# Flask-Web3

*Release 0.1.1*

Jul 08, 2018

# Contents

Welcome to Flask-Web3's documentation.

CHAPTER 1

# User's Guide

## 1.1 About Flask-Web3

Flask-Web3 is a flask extension allowing to smoothly integrate a flask application with web3.py. This package is intended for developers will to build a Flask application that interacts with an Ethereum client.

This page gives a good introduction to Flask-Web3. If not yet install please refer to the Installation section.

It is recommended that you have some light knowledge of web3.py before you try working with Flask-Web3.

### 1.1.1 A simple example

```
>>> from flask import Flask, jsonify
>>> from flask_web3 import current_web3, FlaskWeb3

# Declare Flask application
>>> app = Flask(__name__)

# Set Flask-Web3 configuration
>>> app.config.update({'ETHEREUM_PROVIDER': 'http', 'ETHEREUM_ENDPOINT_URI': 'http://
↪localhost:8545'})

# Declare Flask-Web3 extension
>>> web3 = FlaskWeb3(app=app)

# Declare route
>>> @app.route('/blockNumber')
... def block_number():
...     return jsonify({'data': current_web3.eth.blockNumber})
```

You can notice that Flask-Web3 gives you an application context bound variable `current_web3` that is accessible from any active flask application context.

### 1.1.2 An advanced example

You may like to declare your Flask-Web3 extension from a customize Web3 class with enhanced logic.

```
>>> from flask import Flask, jsonify
>>> from flask_web3 import current_web3, FlaskWeb3
>>> from web3 import Web3

# Declare Flask application
>>> app = Flask(__name__)
>>> app.config.update({'ETHEREUM_PROVIDER': 'http', 'ETHEREUM_ENDPOINT_URI': 'http://
↪localhost:8545'})

# Declare a custom Web3 class
>>> class CustomWeb3(Web3):
...     def customBlockNumber():
...         return self.eth.blockNumber

# Associate a custom FlaskWeb3 extension
>>> class CustomFlaskWeb3(FlaskWeb3):
...     web3_class = CustomWeb3

# Declare customized web3 extension
>>> web3 = CustomFlaskWeb3(app=app)
>>> isinstance(web3, CustomWeb3)
True

# Declare route
>>> @app.route('/customBlockNumber')
... def last_odd_block_number():
...     return jsonify({'data': current_web3.customBlockNumber()})
```

### 1.1.3 Flask-Web3 configuration

| Key | Comment | Default |
|---|---|---|
| `ETHEREUM_PROVIDER` | Type of Ethereum provider to use can be one of `http`, `ipc`, `ws` or `test` | `http` |
| `ETHEREUM_ENDPOINT_URI` | Endpoint URI of Ethereum client (only useful when provider is `http` or `ws`) | `http` |
| `ETHEREUM_IPC_PATH` | IPC path of Ethereum client (only useful when provider is `ipc`) | None |
| `ETHEREUM_OPTS` | A dictionary containing extra options fed to the provider when declaring it | `{}` |

CHAPTER 2

# API Reference

## 2.1 API

This part of the documentation covers all the interfaces of Flask-Web3.

### 2.1.1 Extension

**class** flask_web3.extension.**FlaskWeb3**(**args*, *app=None*, *create_provider=<function create_provider>*, ***kwargs*)

Main class for declaring a flask extension

**Parameters**

- **app** (*flask.Flask*) – Flask application or blueprint object to extend
- **create_provider** (*A function taking a :class:flask.Flask application configuration as parameter*) – Function used to create a Web3 provider

**init_app**(*app*)

Initialize application

**Parameters** **app** (*flask.Flask*) – Flask application or blueprint object to extend

### 2.1.2 Utils

flask_web3.utils.**create_provider**(*config*)

Create a web3.py provider

**Parameters** **config** (*dict*) – Provider configuration

CHAPTER 3

# Contributing

If you are interested in contributing to the project please refer to *Contributing guidelines*

## 3.1 Contributing guidelines

### 3.1.1 Feature Requests, Bug Reports, and Feedback...

...should all be reported on the GitHub Issue Tracker .

#### Reporting issues

- Describe what you expected to happen.
- If possible, include a minimal, complete, and verifiable example to help
- Describe what actually happened. Include the full traceback if there was an exception.

### 3.1.2 Setting-Up environment

#### Requirements

1. Having the latest version of `git` installed locally
2. Having Python 3.6 installed locally
3. Having `virtualenv` installed locally

   To install `virtualenv` you can run the following command

   ```
   $ pip install virtualenv
   ```

4. Having `docker` and `docker-compose` installed locally

5. Having `pip` environment variables correctly configured

   Some of the package's dependencies of the project could be hosted on a custom PyPi server. In this case you need to set some environment variables in order to make `pip` inspect the custom pypi server when installing packages.

   To set `pip` environment variables on a permanent basis you can add the following lines at the end of your `\.bashrc` file (being careful to replace placeholders)

```
# ~/.bashrc

...

# Indicate to pip which pypi server to download from
export PIP_TIMEOUT=60
export PIP_INDEX_URL=<custom_pypi_protocol>://<user>:<password>@<custom_pypi_host>
export PIP_EXTRA_INDEX_URL=https://pypi.python.org/simple
```

#### First time setup

- Clone the project locally
- Create development environment using Docker or Make

```
$ make init
```

### 3.1.3 Project organisation

The project

```
.
├── flask_web3/           # Main package source scripts (where all functional python␣
↪scripts are stored)
├── docs/                 # Docs module containing all scripts required by sphinx␣
↪to build the documentation
├── tests/                # Tests folder where all test modules are stores
├── .coveragerc           # Configuration file for coverage
├── .gitignore            # List all files pattern excluded from git's tracking
├── .gitlab-ci.yml        # GitLab CI script
├── AUTHORS               # List of authors of the project
├── CHANGES               # Changelog listing every changes from a release to␣
↪another
├── CONTRIBUTING.rst      # Indicate the guidelines that should be respected when␣
↪contributing on this project
├── LICENSE               # License of the project
├── Makefile              # Script implement multiple commands to facilitate␣
↪developments
├── README.rst            # README.md of your project
├── setup.cfg             # Configuration of extra commands that will be installed␣
↪on package setup
├── setup.py              # File used to setup the package
└── tox.ini               # Configuration file of test suite (it runs test suite␣
↪in both Python 3.5 and 3.6 environments)
```

### 3.1.4 Coding

#### Development Workflow

Please follow the next workflow when developing

- Create a branch to identify the feature or issue you will work on (e.g. `feature/my-feature` or `hotfix/2287`)
- Using your favorite editor, make your changes, committing as you go and respecting the AngularJS Commit Message Conventions
- Follow PEP8 and limit script's line length to **120 characters**. See *testing-linting*
- Include tests that cover any code changes you make. See *running-test* and *running-coverage*
- Update `setup.py` script with all dependencies you introduce. See *adding-dependency* for precisions
- Write clear and exhaustive docstrings. Write docs to precise how to use the functionality you implement. See *writing-docs*
- Update changelog with the modifications you proceed to. See *updating-changelog*
- Your branch will soon be merged ! :-)

#### Testing

#### Running tests

Run test suite in by running

```
$ make test
```

#### Running coverage

Please ensure that all the lines of source code you are writing are covered in your test suite. To generate the coverage report, please run

```
$ make coverage
```

Read more about coverage.

Running the full test suite with `tox` will combine the coverage reports from all runs.

#### Testing linting

To test if your project is compliant with linting rules run

```
$ make test-lint
```

To automatically correct linting errors run

```
$ make lint
```

### Running full test suite

Run test suite in multiple distinct python environment with following command

```
$ make tox
```

### Writing documentation

Write clear and exhaustive docstrings in every functional scripts.

This project uses sphinx to build documentations, it requires docs file to be written in `.rst` format.

To build the documentation, please run

```
$ make docs
```

### Precisions

### Updating changelog

Every implemented modifications on the project from a release to another should be documented in the changelog `CHANGES.rst` file.

The format used for a release block is be the following

```
Version <NEW_VERSION>
---------------------

Released on <NEW_VERSION_RELEASED_DATE>, codename <NEW_VERSION_CODENAME>.

Features

- Feature 1
- Feature 2
- Feature 3

Fixes

- Hotfix 1 (``#134``)
- Hotfix 2 (``#139``)

.. _#134: https://github.com/nmvalera/flask-web3/issues/134
.. _#139: https://github.com/nmvalera/sandbox/flask-web3/issues/139
```

Be careful to never touch the header line as well as the release's metadata sentence.

```
Version <NEW_VERSION>
---------------------

Released on <NEW_VERSION_RELEASED_DATE>, codename <NEW_VERSION_CODENAME>.
```

### Adding a new dependency

When adding a new package dependency it should be added in `setup.py` file in the `install_requires` list

The format should be `dependency==1.3.2`.

**When adding a dev dependency (e.g. a testing dependency) it should be added in**

- `setup.py` file in the `extra_requires` `dev` list
- `tox.ini` file in the `[testenv]` `deps`

### 3.1.5 Makefile commands

`Makefile` implements multiple handful shell commands for development

#### make init

**Initialize development environment including**

- venv creation
- package installation in dev mode

#### make clean

Clean the package project by removing some files such as `.pyc`, `.pyo`, `*.egg-info`

#### make test-lint

Check if python scripts are compliant with PEP8 rules

#### make lint

Automatically correct PEP8 mistakes contained in the project.

#### make coverage

Run the test suite and computes test coverage. It creates an html report that is automatically open after the commands terminates

#### make tox

Run the test suites in multiple environments

#### make docs

Build documentation from the `docs` folder using sphinx. It generates a build of the documentation in html format located in `docs/_build/html`.

CHAPTER 4

# Additional Notes

Legal information and changelog are here for the interested.

## 4.1 Changelog

Here you can see the full list of changes between each releases of Flask-Web3.

### 4.1.1 Version 0.1.1

Released on July 8th 2018

Fix

- Travis deployment password

### 4.1.2 Version 0.1.0

Released on July 8th 2018

Features

- Implement Flask-Web3 extension with possibility to customize the base Web3 class used
- Implement `create_provider` utility function that create a provider from a flask like configuration object
- Implement `current_web3` which is an Flask application context bound object

### 4.1.3 Version 0.0.0

Unreleased

Chore

- Project: Initialize project

## 4.2 License

### 4.2.1 Authors

Flask-Web3 is developed and maintained by the ConsenSys France team and community contributors. The core maintainers are:

- Nicolas Maurice (nmvalera)

### 4.2.2 General License Definitions

The following section contains the full license texts for Flask-Web3 and the documentation.

- "AUTHORS" hereby refers to all the authors listed in the *Authors* section.
- The "*License*" applies to all the source code shipped as part of Flask-Web3 (Flask-Web3 itself as well as the examples and the unit tests) as well as documentation.

### 4.2.3 License



Redistribution and use in source and binary forms of the software as well as documentation, 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 Flask-Web3 nor the names of the contributors may not be used to endorse or promote products derived from this software without specific prior written permission.

CHAPTER 5

# Indices and tables

- genindex
- modindex
- search

# Python Module Index

## f

# Index