
django-split-settings

Release 1.3.1

Nikita Sobolev

Apr 05, 2024

CONTENTS

1	Installation	3
2	Usage	5
3	Tips and tricks	7
4	Updating BASE_DIR	9
5	Do you want to contribute?	11
6	Version history	13
6.1	API Reference	13
	Python Module Index	19
	Index	21

Organize Django settings into multiple files and directories. Easily override and modify settings. Use wildcards in settings file paths and mark settings files as optional.

Read [this blog post](#) for more information. Also, check this [example project](#).

While this package will most likely work with the most versions of `django`, we [officially support](#):

- 4.2
- 5.0
- 5.1

This package has no dependencies itself.

In case you need older `python` / `django` versions support, then consider using older versions of `django-split-settings`.

INSTALLATION

```
pip install django-split-settings
```


USAGE

Replace your existing `settings.py` with a list of components that make up your Django settings. Preferably create a settings package that contains all the files.

Here's a minimal example:

```
from split_settings.tools import optional, include

include(
    'components/base.py',
    'components/database.py',
    optional('local_settings.py')
)
```

In the example, the files `base.py` and `database.py` are included in that order from the subdirectory called `components/`. `local_settings.py` in the same directory is included if it exists.

Note: The local context is passed on to each file, so each following file can access and modify the settings declared in the previous files.

We also made an in-depth [tutorial](#).

TIPS AND TRICKS

You can use wildcards in file paths:

```
include('components/my_app/*.py')
```

Note that files are included in the order that `glob` returns them, probably in the same order as what `ls -U` would list them. The files are NOT in alphabetical order.

You can modify common settings in environment settings simply importing them

```
# local_settings.py
from components.base import INSTALLED_APPS

INSTALLED_APPS += (
    'raven.contrib.django.raven_compat',
)
```


UPDATING BASE_DIR

The `django create-project` command will create a variable in your `settings.py` called `BASE_DIR`, which is often used to locate static files, media files, and templates.

```
# Created by django create-project  
BASE_DIR = os.path.dirname(os.path.dirname(os.path.abspath(__file__)))  
STATIC_ROOT = os.path.join(BASE_DIR, "staticfiles/")  
MEDIA_ROOT = os.path.join(BASE_DIR, "mediafiles/")
```

The expression for `BASE_DIR` means: get the path to the current file (`settings.py`), get the parent folder (whatever you named your project), get the parent folder (the root of the project). So `STATIC_ROOT` will then be evaluated to `/staticfiles` (with `/` meaning the root of your project/repo).

With `django-split-settings` `settings` is now a module (instead of a file), so `os.path.dirname(os.path.dirname(os.path.abspath(__file__)))` will evaluate to `/whatever-you-named-your-project` as opposed to `/.`

To fix this `BASE_DIR` needs to be set to the parent folder of `/whatever-you-named-your-project`:

```
BASE_DIR = os.path.dirname(os.path.dirname(os.path.dirname(os.path.abspath(__file__))))
```


DO YOU WANT TO CONTRIBUTE?

Read the [CONTRIBUTING.md](#) file.

VERSION HISTORY

See [CHANGELOG.md](#) file.

6.1 API Reference

6.1.1 API

Organize Django settings into multiple files and directories.

Easily override and modify settings. Use wildcards and optional settings files.

class `_Optional`

Bases: `str`

Wrap a file path with this class to mark it as optional.

Optional paths don't raise an `OSError` if file is not found.

optional(*filename*)

This function is used for compatibility reasons.

It masks the old *optional* class with the name error. Now *invalid-name* is removed from *pylint*.

Parameters

filename (`Optional[str]`) – the filename to be optional.

Return type

`str`

Returns

New instance of `_Optional`.

include(*args, scope=None)

Used for including Django project settings from multiple files.

Parameters

- ***args** (`str`) – File paths (`glob` - compatible wildcards can be used).
- ****kwargs** – Settings context: `scope=globals()` or `None`.

Raises

OSError – if a required settings file is not found.

Return type

`None`

Usage example:

```
from split_settings.tools import optional, include

include(
    'components/base.py',
    'components/database.py',
    optional('local_settings.py'),

    scope=globals(), # optional scope
)
```

6.1.2 Version history

We follow Semantic Version.

1.3.1

Bugfixes

- Fixes getting the last stack item performance #532

1.3.0

Features

- Drops python3.8 support
- Adds python3.11 and python3.12 support
- Adds django5.0 support
- Uses OSError instead of IOError alias
- Converts `include(*args, **kwargs)` to `include(*args, scope=...)`, because other kwargs were not supported anyway

1.2.0

Features

- Adds python3.10 support
- Drops python3.6 support
- Adds django4.1 support

1.1.0

Features

- Adds python3.9 support
- Adds django3.1 support

Misc

- Moves to Github Actions

1.0.1

Bugfixes

- Fixes that django's dev server was not catching split setting filechanges

1.0.0

Breaking changes:

- Drops python2 support
- Drops django2.0 support

Improvements:

- Moves to poetry
- Adds mypy support
- Adds wemake-python-styleguide support
- Adds extra CI checks: safety, doc8
- Adds py.typed file to package type information

0.3.1

Improvements:

- Added support for django till to 2.2 version.

0.3.0

Improvements:

- Added Django==2.0
- Removed old versions of Django from test matrix
- Removed python3.4 from test matrix
- Documentation updates
- Adds more flake8 plugins to enforce strict style

Bugs:

- Fixes Windows problems via #21

0.2.5

Improvements:

- Added python3.6 and Django==1.11
- Fixed tests/settings structure with basic/ folder
- Added documentation, which is built with Sphinx
- Updated README.rst with new logo
- Updated README.rst with docs badge
- Updated CONTRIBUTING.rst with new information

Bugs:

- Updated README.rst to be compatible with PyPI

0.2.4

- Changed the default Django version in the requirements from `>= 1.5.1` to `>= 1.5`
- Added `setup.cfg` to support `python setup.py test` command
- Refactored how the tests work
- Added `tests/conftest.py` file with the fixtures, used fixtures widely
- Changed all test to be functions instead of classes
- Added new classifiers
- Added `pytest-env` to read env variables from `setup.cfg`
- Removed `run_coveralls.py`, added `after_success` section in `.travis.yml`
- Changed the `README.rst` to be shorter

0.2.3

- Added `django@1.10` support
- Now `include` function finds parent `globals()` scope automatically if not provided
- Added protection against infinite recursion
- Added tests for stackable settings definition. See `tests/settings/stacked/`
- Added tests for the new functionality
- Added tests for `django@1.10` in `tox` and `travis`
- Removed 3.2 and 3.3 from `setup.py` since these versions were not tested anyway

0.2.2

- Now supporting unicode filenames, fixes [#9](#)
- Tests structure is changed
- Removed example
- Changed how MANIFEST.in is defined

0.2.1

- Changed optional to be a function.
- Added test_tools.py, achieved 100% in coverage.
- Removed setuptools-git from setup.py, now Manifest is only way to provide dist sources.
- Added run_coveralls.py to work on both CI and local tests.
- Style fixes.

0.2.0

- Now tox is used for testing.
- Added coverage information and badge.
- Removed pep8 utility, now using pylint.

0.1.3

- Python 3.5 support, Django 1.9 test-support, documentation updates.

0.1.2

- Fixed Python 3 compatibility. Fixed [#7](#).

0.1.1

- Fixed issue [#1](#): now works with Gunicorn, too

0.1.0

- Initial version

PYTHON MODULE INDEX

S

`split_settings.tools`, [13](#)

Symbols

`_Optional` (*class in `split_settings.tools`*), 13

I

`include()` (*in module `split_settings.tools`*), 13

M

module
 `split_settings.tools`, 13

O

`optional()` (*in module `split_settings.tools`*), 13

S

`split_settings.tools`
 module, 13