django-reinhardt Documentation

Release 0.1.0

Hyuntak Joo

December 02, 2016

Contents

Contents:

CHAPTER 1

django-reinhardt

There are many object permission backends like django-guardian or django-permission.

But some time, it is needed to define permissions as not just object-user relationship.

django-reinhardt make you handle object permissions by defining methods in your django model

- Free software: MIT license
- Documentation: https://django-reinhardt.readthedocs.io.

1.1 Installation

Use pip like:

```
$ pip install django-reinhardt
```

1.2 Usage

Add extra authorization backends in your settings.py:

```
AUTHENTICATION_BACKENDS = (
    'django.contrib.auth.backends.ModelBackend', # default
    'reinhardt.backends.PermissionBackend',
)
```

It's done. you don't need to add any app or migrate anything.

Assume that `Inquiry` model needs to have two permission: `change_inqury`, `view_inquiry`

```
class Inquiry(models.Model):

    writer = models.ForeignKey(settings.AUTH_USER_MODEL)
    text = models.TextField()
    pub_date = models.DateTimeField(auto_now_add=True)

    @object_permission(codename='change_inquiry')
    def is_changeable_by(self, user):
        return self.writer == user or user.is_staff

    @object_permission(codename='view_inquiry')
    def is_viewable_by(self, user):
        return self.writer == user
```

Then you can just define methods having `user` parameter, decorated by `object_permission`.

Now the following codes will work as expected:

```
user1 = get_user_model().objects.create(
    username='nanase'
)
user2 = get_user_model().objects.create(
    username='maiyan'
)
user3 = get_user_model().objects.create(
    username='ikuta'
)
inquiry = Inquiry.objects.create(
    writer=self.user1,
    text='How can I delete my account?'
)

assert user1.has_perm('yourapp.change_inquiry', obj=inquiry) == True
assert user2.has_perm('yourapp.view_inquiry', obj=inquiry) == False
assert user3.has_perm('yourapp.change_inquiry', obj=inquiry) == False
assert user3.has_perm('yourapp.view_inquiry', obj=inquiry) == True
```

1.3 Credits

This package was created with Cookiecutter and the audreyr/cookiecutter-pypackage project template.

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:

2.1 Types of Contributions

2.1.1 Report Bugs

Report bugs at https://github.com/momamene/reinhardt/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.

2.1.2 Fix Bugs

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

2.1.3 Implement Features

Look through the GitHub issues for features. Anything tagged with "enhancement" and "help wanted" is open to whoever wants to implement it.

2.1.4 Write Documentation

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

2.1.5 Submit Feedback

The best way to send feedback is to file an issue at https://github.com/momamene/django-reinhardt/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 :)

2.2 Get Started!

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

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

```
$ git clone git@github.com:your_name_here/django-reinhardt.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-reinhardt
$ cd django-reinhardt/
$ 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 django-reinhardt tests
$ python setup.py test or 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.

2.3 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 3.3, 3.4 and 3.5. Check https://travis-ci.org/momamene/django-reinhardt/pull_requests and make sure that the tests pass for all supported Python versions.

2.4 Tips

To run a subset of tests:

```
$ py.test
```

CHAPTER 3

Indices and tables

- genindex
- modindex
- search