django-flexisettings Documentation

Release 0.2.5

ledapei

April 11, 2013

CONTENTS

Django flexible settings with running environment support, separate security files and project layout detection.

Features:

- get security configuration (passwords, secret key, API keys, etc) out of the main settings file
- support multiple environments in a flexible way
- automatic discovery and configuration of common project layouts

CHAPTER

ONE

DOCUMENTATION

1.1 Installation and configuration

1.1.1 Installation

```
$ pip install django-flexisettings
```

No need to declare flexisettings in `INSTALLED_APPS`.

1.1.2 Configuration

Development

Edit `manage.py`, modify the value of `DJANGO_SETTINGS_MODULE` to point at `flexisettings.settings` and add `FLEXI_WRAPPED_MODULE` to point at your project's settings:

```
[...]
if __name__ == "__main__":
    os.environ.setdefault("DJANGO_SETTINGS_MODULE", "flexisettings.settings")
    os.environ.setdefault("FLEXI_WRAPPED_MODULE", "myproject.settings")
[...]
```

WSGI application

Edit `myproject/wsgi.py`, modify the value of `DJANGO_SETTINGS_MODULE` to point at `flexisettings.settings` and add `FLEXI_WRAPPED_MODULE` to point at your project's settings:

```
[...]
import os

os.environ.setdefault("DJANGO_SETTINGS_MODULE", "flexisettings.settings")
os.environ.setdefault("FLEXI_WRAPPED_MODULE", "myproject.settings")
[...]
```

Gunicorn

Following django's recommandations on how to use gunicorn, the project will run in WSGI mode.

First edit the `myproject/wsgi.py` file as shown in the *WSGI application* paragraph.

Command line

```
$ cd /path/to/myproject
$ /path/to/venv/bin/python gunicorn --workers=42 myproject.wsgi:application
```

Debian

The configuration file for gunicorn should be saved as `/etc/gunicorn.d/myproject` and contain:

```
CONFIG = {
    'python': '/path/to/venv/bin/python',
    'working_dir': '/path/to/myproject',
    'user': 'www-data',
    'group': 'www-data',
    'args': (
        'myproject.wsgi:application',
    ),
}
```

Extra options that could be useful in the `args` tuple:

- `'--bind=ip:port'`: set which ip and port the process whould bind to
- `'--workers=#'`: set the number of workers to spawn

For more information on gunicorn configuration, read the docs.

1.2 Settings

1.2.1 Settings options

Settings options default values are set to ensure consistency with django settings behavior. Most of the smartness, unicorns and rainbows are not enabled by default.

DJANGO_SETTINGS_MODULE

Environment variable to set to point at `flexisettings.settings`, more details in the django documentation. This can be set by environment variable either on the command line, in `manage.py` or in `myproject/wsgi.py`.

FLEXI_WRAPPED_MODULE

Environment variable to set to point at your django settings, this environment variable has to be set for `flexisettings` to work. This can be set by environmentvariable either on the command line, in `manage.py` or in `myproject/wsgi.py`.

FLEXI_RUN_ENV

Default: `None`

String defining the running environment, used when trying to load settings depending on the defined running environment.

FLEXI_SYS_PATH

Default: `['apps', 'lib']`

List of paths to add to `sys.path` if they exist, this will be done if `FLEXI_LAYOUT_DISCOVERY` is `True`.

FLEXI_AUTORELOAD

Default: `True` (this is django default behavior)

Boolean affecting the autoreload machinery for settings. If set to `True`, it should make the django autoreload code work when changing any settings file found in `flexisettings.settings._wrapped_modules`. If set to `False`, the autoreload will not work on settings. The code is a bit tricky, not guaranteed to work and might have unsuspected effects, hence the possibility to disable.

FLEXI_LAYOUT_DISCOVERY

Default: `False`

Boolean that determines if `flexisettings` tries to be smart about your project layout. It will add any `FLEXI_MEDIA_FOLDER`, `FLEXI_STATIC_FOLDER`, `FLEXI_TEMPLATE_FOLDERS` folder to the appropriate configuration variables if they are not already set.

FLEXI_MEDIA_FOLDER

Default: `'media'`

The media folder name to look for when doing layout discovery.

FLEXI_STATIC_FOLDER

Default: `'static'`

The static folder name to look for when doing layout discovery.

FLEXI_TEMPLATE_FOLDERS

Default: `('templates', )`

A tuple of templates folder names to look for when doing layout discovery.

FLEXI_SITE_ROOT

Default: `FLEX_PROJECT_ROOT`

Path to the site root, everything that should not be tracked in a VCS but is still part of your website. For example, the `MEDIA_ROOT` folder should reside in the site root.

FLEXI_PROJECT_ROOT

Default: `dirname()` of the settings folder

Path to the django project, basically everything that should be tracked in a VCS.

1.2.2 Example configuration

- edit `env.py` to set the working environment, let's use `prod` here:

```
$ cat myproject/env.py
# environment declaration
FLEXI_RUN_ENV = 'prod'
```

- edit `security.py` to set the `SECRET_KEY` value:

```
$ grep SECRET_KEY myproject/security.py
SECRET_KEY = 'alongandcomplexsecretstring'
```

- set security variables in `security_prod.py` to be used in any settings file:

```
$ grep DEFAULTDB myproject/security_prod.py
DEFAULTDB_NAME = 'dbname'
DEFAULTDB_USER = 'dbuser'
DEFAULTDB_PWD = 'secret'
```

- edit `settings_prod.py` to override generic settings like `DATABASES`, `MEDIA_ROOT`, `TIME_ZONE`:

```
$ cat myproject/settings/settings_prod.py
[...]
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.postgresql_psycopg2',
        'NAME': DEFAULTDB_NAME,
        'USER': DEFAULTDB_USER,
        'PASSWORD': DEFAULTDB_PWD,
        'HOST': '',
        'PORT': '',
    }
}
[...]
```

1.3 Internals

1.3.1 Layout

```
myproject/__init__.py
          env.py
          security.py
          security_FLEXI_RUN_ENV.py
          settings.py
          settings_FLEXI_RUN_ENV.py
```

- `env.py` has a single variable `FLEXI_RUN_ENV` that defines a running environment. This variable is then used to call different profiles of configuration depending on the running environment like `dev`, `stage` and `prod`. The `FLEXI_RUN_ENV` variable can also be set by environment, the environment takes precedence over `env.py`. Example:

```
$ export FLEXI_RUN_ENV='dev'; python manage.py runserver
```

- `security.py` holds all the common security variables for the project, probably a good place for `SECRET_KEY`.

- `security_FLEXI_RUN_ENV` specific environment security variables, dedicated database passwords should go there. Setting defined here will override settings defined in `security.py`.
- `settings.py` the classic settings file found in all django projects.
- `settings_FLEXI_RUN_ENV.py` gathers local settings for a given running environment, settings defined here will override settings defined in `settings.py`.

1.3.2 Loading order

The modules are loaded in the following order:

1. `myproject/env.py`
2. `myproject/security.py`
3. `myproject/security_FLEXI_RUN_ENV.py`
4. `myproject/settings.py`
5. `myproject/settings_FLEXI_RUN_ENV.py`

If the `FLEXI_RUN_ENV` variable is false in python, the only settings files read are `security.py` and `settings.py`.

1.4 Security

1.4.1 Security files

A very simple way to make sure that passwords are not pushed in your VCS is to exclude any file matching `myproject/security*`. It would also be a good idea to reduce the access to such files by removing read rights for users other than the one running django.

1.4.2 VCS

Files that should not be pushed to your VCS are:

- `env.py` : to allow for multiple environments to run at the same time. If you're using git, you can add `myproject/env.py` in `.gitignore`.
- any security file : repeat after me, **any** security data in your VCS is a **bad idea**. If you're using git you can add `myproject/security*.py` in `.gitignore`.

1.5 Debugging

To help in the debugging process, it is possible to know which files have been included in the final settings object by accessing `flexisettings.settings._wrapped_modules`. An example from the unit tests, the project name is 'testProject' and the running environment is 't':

```
$ python manage.py shell
[...]
>>> import flexisettings.settings
>>> flexisettings.settings._wrapped_modules
{'testProject.env': '/path/to/django-flexisettings/t/testProject/env.py',
```

```
 'testProject.security': '/path/to/django-flexisettings/t/testProject/security.py',
 'testProject.settings': '/path/to/django-flexisettings/t/testProject/settings.py',
 'testProject.settings_t': '/path/to/django-flexisettings/t/testProject/settings_t.py'}
```

1.6 License

django-flexisettings is published under a 3-clause BSD license, see LICENSE file in the project.

1.6.1 LICENSE

Verbatim of `LICENSE` file:

CHAPTER

TWO

REFERENCES

- Django project
- Django settings
- Splitting up the settings file
- PyPI django-flexisettings

CHAPTER

THREE

PROJECT LAYOUTS

- Django Project Conventions, Revisited (Zachary Voase)
- django-resusable-app
- pinax-project-zero
- Django Project Structure
- A general django project structure or folder layout

CHAPTER

FOUR

SEARCH

- *search*