svSite Documentation

Release 0.6

Mark

April 18, 2016

Contents

svSite is a project to create reusable website code that is usable for small or medium sized associations. It will be usable for communication within the organisation, creating and promoting events, displaying news and static information, etc.

CHAPTER 1

Status

Under development; not usable yet. Please stand by for release v1.0!

CHAPTER 2

Resources

- Documentation: http://svsite.readthedocs.org/
- Bugs, features: https://github.com/mverleg/svsite/issues

CHAPTER 3

License

The code is available under the BSD License. You can free (and encouraged) to use it as you please, but at your own risk.

CHAPTER 4

Using

4.1 Installation

svSite should provide a fully functional site with minimal work.

Although later on you may want to personalize the look (info), which will take time. That's inevitable.

To get svSite running, follow the steps in the appropriate section.

4.1.1 Linux / bash

Installing dependencies

For this to work, you will need `python3-dev` including `pip` and a database (`sqlite3` is default and easy, but slow). Things will be easier and better with `virtualenv` or `pew` and `git`, so probably get those too. You'll also need `libjpeg-dev` and the dev version of Python because of `pillow`. You can install them with:

```
sudo apt-get install python3-dev sqlite3 git libjpeg-dev python-pip
sudo apt-get install postgresql libpq-dev       # for postgres, only if you want that database
sudo apt-get install mysql-server mysql-client  # for mysql, only if you want that database
```

Get the code. The easiest way is with git, replacing `SITENAME`:

```
git clone https://github.com/mverleg/svsite.git SITENAME
```

Enter the directory (`cd SITENAME`).

Starting a virtual environment is recommended (but optional), as it keeps this project's Python packages separate from those of other projects. If you know how to do this, just do it your way. This is just one of the convenient ways:

```
sudo pip install -U pew
pew new --python=python3 sv
```

If you skip this step, everything will be installed system-wide, so you need to prepend `sudo` before any *pip* command. Also make sure you're installing for Python 3.

Install the necessary Python dependencies through:

```
pip install -r dev/requires.pip
pip install psycopg2     # for postgres, only if you want that database
pip install mysqlclient  # for mysql, only if you want that database
```

Development

If you want to run tests, build the documentation or do anything other than simply running the website, you should install (otherwise skip it):

```
pip install -r dev/requires_dev.pip  # optional
```

Database

We need a database. SQLite is used by default, which you could replace now or later (see local settings) for a substantial performance gain. To create the structure and an administrator, type this and follow the steps:

```
python3 source/manage.py migrate --settings=base.settings_migration
python3 source/manage.py migrate --settings=settings
python3 source/manage.py createsuperuser
```

Static files

Then there are static files we need, which are handles by bower by default [1]. On Ubuntu, you can install bower using:

```
sudo apt-get install nodejs
npm install bower
```

After that, install the static files and connect them:

```
python3 source/manage.py bower install
python3 source/manage.py collectstatic --noinput
```

Starting the server

Then you can start the test-server. This is not done with the normal `runserver` command but with

```
python3 source/manage.py runsslserver localhost.markv.nl:8443 --settings=base.settings_development
```

We use this special command to use a secure connection, which is enforced by default. In this test mode, an unsigned certificate is used, so you might have to add a security exception.

You can replace the url and port. You can stop the server with `ctrl+C`.

Next time

To **(re)start the server** later, go to the correct directory and run:

```
pew workon sv  # only if you use virtualenv
python3 source/manage.py runsslserver localhost.markv.nl:8443 --settings=base.settings_development
```

Note that this is just for development! When the website is going live, you should probably use a webserver such as Apache.

[1] If you don't want to install node and bower, you can easily download the packages listed in *dev/bower/json* by hand and put them in *env/bower*. Make sure they have a *dist* subdirectory where the code lives. You still need to run the `collectstatic` command if you do this.

4.2 Machine-specific settings

Some settings are machine-dependent, so you need to create `local.py` containing these settings. This file should be in the same directory as `settings.py`, so typically `source/local.py`.

At least, your local settings should contain:

```
from os.path import dirname, join

BASE_DIR = dirname(dirname(__file__))

SITE_URL = 'svleo.markv.nl'  #todo: update url
ALLOWED_HOSTS = [SITE_URL, 'localhost']

SECRET_KEY = ''  #todo: generate a long random string

DATABASES = {  #todo: choose some database settings
        'default': {
                'ENGINE': 'django.db.backends.postgresql',
                'NAME': 'database',
                'USER': 'username',
                'PASSWORD': 'PASSWORD',
                'HOST': '127.0.0.1',
                'CONN_MAX_AGE': 120,
        }
}
# alternatively, as a deveopment database:
# DATABASES = {
#       "default": {
#               "ENGINE": "django.db.backends.sqlite3",
#               "NAME": join(BASE_DIR, 'dev', 'data.sqlite3'),
#       }
# }

MEDIA_ROOT = join('data', 'media', 'svleo')
STATIC_ROOT = join('data', 'static', 'svleo')
CMS_PAGE_MEDIA_PATH = join(MEDIA_ROOT, 'cms')
SV_THEMES_DIR = join(BASE_DIR, 'themes')
```

You can create a secret key using random.org (join both together), or generate a better one yourself with bash:

```
</dev/urandom tr -dc '1234567890!@#$%&*--+=__qwertQWERTasdfgASDFGzxcvbZXCVB' | head -c 32
```

You might also want to have a look at some of these:

```
SV_DEFAULT_THEME = 'standard'

TIME_ZONE = 'Europe/Amsterdam'
LANGUAGE_CODE = 'nl'
LANGUAGES = (
        ('nl', ('Dutch')),  # using gettext_noop here causes a circular import
        ('en', ('English')),
)

CACHES = {
        'default': {
                'BACKEND': 'django.core.cache.backends.memcached.MemcachedCache',
                'LOCATION': '127.0.0.1:11211',
        }
```

```
}

# You have to redifine TEMPLATES if you want to add a template path or change TEMPLATE_DEBUG

INTERNAL_IPS = []  # these ips are treated differently if a problem occurs

# more info: https://docs.djangoproject.com/en/dev/topics/logging/#configuring-logging
LOGGING = {
        'version': 1,
        'disable_existing_loggers': False,
        'handlers': {
                'file': {
                        'level': 'DEBUG',
                        'class': 'logging.FileHandler',
                        'filename': '/path/to/django/debug.log',  # change this path
                },
        },
        'loggers': {
                'django': {
                        'handlers': ['file'],
                        'level': 'DEBUG',
                        'propagate': True,
                },
        },
}

SESSION_COOKIE_SECURE = CSRF_COOKIE_SECURE = False

DEBUG = FILER_DEBUG = False
```

You can change other Django settings, particularly it might be worthwhile to have a look at globalization settings.

4.3 Layout

svSite should provide a fully functional site (with minimal work), but you may want to personalize the look, which will take time. That's inevitable. Here is some information on how to do it.

4.3.1 Theme requirements

To create your own theme, these are the requirements:

- The files should be organized into directories `templates`, `static` and `info`:
  - The `templates` directory should contain `base.html` holding the theme body and optionally `head.html` holding anything in `<head>` (you might want to include `default_head.html`). Except for that, it can contain any templates you want (these are only used if you explicitly include them).
  - The `static` directory should contain any static files you use (see below on how to use them).
  - The `info` directory can contain any of these files: `readme.rst`, `description.rst`, `credits.rst` and `license.txt`. Other files can be included but nothing special happens with them.
- Include static css/js files using:

```
    {% load addtoblock from sekizai_tags %}
    {% addtoblock "css" %}
```

```
        <link rel="stylesheet" href="{% static THEME_PREFIX|add:'/css/style.css' %}">
{% endaddtoblock "css" %}
```

and other static files:

```
{% load static from staticfiles %}
<img src="{% static THEME_PREFIX|add:'/logo.png' %}" />
```

You can also hard-code `{% static 'theme_name/logo.png' %}`. This behaves differently in case another theme extends this one.

- For the `base.html` template:
  - It should not extend anything (it is itself included).
  - It should define precisely these placeholders:

```
{% placeholder "header" %}
{% placeholder "top-row" %}
{% placeholder "content" %}
{% placeholder "sidebar" %}
{% placeholder "bottom-row" %}
```

  - It should `{% include include_page %}` if it's set, e.g. a structure like this:

```
{% if page_include %}
        {% include page_include %}
{% else %}
        {% placeholder "content" %}
{% endif %}
```

  - You do not need to define `{% block %}` . You won't be able to extend them since Django doesn't let you extend blocks from included templates.

CHAPTER 5

Extending

5.1 Contribute

Say you want to make some changes. Perhaps they turn out great, and you want to share them. That is greatly appreciated, and this page tells you how to do it!

5.1.1 Linux / bash

You will need to make sure you can push code to Github by setting up ssh keys. Then fork the svsite repository and follow these steps.

You will need `python3`, `pip`, a database (`sqlite3` is default and easy, but slow), `virtualenv`, `git`, `elasticsearch` and some SSL packages. Just type:

```
sudo apt-get install python3 sqlite3 python-virtualenv git build-essential libssl-dev libffi-dev pyth
```

Get your copy of the svsite code:

```
git clone git@github.com:YOUR_SVSITE_FORK.git
```

Go to the directory, start a virtualenv and install:

```
virtualenv -p python3 env
source env/bin/activate
pip install --requirement dev/pip_freeze.txt
pip install --no-deps --editable .
```

To create the database and superuser:

```
python3 source/manage.py migrate
python3 source/manage.py createsuperuser
```

You might want to run the tests:

```
py.test tests
```

Then you can start the server on localhost:

```
python3 source/manage.py runserver_plus --cert dev/cert
```

You can now open the site in a browser. It is running on localhost over https on port 8000. The server prepends `www`, so use a domain that works with that prefix. For example,

> https://www.localhost.markv.nl:8000/

This refers to your localhost (127.0.0.1). The first time you will probably need to add a security exception, as this is a debug SSL certificate.

Now you are ready to make your chances!

After you are done and have tested your changes (and converted space-indents to tabs), you can suggest it for inclusion into svsite by means of a pull request

5.1.2 External services

There is a minimal api for building some external services, which is described in `integration_api`.

5.1.3 A general note

Good luck! **why-we-never-forget-our-fellow-coders_**

5.2 Models

The models and their relations can be seen in this graph:

With *graphviz* and *django-extensions* you can generate this image yourself:

```
python source/manage.py graph_models --all --settings=base.settings_development | grep -v '^  //' | g
```

5.3 Design notes (hacks)

Some parts are less than elegant. Although, at the time of writing, it seems there may not be a better way, it warrants a warning anyway.

5.3.1 Migrating

Clean migrations don't quite work for some cms addons. Find the migration info in the installation documentation.

5.3.2 Themes

Djangocms seems not designed to handle dynamic templates, so a fixed template is used that dynamically includes the theme template based on a context variable.

Since *djangocms* uses *sekizai*, which must have it's *render_block* be in the top template, it is necessary to have the <*head*> and <*body*> in this top template, and to include only the rest of the content of these tags.

Furthermore, the CMS does some kind of pre-render without context to find the placeholders to be filled. This means placeholders cannot depend on the theme (=context). Placeholders are defined in `default_body.html` and themes should match those.

5.3.3 Special pages

This relates to those pages (e.g. search results) that should not be plugins in the CMS, but should be integrated into it anyway (to be in the menu, be moved and allow placeholders).

What I would have preferred to do would be to have such pages (as apphooks) extend the main template and overwrite `{% block content %}`. However, because of themes, `{% block content %}` is necessarily defined in an `{% include %}` file. Django cannot extend blocks defined in included files (regrettably) since they are each rendered separately (not so much 'included'), making the block useless.

The 'solution' used is to force templates to include a dynamic template instead of the *content* placeholder for such pages.

```
{% if page_include %}
        {% include page_include %}
{% else %}
        {% placeholder "content" %}
{% endif %}
```

There is a special version of *render*, namely *base.render_cms_special*, that you can use like this:

```
def my_view(request):
        value = 'do some query or something'
        return render_cms_special(request, 'my_template.html', dict(
                key=value,
        ))
```

It is important to note that `my_template.html` in this example should render *just* the content part, not the full page. Don't `{% extend %}` the base template (or anything, for that matter); this is done automatically.

In order for placeholderes to show up and for things to be integrated into the cms, you will need to add this view/app as an app-hook (this is the normal way; the only difference is that you should use `render_cms_special`).

5.3.4 Users & groups

#todo - build-in Django groups - CMS users