django-groups-cache Documentation

Release 1.0.2

Ryan Castner

Jan 06, 2018

Contents

Contents:

CHAPTER 1

django-groups-cache

Caches the groups a user is in so requests don't have to make calls to the database to check group status.

1.1 Support

Currently supporting Django 1.8, 1.11, and 2.0 with Python 2.7/Python 3 where support for versions aligns with Django's support. See the *tox.ini* file for specific Python and Django version pairings.

1.2 Documentation

The full documentation is at https://django-groups-cache.readthedocs.io.

1.3 Quickstart

Install django-groups-cache:

```
pip install django-groups-cache
```

Add it to your *INSTALLED_APPS*:

```
INSTALLED_APPS = (
    ...
    'groups_cache.apps.GroupsCacheConfig',
    ...
)
```

Add the middleware to your *MIDDLEWARE_CLASSES*:

```
MIDDLEWARE_CLASSES = (
    'django.contrib.sessions.middleware.SessionMiddleware',
    'django.contrib.auth.middleware.AuthenticationMiddleware',
    ...
    'groups_cache.middleware.GroupsCacheMiddleware',
)
```

Checking in a Django Template if the user is in a group name:

```
{% if "admins" in request.groups_cache %}
  <a href="/admin">Admin Panel</a>
{% endif %}

# or use templatetag, note that templatetag is slower

{% load has_group %}

{% if request.user|has_group:"admins" %}
  <a href="/admin">Admin Panel</a>
{% endif %}
```

Turn on caching:

```
CACHES = {
    'default': {
        'BACKEND': 'django.core.cache.backends.memcached.MemcachedCache',
        'LOCATION': '127.0.0.1:11211',
    }
}
```

See https://docs.djangoproject.com/en/1.10/topics/cache/#memcached for more details on setting up memcached.

Note**

Using *django.core.cache.backends.locmem.LocMemCache* is not safe for production unless you are only running a single process (and odds are you aren't).

See https://docs.djangoproject.com/en/1.10/topics/cache/#local-memory-caching for more details.

1.4 Features

- Adds *groups_cache* property to *request* object
- Provides templatetag *has_group*
- Invalidates cache on *User* or *Group* model changes and on m2m *groups* ManyToManyField changes
- Fully tested with high coverage

1.5 Running Tests

```
source <YOURVIRTUALENV>/bin/activate
(myenv) $ pip install tox
(myenv) $ tox
```

1.6 Credits

Tools used in rendering this package:

- Cookiecutter
- cookiecutter-djangopackage

CHAPTER 2

Installation

At the command line:

```
$ easy_install django-groups-cache
```

Or, if you have virtualenvwrapper installed:

```
$ mkvirtualenv django-groups-cache
$ pip install django-groups-cache
```

CHAPTER 3

Usage

To use django-groups-cache in a project, add it to your *INSTALLED_APPS*:

```
INSTALLED_APPS = (
    ...
    'groups_cache.apps.GroupsCacheConfig',
    ...
)
```

Add django-groups-cache's URL patterns:

```
from groups_cache import urls as groups_cache_urls


urlpatterns = [
    ...
    url(r'^', include(groups_cache_urls)),
    ...
]
```

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/audiolion/django-groups-cache/issues.

If you are reporting a bug, please include:

- Operating system 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 issue tracker for bugs. Anything tagged with "bug" is open to whoever wants to implement it.

4.1.3 Implement Features

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

4.1.4 Write Documentation

django-groups-cache could always use more documentation, whether as part of the official django-groups-cache 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/audiolion/django-groups-cache/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-groups-cache* for local development.

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

```
$ git clone git@github.com:your_name_here/django-groups-cache.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-groups-cache
$ cd django-groups-cache/
$ 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 groups_cache 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 2.7, 3.3, and for PyPy. Check https://travis-ci.org/audiolion/django-groups-cache/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_groups_cache
```

CHAPTER 5

Credits

5.1 Development Lead

- Ryan Castner <castner.rr@gmail.com>

5.2 Contributors

None yet. Why not be the first?

CHAPTER 6

History

6.1 0.5.5 (2017-01-13)

- Bug fix for templatetag *has_group*
- Tests added for 100% coverage of templatetag folder

6.2 0.5.5 (2017-01-12)

- Omit urls.py from coverage report (not used but needed for django package)
- Omit apps.py from coverage report (default apps file)

6.3 0.5.4 (2017-01-12)

- Removal of py32/django1.8 support from Travis CI build

6.4 0.5.3 (2017-01-12)

- Fixing .travis.yml file and CI builds

6.5 0.5.2 (2017-01-12)

- Typo in README.rst

6.6 0.5.1 (2017-01-12)

- Add codecov.io support
- Documentation updates

6.7 0.5.0 (2017-01-12)

- Add requirements.txt to tox.ini so tests can run
- Fix broken compatibility with Django 1.8/1.9 due to backwards incompatible changes in 1.9/1.10 code

6.8 0.4.0 (2017-01-12)

- Add test suite that generates 100% coverage
- Fixed a bug found by test suite with cache not invalidating on the *groups* ManyToManyField changing

6.9 0.3.1 (2017-01-11)

- Documentation updates

6.10 0.3.0 (2017-01-11)

- First stable and working release on PyPI.

6.11 0.1.0 (2017-01-11)

- First release on PyPI.