# django-columns Documentation

***Release 0.1.0***

**Audrey Roy Greenfeld**

December 19, 2016

Contents

Contents:

CHAPTER 1

# django-columns

Django template filter for splitting a list into columns.

## 1.1 Documentation

The full documentation is at https://django-columns.readthedocs.org.

## 1.2 Quickstart

1. Install the package. At the command line:

```
$ pip install django-columns
```

2. Add *columns* to *INSTALLED_APPS*.

3. Using django-columns is easy. Front-end developers and designers will find it particularly useful.

To split a list into 2 lists, to fill 2 Bootstrap grid columns:

```
{% load columns %}

<div class="row">
    {% for col in mylist|columns:2 %}
        <div class="col-md-6">
            {% for item in col %}
                <div class="item">{{ item }}</div>
            {% endfor %}
        </div><!-- /col-md-6 -->
    {% endfor %}
</div><!-- /row -->
```

Similarly, to split a list into 3 lists, to fill 3 columns:

```
{% load columns %}

<div class="row">
    {% for col in mylist|columns:3 %}
        <div class="col-md-4">
            {% for item in col %}
                <div class="item">{{ item }}</div>
            {% endfor %}
```

```
        </div><!-- /col-md-4 -->
    {% endfor %}
</div><!-- /row -->
```

These examples use Bootstrap-style columns, but you can use any other grid framework with django-columns.

CHAPTER 2

# Installation

1. Install the package. At the command line:

```
$ pip install django-columns
```

2. Add *columns* to *INSTALLED_APPS*.

CHAPTER 3

# Usage

A common use case is for splitting a list into 2 lists, to fill 2 columns:

```
{% load columns %}

<div class="row">
    {% for col in mylist|columns:3 %}
        <div class="col-md-4">
            {% for item in col %}
                <div class="item">{{ item }}</div>
            {% endfor %}
        </div><!-- /col-md-4 -->
    {% endfor %}
</div><!-- /row -->
```

CHAPTER 4

# Contributing

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

You can contribute in many ways:

## 4.1 Types of Contributions

### 4.1.1 Report Bugs

Report bugs at https://github.com/audreyr/django-columns/issues.

If you are 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.

### 4.1.2 Fix Bugs

Look through the GitHub issues for bugs. Anything tagged with "bug" is open to whoever wants to implement it.

### 4.1.3 Implement Features

Look through the GitHub issues for features. Anything tagged with "feature" is open to whoever wants to implement it.

### 4.1.4 Write Documentation

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

### 4.1.5 Submit Feedback

The best way to send feedback is to file an issue at https://github.com/audreyr/django-columns/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 contributions are welcome :)

## 4.2 Get Started!

Ready to contribute? Here's how to set up *django-columns* for local development.

1. Fork the *django-columns* repo on GitHub.
2. Clone your fork locally:

```
$ git clone git@github.com:your_name_here/django-columns.git
```

3. Install your local copy into a virtualenv. Assuming you have virtualenvwrapper installed, this is how you set up your fork for local development:

```
$ mkvirtualenv django-columns
$ cd django-columns/
$ python setup.py develop
```

4. Create a branch for local development:

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

Now you can make your changes locally.

5. When you're done making changes, check that your changes pass flake8 and the tests, including testing other Python versions with tox:

```
$ flake8 columns tests
$ python setup.py test
$ tox
```

To get flake8 and tox, just pip install them into your virtualenv.

6. 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
```

7. Submit a pull request through the GitHub website.

## 4.3 Pull Request Guidelines

Before you submit a pull request, check that it meets these guidelines:

1. The pull request should include tests.
2. If the pull request adds functionality, the docs should be updated. Put your new functionality into a function with a docstring, and add the feature to the list in README.rst.
3. The pull request should work for Python 2.6, 2.7, and 3.3, and for PyPy. Check https://travis-ci.org/audreyr/django-columns/pull_requests and make sure that the tests pass for all supported Python versions.

## 4.4 Tips

To run a subset of tests:

```
$ python -m unittest tests.test_columns
```

CHAPTER 5

# Credits

## 5.1 Original Snippet

This is originally based on a snippet by Chris Beaven (https://github.com/smileychris). (It has been altered quite a bit. Don't expect the same behavior!)

## 5.2 Development Lead

- Audrey Roy Greenfeld <audreyr@gmail.com>

## 5.3 Contributors

None yet. Why not be the first?

CHAPTER 6

# History

## 6.1 0.1.0 (2014-05-12)

- First release on PyPI.