



Поиск проектов




[Помощь](#)[Спонсоры](#)[Войти](#)[Зарегистрироваться](#)

python-decouple 3.8

Последняя
версия`pip install python-decouple` Выпущен: 2 мар. 2023
г.

Strict separation of settings from code.


Навигация

 [Описание
проекта](#) [История
выпусков](#) [Загрузка
файлов](#)

Ссылки проекта

 [Homepage](#)

Статистика

Смотрите
статистику этого
проекта на
[Libraries.io](#)  или

Описание проекта

Развязка Python: строгое отделение настроек от кода

Развязка помогает вам упорядочить ваши настройки таким образом, чтобы вы могли изменять параметры без необходимости повторного развертывания вашего приложения.

Это также упрощает для вас:

1. сохраняйте параметры в файлах `ini` или `.env`;
2. определите исчерпывающие значения по умолчанию;
3. правильное преобразование значений в правильный тип данных;
4. имейте только один конфигурационный модуль для управления всеми вашими экземплярами.

Изначально он был разработан для Django, но стал независимым универсальным инструментом для отделения настроек от кода.

В нашем
общедоступном
наборе данных на
Google BigQuery [↗](#)

Метаданные

Лицензия: MIT
License (MIT)

Автор: Henrique
Bastos [✉](#)

Сопровождающие



энрикебасто
с

Классификаторы

Статус разработки

- 5 -
Производственный
/ стабильный

Фреймворк

- Django
- Колба

Целевая аудитория

- Разработчики

Лицензия

- Одобрен OSI ::
Лицензия MIT

Естественный язык

- Английский

Операционная система

- Не зависит от
операционной
системы

build no longer available [pypi v3.8](#)

Краткие сведения

- Развязка Python: строгое отделение настроек от кода
- Почему?
 - Почему бы просто не использовать переменные окружения?
- Использование
 - Кодировки
 - Где хранятся данные настроек?
 - Ini-файл
 - Env-файл
 - Пример: Как мне использовать его с Django?
 - Внимание с *неопределенными* параметрами
 - Переопределение конфигурационных файлов переменными среды
- Как это работает?
 - Понимание приведенного аргумента
 - Встроенный Csv-помощник
 - Встроенный помощник по выбору
- Помочь
- Лицензия

Почему?

Файлы настроек в веб-фреймворках хранят множество различных типов параметров:

- Локализация и i18n;
- Промежуточные программы и установленные приложения;
- Обработчики ресурсов для базы данных, Memcached и других служб резервного копирования;
- Учетные данные для доступа к внешним сервисам, таким как Amazon S3 или Twitter;
- Значения для каждого развертывания, такие как каноническое имя хоста для экземпляра.

Первые 2 являются *настройками проекта*, а последние 3 - *настройками экземпляра*.

Язык
программирования

- Python
- Python :: 3

Тема

- Разработка
программного
обеспечения ::
Библиотеки



AWS является
постоянным
спонсором
Python Software
Foundation.

Спонсор PSF
Обслуживается
этично

Вы должны иметь возможность изменять *настройки экземпляра* без повторного развертывания вашего приложения.

Почему бы просто не использовать переменные окружения?

Envvars работает, но поскольку `os.environ` возвращает только строки, это сложно.

Допустим, у вас есть *envvar* `DEBUG=False`. Если вы запустите:

```
if os.environ['DEBUG']:  
    print True  
else:  
    print False
```

Он выведет `True`, потому что `os.environ['DEBUG']` возвращает строку `"False"`. Поскольку это непустая строка, она будет оценена как `True`.

Развязка предоставляет решение, которое не похоже на обходной путь: `config('DEBUG', приведение=bool)`.

Использование

Установить:

```
pip install python-decouple
```

Затем используйте его на своем `settings.py`.

1. Импортируйте объект `config`:

```
from decouple import config
```

2. Извлекает параметры конфигурации:

```
SECRET_KEY = config('SECRET_KEY')
DEBUG = config('DEBUG', default=False, cast=bool)
EMAIL_HOST = config('EMAIL_HOST', default='localhost')
EMAIL_PORT = config('EMAIL_PORT', default=25, cast=int)
```

Кодировки

Кодировкой Decouple по умолчанию является *UTF-8*.

Но вы можете указать предпочитаемую вами кодировку.

Поскольку *config* является ленивым и открывает файл конфигурации только при первой необходимости, у вас есть возможность изменить его кодировку сразу после импорта.

```
from decouple import config
config.encoding = 'cp1251'
SECRET_KEY = config('SECRET_KEY')
```

Если вы хотите вернуться к кодировке вашей системы по умолчанию, используйте:

```
import locale
from decouple import config
config.encoding = locale.getpreferredencoding(False)
SECRET_KEY = config('SECRET_KEY')
```

Где хранятся данные настроек?

Развязка поддерживает как *.ini*, так и *.env* файлы.

Ini-файл

Просто создайте `settings.ini` рядом с вашим модулем конфигурации в форме:

```
[settings]
DEBUG=True
TEMPLATE_DEBUG=%(DEBUG)s
SECRET_KEY=ARANDOMSECRETKEY
DATABASE_URL=mysql://myuser:mypassword@myhost/mydatabase
PERCENTILE=90%%
#COMMENTED=42
```

Примечание: Поскольку `ConfigParser` поддерживает интерполяцию строк, для представления символа `%` его нужно экранировать как `%%`.

Env-файл

Просто создайте текстовый файл `.env` в корневом каталоге вашего репозитория в форме:

```
DEBUG=True
TEMPLATE_DEBUG=True
SECRET_KEY=ARANDOMSECRETKEY
DATABASE_URL=mysql://myuser:mypassword@myhost/mydatabase
PERCENTILE=90%
#COMMENTED=42
```

Пример: Как мне использовать его с Django?

Учитывая, что у меня есть файл `.env` в корневом каталоге моего репозитория, вот фрагмент моего `settings.py`.

Я также рекомендую использовать `pathlib` и `dj-database-url`.

```
# coding: utf-8
from decouple import config
from unipath import Path
from dj_database_url import parse as db_url

BASE_DIR = Path(__file__).parent

DEBUG = config('DEBUG', default=False, cast=bool)
```

```

TEMPLATE_DEBUG = DEBUG

DATABASES = {
    'default': config(
        'DATABASE_URL',
        default='sqlite:/// ' + BASE_DIR.child('db.sqlite3').path,
        cast=db_url
    )
}

TIME_ZONE = 'America/Sao_Paulo'
USE_L10N = True
USE_TZ = True

SECRET_KEY = config('SECRET_KEY')

EMAIL_HOST = config('EMAIL_HOST', default='localhost')
EMAIL_PORT = config('EMAIL_PORT', default=25, cast=int)
EMAIL_HOST_PASSWORD = config('EMAIL_HOST_PASSWORD', default='')
EMAIL_HOST_USER = config('EMAIL_HOST_USER', default='')
EMAIL_USE_TLS = config('EMAIL_USE_TLS', default=False, cast=bool)

# ...

```

Внимание с неопределенными параметрами

В приведенном выше примере все параметры конфигурации, кроме `SECRET_KEY = config('SECRET_KEY')`, имеют значение по умолчанию на случай, если оно не существует в `.env` файле.

Если `SECRET_KEY` отсутствует в `.env`, *развязка* вызовет ошибку `UndefinedValueError`.

Эта политика *быстрого сбоя* помогает вам избежать преследования за неправильным поведением, когда вы в конечном итоге забываете параметр.

Переопределение конфигурационных файлов переменными среды

Иногда может потребоваться изменить значение параметра без необходимости редактировать файлы `.ini` или `.env`.

Начиная с версии 3.0, *развязка* учитывает *способ unix*. Поэтому переменные среды имеют приоритет над файлами

конфигурации.

Чтобы переопределить параметр конфигурации, вы можете просто сделать:

```
DEBUG=True python manage.py
```

Как это работает?

Развязка всегда выполняет поиск параметров в этом порядке:

1. Переменные среды;
2. Репозиторий: файл `ini` или `.env`;
3. Аргумент по умолчанию передан в `config`.

Есть 4 класса, которые творят волшебство:

- **Конфигурация**

Координирует весь процесс извлечения конфигурации.

- **Репозиторий**

Может считывать значения из `os.environ` и `ini`-файлов в таком порядке.

Примечание: Начиная с версии 3.0, развязка учитывает приоритет переменных среды `unix` над конфигурационными файлами.

- **RepositoryEnv**

Может считывать значения из файлов `os.environ` и `.env`.

Примечание: Начиная с версии 3.0, развязка учитывает приоритет переменных среды `unix` над конфигурационными файлами.

- **Автоконфигурация**

Это ленивая `конфигурации`, которая определяет, какой репозиторий конфигурации вы используете.

Он рекурсивно выполняет поиск по пути к вашему конфигурационному модулю в поисках файла `settings.ini` или `.env`.

Необязательно, он принимает аргумент `search_path`, чтобы явно определить, где начинается поиск.

Объект `config` - это экземпляр `AutoConfig`, который создает экземпляры `конфигурации` с соответствующим `репозиторием` при первом ее использовании.

Понимание приведенного аргумента

По умолчанию все значения, возвращаемые `decouple`, являются `строками`, в конце концов, они считываются из `текстовых файлов` или `envvars`.

Однако в вашем коде Python может ожидать некоторый другой тип значения, например:

- `DEBUG` в Django ожидает логическое значение `True` или `False`.
- `EMAIL_PORT` в Django ожидает `целое` число.
- `ALLOWED_HOSTS` в Django ожидает `список` имен хостов.
- `SECURE_PROXY_SSL_HEADER` в Django ожидает `кортеж` с двумя элементами, именем заголовка для поиска и требуемым значением.

Чтобы удовлетворить эту потребность, функция `config` принимает аргумент `приведения`, который принимает любой `вызываемый` аргумент, который будет использоваться для `преобразования` строкового значения во что-то другое.

Давайте посмотрим несколько примеров для вышеупомянутых случаев:

```
>>> os.environ['DEBUG'] = 'False'
>>> config('DEBUG', cast=bool)
False

>>> os.environ['EMAIL_PORT'] = '42'
>>> config('EMAIL_PORT', cast=int)
```


42

```
>>> os.environ['ALLOWED_HOSTS'] = '.localhost, .herokuapp.com'
>>> config('ALLOWED_HOSTS', cast=lambda v: [s.strip() for s in v.split(',')])
['.localhost', '.herokuapp.com']

>>> os.environ['SECURE_PROXY_SSL_HEADER'] = 'HTTP_X_FORWARDED_PROTO=https'
>>> config('SECURE_PROXY_SSL_HEADER', cast=Csv(post_processor=lambda v: [s.strip() for s in v.split(',')]))
(['HTTP_X_FORWARDED_PROTO', 'https'])
```

Как вы можете видеть, `приведение` в действие очень гибко. Но последний пример получился немного сложным.

Встроенный Csv-помощник

Чтобы учесть сложность последнего примера, *Decouple* поставляется с расширяемым *Csv-хелпером*.

Давайте улучшим последний пример:

```
>>> from decouple import Csv
>>> os.environ['ALLOWED_HOSTS'] = '.localhost, .herokuapp.com'
>>> config('ALLOWED_HOSTS', cast=Csv())
['.localhost', '.herokuapp.com']
```

У вас также может быть значение по умолчанию, которое должно быть строкой для обработки в *Csv* формате.

```
>>> from decouple import Csv
>>> config('ALLOWED_HOSTS', default='127.0.0.1', cast=Csv())
['127.0.0.1']
```

Вы также можете параметризовать *Csv Helper* для возврата других типов данных.

```
>>> os.environ['LIST_OF_INTEGERS'] = '1,2,3,4,5'
>>> config('LIST_OF_INTEGERS', cast=Csv(int))
[1, 2, 3, 4, 5]

>>> os.environ['COMPLEX_STRING'] = '%virtual_env%\t *imp'
>>> csv = Csv(cast=lambda s: s.upper(), delimiter='\t',
```

```
>>> csv(os.environ['COMPLEX_STRING'])
['VIRTUAL_ENV', 'IMPORTANT STUFF', 'TRAILING SPACES']
```

По умолчанию `Csv` возвращает `список`, но вы можете получить `кортеж` или что угодно другое, используя аргумент `post_process`:

```
>>> os.environ['SECURE_PROXY_SSL_HEADER'] = 'HTTP_X_FORWARDED_PROTO=https'
>>> config('SECURE_PROXY_SSL_HEADER', cast=Csv(post_process=lambda x: tuple(x)))
('HTTP_X_FORWARDED_PROTO', 'https')
```

Встроенный помощник по выбору

Позволяет выполнять приведение и проверку на основе списка вариантов. Например:

```
>>> from decouple import config, Choices
>>> os.environ['CONNECTION_TYPE'] = 'usb'
>>> config('CONNECTION_TYPE', cast=Choices(['eth', 'usb', 'serial']))
'usb'

>>> os.environ['CONNECTION_TYPE'] = 'serial'
>>> config('CONNECTION_TYPE', cast=Choices(['eth', 'usb', 'serial']))
Traceback (most recent call last):
...
ValueError: Value not in list: 'serial'; valid values are 'eth', 'usb'
```

Вы также можете параметризовать *помощник выбора* для приведения к другому типу:

```
>>> os.environ['SOME_NUMBER'] = '42'
>>> config('SOME_NUMBER', cast=Choices([7, 14, 42], cast=int))
42
```

Вы также можете использовать кортеж вариантов, подобный Django:

```
>>> USB = 'usb'
>>> ETH = 'eth'
>>> BLUETOOTH = 'bluetooth'
>>>
>>> CONNECTION_OPTIONS = (
...     (USB, 'USB'),
...     (ETH, 'Ethernet'),
...     (BLUETOOTH, 'Bluetooth'),)
...
>>> os.environ['CONNECTION_TYPE'] = BLUETOOTH
>>> config('CONNECTION_TYPE', cast=Choices(choices=CONNECTION_OPTIONS,
...     'bluetooth'))
```

Помочь

Ваш вклад приветствуется.

Настройте свою среду разработки:

```
git clone git@github.com:henriquebastos/python-decouple
cd python-decouple
python -m venv .venv
source .venv/bin/activate
pip install -r requirements.txt
tox
```

Decouple поддерживает как Python 2.7, так и 3.6.

Убедитесь, что у вас установлены оба.

Я использую [pyenv](#) для управления несколькими версиями Python, и я описал настройку своего рабочего пространства в этой статье: [The definitive guide to setup my Python workspace](#)

Вы можете отправлять запросы на извлечение и проблемы для обсуждения. Однако я рассматриваю только объединение протестированного кода.

Лицензия

The MIT License (MIT)

Авторское право (с) 2017 Энрике Бастос <энрике в bastos dot net>

Настоящим предоставляется бесплатное разрешение любому лицу, получающему копию этого программного обеспечения и связанных с ним файлов документации ("Программное обеспечение"), осуществлять операции с Программным обеспечением без ограничений, включая, без ограничения, права на использование, копирование, модификацию, объединение, публикацию, распространение, сублицензирование и / или продажу копий Программного обеспечения, а также разрешать лицам, которым предоставляется Программное обеспечение, делать это при соблюдении следующих условий:

Вышеуказанное уведомление об авторских правах и данное уведомление о разрешении должны быть включены во все копии или существенные части Программного обеспечения.

ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ ПРЕДОСТАВЛЯЕТСЯ "КАК ЕСТЬ", БЕЗ КАКИХ-ЛИБО ГАРАНТИЙ, ЯВНЫХ ИЛИ ПОДРАЗУМЕВАЕМЫХ, ВКЛЮЧАЯ, НО НЕ ОГРАНИЧИВАЯСЬ ГАРАНТИЯМИ ТОВАРНОЙ ПРИГОДНОСТИ, ПРИГОДНОСТИ ДЛЯ ОПРЕДЕЛЕННОЙ ЦЕЛИ И ОТСУТСТВИЯ НАРУШЕНИЙ. АВТОРЫ ИЛИ ПРАВООБЛАДАТЕЛИ НИ ПРИ КАКИХ ОБСТОЯТЕЛЬСТВАХ НЕ НЕСУТ ОТВЕТСТВЕННОСТИ ЗА КАКИЕ-ЛИБО ПРЕТЕНЗИИ, УБЫТКИ ИЛИ ИНУЮ ОТВЕТСТВЕННОСТЬ, БУДЬ ТО В РЕЗУЛЬТАТЕ ДЕЙСТВИЯ КОНТРАКТА, ДЕЛИКТА ИЛИ ИНЫМ ОБРАЗОМ, ВЫТЕКАЮЩИЕ ИЗ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ ИЛИ ИСПОЛЬЗОВАНИЯ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ ИЛИ В СВЯЗИ С НИМ.



Помощь

Установка пакетов [↗](#)
Загрузка пакетов [↗](#)
Руководство пользователя [↗](#)
Project name retention [↗](#)
Часто задаваемые вопросы

О PyPI

PyPI в Twitter [↗](#)
Панель мониторинга
инфраструктуры [↗](#)
Статистика
Логотипы и товарные знаки
Наши спонсоры

Внесение вклада в PyPI

Использование PyPI

- Программные ошибки и обратная связь
- Внесение вклада на GitHub'e
- Перевод PyPI
- Спонсор PyPI
- Вклад в разработку

- Нормы поведения
- Сообщить о проблеме безопасности
- Политика конфиденциальности
- Условия пользования
- Политика допустимого использования

Статус: Service Under Maintenance

Разработано и поддерживается сообщество Python'а для сообщества Python'а.
Пожертвуйте сегодня!

PyPI", "Python Package Index" и логотипы блоков являются зарегистрированными
товарными знаками Python Software Foundation.

© 2023 Python Software Foundation
Карта сайта

Переключиться на настольную версию

English

español

français

日本語

português (Brasil)

українська

Ελληνικά

Deutsch

中文 (简体)

中文 (繁體)

> русский

עברית

esperanto



AWS
Cloud computing
and Security
Sponsor

Datadog
Monitoring

Fastly
CDN

Google
Download
Analytics

Microsoft
PSF Sponsor

Pingdom
Monitoring

Sentry
Error logging

StatusPage
Status page