youCKAN Documentation

Release 0.1.0.dev

Axel Haustant

May 26, 2014

Contents

A social frontend to CKAN

CHAPTER 1

Compatibility

YouCKAN requires Python 2.7

CHAPTER 2

Installation

You can install youckan with pip:

```
$ pip install youckan
```

or with easy_install:

```
$ easy_install youckan
```

YouCKAN use yuglify to compress assets and less to compile LESS to CSS files. You can install them with npm:

```
$ npm install -g less yuglify
```

CHAPTER 3

Documentation

The documentation is hosted on Read the Docs

Contents:

3.1 Configuration

YouCKAN use a single ini file for its configuration

3.1.1 Sample

You can generate a sample configuration by running:

```
$ youckan genconf
# or
$ youckan-auth genconf
```

You will be asked some questions and have as a result the following ini file:

```
[site]
debug = false
secret = +hant#30d^df=1$vbolo3p+6t6xdio5312@a63-8b^96q3n-u@
allowed_hosts =
admins =
language = fr
timezone = Europe/Paris

[db]
default = sqlite://youckan.sqlite
ckan = postgres://ckan_default:ckan_default@localhost/ckan_default

[email]
webmaster = webmaster@youckan
admin = admin@youckan

[etalab]
domain = my-domain.com
ckan_url = http://ckan.{domain}
home_url = http://www.{domain}
wiki_url = http://wiki.{domain}
wiki_api_url = http://wiki.{domain}/api.php
```

```
questions_url = http://questions.{domain}

[path]
static = staticroot
static_url = /static/
media = media
media_url = /media/

[social:twitter]
key =
secret =

[social:google]
key =
secret =

[social:linkedin]
key =
secret =

[log]
level = warning
file = {name}.log

[celery]
broker = django://
backend = database
```

Feel free to customize it for your needs.

3.1.2 Advanced customization

YouCKAN configuration is extensible as its only standard django configuration.

You can either extend an existing configuration:

```
from youckan.settings import *

MY_OVERRIDEN_SETTING = 'my.value'
```

or start a new one from scratch.

3.2 Deployment

3.2.1 Environment creation

```
$ mkdir -p $YOUCKAN_HOME
$ cd $YOUCKAN_HOME
$ virtualenv .
$ source bin/activate
$ pip install youckan
$ mkdir media
```

As a postgresql administrator:

```
$ createuser youckan -P
$ createdb youckan -O youckan -E UTF8
```

3.2.2 Configuration

```
$ youckan genconf --ini
$ vim youckan.ini
```

3.2.3 Initialisation

```
$ youckan init [--noinput]
```

If `--noinput` is specified, no questions will be asked and the initialization will run in unattended mode.

You can create a super user at anytime with:

```
$ youckan createsuperuser
```

3.2.4 Upgrade

Upgrading is as easy as upgrading the youckan package and rerunning the initialization:

```
$ pip install -U youckan
$ youckan init --noinput
```

3.2.5 NGinx + uWSGI

```
$ youckan genconf --nginx
# or
$ youckan genconf --nginx --uwsgi
```

3.2.6 Apache 2 + mod

```
$ youckan genconf --apache
# or
$ youckan genconf --apache --uwsgi
```

3.3 YouCKAN SSO

3.4 API

3.5 Contributing

YouCKAN is open-source and very open to contributions.

3.5.1 Submitting issues

Issues are contributions in a way so don't hesitate to submit reports on the official bugtracker.

Provide as much informations as possible to specify the issues:

- the YouCKAN version used
- a stacktrace
- installed applications list
- ...

3.5.2 Submitting patches (bugfix, features, ...)

If you want to contribute some code:

1. fork the official YouCKAN repository
2. create a branch with an explicit name (like `my-new-feature` or `issue-XX`)
3. do your work in it
4. rebase it on the master branch from the official repository (cleanup your history by performing an interactive rebase)
5. submit your pull-request

There are some rules to follow:

- your contribution should be documented (if needed)
- your contribution should be tested and the test suite should pass successfully
- your code should be mostly PEP8 compatible with a 120 characters line length
- your contribution should support both Python 2 and 3 (use `tox` to test)

You need to install some dependencies to hack on YouCKAN:

```
$ pip install -r requirements/all.pip
```

A fabfile is provided to simplify the common tasks:

```
$ fab -l
Available commands:

coverage    Run the test suite with coverage
datamig     Generate a south data migration for an application
debug       Run Development server.
dist        Build a source distribution
doc         Generate the documentation.
gdist       Build a source distribution with git version
i18n        Generate translation files (.mo)
i18n_build  Compile translation files (.po)
init        Initialize database and user.
mig         Generate a south migration for an application
pep8        Run the PEP8 report
pylint      Run the pylint report
serve       Run Development server.
sso         Run Development server.
syncdb      Synchronize database and generate changesets
```

```
test        Run only project tests (exclude those from Django and third-party applications).
test_all    Run all tests (including those from Django and third-party applications).
update      Update all dependencies and database
update_js   Update javascript dependencies
update_py   Update python dependencies.
work        Run the development worker
```

To ensure everything is fine before submission, use `tox`. It will run the test suite on all the supported Python version and ensure the documentation is generating.

```
$ tox
```

You also need to ensure your code is PEP8 compliant (following the project rules: see `pep8.rc` file):

```
$ fab pep8
```

Don't forget client-side code and tests.

You can run the javascript test suite in the browser (http://localhost:8000/tests).

Note: minification use `yuglify` so you need to install it before: `npm install -g yuglify`

3.6 Changelog

3.6.1 Current

- Nothing yet

CHAPTER 4

Indices and tables

- *genindex*
- *modindex*
- *search*