Django-environ Documentation

Release 0.4.3

joke2k

Aug 18, 2017

Contents

Django-environ allows you to utilize 12factor inspired environment variables to configure your Django application.

CHAPTER 1

Django-environ

Django-environ allows you to utilize 12factor inspired environment variables to configure your Django application.

This module is a merge of:

- envparse
- honcho
- dj-database-url
- dj-search-url
- dj-config-url
- django-cache-url

and inspired by:

- 12factor
- 12factor-django
- Two Scoops of Django

This is your *settings.py* file before you have installed **django-environ**

```
import os
SITE_ROOT = os.path.dirname(os.path.dirname(os.path.dirname(os.path.realpath(__file__
↪))))

DEBUG = True
TEMPLATE_DEBUG = DEBUG

DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.postgresql_psycopg2',
        'NAME': 'database',
        'USER': 'user',
        'PASSWORD': 'githubbedpassword',
```

```
        'HOST': '127.0.0.1',
        'PORT': '8458',
    },
    'extra': {
        'ENGINE': 'django.db.backends.sqlite3',
        'NAME': os.path.join(SITE_ROOT, 'database.sqlite')
    }
}

MEDIA_ROOT = os.path.join(SITE_ROOT, 'assets')
MEDIA_URL = 'media/'
STATIC_ROOT = os.path.join(SITE_ROOT, 'static')
STATIC_URL = 'static/'

SECRET_KEY = '...im incredibly still here...'

CACHES = {
    'default': {
        'BACKEND': 'django.core.cache.backends.memcached.MemcachedCache',
        'LOCATION': [
            '127.0.0.1:11211', '127.0.0.1:11212', '127.0.0.1:11213',
        ]
    },
    'redis': {
        'BACKEND': 'django_redis.cache.RedisCache',
        'LOCATION': '127.0.0.1:6379/1',
        'OPTIONS': {
            'CLIENT_CLASS': 'django_redis.client.DefaultClient',
            'PASSWORD': 'redis-githubbed-password',
        }
    }
}
```

After:

```
import environ
root = environ.Path(__file__) - 3 # three folder back (/a/b/c/ - 3 = /)
env = environ.Env(DEBUG=(bool, False),) # set default values and casting
environ.Env.read_env() # reading .env file

SITE_ROOT = root()

DEBUG = env('DEBUG') # False if not in os.environ
TEMPLATE_DEBUG = DEBUG

DATABASES = {
    'default': env.db(), # Raises ImproperlyConfigured exception if DATABASE_URL not␣
↪in os.environ
    'extra': env.db('SQLITE_URL', default='sqlite:////tmp/my-tmp-sqlite.db')
}

public_root = root.path('public/')

MEDIA_ROOT = public_root('media')
MEDIA_URL = 'media/'
STATIC_ROOT = public_root('static')
STATIC_URL = 'static/'
```

```
SECRET_KEY = env('SECRET_KEY') # Raises ImproperlyConfigured exception if SECRET_KEY
↪not in os.environ

CACHES = {
    'default': env.cache(),
    'redis': env.cache('REDIS_URL')
}
```

You can also pass `read_env()` an explicit path to the `.env` file.

Create a `.env` file:

```
DEBUG=on
# DJANGO_SETTINGS_MODULE=myapp.settings.dev
SECRET_KEY=your-secret-key
DATABASE_URL=psql://urser:un-githubbedpassword@127.0.0.1:8458/database
# SQLITE_URL=sqlite:///my-local-sqlite.db
CACHE_URL=memcache://127.0.0.1:11211,127.0.0.1:11212,127.0.0.1:11213
REDIS_URL=rediscache://127.0.0.1:6379/1?client_class=django_redis.client.
↪DefaultClient&password=redis-un-githubbed-password
```

CHAPTER 2

How to install

```
$ pip install django-environ
```

CHAPTER 3

How to use

There are only two classes, `environ.Env` and `environ.Path`

```
>>> import environ
>>> env = environ.Env(
        DEBUG=(bool, False),
    )
>>> env('DEBUG')
False
>>> env('DEBUG', default=True)
True

>>> open('.myenv', 'a').write('DEBUG=on')
>>> environ.Env.read_env('.myenv') # or env.read_env('.myenv')
>>> env('DEBUG')
True

>>> open('.myenv', 'a').write('\nINT_VAR=1010')
>>> env.int('INT_VAR'), env.str('INT_VAR')
1010, '1010'

>>> open('.myenv', 'a').write('\nDATABASE_URL=sqlite:///my-local-sqlite.db')
>>> env.read_env('.myenv')
>>> env.db()
{'ENGINE': 'django.db.backends.sqlite3', 'NAME': 'my-local-sqlite.db', 'HOST': '',
↪'USER': '', 'PASSWORD': '', 'PORT': ''}

>>> root = env.path('/home/myproject/')
>>> root('static')
'/home/myproject/static'
```

See cookiecutter-django for a concrete example on using with a django project.

CHAPTER 4

Supported Types

- str
- bool
- int
- float
- json
- list (FOO=a,b,c)
- tuple (FOO=(a,b,c))
- dict (BAR=key=val,foo=bar) #environ.Env(BAR=(dict, {}))
- dict (BAR=key=val;foo=1.1;baz=True) #environ.Env(BAR=(dict(value=unicode, cast=dict(foo=float,baz=bool)), {}))
- url
- path (environ.Path)
- **db_url**
  - PostgreSQL: postgres://, pgsql://, psql:// or postgresql://
  - PostGIS: postgis://
  - MySQL: mysql:// or mysql2://
  - MySQL for GeoDjango: mysqlgis://
  - SQLITE: sqlite://
  - SQLITE with SPATIALITE for GeoDjango: spatialite://
  - Oracle: oracle://
  - Redshift: redshift://
  - LDAP: ldap://

- **cache_url**
  - Database: dbcache://
  - Dummy: dummycache://
  - File: filecache://
  - Memory: locmemcache://
  - Memcached: memcache://
  - Python memory: pymemcache://
  - Redis: rediscache://
- **search_url**
  - ElasticSearch: elasticsearch://
  - Solr: solr://
  - Whoosh: whoosh://
  - Xapian: xapian://
  - Simple cache: simple://
- **email_url**
  - SMTP: smtp://
  - SMTP+SSL: smtp+ssl://
  - SMTP+TLS: smtp+tls://
  - Console mail: consolemail://
  - File mail: filemail://
  - LocMem mail: memorymail://
  - Dummy mail: dummymail://

CHAPTER 5

Tips

In order to use unsafe characters you have to encode with `urllib.parse.encode` before you set into `.env` file.

See https://perishablepress.com/stop-using-unsafe-characters-in-urls/ for reference.

In order to set email configuration for django you can use this code:

```
EMAIL_CONFIG = env.email_url(
    'EMAIL_URL', default='smtp://user@:password@localhost:25')

vars().update(EMAIL_CONFIG)
```

SQLite connects to file based databases. The same URL format is used, omitting the hostname, and using the "file" portion as the filename of the database. This has the effect of four slashes being present for an absolute file path: sqlite:////full/path/to/your/database/file.sqlite.

CHAPTER 6

Tests

```
$ git clone git@github.com:joke2k/django-environ.git
$ cd django-environ/
$ python setup.py test
```

CHAPTER 7

License

Django-environ is licensed under the MIT License - see the LICENSE_FILE file for details

CHAPTER 8

Changelog

- Confirm support for Django 1.11.
- Support for Redshift database URL
- Fix uwsgi settings reload problem (#55)
- Update support for django-redis urls (#109)
- Fix for unsafe characters into URLs
- Clarifying warning on missing or unreadable file. Thanks to @nickcatal
- Add support for Django 1.10.
- Fix support for Oracle urls
- Fix support for django-redis
- Fix non-ascii values (broken in Python 2.x)
- New email schemes - smtp+ssl and smtp+tls (smtps would be deprecated)
- redis_cache replaced by django_redis
- Add tuple support. Thanks to @anonymouzz
- Add LDAP url support for database (django-ldapdb)
- Fix psql/pgsql url
- Add cache url support
- Add email url support
- Add search url support
- Rewriting README.rst
- environ/environ.py: Env.__call__ now uses Env.get_value instance method
- environ/environ.py, environ/test.py, environ/test_env.txt: add advanced float parsing (comma and dot symbols to separate thousands and decimals)

- README.rst, docs/index.rst: fix TYPO in documentation
- initial release

CHAPTER 9

Credits

- 12factor
- 12factor-django
- Two Scoops of Django
- rconradharris / envparse
- kennethreitz / dj-database-url
- migonzalvar / dj-email-url
- ghickman / django-cache-url
- dstufft / dj-search-url
- julianwachholz / dj-config-url
- nickstenning / honcho
- envparse
- Distribute
- modern-package-template

CHAPTER 10

environ.Env

class environ.environ.**Env**(***scheme*)

Provide scheme-based lookups of environment variables so that each caller doesn't have to pass in *cast* and *default* parameters.

Usage::

```
env = Env(MAIL_ENABLED=bool, SMTP_LOGIN=(str, 'DEFAULT'))
if env('MAIL_ENABLED'):
    ...
```

BOOLEAN_TRUE_STRINGS = ('true', 'on', 'ok', 'y', 'yes', '1')

DB_SCHEMES = {'mysql2': 'django.db.backends.mysql', 'postgres': 'django.db.backends.postgresql_psycopg2', 'redshift'

DEFAULT_DATABASE_ENV = 'DATABASE_URL'

CACHE_SCHEMES = {'memcache': 'django.core.cache.backends.memcached.MemcachedCache', 'filecache': 'django.core

DEFAULT_CACHE_ENV = 'CACHE_URL'

EMAIL_SCHEMES = {'smtp+ssl': 'django.core.mail.backends.smtp.EmailBackend', 'memorymail': 'django.core.mail.bac

DEFAULT_EMAIL_ENV = 'EMAIL_URL'

SEARCH_SCHEMES = {'simple': 'haystack.backends.simple_backend.SimpleEngine', 'solr': 'haystack.backends.solr_bac

DEFAULT_SEARCH_ENV = 'SEARCH_URL'

__call__(*var*, *cast=None*, *default=<NoValue>*, *parse_default=False*)

str(*var*, *default=<NoValue>*)

Return type *str*

bool(*var*, *default=<NoValue>*)

Return type *bool*

int(*var*, *default=<NoValue>*)

Return type *int*

float (*var*, *default=<NoValue>*)

Return type *float*

json (*var*, *default=<NoValue>*)

Returns Json parsed

list (*var*, *cast=None*, *default=<NoValue>*)

Return type *list*

dict (*var*, *cast=<type 'dict'>*, *default=<NoValue>*)

Return type *dict*

url (*var*, *default=<NoValue>*)

Return type urlparse.ParseResult

db_url (*var='DATABASE_URL'*, *default=<NoValue>*, *engine=None*)
Returns a config dictionary, defaulting to DATABASE_URL.

Return type *dict*

cache_url (*var='CACHE_URL'*, *default=<NoValue>*, *backend=None*)
Returns a config dictionary, defaulting to CACHE_URL.

Return type *dict*

email_url (*var='EMAIL_URL'*, *default=<NoValue>*, *backend=None*)
Returns a config dictionary, defaulting to EMAIL_URL.

Return type *dict*

search_url (*var='SEARCH_URL'*, *default=<NoValue>*, *engine=None*)
Returns a config dictionary, defaulting to SEARCH_URL.

Return type *dict*

path (*var*, *default=<NoValue>*, ***kwargs*)

Return type *Path*

classmethod **read_env** (*env_file=None*, ***overrides*)
Read a .env file into os.environ.

If not given a path to a dotenv path, does filthy magic stack backtracking to find manage.py and then find the dotenv.

http://www.wellfireinteractive.com/blog/easier-12-factor-django/

https://gist.github.com/bennylope/2999704

classmethod **db_url_config** (*url*, *engine=None*)
Pulled from DJ-Database-URL, parse an arbitrary Database URL. Support currently exists for PostgreSQL, PostGIS, MySQL, Oracle and SQLite.

SQLite connects to file based databases. The same URL format is used, omitting the hostname, and using the "file" portion as the filename of the database. This has the effect of four slashes being present for an absolute file path:

```
>>> from environ import Env
>>> Env.db_url_config('sqlite:////full/path/to/your/file.sqlite')
{'ENGINE': 'django.db.backends.sqlite3', 'HOST': '', 'NAME': '/full/path/to/
↪your/file.sqlite', 'PASSWORD': '', 'PORT': '', 'USER': ''}
```

```
>>> Env.db_url_config('postgres://uf07k1i6d8ia0v:wegauwhgeuioweg@ec2-107-21-
↪253-135.compute-1.amazonaws.com:5431/d8r82722r2kuvn')
{'ENGINE': 'django.db.backends.postgresql_psycopg2', 'HOST': 'ec2-107-21-253-
↪135.compute-1.amazonaws.com', 'NAME': 'd8r82722r2kuvn', 'PASSWORD':
↪'wegauwhgeuioweg', 'PORT': 5431, 'USER': 'uf07k1i6d8ia0v'}
```

classmethod **cache_url_config**(*url*, *backend=None*)
Pulled from DJ-Cache-URL, parse an arbitrary Cache URL.

Parameters

- **url** –
- **backend** –

Returns

classmethod **email_url_config**(*url*, *backend=None*)
Parses an email URL.

classmethod **search_url_config**(*url*, *engine=None*)

get_value(*var*, *cast=None*, *default=<NoValue>*, *parse_default=False*)
Return value for given environment variable.

Parameters

- **var** – Name of variable.
- **cast** – Type to cast return value as.
- **default** – If var not present in environ, return this instead.
- **parse_default** – force to parse default..

Returns Value from environment or default (if set)

classmethod **parse_value**(*value*, *cast*)
Parse and cast provided value

Parameters

- **value** – Stringed value.
- **cast** – Type to cast return value as.

Returns Casted value

CHAPTER 11

environ.Path

class environ.environ.**Path**(*start=''*, **paths*, ***kwargs*)

Inspired to Django Two-scoops, handling File Paths in Settings.

```
>>> from environ import Path
>>> root = Path('/home')
>>> root, root(), root('dev')
(<Path:/home>, '/home', '/home/dev')
>>> root == Path('/home')
True
>>> root in Path('/'), root not in Path('/other/path')
(True, True)
>>> root('dev', 'not_existing_dir', required=True)
Traceback (most recent call last):
environ.environ.ImproperlyConfigured: Create required path: /home/not_existing_dir
>>> public = root.path('public')
>>> public, public.root, public('styles')
(<Path:/home/public>, '/home/public', '/home/public/styles')
>>> assets, scripts = public.path('assets'), public.path('assets', 'scripts')
>>> assets.root, scripts.root
('/home/public/assets', '/home/public/assets/scripts')
>>> assets + 'styles', str(assets + 'styles'), ~assets
(<Path:/home/public/assets/styles>, '/home/public/assets/styles', <Path:/home/
↪public>)
```

root -> Retrieve absolute path

__call__(**paths*, ***kwargs*)

Retrieve the absolute path, with appended paths

Parameters

- **paths** – List of sub path of self.root
- **kwargs** – required=False

path(**paths*, ***kwargs*)

Create new Path based on self.root and provided paths.

Parameters

- **paths** – List of sub paths
- **kwargs** – required=False

Return type *Path*

file(*name*, **args*, ***kwargs*)
Open a file.

Parameters

- **name** – Filename appended to self.root
- **args** – passed to open()
- **kwargs** – passed to open()

Return type *file*

CHAPTER 12

Indices and tables

- genindex
- modindex
- search

Python Module Index

e

Index

Symbols

B

C

D

E

F

G

I

J

L

P

R

S

U