

---

# rf-django-envconfig

*Release 0.1.0*

December 15, 2016



<b>1</b>	<b>Overview</b>	<b>1</b>
1.1	Related work . . . . .	1
<b>2</b>	<b>Installation</b>	<b>3</b>
<b>3</b>	<b>Usage</b>	<b>5</b>
<b>4</b>	<b>Reference</b>	<b>7</b>
4.1	rf_django_envconfig . . . . .	7
<b>5</b>	<b>Contributing</b>	<b>9</b>
5.1	Bug reports . . . . .	9
5.2	Documentation improvements . . . . .	9
5.3	Feature requests and feedback . . . . .	9
5.4	Development . . . . .	9
<b>6</b>	<b>Authors</b>	<b>11</b>
<b>7</b>	<b>Changelog</b>	<b>13</b>
7.1	0.1.0 (2016-12-15) . . . . .	13
<b>8</b>	<b>Indices and tables</b>	<b>15</b>
	<b>Python Module Index</b>	<b>17</b>



---

## Overview

---

docs	
tests	
package	

Helpers to configure Django settings by environment variables

The goal of this package is to help making your Django settings configurable via environment variables, while keeping everything as simple and decoupled as possible.

## 1.1 Related work

- <https://12factor.net/>
- <https://github.com/doismellburning/django12factor>
- <https://github.com/joke2k/django-enviro>

### 1.1.1 Installation

```
pip install rf-django-envconfig
```

### 1.1.2 Documentation

<https://rf-django-envconfig.readthedocs.io/>

### 1.1.3 Development

To run the all tests run:

```
tox
```

Note, to combine the coverage data from all the tox environments run:

Windows	<code>set PYTEST_ADDOPTS=--cov-append tox</code>
Other	<code>PYTEST_ADDOPTS=--cov-append tox</code>

---

## Installation

---

At the command line:

```
pip install rf-django-envconfig
```





---

### Usage

---

To use rf-django-envconfig in a project:

```
import rf_django_envconfig
```



---

## Reference

---

### 4.1 rf\_django\_envconfig

`rf_django_envconfig.get_bool(env, name, default=False)`

Get a boolean value from the environment

If name is not found in env, return True if default evaluates to True, otherwise return False.

The following values are considered False:

- 'False'
- 'false'
- 'Off'
- 'off'
- '0'
- 'No'
- 'no'
- ''

Any other value is considered True.

`rf_django_envconfig.get_db_url(env, name, default='')`

Get a settings.DATABASES entry from a URL in the environment

This is just a thin convenience wrapper around dj-database-url.

If name is not found in env, return the result of parsing default.

```
DATABASES = {
    'default': get_db_url(
        os.environ,
        'DATABASE_URL',
        default='postgres://myuser:mypassword@localhost/myproject'
    )
}
```

`rf_django_envconfig.get_list(env, name, default=None)`

Get a list from the environment

The input is assumed to be a comma-separated list of strings.

If name is not found in env, return default. Note that default is returned as-is, so you should usually specify it as a list:

```
ALLOWED_HOSTS = get_list(os.environ, 'ALLOWED_HOSTS', ['localhost', ])
```

---

## Contributing

---

Contributions are welcome, and they are greatly appreciated! Every little bit helps, and credit will always be given.

### 5.1 Bug reports

When [reporting a bug](#) please include:

- Your operating system name and version.
- Any details about your local setup that might be helpful in troubleshooting.
- Detailed steps to reproduce the bug.

### 5.2 Documentation improvements

rf-django-envconfig could always use more documentation, whether as part of the official rf-django-envconfig docs, in docstrings, or even on the web in blog posts, articles, and such.

### 5.3 Feature requests and feedback

The best way to send feedback is to file an issue at <https://github.com/rfleschenberg/rf-django-envconfig/issues>.

If you are proposing a feature:

- Explain in detail how it would work.
- Keep the scope as narrow as possible, to make it easier to implement.
- Remember that this is a volunteer-driven project, and that code contributions are welcome :)

### 5.4 Development

To set up *rf-django-envconfig* for local development:

1. Fork [rf-django-envconfig](#) (look for the “Fork” button).
2. Clone your fork locally:

```
git clone git@github.com:your_name_here/rf-django-envconfig.git
```

3. Create a branch for local development:

```
git checkout -b name-of-your-bugfix-or-feature
```

Now you can make your changes locally.

4. When you're done making changes, run all the checks, doc builder and spell checker with `tox` one command:

```
tox
```

5. Commit your changes and push your branch to GitHub:

```
git add .
git commit -m "Your detailed description of your changes."
git push origin name-of-your-bugfix-or-feature
```

6. Submit a pull request through the GitHub website.

### 5.4.1 Pull Request Guidelines

If you need some code review or feedback while you're developing the code just make the pull request.

For merging, you should:

1. Include passing tests (run `tox`)<sup>1</sup>.
2. Update documentation when there's new API, functionality etc.
3. Add a note to `CHANGELOG.rst` about the changes.
4. Add yourself to `AUTHORS.rst`.

### 5.4.2 Tips

To run a subset of tests:

```
tox -e envname -- py.test -k test_myfeature
```

To run all the test environments in *parallel* (you need to `pip install detox`):

```
detox
```

---

<sup>1</sup> If you don't have all the necessary python versions available locally you can rely on Travis - it will run the tests for each change you add in the pull request.  
It will be slower though ...

---

**Authors**

---

- René Fleschenberg - <https://fleschenberg.net>





---

## Changelog

---

### 7.1 0.1.0 (2016-12-15)

- First release on PyPI.



---

## Indices and tables

---

- `genindex`
- `modindex`
- `search`



## r

`rf_django_envconfig`, [7](#)



## G

`get_bool()` (in module `rf_django_envconfig`), [7](#)  
`get_db_url()` (in module `rf_django_envconfig`), [7](#)  
`get_list()` (in module `rf_django_envconfig`), [7](#)

## R

`rf_django_envconfig` (module), [7](#)