
Frater Documentation

John Henning

Apr 10, 2020

CONTENTS:

1	Getting Started	1
1.1	Configs	1
1.2	Components	1
1.3	Streams	2
2	Installation Guide	3
3	API	5
3.1	frater.client package	5
3.2	frater.component package	5
3.3	frater.config package	6
3.4	frater.data_store package	6
3.5	frater.data_type package	6
3.6	frater.dependency package	7
3.7	frater.factory package	7
3.8	frater.io package	8
3.9	frater.logging package	8
3.10	frater.server package	8
3.11	frater.stream package	9
3.12	frater.system package	9
3.13	frater.utilities package	9
3.14	frater.validation package	10
4	Indices and tables	11
	Python Module Index	13
	Index	15

GETTING STARTED

1.1 Configs

1.2 Components

1.2.1 IOComponent

To start off, we'll show how to work with probably the most common type of Component: `IOComponent`. `IOComponents` can be used for building components that require input and output streams.

Let's take a look with the following example where we build a simple summation component that adds up all the data that it receives over time.

```
from dataclasses import dataclass, field
from typing import List

from frater.component import ComponentState, IOComponent, IOComponentConfig, \
    ↳ComponentBuilder
from frater.stream import InputStream, OutputStream, StreamConfig, StreamState

@dataclass
class SummationComponentState(ComponentState):
    total: int = 0

class SummationComponent(IOComponent):
    def __init__(self, config: IOComponentConfig, input_stream: InputStream, output_
    ↳stream: OutputStream):
        super(SummationComponent, self).__init__(config, input_stream, output_stream)

    def init_state(self):
        return SummationComponentState()

    def process(self, data):
        self.state.total += data
        return self.state.total
```

Here we've defined the `SummationComponent` with its corresponding state `SummationComponentState` that will hold the state info of

1.3 Streams

INSTALLATION GUIDE

3.1 frater.client package

3.1.1 Submodules

frater.client.api module

```
class API (host, port, protocol='http://')  
    Bases: object  
    property api_url  
    build_endpoint_url (endpoint)  
    get (endpoint, params=None)  
    post (endpoint, data)  
    delete (endpoint, data)  
    patch (endpoint, data)  
    put (endpoint, data)
```

frater.client.component module

frater.client.system module

3.2 frater.component package

3.2.1 Subpackages

frater.component.builder package

Submodules

frater.component.builder.builder module

frater.component.component package

Submodules

`frater.component.component.batch_component` module

`frater.component.component.component` module

`frater.component.component.input_component` module

`frater.component.component.io_component` module

`frater.component.component.output_component` module

`frater.component.component.window_component` module

`frater.component.manager` package

Submodules

`frater.component.manager.manager` module

`frater.component.manager.thread` module

3.3 frater.config package

3.3.1 Submodules

`frater.config.config` module

3.4 frater.data_store package

3.4.1 Subpackages

`frater.data_store.file_store` package

Submodules

`frater.data_store.file_store.file_store` module

`frater.data_store.file_store.image_store` module

3.5 frater.data_type package

3.5.1 Submodules

`frater.data_type.data_type` module

frater.data_type.factory module

3.6 frater.dependency package

3.6.1 Submodules

frater.dependency.dependency module

3.7 frater.factory package

class Factory

Bases: object

register (*key*)

register_item (*key, value*)

unregister (*key*)

get (*key*)

get_registered_keys ()

get_registered_values ()

class ObjectFactory (*base_type=<class 'object'>*)

Bases: *frater.factory.factory.Factory*

get_registered_objects ()

register (*key*)

register_class (*key, derived*)

3.7.1 Submodules

frater.factory.factory module

class Factory

Bases: object

register (*key*)

register_item (*key, value*)

unregister (*key*)

get (*key*)

get_registered_keys ()

get_registered_values ()

frater.factory.object_factory module

```
class ObjectFactory (base_type=<class 'object'>)  
    Bases: frater.factory.factory.Factory  
    get_registered_objects ()  
    register (key)  
    register_class (key, derived)
```

3.8 frater.io package

3.8.1 Submodules

frater.io.deserializers module

frater.io.serializers module

3.9 frater.logging package

3.9.1 Subpackages

frater.logging.handler package

Submodules

frater.logging.handler.config module

frater.logging.handler.factory module

frater.logging.handler.kafka module

frater.logging.handler.standard module

3.9.2 Submodules

frater.logging.logger module

frater.logging.summary module

3.10 frater.server package

3.10.1 Submodules

frater.server.component module

frater.server.middleware module

frater.server.server_manager module

frater.server.system module

3.11 frater.stream package

3.11.1 Submodules

frater.stream.build module

frater.stream.factory module

frater.stream.json module

frater.stream.kafka module

frater.stream.log module

frater.stream.mongo module

frater.stream.stream module

frater.stream.stream_state module

3.12 frater.system package

3.12.1 Subpackages

frater.system.manager package

Submodules

frater.system.manager.manager module

3.13 frater.utilities package

3.13.1 Subpackages

frater.utilities.network package

Submodules

frater.utilities.network.config module

`frater.utilities.network.handler` module

3.13.2 Submodules

`frater.utilities.function` module

`frater.utilities.image` module

`frater.utilities.interpolation` module

`frater.utilities.json` module

`frater.utilities.kafka` module

`frater.utilities.singleton` module

`frater.utilities.url` module

3.14 `frater.validation` package

3.14.1 Submodules

`frater.validation.error` module

exception `ValidationError`

Bases: `Exception`

`frater.validation.json` module

`validate_json` (*_func=None, default=None, completion=True*)

`validate_helper` (*data, default*)

`add_defaults` (*data, default*)

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

f

- frater, 5
- frater.client.api, 5
- frater.factory, 7
- frater.factory.factory, 7
- frater.factory.object_factory, 8
- frater.validation, 10
- frater.validation.error, 10
- frater.validation.json, 10

A

add_defaults() (in module *frater.validation.json*),
10
API (class in *frater.client.api*), 5
api_url() (API property), 5

B

build_endpoint_url() (API method), 5

D

delete() (API method), 5

F

Factory (class in *frater.factory*), 7
Factory (class in *frater.factory.factory*), 7
frater
 module, 5
frater.client.api
 module, 5
frater.factory
 module, 7
frater.factory.factory
 module, 7
frater.factory.object_factory
 module, 8
frater.validation
 module, 10
frater.validation.error
 module, 10
frater.validation.json
 module, 10

G

get() (API method), 5
get() (Factory method), 7
get_registered_keys() (Factory method), 7
get_registered_objects() (ObjectFactory
 method), 7, 8
get_registered_values() (Factory method), 7

M

module

frater, 5
frater.client.api, 5
frater.factory, 7
frater.factory.factory, 7
frater.factory.object_factory, 8
frater.validation, 10
frater.validation.error, 10
frater.validation.json, 10

O

ObjectFactory (class in *frater.factory*), 7
ObjectFactory (class in
 frater.factory.object_factory), 8

P

patch() (API method), 5
post() (API method), 5
put() (API method), 5

R

register() (Factory method), 7
register() (ObjectFactory method), 7, 8
register_class() (ObjectFactory method), 7, 8
register_item() (Factory method), 7

U

unregister() (Factory method), 7

V

validate_helper() (in module
 frater.validation.json), 10
validate_json() (in module *frater.validation.json*),
10
ValidationError, 10