Flask-OpenVidu Documentation

Release 0.1.0

Marcell Pünkösd

Dec 28, 2022

Contents:

Adds OpenVidu support to your Flask application through PyOpenVidu.

- Free software: MIT license
- Documentation: https://flask-openvidu.readthedocs.io.

CHAPTER 1

Simple example

A basic Flask app that lists the currently active sessions on the server:

```
from flask import Flask
from flask_openvidu import OpenVidu

app = Flask(__name__)

app.config["OPENVIDU_URL"] = "https://example.com:4443/"
app.config["OPENVIDU_SECRET"] = "your_secret"

ov = OpenVidu(app)

@app.route('/sessions')
def sessions():
    text = ""
    for session in ov.connection.sessions:
        text += session.id + "\n"

    return text
```

CHAPTER 2

Credits

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

2.1 Documentation

2.1.1 Installation

Stable release

To install Flask-OpenVidu, run this command in your terminal:

```
$ pip install flask_openvidu
```

This is the preferred method to install Flask-OpenVidu, as it will always install the most recent stable release.

If you don't have pip installed, this Python installation guide can guide you through the process.

From sources

The sources for Flask-OpenVidu can be downloaded from the Github repo.

You can either clone the public repository:

```
$ git clone git://github.com/marcsello/flask_openvidu
```

Or download the tarball:

```
$ curl -OJL https://github.com/marcsello/flask_openvidu/tarball/master
```

Once you have a copy of the source, you can install it with:

```
$ python setup.py install
```

2.1.2 Usage

To use Flask-OpenVidu in a project, you have to import it to your Flask project:

```
from flask_openvidu import OpenVidu
```

Configuration is handled by Flask's configuration solution. See configuration for more details:

```
app = Flask(__name__)
app.config["OPENVIDU_URL"] = "https://example.com:4443/"
app.config["OPENVIDU_SECRET"] = "your_secret"
```

In order to use this object when handling requests, you have to bind it to the application like this:

```
openvidu = OpenVidu(app)
```

The so called factory pattern is supported as well:

```
openvidu = OpenVidu()

openvidu.init_app(app)
```

After everything is initialized properly, you can access to a PyOpenVidu object:

```
@app.route('/sessions')
def sessions():
    text = ""
    for session in openvidu.connection.sessions:
        text += session.id + "\n"

    return text
```

Currently a new OpenVidu object is created for every request which is valid in that request context.

2.1.3 Configuration

The configuration values supported by Flask-OpenVidu are described in the following table:

Configuration	Default	Description
OPENVIDU_URL	None	The url to reach your OpenVidu Server instance. Typically something like https://localhost:4443/ .
OPENVIDU_SECRET	None	Secret for your OpenVidu server.

2.1.4 Module Overview

flask_openvidu package

OpenVidu Class

Main module.

class flask_openvidu.flask_openvidu.**OpenVidu**(*app=None*)
Bases: object

This class provides an OpenVidu object configured by Flask.

Initialize the OpenVidu object according to Flask config. Factory pattern is supported as well. See *init_app()*.

Note: If app provided, an initial fetch() will be issued, as the OpenVidu object is created.

Parameters **app** – Optional Flask application to be bound.

connect() → pyopenvidu.openvidu.OpenVidu
Creates a new openvidu session instance that belongs to the current Flask application.

Returns an OpenVidu instance configured according to the Flask configuration.

connection
Get or create the OpenVidu instance belongs to the current Flask application.

Because of constructing a new object for every request, fetch() will be automatically called at the first time accessing to it in the request context. This means only a single fetch() call during the handle of each request.

Returns an OpenVidu instance configured according to the Flask configuration.

init_app(*app*)
Initialize the OpenVidu object according to Flask config.

Note: Calling this function will do an initial fetch() call, as the OpenVidu object is created.

Parameters **app** – Flask application to be bound.

2.1.5 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/marcsello/flask-openvidu/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” and “help wanted” is open to whoever wants to implement it.

Implement Features

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

Write Documentation

Flask-OpenVidu could always use more documentation, whether as part of the official Flask-OpenVidu 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/marcsello/flask-openvidu/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 *flask_openvidu* for local development.

1. Fork the *flask-openvidu* repo on GitHub.
2. Clone your fork locally:

```
$ git clone git@github.com:your_name_here/flask-openvidu.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 flask_openvidu
$ cd flask-openvidu/
$ 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 flask_openvidu tests
$ python setup.py test or pytest
$ 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 3.7 and 3.8, and for PyPy. Check https://travis-ci.com/marcsello/flask_openvidu/pull_requests and make sure that the tests pass for all supported Python versions.

Tips

To run a subset of tests:

```
$ pytest
```

Deploying

A reminder for the maintainers on how to deploy. Make sure all your changes are committed (including an entry in HISTORY.rst). Then run:

```
$ bump2version patch # possible: major / minor / patch
$ git push
$ git push --tags
```

Travis will then deploy to PyPI if tests pass.

2.1.6 Credits

Development Lead

- Marcell Pünkösd <punkosdmarcell@rocketmail.com>

Contributors

None yet. Why not be the first?

2.1.7 History

0.1.0 (2020-04-29)

- First release on PyPI.

2.2 Indices and tables

- genindex
- modindex
- search

Python Module Index

f

Index