mozilla-django-oidc Documentation

Release 0.1.8

Mozilla

Jun 15, 2017

Contents

Contents:

CHAPTER 1

Installation

At the command line:

```
$ pip install mozilla-django-oidc
```

Warning: We highly recommend to avoid using Django's cookie-based sessions because they might open you up to replay attacks.

Note: You can find more info about cookie-based sessions in Django's documentation.

Quick start

After installation, you'll need to configure your site to use `mozilla-django-oidc`. Start by making the following changes to your `settings.py` file.

```
# Add 'mozilla_django_oidc' to INSTALLED_APPS
INSTALLED_APPS = (
    # ...
    'django.contrib.auth',
    'mozilla_django_oidc',  # Load after auth
    # ...
)

# Add 'mozilla_django_oidc' authentication backend
AUTHENTICATION_BACKENDS = (
    # ...
    'django.contrib.auth.backends.ModelBackend',
    'mozilla_django_oidc.auth.OIDCAuthenticationBackend',
```

```
    # ...
)
```

Next, edit your `urls.py` and add the following:

```
urlpatterns = patterns(
    # ...
    url(r'^oidc/', include('mozilla_django_oidc.urls')),
    # ...
)
```

Then you need to add the login link to your Django templates. For example:

```
<html>
  <body>
    {% if user.is_authenticated %}
      <p>Current user: {{ user.email }}</p>
    {% else %}
      <a href="{% url 'oidc_authentication_init' %}">Login</a>
    {% endif %}
  </body>
</html>
```

You also need to configure some OpenID connect related settings too. Please add the following to your `settings.py`:

```
OIDC_OP_AUTHORIZATION_ENDPOINT = "<URL of the OIDC OP authorization endpoint>"
OIDC_OP_TOKEN_ENDPOINT = "<URL of the OIDC OP token endpoint>"
OIDC_OP_USER_ENDPOINT = "<URL of the OIDC OP userinfo endpoint>"
OIDC_OP_CLIENT_ID = "<OP issued client id>"
OIDC_OP_CLIENT_SECRET = "<OP issued client secret>"
SITE_URL = "<FQDN that users access the site from eg. http://127.0.0.1:8000/ >"
```

Finally let your OpenID connect OP know about your callback URL. In our example this is: `http://127.0.0.1:8000/oidc/callback/`.

CHAPTER 2

Settings

This document describes the Django settings that can be used to customize the configuration of `mozilla-django-oidc`.

SITE_URL

Default No default

URL that users access your site from. Make sure that you provide the protocol, domain, path and port if needed (e.g. `<protocol>://<domain>:<port>/<path>`)

Note: This does not have to be a publicly accessible URL, so local URLs like `http://localhost:8000` or `http://127.0.0.1` are acceptable as long as they match what you are using to access your site.

OIDC_OP_AUTHORIZATION_ENDPOINT

Default No default

URL of your OpenID Connect provider authorization endpoint.

OIDC_OP_TOKEN_ENDPOINT

Default No default

URL of your OpenID Connect provider token endpoint

OIDC_OP_USER_ENDPOINT

Default No default

URL of your OpenID Connect provider userinfo endpoint

OIDC_RP_CLIENT_ID

Default No default

OpenID Connect client ID provided by your OP

OIDC_RP_CLIENT_SECRET

Default No default

OpenID Connect client secret provided by your OP

OIDC_RP_CLIENT_SECRET_ENCODED

Default `False`

Controls whether your client secret requires base64 decoding for verification

OIDC_VERIFY_JWT

Default `True`

Controls whether the OpenID Connect client verifies the signature of the JWT tokens

OIDC_USE_NONCE

Default `True`

Controls whether the OpenID Connect client uses nonce verification

OIDC_VERIFY_SSL

Default `True`

Controls whether the OpenID Connect client verifies the SSL certificate of the OP responses

OIDC_CREATE_USER

Default `True`

Enables or disables automatic user creation during authentication

OIDC_STATE_SIZE

Default `32`

Sets the length of the random string used for OpenID Connect state verification

OIDC_NONCE_SIZE

Default `32`

Sets the length of the random string used for OpenID Connect nonce verification

OIDC_REDIRECT_FIELD_NAME

Default `next`

Sets the GET parameter that is being used to define the redirect URL after succesful authentication

OIDC_CALLBACK_CLASS

Default `mozilla_django_oidc.views.OIDCAuthenticationCallbackView`

Allows you to substitute a custom class-based view to be used as OpenID Connect callback URL.

Note: When using a custom callback view, it is generally a good idea to subclass the default `OIDCAuthenticationCallbackView` and override the methods you want to change.

LOGIN_REDIRECT_URL

Default `/accounts/profile`

Path to redirect to on successful login. If you don't specify this, the default Django value will be used.

LOGIN_REDIRECT_URL_FAILURE

Default /

Path to redirect to on an unsuccessful login attempt.

LOGOUT_REDIRECT_URL

Default /

Path to redirect to on logout.

CHAPTER 3

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:

Types of Contributions

Report Bugs

Report bugs at https://github.com/mozilla/mozilla-django-oidc/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.

Fix Bugs

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

Implement Features

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

Write Documentation

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

Submit Feedback

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

Get Started!

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

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

```
$ git clone git@github.com:your_name_here/mozilla-django-oidc.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 mozilla-django-oidc
$ cd mozilla-django-oidc/
$ 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 mozilla_django_oidc 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.

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/mozilla/mozilla-django-oidc/pull_requests and make sure that the tests pass for all supported Python versions.

Tips

To run a subset of tests:

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

CHAPTER 4

Credits

Development Lead

- Tasos Katsoulas <akatsoulas@mozilla.com>
- John Giannelos <jgiannelos@mozilla.com>

Contributors

None yet. Why not be the first?

CHAPTER 5

History

0.1.0 (2016-10-12)

- First release on PyPI.

Index