succulent

Release 0.4.4

Tadej Lahovnik, Iztok Fister Jr.

Jul 16, 2026

USER DOCUMENTATION

1 General outline of the framework **3**

2 Detailed insights **5**

3 Documentation **7**

3.1 Getting started 7
3.2 Installation 9
3.3 Testing 9
3.4 Documentation 9
3.5 API Reference 10
3.6 License 14
3.7 Contributor Covenant Code of Conduct 15

Python Module Index **17**

Index **19**

succulent – A lightweight framework for collecting and processing data from HTTP POST requests

- **Free software:** MIT license
- **Github repository:** https://github.com/firefly-cpp/succulent
- **Python versions:** 3.7.x, 3.8.x, 3.9.x, 3.10.x, 3.11x, 3.12x, 3.13x

CHAPTER

ONE

GENERAL OUTLINE OF THE FRAMEWORK

Sending sensor measurements, data, or GPS positions from embedded devices, microcontrollers, and smartwatches to the central server is sometimes complicated and tricky. Setting up the primary data collection scripts can be time-consuming (selecting a protocol, framework, API, testing it, etc.). Usually, scripts are written for a specific task; thus, they are not easily adaptive to other tasks. succulent is a pure Python framework that simplifies the configuration, management, collection, and preprocessing of data collected via POST requests. The inspiration for the framework comes from the practical data collection challenges in smart agriculture. The main idea of the framework was to speed up the process of configuring different collected parameters and providing several useful functions for data transformations. The framework allows users to configure the whole endpoint for data collection in several minutes and thus not spend time on server-side scripts.

CHAPTER

TWO

DETAILED INSIGHTS

The current version includes (but is not limited to) the following functions:

- Request URL generation for data collection
- Data collection from POST requests
- Storing data in different formats (CSV, JSON, SQLite, XML, images)
- Defining boundaries for collected data (min, max)

CHAPTER

THREE

DOCUMENTATION

The documentation is organized into the following sections:

- *User documentation*
- *Developer documentation*
- *API Reference*
- *About*

3.1 Getting started

This section demonstrates the usage of the succulent framework.

3.1.1 Installation

pip

To install `succulent` with pip, use:

```
pip install succulent
```

Alpine Linux

To install `succulent` on Alpine Linux, use:

```
$ apk add py3-succulent
```

Arch Linux

To install `succulent` on Arch Linux, use an AUR helper:

```
$ yay -Syyu python-succulent
```

Fedora Linux

To install `succulent` on Fedora, use:

```
$ dnf install python3-succulent
```

3.1.2 Usage

Example

```
from succulent.api import SucculentAPI
api = SucculentAPI(host='0.0.0.0', port=8080, config='configuration.yml', format='csv')
api.start()
```

3.1.3 Configuration

Data collection

In the root directory, create a `configuration.yml` file and define the following:

```
data:
  - name: # Measure name
    min:  # Minimum value (optional)
    max:  # Maximum value (optional)
```

To collect images, create a `configuration.yml` file in the root directory and define the following:

```
data:
  - key: # Key in POST request
```

To store data collection timestamps, define the following setting in the `configuration.yml` file in the root directory:

```
timestamp: true # false by default
```

To access the URL for data collection, send a GET request (or navigate) to http://localhost:8080/measure.

To restrict access to the collected data, define the following setting in the `configuration.yml` file in the root directory:

```
password: 'password' # Password for data access
```

To store data using a password, append the password parameter to the request URL: `?password=password`.

Data access

To access data via the Succulent API, define the following setting in the `configuration.yml` file in the root directory:

```
results:
  - enable: true # false by default
```

To access the collected data, send a GET request (or navigate) to http://localhost:8080/data. To access password-protected data, append the password parameter to the request URL: `?password=password`.

Data export

To export the data, enable the export option in the configuration file:

```
export:
  - enable: true # false by default
```

To export the data, send a GET request (or navigate) to http://localhost:8080/export. To export password-protected data, append the password parameter to the request URL: `?password=password`. The data will be downloaded in the format specified in the configuration file.

3.2 Installation

3.2.1 Development environment

Requirements

- Python: https://www.python.org
- Poetry: https://python-poetry.org/docs

After installing Poetry and cloning the project from GitHub, execute the following command in the root directory of the cloned project:

```
$ poetry install
```

All of the project's dependencies should be installed and the project should be ready for further development. Note that Poetry creates a separate virtual environment for the project.

Development dependencies

List of succulent's dependencies:

Package	Version
pandas	^2.0.1
pyyaml	^6.0
flask	^2.3.2
xmltodict	^0.13.0
lxml	^5.1.0

List of succulent's development dependencies:

Package	Version
pytest	^6.2
sphinx	^4.4.0
sphinx-rtd-theme	^1.0.0
sphinxcontrib-bibtex	^2.4.1

3.3 Testing

Before making a pull request, provide tests for added features or bug fixes.

Any failed test cases should be resolved before the changes are merged into the main branch.

To check if all tests pass locally, run the following command:

```
$ poetry run pytest
```

3.4 Documentation

To locally generate and preview documentation, run the following commands in the root directory of the project:

```
poetry install
poetry run sphinx-build ./docs ./docs/_build
```

If the documentation builds successfully, it can be previewed in the `docs/_build` folder by opening the `index.html` file.

3.5 API Reference

3.5.1 API

class `succulent.api.`**`SucculentAPI`**(*config*, *host='0.0.0.0'*, *port=8080*, *format='csv'*)

Bases: `object`

Succulent API server.

Parameters

- **host** (*str*) – The host address to run the API server.
- **port** (*int*) – The port number to run the API server.
- **config** (*str*) – Path to the configuration file.
- **format** (*str, optional*) – The format of the data ('csv', 'json', 'sqlite', or 'image').

host

The host address to run the API server.

Type
str

port

The port number to run the API server.

Type
int

config

Configuration file.

Type
dict

format

The format of the data ('csv', 'json', 'sqlite', or 'image').

Type
str

app

Flask application.

Type
Flask

processing

Instance of the Processing class.

Type
Processing

data()

Display the stored data.

Returns
JSON response with the stored data.

Return type
Response

export()

Export the stored data to a CSV file.

index()

Generate index HTML page with information about the API.

Returns
HTML response with the index page.

Return type
Response

measure()

Process the incoming request and store the timestamp.

Returns
JSON response with a success message and timestamp.

Return type
Response

start()

Start the Flask application on the specified host and port.

Returns
None

url()

Generate URL with parameters for measurements.

Returns
JSON response with the generated URL.

Return type
Response

3.5.2 Configuration

class succulent.configuration.**Configuration**(*path*)

Bases: `object`

Class for loading configuration settings from a YAML file.

Parameters
path (*str*) – The path to the YAML configuration file.

path

The path to the YAML configuration file.

Type
str

load_config()

Load the configuration settings from the YAML file.

Returns
The loaded configuration settings as a dictionary.

Return type
dict

Raises
FileNotFoundError – The specified file does not exist.

3.5.3 Processing

class succulent.processing.**Processing**(*format*, *config*, *unittest=False*)

Bases: object

Class for processing and storing data.

Parameters

- **format** (*str*) – The format of the data ('csv', 'json', 'sqlite', 'image', or 'xml').
- **config** (*dict*) – The configuration settings for data processing.
- **unittest** (*bool, optional*) – If True, the class is being used for unit testing. Default is False.

directory

The path to storage location.

Type
str

format

The format of the data ('csv', 'json', 'sqlite', 'image', or 'xml').

Type
str

columns

List of feature names in the data.

Type
list

boundaries

List of minimum and maximum boundaries for each feature in the data.

Type
list

password

The password for data access. Default is None.

Type
str

timestamp

Enable timestamp storage for the data collection.

Type
bool

`enable_results`

Enable access to the collected data via the API.

Type
bool

`enable_export`

Enable collected data export via the API.

Type
bool

`df`

The DataFrame to hold the data (for 'csv', 'json', and 'sqlite' formats).

Type
pandas.DataFrame

`key`

The key to retrieve the image file from the request (for 'image' format).

Type
str

Raises
`ValueError` – Invalid format is provided.

`boundary`(*value*, *boundary*, *column*)

Check if the value is within the specified boundary for a column.

Parameters

- `value` (*float*) – The value to check.
- `boundary` (*dict*) – A dictionary containing 'min' and/or 'max' boundaries for the column.
- `column` (*str*) – The name of the column being checked.

Raises
`ValueError` – Value is outside the specified boundary.

Returns
The original value (within the boundary).

Return type
float

`data`(*req*)

Generate results based on the stored data.

Returns
The results as a dictionary.

Return type
dict

export(*req*)

Export the stored data.

Returns
The exported data as a dictionary.

Return type
dict

parameters()

Generate URL parameters based on data columns.

Returns
URL parameters as a string.

Return type
str

process(*req*)

Process the incoming request and store the data accordingly.

Parameters
req (*Request*) – The incoming request object.

Raises
ValueError – Unauthorised access.

Returns
None

3.6 License


```
MIT License

Copyright (c) 2023-2024 Iztok Fister Jr.

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the "Software"), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all
copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
SOFTWARE.
```

3.7 Contributor Covenant Code of Conduct

3.7.1 Our Pledge

In the interest of fostering an open and welcoming environment, we as contributors and maintainers pledge to making participation in our project and our community a harassment-free experience for everyone, regardless of age, body size, disability, ethnicity, gender identity and expression, level of experience, nationality, personal appearance, race, religion, or sexual identity and orientation.

3.7.2 Our Standards

Examples of behavior that contributes to creating a positive environment include:

- Using welcoming and inclusive language
- Being respectful of differing viewpoints and experiences
- Gracefully accepting constructive criticism
- Focusing on what is best for the community
- Showing empathy towards other community members

Examples of unacceptable behavior by participants include:

- The use of sexualized language or imagery and unwelcome sexual attention or advances
- Trolling, insulting/derogatory comments, and personal or political attacks
- Public or private harassment
- Publishing others' private information, such as a physical or electronic address, without explicit permission
- Other conduct which could reasonably be considered inappropriate in a professional setting

3.7.3 Our Responsibilities

Project maintainers are responsible for clarifying the standards of acceptable behavior and are expected to take appropriate and fair corrective action in response to any instances of unacceptable behavior.

Project maintainers have the right and responsibility to remove, edit, or reject comments, commits, code, wiki edits, issues, and other contributions that are not aligned to this Code of Conduct, or to ban temporarily or permanently any contributor for other behaviors that they deem inappropriate, threatening, offensive, or harmful.

3.7.4 Scope

This Code of Conduct applies both within project spaces and in public spaces when an individual is representing the project or its community. Examples of representing a project or community include using an official project e-mail address, posting via an official social media account, or acting as an appointed representative at an online or offline event. Representation of a project may be further defined and clarified by project maintainers.

3.7.5 Enforcement

Instances of abusive, harassing, or otherwise unacceptable behavior may be reported by contacting the project team. The project team will review and investigate all complaints, and will respond in a way that it deems appropriate to the circumstances. The project team is obligated to maintain confidentiality with regard to the reporter of an incident. Further details of specific enforcement policies may be posted separately.

Project maintainers who do not follow or enforce the Code of Conduct in good faith may face temporary or permanent repercussions as determined by other members of the project's leadership.

3.7.6 Attribution

This Code of Conduct is adapted from the Contributor Covenant, version 1.4, available at http://contributor-covenant.org/version/1/4.

Homepage: http://contributor-covenant.org
Version: http://contributor-covenant.org/version/1/4

PYTHON MODULE INDEX

S

succulent, **??**
succulent.api, 10
succulent.configuration, 11
succulent.processing, 12

INDEX

A

app (*succulent.api.SucculentAPI attribute*), 10

B

boundaries (*succulent.processing.Processing attribute*), 12
boundary() (*succulent.processing.Processing method*), 13

C

columns (*succulent.processing.Processing attribute*), 12
config (*succulent.api.SucculentAPI attribute*), 10
Configuration (*class in succulent.configuration*), 11

D

data() (*succulent.api.SucculentAPI method*), 10
data() (*succulent.processing.Processing method*), 13
df (*succulent.processing.Processing attribute*), 13
directory (*succulent.processing.Processing attribute*), 12

E

enable_export (*succulent.processing.Processing attribute*), 13
enable_results (*succulent.processing.Processing attribute*), 13
export() (*succulent.api.SucculentAPI method*), 11
export() (*succulent.processing.Processing method*), 13

F

format (*succulent.api.SucculentAPI attribute*), 10
format (*succulent.processing.Processing attribute*), 12

H

host (*succulent.api.SucculentAPI attribute*), 10

I

index() (*succulent.api.SucculentAPI method*), 11

K

key (*succulent.processing.Processing attribute*), 13

L

load_config() (*succulent.configuration.Configuration method*), 11

M

measure() (*succulent.api.SucculentAPI method*), 11
module
    succulent, 1
    succulent.api, 10
    succulent.configuration, 11
    succulent.processing, 12

P

parameters() (*succulent.processing.Processing method*), 14
password (*succulent.processing.Processing attribute*), 12
path (*succulent.configuration.Configuration attribute*), 11
port (*succulent.api.SucculentAPI attribute*), 10
process() (*succulent.processing.Processing method*), 14
Processing (*class in succulent.processing*), 12
processing (*succulent.api.SucculentAPI attribute*), 10

S

start() (*succulent.api.SucculentAPI method*), 11
succulent
    module, 1
succulent.api
    module, 10
succulent.configuration
    module, 11
succulent.processing
    module, 12
SucculentAPI (*class in succulent.api*), 10

T

timestamp (*succulent.processing.Processing attribute*), 12

U

`url()` (*succulent.api.SucculentAPI method*), 11