Flask-OAuth2Server Documentation

Release 0.1.0.dev0

CERN

December 25, 2016

Contents

CHAPTER 1

Flask-OAuth2Server

Flask-OAuth2Server allows you to quickly add an OAuth2 provider to your Flask application.

CHAPTER 2

User's Guide

This part of the documentation will show you how to get started in using Flask-OAuth2Server with Flask.

2.1 Installation

Install Flask-OAuth2Server with `pip`

```
pip install flask-oauth2server
```

The development version can be downloaded from its page at GitHub.

```
git clone https://github.com/inveniosoftware/flask-oauth2server.git
cd flask-oauth2server
python setup.py develop
source run-tests.sh
```

2.1.1 Requirements

Flask-OAuth2Server has the following dependencies:

- Flask
- six

Flask-OAuth2Server requires Python version 2.6, 2.7 or 3.3+

2.2 Quickstart

This guide assumes you have successfully installed Flask-OAuth2Server and a working understanding of Flask. If not, follow the installation steps and read about Flask at http://flask.pocoo.org/docs/.

2.2.1 A Minimal Example

A minimal Flask-OAuth2Server usage example looks like this. First create the application and initialize the extension:

```
>>> from flask import Flask
>>> from flask_oauth2server import OAuth2Server
>>> app = Flask('myapp')
>>> ext = OAuth2Server(app=app)
```

2.2.2 Some Extended Example

Flask-OAuth2Server also has support for ...

```
# -*- coding: utf-8 -*-
#
# This file is part of Flask-OAuth2Server
# Copyright (C) 2014 CERN.
#
# Flask-OAuth2Server is free software; you can redistribute it and/or
# modify it under the terms of the Revised BSD License; see LICENSE
# file for more details.

"""Helper module to create an oauthclient for testing purposes."""

from unittest import TestCase
from flask import url_for, request, session, jsonify, abort, Flask
from flask_oauthlib.client import OAuth


class FlaskTestCase(TestCase):
    """
    Mix-in class for creating the Flask application
    """

    def setUp(self):
        app = Flask(__name__)
        app.config['DEBUG'] = True
        app.config['TESTING'] = True
        app.logger.disabled = True
        self.app = app


def create_client(app, name, **kwargs):
    """Helper function to create a OAuth2 client to test an OAuth2 provider."""
    default = dict(
        consumer_key='confidential',
        consumer_secret='confidential',
        request_token_params={'scope': 'test:scope'},
        base_url=app.config['CFG_SITE_SECURE_URL'],
        request_token_url=None,
        access_token_method='POST',
        access_token_url='%s/oauth/token' % app.config['CFG_SITE_SECURE_URL'],
        authorize_url='%s/oauth/authorize' % app.config['CFG_SITE_SECURE_URL'],
    )
    default.update(kwargs)

    oauth = OAuth(app)
    remote = oauth.remote_app(name, **default)

    @app.route('/oauth2test/login')
    def login():
```

```
        return remote.authorize(callback=url_for('authorized', _external=True))

    @app.route('/oauth2test/logout')
    def logout():
        session.pop('confidential_token', None)
        return "logout"

    @app.route('/oauth2test/authorized')
    @remote.authorized_handler
    def authorized(resp):
        if resp is None:
            return 'Access denied: error=%s' % (
                request.args.get('error', "unknown")
            )
        if isinstance(resp, dict) and 'access_token' in resp:
            session['confidential_token'] = (resp['access_token'], '')
            return jsonify(resp)
        return str(resp)

    def get_test(test_url):
        if 'confidential_token' not in session:
            abort(403)
        else:
            ret = remote.get(test_url)
            if ret.status != 200:
                return abort(ret.status)
            return ret.raw_data

    @app.route('/oauth2test/test-ping')
    def test_ping():
        return get_test(url_for("oauth2server.ping"))

    @app.route('/oauth2test/test-info')
    def test_info():
        return get_test(url_for('oauth2server.info'))

    @app.route('/oauth2test/test-invalid')
    def test_invalid():
        return get_test(url_for('oauth2server.invalid'))

    @remote.tokengetter
    def get_oauth_token():
        return session.get('confidential_token')

    return remote
```

2.3 User Guide

CHAPTER 3

API Reference

If you are looking for information on a specific function, class or method, this part of the documentation is for you.

3.1 API Docs

CHAPTER 4

Additional Notes

Notes on how to contribute, legal information and changelog are here for the interested.

4.1 Contributing

See <http://inveniosoftware.org/wiki/Development/Contributing> for now.

4.2 Changelog

Here you can see the full list of changes between each Flask-OAuth2Server release.

4.2.1 Version 0.1

- Initial public release

4.3 License

Flask-OAuth2Server is free software; you can redistribute it and/or modify it under the terms of the Revised BSD License quoted below.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

- Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
- Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
- Neither the name of the copyright holder nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

4.3.1 Authors

Flask-OAuth2Server is developed for use in Invenio digital library software.

Contact us at info@inveniosoftware.org

Contributors

- Lars Holm Nielsen <lars.holm.nielsen@cern.ch>
- Roman Chyla <roman.chyla@gmail.com>
- Konstantinos Kostis <konstantinos.kostis@cern.ch>
- Jiri Kuncar <jiri.kuncar@cern.ch>
- Eirini Psallida <eirini.psallida@cern.ch>