Get started Open in app

Nithin K

Follow 17 Followers About

Separating Django application config and secrets from code : python decouple

Nithin K Jun 26, 2018 · 2 min read

A web application whether small or large will be having a lot of app secrets such as SECRET_KEY, api keys of different services used, database credentials etc. The convenient way for anyone to use these is to hardcode them in source code. It works, but it is also the most insecure way. We often push the code to repositories for version management or we share it among others resulting in exposing the app secrets. The easiest and secure way is to use these secrets as environment variable and import them directly in app. Python decouple manages this for you. It also helps in managing application configurations and secrets based on the development environments (DEV/PROD/STAGING).

> *If you already did the mistake of pushing code with sensitive information to github refer https://medium.com/@nithinkvijayan/https-medium-com-nithinkvijayan-removing-sensitive-information-from-a-git-repo-f13b5d9787da*

Decouple was originally designed for Django, but currently exist as a independent tool for separating setting from code. It helps in changing settings of an application without needing to build and redeploy the entire app.

Installation and usage

Install python-decouple using

```
pip install python-decouple.
```

Once installed you can use it by importing config object of decouple module

```
from decouple import config
```

Storing application settings

The application settings for an instance, such as ALLOWED_HOSTS or DEBUG mode etc can be stored either in environment variables or in a .env file at project root. Decouple will search for these in below order

1. Environment variables
2. In .env file at project root

The content of the file should be just variables and values per line. Anything which you want to import in setting can go here. Let it be configuration or secrets.

```
DEBUG=True
EMAIL_PORT=25
EMAIL_HOST=smtp.mydomain.com
DATABASE_URL=mysql://user:pass@host/mydb
API_KEY=myapitoken
ALLOWED_HOSTS=mydomain.com,localhost
```

> *Tip: You can use .env file for a local development to define env variables. While on production deployment use environment variables of the instance. Also remember to add .env to your gitignore file*

Using the stored values in application

The values stored in env file or environment variables can be used in application via config object. The argument to config object should be same as the variable name defined earlier. In my settings.py i can use below code.

```
from decouple import config

DATABASE_URL = config('DATABASE_URL')
EMAIL_HOST = config('EMAIL_HOST')
API_KEY = config('API_KEY')
```

By default config returns a string. But Django needs values for settings such as DEBUG to be a boolean and EMAIL_PORT to be a integer. This can be done by passing an additional parameter 'cast' to config so that value will be converted to specified type before assignment.

```
DEBUG = config('DEBUG', cast=bool)
EMAIL_PORT = config('EMAIL_PORT', cast=int)
```

In case of ALLOWED_HOSTS the type should be a list. We can do that as well using the builtin Csv Helper

```
from decouple import Csv

ALLOWED_HOSTS = config('ALLOWED_HOSTS', cast=Csv())
```

Or for tuple

```
HOSTS = config('HOSTS', cast=Csv(tuple_=True))
```

Supplying default values

We can also supply default values to use so that the corresponding variable will be set to that value if the mentioned key is absent.

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

For more on decouple visit https://github.com/henriquebastos/python-decouple/

Django Python Python Decouple Web Development Web App Security