items for sponsors.

Example snippet from `settings.py`:

```
WAFER_DYNAMIC_MENUS = (
    'wafer.pages.models.page_menus',
    'wafer.sponsors.models.sponsor_menu',
)
```

6.4 Page menus

Page menus are generated by `wafer.pages.models.page_menus`. They appear when a page is marked for inclusion in the navigation menu. Each page selected for inclusion as a menu entry with the page name that links to the page. If the page has a parent, it appears in a sub-menu named after its root ancestor.

6.5 Sponsor menu

The sponsor menu is a single sub-menu named `sponsors`. It lists the sponsors in order of precedence with links to their sponsor pages and includes links to the full list of sponsors and the list of sponsorship packages at the bottom of the sub-menu.

CHAPTER

SEVEN

STATIC SITE GENERATION

7.1 Usage

The `manage.py build` command will generate a static version of the site using django-bakery.

The static site will include pages, talks, sponsors and user details.

You need to exclude container pages used for the menus from the static site using the "exclude from static" option in the admin interface, otherwise it will attempt to create files with the same name as the containing directories and the export will fail. If this happens, simply correct the problematic pages and rerun the command.

We suggest setting `WAFER_HIDE_LOGIN` to `True` when generating the static site so there is no login button on the static site.

CHAPTER

EIGHT

SETTINGS

Wafer has several Django settings that control its behaviour. It attempts to provide reasonable defaults for these, (and Django in general), in the `wafer.settings` module, so you can import this in your app's `settings.py`, and then override things you want to change.

8.1 Wafer's settings

CODE_HOSTING_ENTRIES
A dictionary of code hosting sites for the user profile. Entries should be of the form: **keyname**: "Description" .

Each entry in the dictionary will be added as a form field on the profile page - entries are assumed to be for urls to the approriate site.

Entries in this dictionary will be grouped together on the edit profile form and on the profile display.

PAGE_DIR
The directory that the `load_pages` management command will load pages from. Should be an absolute path with a trailing `/`.

SOCIAL_MEDIA_ENTRIES
A dictionary of social sites for the user profile. Entries should be of the form: **keyname**: "Description" .

Each entry in the dictionary will be added as a form field on the profile page - entries are assumed to be for urls to the approriate site.

Entries in this dictionary will be grouped together on the edit profile form and on the profile display.

WAFER_CACHE
The name of the Django cache backend that wafer can use. Defaults to `'wafer_cache'`.

WAFER_CONFERENCE_ACRONYM
The abbreviated name of the conference.

WAFER_DEFAULT_GROUPS
A list of groups that any new user is automatically added to. This can be used to tweak the default permissions available to website users by creating groups with the required access.

WAFER_DYNAMIC_MENUS
A list of functions to call to generate additional menus.

WAFER_GITHUB_CLIENT_ID
The client ID for GitHub SSO. Used when `WAFER_SSO` contains `'github'`. If you set this up, they'll provide you with one.

WAFER_GITHUB_CLIENT_SECRET
The secret for GitHub SSO. Used when `WAFER_SSO` contains `'github'`. If you set this up, they'll provide you with one.

WAFER_GITLAB_CLIENT_ID
The client ID for GitLab SSO. Used when `WAFER_SSO` contains `'gitlab'`. If you set this up, they'll provide you with one.

WAFER_GITLAB_CLIENT_SECRET
The secret for GitLab SSO. Used when `WAFER_SSO` contains `'gitlab'`. If you set this up, they'll provide you with one.

WAFER_GITLAB_HOSTNAME
The hostname of the GitLab instance used for SSO. Defaults to `gitlab.com`. Used when `WAFER_SSO` contains `'gitlab'`.

WAFER_HIDE_LOGIN
A boolean flag. When `True`, the login link in the menu is hidden. This is useful to set, before making a site static.

WAFER_MENUS
Static menu structure for the site. This is a list of dicts, with the keys:

label
The text in the menu.

url
The URL to link to.

items
An optional list of similar dicts, making up a submenu.

WAFER_PAGE_MARKITUP_FILTER
Configuration for django-markitup. The type of markup used for pages, only.

`MARKITUP_FILTER` is used for rendering other objects. This allows a more relaxed security configuration for pages, where XSS is less of a risk, and embedded HTML markup can be useful for styling.

WAFER_PUBLIC_ATTENDEE_LIST
A boolean flag. When `True`, all registered users' profiles are publicly visible. Otherwise, only users with associated public talks have public profiles.

WAFER_REGISTRATION_MODE
The mechanisms users will register for the conference, with. Possible options are:

'ticket'
For Quicket integration. The default.

'custom'
For your own implementation. See `WAFER_USER_IS_REGISTERED`.

WAFER_REGISTRATION_OPEN
A boolean flag. When `True`, users can register for the conference. (Note, this is not the same as signing up for an account on the website.)

WAFER_SSO
A list of SSO mechanisms in use. Possible options are: `'github'`, `'gitlab'`.

WAFER_TALK_FORM
The form used for talk/event submission. There is a reasonable default form, but this can be changed to customise the submission process.

WAFER_TALK_LANGUAGES
A tuple of tuples, indicating the languages that users can select when submitting talks. Each tuple has the language code as the first element, and the language name as the second element. Example: `(("en", "English"), ("pt", "Portuguese"))`. The first language listed will be considered the default language, and will be selected by default on new submissions.

WAFER_TALK_REVIEW_SCORES
A tuple of 2 integers. The range of values for talk reviews. Inclusive.

WAFER_TALKS_OPEN
A boolean flag. When `True`, users can submit talks.

WAFER_TICKETS_SECRET
The secret for the Quicket API. Used when `WAFER_REGISTRATION_MODE` is `'ticket'`.

WAFER_USER_IS_REGISTERED
A function, which takes a user, and determines if they have registered for attendance at the conference. It should return a boolean result. The default function checks for a Quicket ticket.

WAFER_USER_TICKET_TYPES
A function which returns a list of ticket types associated with a user. This is intende to help track remote vs in-person tickets and similar cases. It should return a list of ticket type descriptions. The default function returns the types of any Quicket tickets associated with the user.

WAFER_VIDEO
A boolean flag. When `True`, the default talk submission form will ask for a video release from the submitter.

WAFER_VIDEO_LICENSE
The name of the license that the conference's videos will be released under. Talk submitters will be asked to release their video under this license.

WAFER_VIDEO_LICENSE_URL
Link to the full text of `WAFER_VIDEO_LICENSE`.

WAFER_VIDEO_REVIEWER
A boolean flag. When `True`, the default talk submission form will ask for the email address of someone who will review the talk's video, once it is ready to publish.

8.2 Third party settings

Some libraries that wafer uses have settings that you may want to configure. This is a non-complete list of them, see the individual project's documentation for more details.

ACCOUNT_ACTIVATION_DAYS
Used by django-registration-redux. Number of days that users have to click the account activation link that was emailed to them.

MARKITUP_FILTER
Configuration for django-markitup. The type of markup used for talk abstracts, user profiles, and other things. Also, configuration for the conversion, such as allowing arbitrary HTML embedding.

`WAFER_PAGE_MARKITUP_FILTER` is used for rendering pages, which usually have a lower security risk to other markup on the site.

BUILD_DIR
Used by django-bakery. The directory that static versions of the sites are rendered to.

REGISTRATION_OPEN
Boolean flag. Used by django-registration-redux. When `True`, user sign-up is permitted.

REGISTRATION_FORM

Dotted path. Used by django-registration-redux. We provide `wafer.registration.forms.WaferRegistrationForm` to validate usernames.

CHAPTER

NINE

TRANSLATIONS

9.1 Translating wafer

Translations for wafer are managed at weblate.org

9.2 Managing translations

The summary is:

- Add weblate as a remote `git remote add weblate https://hosted.weblate.org/git/wafer/wafer/`
- Pull weblate updates `git remote update weblate`
- Merge translations into wafer `git merge weblate/master`
- Fix any merge issues and create a PR on github

See the **`weblate`_** documentation for more details on how to pull and merge translations.

To regenerate the django.pot file, use `./manage.py makemessages --keep-pot`.

CHAPTER

TEN

INDICES AND TABLES

- genindex
- modindex
- search