django Organice Documentation

Release 0.4.0

Peter Bittner <django@bittner.it>

September 07, 2016

Contents

1 Contents **3**
- 1.1 django Organice . . . 3
- 1.2 Installation . . . 5
- 1.3 Configuration . . . 8
- 1.4 Themes . . . 10
- 1.5 User Manual . . . 10
- 1.6 Contributing to django Organice . . . 11
- 1.7 Alternatives to django Organice . . . 13
- 1.8 History . . . 13

2 Indices and tables **17**

All-in-one collaboration solution. For non-profit organizations, sports clubs, small to medium-sized businesses.

django Organice is a nice way to run a collaboration platform, an intranet, and websites for your business. It's a compilation of the best Django packages, preconfigured for getting you started quickly. Being powered by the well-known Django Web framework it's a safe choice for your IT investment providing an easily extensible architecture.

Also available in the cloud. Nice, very nice, http://organice.io

CHAPTER 1

Contents

1.1 django Organice

All-in-one collaboration solution. For non-profit organizations, sports clubs, small to medium-sized businesses.

django Organice is a nice way to run a collaboration platform, an intranet, and websites for your business. It's a compilation of the best Django packages, preconfigured for getting you started quickly. Being powered by the well-known Django Web framework it's a safe choice for your IT investment providing an easily extensible architecture.

1.1.1 Fundamental Features

- Enables everyone to contribute to your content—while retaining your full control.
- Supports all four common collaboration and support patterns in several ways:
  1. *Active contribution* (create content, contribute corrections)
  2. *Passive contribution* (report issues, assign tasks, provide feedback)
  3. *Active consumption* (search and browse for content)
  4. *Passive consumption* (subscribe to notifications)
- Continually involves, helps and optimally supports your collaborators by sending out reminders, editing and submission hints to avoid losing traction or interest.

1.1.2 Core Components

Major components of django Organice are:

- Django Web framework
- full-featured content management (django CMS)
- full-featured versatile blog (Zinnia, cmsplugin)
- events with event calendar (via Zinnia)
- flexible contact pages (cmsplugin)
- link collection pages

- full-featured newsletter (Emencia)
- todo/issue lists
- social login and user profiles (allauth)
- generic analytics support (analytical)
- web analytics respecting your privacy (Piwik integration)
- multiple themes (available for free at http://organice.io/themes)
- multi-language support
- multiple websites support
- database support for PostgreSQL, MySQL, Oracle, and more

1.1.3 Roadmap

We are thrilled to get these awesome features out to you:

- group spaces with automatic mailing list (groupname@example.com)
- documentation area / wiki (probably by use of CMS functionality)
- discussion forum / Q&A board (Askbot, Misago)
- content download for magazine generation
- unobtrusive, integrated, transparent document and asset management
- DropBox, ownCloud, Windows OneDrive, Google Drive, Apple iCloud, etc. integration
- fully integrated search (Haystack)
- live chat or chat integration
- surveys (django-crowdsourcing, ntusurvey)

1.1.4 Who is using django Organice?

Examples of websites running django Organice:

- Organice.io website
- Organice demo site
- Kreuzlingen Rudolf Steiner school

1.1.5 Download and Contributions

Official repositories: (kept in sync)

1. Bitbucket: https://bitbucket.org/organice/django-organice
2. GitHub: https://github.com/Organice/django-organice

1.1.6 Getting Help

- Documentation is available at http://docs.organice.io
- Questions? Please use StackOverflow. Tag your questions with `django-organice`.
- Found a bug? Please use either the Bitbucket or GitHub issue tracker (you choose)
- Need support? You're welcome to use our Gitter chat room.

1.2 Installation

This document assumes you are familiar with basic Python and Django development and their tools. If not, please read up on pip, virtualenv, and virtualenvwrapper first. A basic understanding is sufficient.

1.2.1 Requirements

- Python 2.7, 3.3, or 3.4

All other dependencies are resolved by the django Organice installer. Most of those dependencies are intentionally not pinned on their version number to allow a liberal upgrade path. This means you will get more up-to-date packages installed, but they may break your setup or not install at all.

Confirmed, working dependencies are documented in the History for each release. If anything goes wrong during the installation described below try installing those requirements in your virtual environment *before* you install `django-organice`:

```
$ pip install -r docs/requirements.txt
```

Recommended for installation

- pip
- virtualenv
- virtualenvwrapper

1.2.2 Installing django Organice

1. We recommend preparing a virtual environment for running django Organice:

   ```
   $ mkvirtualenv example
   $ workon example
   ```

   The prompt will change to something like `(example)~$` to reflect that your new virtual environment is active.

2. The easiest way is using `pip` for installation:

   ```
   $ pip install django-organice
   ```

This will pull the latest django Organice package from the Internet and install all dependencies automatically.

If you're a developer you may want to run django Organice with the latest sources: (don't do this as a user)

```
$ git clone https://github.com/Organice/django-organice.git
$ cd django-organice
$ python setup.py install
```

or, alternatively, using pip:

```
$ pip install git+https://github.com/Organice/django-organice.git#egg=django-organi
```

3. Install the adapter suitable for your database (PostgreSQL `psycopg2`, MySQL `MySQL-python`, Oracle `cx_Oracle`, etc.), e.g.

```
$ pip install psycopg2
```

The Django project recommends PostgreSQL.

Note: You can skip this step if you decide to use SQLite, e.g. for evaluation purposes.

4. Run the Organice setup command to create your new project: (e.g. *example*)

```
$ organice-setup example
```

5. Edit your settings in `example/settings/common.py`, `example/settings/develop.py`, etc. See the Django documentation on settings if you're not familiar with it. The `develop` settings are used by your project by default (local development), `common` is included in all profiles.

6. Initialize your database:

```
$ python manage.py organice bootstrap
```

This will prepare the database and add some sample content. If you'd rather wish to start with a clean database run to `migrate` only instead:

```
$ python manage.py migrate
```

7. Start your Django project:

```
$ python manage.py runserver
```

You can now point your browser to http://127.0.0.1:8000/ and start developing your project locally.

Note: If you're planning to create your content locally make sure you use the same database engine for local development and production. Your plan of moving the whole database content from development to production will give you major headaches otherwise. And, use Sqlite for evaluating only!

1.2.3 Initial Configuration

1. Follow the instructions given to you by the django Organice installer `organice-setup` after setup has completed. You have to adapt some values in your project settings!
2. If you want your site to use a language other than English, or you want to use several languages: Adapt the values of `LANGUAGE_CODE` and `LANGUAGES`, and set `USE_I18N = True` in your project settings.
3. After installation django Organice is configured, but unless you ran the `bootstrap` management command the database is blank without any content. You can add some sample content and other data running one or all of the following commands:

```
$ python manage.py organice initauth  # prepare social auth provider configuration
$ python manage.py organice initcms   # add pages for your website
$ python manage.py organice initblog  # add blog categories and posts
```

4. Alternatively, add your first pages, blog posts, and newsletter data manually:
   - Add some pages and navigation in the Django administration at Cms > Pages, and publish your changes.
   - Surf your new website, and fill your new pages with content using the front-end editing feature.
   - Surf to `/blog/` on your website, and start adding Blog posts.
   - Add a user in the Django administration at Newsletter > Contacts.
   - Add `localhost` (or appropriate server) to Newsletter > SMTP servers.
   - To allow subscribing from the website (from `/newsletter/subscribe`) add a list to Newsletter > Mailing lists.
   - Finally, add your first newsletter to Newsletter > Newsletters.
   - For adding templates to Emencia Newsletter please consult the related section in the TinyMCE 3.x documentation.
   - To add a link page simply use the "bookmarks" template for a page. You have to add categories and links in the Django Admin. (NOTE: This feature may be replaced by an integration of social bookmarking services in future.)
5. For sending newsletters to work you must configure a cronjob polling on `python manage.py send_newsletter` every half an hour. If that was just Greek to you go ask your server admin for help. She knows!

1.2.4 Deployment to Production

During the installation `organice-setup` prepared 3 different environments that help you with deployment:

```
example
-- settings
|   -- __init__.py
|   -- common.py
|   -- develop.py
```

```
|   -- staging.py
|   -- production.py
```

This modularized setup is described in Solution 2 of Tommy Jarnac's blog on Django settings best practices [1]. The `develop` settings are active by default (for local development), `common` is included by all profiles.

For deployment to environments other than `develop` the settings module location must be overridden by setting the Django environment variable `DJANGO_SETTINGS_MODULE`. For example, if you use Apache as your Django web server adapt your Apache configuration file for `example` with:

```
SetEnv DJANGO_SETTINGS_MODULE example.settings.production
```

Note: To test different settings locally you can start the Django webserver with the `--settings` option:

```
$ python manage.py runserver --settings=example.settings.staging
```

Finally, make sure you also have consulted the deployment checklist of the Django project and follow their best practices.

1.3 Configuration

django Organice comes with sensible defaults for almost anything. Yet still, you can customize its behavior with the help of the settings listed in this section.

1.3.1 Settings

You may define any of the following options in your project's `settings` to override the default value.

ORGANICE_URL_PATH_ADMIN

Default `'admin'`

The URL path for accessing the Django Administration backend (e.g. `www.example.com/admin`). Must be non-empty. Use an identifier only, no white space, no leading or trailing slash.

ORGANICE_URL_PATH_BLOG

Default `'blog'`

The URL path for the blog's start page (e.g. `www.example.com/blog`). Must be non-empty. Use an identifier only, no white space, no leading or trailing slash.

[1] David Cramer from DISQUS has described a similar solution at http://justcramer.com/2011/01/13/settings-in-django/

ORGANICE_URL_PATH_NEWSLETTER

Default `'newsletter'`

The URL path for accessing newsletter functionality on the front-end (e.g. `www.example.com/newsletter`). Must be non-empty. Use an identifier only, no white space, no leading or trailing slash.

ORGANICE_URL_PATH_TODO

Default `'todo'`

The URL path for accessing todo list functionality on the front-end (e.g. `www.example.com/todo`). Must be non-empty. Use an identifier only, no white space, no leading or trailing slash.

1.3.2 Third Party Settings

ACCOUNT_ADAPTER

Default (see django-allauth configuration)

The authentication adapter used by Organice. The `organice-setup` command sets its value to `'organice.auth.adapters.AccountAdapter'`, which ensures that the CMS editorial workflow is activated for every new user. This is originally a setting from `django-allauth` (see the Advanced Usage chapter of the allauth docs). Must be a valid dotted module path. *Remove this setting to deactivate the editorial workflow for guest users with email signup.*

SOCIALACCOUNT_ADAPTER

Default (see django-allauth configuration)

The authentication adapter used by Organice. The `organice-setup` command sets its value to `'organice.auth.adapters.SocialAccountAdapter'`, which ensures that the CMS editorial workflow is activated for every new user. This is originally a setting from `django-allauth` (see the Advanced Usage chapter of the allauth docs). Must be a valid dotted module path. *Remove this setting to deactivate the editorial workflow for guest users with social signup.*

Analytics Providers

The analytics services supported by `django-analytical` are enabled by setting various service properties in your settings file. The properties are documented in their documentation. For security reasons you shouldn't add those to your common settings file, but to `settings.production`.

Maps (Plugin for django CMS)

All map providers require that you configure an API key. See the djangocms-maps configuration for details.

1.4 Themes

One of the nice things of django Organice is that themes are handled as separate projects. In fact, they are pluggable Django apps composed of assets and templates that you can simply install and activate.

1.4.1 Official Themes

Here is a list of django Organice themes officially supported by us:

1. RSSK Theme: django-organice-theme-rssk
2. Fullpage Theme: django-organice-theme-fullpage

If you have a nice theme and would like to include it in this list let us know by e-mail or make a pull request on this page of the documentation.

Mother Theme

`django-organice-theme` is the mother of all themes for django Organice. This theme is installed automatically when you install django Organice. From the development perspective all themes are derived from the mother theme, which contains a collection of static files (assets) and templates, as well as a `Makefile` for asset management. The mother theme is composed of:

- bootstrap-sass (Sass version of Twitter Bootstrap v3)
- Compass (CSS authoring framework using Sass)
- UglifyJS v2 (JavaScript minifier)

The `Makefile` also supports you with updating those components on your development system.

1.4.2 Rolling Your Own Theme

Preparations:

- Visit http://organice.io/themes and find a theme that is as close as it gets of what you want.
- Go to that theme's repository page, make a copy of the whole project, and rename it (e.g. to `mytheme`).

Loop until you're happy:

- Add or adapt the style sheet (`.scss`), JavaScript (`.js`), and other files in `mytheme/static/`.
- Run `make assets` in order to compile the Sass files to CSS, and combine and minify both CSS and JavaScript.
- Adapt the template files in `mytheme/templates/`, and test the results on your development system.

1.5 User Manual

django Organice is composed of the following main components:

1. *Content Management* (Cms)

2. *Blog* (Zinnia)
3. *Newsletter*

1.5.1 Content Management

Editing your website is not much different from surfing the web. When you're logged in to your website you will have the django CMS toolbar on top of the page. Press the blue "Edit" button to go into Edit mode. When you now hover over the website elements you'll notice tooltips ("Double-click to edit"). Double-click on those areas, and an editor window will open up to allow you to modify the content.

You can safely modify content on the page without the website to change. Only you can see the changes, and only as long as you are in Edit mode. When you're happy with the changes press the blue "Publish changes" button on the django CMS toolbar to make your version visible to the world.

For more technical tasks like creating a navigation structure, adding pages, etc. the django CMS toolbar redirects you to the Django administration interface. Want to have an upfront first impression before using it? Check out the (slightly outdated) django CMS video on frontend-editing.

1.5.2 Blog

Writing and publishing interesting articles now and then is called "blogging". It's very similar to the usual editing of your website, but still it's different because you get more features as a natural add-on: Categories, tagging, comments, publication dates, archives (yearly, monthly, daily), related entries, RSS feeds, etc.

On your website you simply go to the Blog area, and click on the "Add" link, which is available when you're logged in. In the blog entry editor you have all functionality you already know from the content management section. Modifying existing blog entries works identical to the usual content changes on your website with the exception of publishing, which is not available with an intermediate draft step: All your changes are immediately visible online.

Note that in django Organice we also use the blog functionality for other use cases that are very similar to the usual blogging, such as event agendas, job postings, press releases, download lists, etc.

1.5.3 Newsletter

Sending out news or updates to a circle of friends or customers is managed by the Newsletter component, which is available from the Django administration interface when you're logged in to your website. Simply click on "Admin > Site Administration" on the django CMS toolbar, and go to the Newsletter section in Django administration.

This component has a couple of powerful features. It's best explained having you watch the interesting French overview video from the Emencia website.

1.6 Contributing to django Organice

Official repositories: (kept in sync)

1. Bitbucket: https://bitbucket.org/organice/django-organice
2. GitHub: https://github.com/Organice/django-organice

Fork any of the repositories, make your code changes or additions, and place a pull request.

1.6.1 How To Get Started

After cloning your own fork from Bitbucket or GitHub make sure you have created and activated a virtual environment for development, then run `make develop` to install packages that help you with your development tasks (tools for testing, translation, docs generation).

Guidelines

The primary interpreter target to develop against is **Python 3**. Ideally, use the highest one the Django package integrated into our project is compatible with (3.4, at the moment). All other supported Python versions are tested by the integration server as soon as you place the pull request. You can run tests locally before pushing using `tox` or `setup.py test`, e.g.

```
$ tox               # run all tests against all supported Python versions
$ tox -e py34,py27  # run all tests against Python 3.4 and 2.7 only
$ ./setup.py -q test -a tests/management  # only run management tests against default py
```

Source code is supposed to satisfy `flake8` default rules (with the exception of line length, which can be up to 120 characters long). A pre-commit hook for Git is installed automatically for your convenience when you run `make develop`, so you shouldn't even be able to commit when `flake8` is not passing. Additional static analysis is conducted by the QA server, and you should make sure that code health goes up (or stays the same) with each contribution.

1.6.2 Help Wanted

- Writing tests (unit tests, BDD tests)
- New features on the roadmap (see README)
- Translations (blog posts, documentation, user interface)

Translation

Translation is done on Transifex for both the project text strings and the documentation.

The multilingual documentation is written in reStructuredText syntax, and built using Sphinx. As a translator you can simply jump on Transifex and get your hands dirty for your language. Some helpful background reading is available from Read the Docs and Transifex support.

1.6.3 Bumping Versions, Package, Release

As for the version numbers of django Organice we use Semantic Versioning. Changes from one to the next release are documented in the History.

1.7 Alternatives to django Organice

With django Organice we strive to make people happy who are not tech or process experts. People who want to collaborate smoothly without having taken part of extensive training programmes in order to understand the tool. Simplicity is key, everything should be intuitive, getting things done easily. Occasional users should be happy to the same extent than daily-habit expert users.

Naturally though, django Organice isn't a fit for everyone. If you feel it can't satisfy your specific needs it may be a good idea to look at the following alternatives. If you happen to know additional offers drop us a note so we can make the list more complete.

1.7.1 Software Alternatives

In alphabetical order.

- Bitrix Intranet (PHP, commercial, ships with source code)
- Coyo (Java/Play, commercial, ships with source code)
- Fairgate (PHP, commercial, Switzerland only)
- Microsoft SharePoint + Yammer [1] (.NET + Rails, commercial)
- Odoo (Python, open source)
- Open Atrium (Drupal, open source)
- teamraum Enterprise (Django, commercial)
- tendenci (Django, open source)

1.7.2 Cloud Competitors

In alphabetical order.

- Bitrix24.com / Bitrix24.de [1]
- huddle
- Igloo
- Odoo
- teamraum
- VereinOnline

1.8 History

1.8.1 0.4

- Use djangocms-maps (instead of djangocms-googlemap)
- Add a simple type of editorial workflow (via user groups and CMS permissions)

[1] http://intranetsdoneright.blogspot.com/2013/11/what-is-social-intranet.html

- Upgrade Zinnia templates (django-organice-theme)
- Upgrade and add more social auth providers (django-allauth)
- Upgrade to Django 1.8.14, django CMS 3.3.2, Zinnia 0.16
- Add ability to generate Nginx deployment configuration

  Changes https://github.com/Organice/django-organice/compare/v0.3...v0.4

  Dependencies https://github.com/Organice/django-organice/blob/v0.4/docs/requirements.txt

1.8.2 0.3

- Disable Contact Page plugin (to be replaced by forms builder) – *planned for v0.5*
- Disable Mediatree component (needs upgrade to Python 3) – *planned for v0.5*
- Disable Newsletter component (needs upgrade to Python 3) – *planned for v0.5*
- Add Python 3 support
- Switch from develop/master model to feature branching model (default branch: `master`)
- Migrate project to Django 1.8.8 and django CMS 3.2
- Add sample content loading (first using fixtures, later converted to management command)
- Integration of Piwik open source analytics (optional via django-analytical)
- Add some test coverage, finally!
- Start with continuous builds (Travis-CI, Shippable)

  Changes https://github.com/Organice/django-organice/compare/v0.2...v0.3

  Dependencies https://github.com/Organice/django-organice/blob/v0.3/docs/requirements.txt

1.8.3 0.2

- Project-level Makefile
- Automation of translation processes with Transifex (for documentation only)
- New options for settings (`ORGANICE_URL_PATH_...`)
- Newsletter editor configuration, newsletter template sample
- Add social login and user profiles (django-allauth)
- Add assets pipeline (bootstrap-sass, Compass, UglifyJS v2)
- Upgrade jQuery to v1.11.0, template overhaul with Bootstrap
- Add language selection dropdown menu
- Migrate theme data (templates, styles, and javascript) and assets pipeline to separate projects
- Generation of server configuration (lighttpd) and more options in organice-setup
- Add media management (django-media-tree)
- Add todo lists (django-todo)

- Add generic analytics (django-analytical)

  Changes https://github.com/Organice/django-organice/compare/v0.1...v0.2

  Dependencies https://github.com/Organice/django-organice/blob/v0.2/docs/requirements.txt

1.8.4 0.1

- Initial release
- Based on Django 1.5.5, django CMS 2.4.3, Zinnia blog 0.13, Emencia newsletter 0.3.dev
- A more natural i18n mechanism than vanilla Django, no language prefix for default language
- Setup script with project generation, deployment settings, custom templates, Bootstrap 3

  Dependencies https://github.com/Organice/django-organice/blob/v0.1/docs/requirements.txt

CHAPTER 2

Indices and tables

- genindex
- search

A

analytics, 9

B

Bitbucket, 11
Blog, 11

C

Content Management, 11

D

database, 6
dependencies, 5

G

GitHub, 11

I

install, 6

N

Newsletter, 11

R

release, 12
Requirements, 5

T

Transifex, 12
translator, 12

V

version numbers, 12
virtual environment, 5