Tvilling Digital

Simen Norderud Jensen

Jun 08, 2019

CONTENTS:

CHAPTER

ONE

MAIN MODULE

The start point of the application.

class main.**Settings**(*settings_module*)
Bases: object

A class for holding the application settings

main.**main**(*args*)
Start the application.

Will be called with command line args if the file is run as a script

CHAPTER

TWO

SRC PACKAGE

2.1 Subpackages

2.1.1 src.blueprints package

2.1.1.1 Submodules

2.1.1.2 src.blueprints.views module

async src.blueprints.views.**blueprint_detail**(*request: aiohttp.web_request.Request*)
Get detailed information for the blueprint with the given id

async src.blueprints.views.**blueprint_list**(*request: aiohttp.web_request.Request*)
List all uploaded blueprints.

Append a blueprint id to get more information about a listed blueprint.

src.blueprints.views.**dumps**(*obj*, ***, *skipkeys=False*, *ensure_ascii=True*, *check_circular=True*, *allow_nan=True*, *cls=None*, *indent=None*, *separators=None*, *default=<function make_serializable>*, *sort_keys=False*, ***kw*)
A version of json.dumps that uses make serializable recursively to make objects serializable

async src.blueprints.views.**retrieve_method_info**(*class_body*, *method_name*, *params_ignore=1*) → Tuple[str, List]
Retrieves docs and parameters from the method

Parameters

- **class_body** – the body of the class the method belongs to
- **method_name** – the name of the method
- **params_ignore** – how many of the first params to ignore, defaults to 1 (only ignore self)

Returns a tuple containing both the docstring of the method and a list of parameters with name and default value

2.1.2 src.clients package

2.1.2.1 Submodules

2.1.2.2 src.clients.models module

class src.clients.models.**Client**
Bases: object

Handles connections to a clients websocket connections

async close()
Will close all the clients websocket connections

dict_repr() → dict
Returns a the number of connections the client has

async receive(*topic*, *bytes*)
Asynchronously transmit data to the clients websocket connections

Will add the data to the buffer and send it when the buffer becomes large enough

Parameters

- **topic** – the topic the data received from
- **bytes** – the data received as bytes

2.1.2.3 src.clients.views module

async src.clients.views.**client**(*request: aiohttp.web_request.Request*)
Show info about the client sending the request

src.clients.views.**dumps**(*obj*, *, *skipkeys=False*, *ensure_ascii=True*, *check_circular=True*, *allow_nan=True*, *cls=None*, *indent=None*, *separators=None*, *default=<function make_serializable>*, *sort_keys=False*, ***kw*)
A version of json.dumps that uses make serializable recursively to make objects serializable

2.1.3 src.datasources package

2.1.3.1 Submodules

2.1.3.2 src.datasources.models module

class src.datasources.models.**UdpDatasource**(*addr: Tuple[str, int]*, *input_byte_format: str*, *input_names: List[str]*, *output_refs: List[int]*, *time_index: int*, *topic: str = None*)
Bases: object

Represents a single UDP datasource

class src.datasources.models.**UdpReceiver**(*kafka_addr: str*)
Bases: asyncio.protocols.DatagramProtocol

Handles all UDP datasources

connection_lost(*exc: Optional[Exception]*) → None
Called when the connection is lost or closed.

The argument is an exception object or None (the latter meaning a regular EOF is received or the connection was aborted or closed).

connection_made(*transport: asyncio.transports.BaseTransport*) → None
Called when a connection is made.

The argument is the transport representing the pipe connection. To receive data, wait for data_received() calls. When the connection is closed, connection_lost() is called.

datagram_received(*raw_data: bytes, addr: Tuple[str, int]*) → None
Filters, transforms and buffers incoming packets before sending it to kafka

error_received(*exc: Exception*) → None
Called when a send or receive operation raises an OSError.

(Other than BlockingIOError or InterruptedError.)

get_sources()
Returns a list of the current sources

set_source(*source_id: str, addr: Tuple[str, int], topic: str, input_byte_format: str, input_names: List[str], output_refs: List[int], time_index: int*) → None
Creates a new datasource object and adds it to sources, overwriting if necessary

Parameters

- **source_id** – the id to use for the datasource
- **addr** – the address the datasource will send from
- **topic** – the topic the data will be put on
- **input_byte_format** – the byte_format of the data that will be received
- **input_names** – the names of the values in the data that will be received
- **output_refs** – the indices of the values that will be transmitted to the topic
- **time_index** – the index of the value that represents the time of the data

src.datasources.models.**generate_catman_outputs**(*output_names: List[str], output_refs, single: bool = False*) → Tuple[List[str], List[int], str]
Generate ouput setup for a datasource that is using the Catman software

Parameters

- **single** – true if the data from Catman is single precision (4 bytes each)
- **output_names** – a list of the names of the input data

2.1.3.3 src.datasources.views module

async src.datasources.views.**datasource_create**(*request: aiohttp.web_request.Request*)
Create a new datasource from post request.

Post parameters:

- id: the id to use for the source
- address: the address to receive data from
- port: the port to receive data from

- output_name: the names of the outputs Must be all the outputs and in the same order as in the byte stream.
- output_ref: the indexes of the outputs that will be used
- time_index: the index of the time value in the output_name list
- byte_format: the python struct format string for the data received. Must include byte order (https://docs.python.org/3/library/struct.html?highlight=struct#byte-order-size-and-alignment) Must be in the same order as name. Will not be used if catman is true.
- catman: set to true to use catman byte format byte_format is not required if set
- single: set to true if the data is single precision float Only used if catman is set to true

returns redirect to created simulation page

async src.datasources.views.**datasource_delete**(*request: aiohttp.web_request.Request*)
Delete the datasource

async src.datasources.views.**datasource_detail**(*request: aiohttp.web_request.Request*)
Information about the datasource with the given id. To delete the datasource append /delete To subscribe to the datasource append /subscribe To start the datasource append /start To stop the datasource append /stop

async src.datasources.views.**datasource_list**(*request: aiohttp.web_request.Request*)
List all datasources.

Listed datasources will contain true if currently running and false otherwise. Append an id to get more information about a listed datasource. Append /create to create a new datasource

async src.datasources.views.**datasource_start**(*request: aiohttp.web_request.Request*)
Start the datasource

async src.datasources.views.**datasource_stop**(*request: aiohttp.web_request.Request*)
Stop the server from retrieving data from the datasource with the given id.

async src.datasources.views.**datasource_subscribe**(*request: aiohttp.web_request.Request*)
Subscribe to the datasource with the given id

async src.datasources.views.**datasource_unsubscribe**(*request: aiohttp.web_request.Request*)
Unsubscribe to the datasource with the given id

src.datasources.views.**dumps**(*obj*, ***, *skipkeys=False*, *ensure_ascii=True*, *check_circular=True*, *allow_nan=True*, *cls=None*, *indent=None*, *separators=None*, *default=<function make_serializable>*, *sort_keys=False*, ***kw*)
A version of json.dumps that uses make serializable recursively to make objects serializable

src.datasources.views.**try_get_source**(*app*, *topic*)
Attempt to get the datasource sending to the given topic

Raises an HTTPNotFound error if not found.

2.1.4 src.fmus package

2.1.4.1 Submodules

2.1.4.2 src.fmus.views module

src.fmus.views.**dumps**(*obj*, ***, *skipkeys=False*, *ensure_ascii=True*, *check_circular=True*, *allow_nan=True*, *cls=None*, *indent=None*, *separators=None*, *default=<function make_serializable>*, *sort_keys=False*, ***kw*)
A version of json.dumps that uses make serializable recursively to make objects serializable

async src.fmus.views.**fmu_detail**(*request: aiohttp.web_request.Request*)
Get detailed information for the FMU with the given id

Append /models to get the 3d models if any

async src.fmus.views.**fmu_list**(*request: aiohttp.web_request.Request*)
List all uploaded FMUs.

Append an FMU id to get more information about a listed FMU.

async src.fmus.views.**fmu_model**(*request: aiohttp.web_request.Request*)
Get a 3d model belonging to the FMU if it exists

async src.fmus.views.**fmu_models**(*request: aiohttp.web_request.Request*)
List the 3d models belonging to the FMU if any exists

Append the models id the get a specific model

2.1.5 src.processors package

2.1.5.1 Submodules

2.1.5.2 src.processors.models module

class src.processors.models.**Processor**(*processor_id: str*, *blueprint_id: str*, *blueprint_path: str*, *init_params: dict*, *topic: str*, *source_topic: str*, *source_format: str*, *min_input_spacing: float*, *min_step_spacing: float*, *min_output_spacing: float*, *processor_root_dir: str*, *kafka_server: str*)
Bases: object

The main process endpoint for processor processes

retrieve_status()
Retrieves the status of the processor process

Can only be called after initialization. Should be run in a separate thread to prevent the connection from blocking the main thread :return: the processors status as a dict

set_inputs(*input_refs*, *measurement_refs*, *measurement_proportions*)
Sets the input values, must not be called before start

Parameters **output_refs** – the indices of the inputs that will be used

set_outputs(*output_refs*)
Sets the output values, must not be called before start

Parameters

- **input_refs** – the indices of the inputs that will be used
- **measurement_refs** – the indices of the input data values that will be used. Must be in the same order as input_ref.
- **measurement_proportions** – list of scales to be used on values before inputting them. Must be in the same order as input_ref.

start(*input_refs*, *measurement_refs*, *measurement_proportions*, *output_refs*, *start_params*)
Starts the process, must not be called before init_results

Parameters

- **input_refs** – the indices of the inputs that will be used
- **measurement_refs** – the indices of the input data values that will be used. Must be in the same order as input_ref.
- **measurement_proportions** – list of scales to be used on values before inputting them. Must be in the same order as input_ref.
- **output_refs** – the indices of the inputs that will be used
- **start_params** – the processors start parameters as a dict

Returns the processors status as a dict

async stop()
Attempts to stop the process nicely, killing it otherwise

class src.processors.models.**Variable**(*valueReference: int*, *name: str*)
Bases: object

A simple container class for variable attributes

src.processors.models.**processor_process**(*connection: multiprocessing.connection.Connection*, *blueprint_path: str*, *init_params: dict*, *processor_dir: str*, *topic: str*, *source_topic: str*, *source_format: str*, *kafka_server: str*, *min_input_spacing: float*, *min_step_spacing: float*, *min_output_spacing: float*)
Runs the given blueprint as a processor

Is meant to be run in a separate process

Parameters

- **connection** – a connection object to communicate with the main process
- **blueprint_path** – the path to the blueprint folder
- **init_params** – the initialization parameters to the processor as a dictionary
- **processor_dir** – the directory the created process will run in
- **topic** – the topic the process will send results to
- **source_topic** – the topic the process will receive data from
- **source_format** – the byte format of the data the process will receive
- **kafka_server** – the address of the kafka bootstrap server the process will use
- **min_input_spacing** – the minimum time between each input to the processor

- **min_step_spacing** – the minimum time between each step function call on the processor
- **min_output_spacing** – the minimum time between each results retrieval from the processor

Returns

2.1.5.3 src.processors.views module

src.processors.views.**dumps**(*obj*, *, *skipkeys=False*, *ensure_ascii=True*, *check_circular=True*, *allow_nan=True*, *cls=None*, *indent=None*, *separators=None*, *default=<function make_serializable>*, *sort_keys=False*, **kw*)

A version of json.dumps that uses make serializable recursively to make objects serializable

async src.processors.views.**processor_create**(*request: aiohttp.web_request.Request*)

Create a new processor from post request.

Post params:

- id:* id of new processor instance max 20 chars, first char must be alphabetic or underscore, other chars must be alphabetic, digit or underscore
- blueprint:* id of blueprint to be used max 20 chars, first char must be alphabetic or underscore, other chars must be alphabetic, digit or underscore
- init_params: the processor specific initialization variables as a json string
- topic:* topic to use as input to processor
- min_output_interval: the shortest time allowed between each output from processor in seconds

async src.processors.views.**processor_delete**(*request: aiohttp.web_request.Request*)

Delete the processor with the given id.

async src.processors.views.**processor_detail**(*request: aiohttp.web_request.Request*)

Get detailed information for the processor with the given id

Append /subscribe to subscribe to the processor Append /unsubscribe to unsubscribe to the processor Append /stop to stop the processor Append /delete to delete the processor Append /outputs to get the outputs of the processor Append /inputs to get the inputs of the processor Append /status to update and get the status of the processor

async src.processors.views.**processor_inputs_update**(*request: aiohttp.web_request.Request*)

Update the processor inputs

Post params:

- input_ref: reference values to the inputs to be used
- measurement_ref: reference values to the measurement inputs to be used for the inputs. Must be in the same order as input_ref.
- measurement_proportion: scale to be used on measurement values before inputting them. Must be in the same order as input_ref.

async src.processors.views.**processor_list**(*request: aiohttp.web_request.Request*)

List all created processors.

Returns a json object of processor id to processor status objects.

Append a processor id to get more information about a listed processor. Append /create to create a new processor instance Append /clear to delete stopped processors

async src.processors.views.**processor_outputs_update**(*request: aiohttp.web_request.Request*)

Update the processor outputs

Post params:

- output_ref: reference values to the outputs to be used

async src.processors.views.**processor_start**(*request: aiohttp.web_request.Request*)

Start a processor from post request.

Post params:

- id:* id of processor instance max 20 chars, first char must be alphabetic or underscore, other chars must be alphabetic, digit or underscore
- start_params: the processor specific start parameters as a json string
- input_ref: list of reference values to the inputs to be used
- output_ref: list of reference values to the outputs to be used
- measurement_ref: list of reference values to the measurement inputs to be used for the inputs. Must be in the same order as input_ref.
- measurement_proportion: list of scales to be used on measurement values before inputting them. Must be in the same order as input_ref.

async src.processors.views.**processor_status**(*request: aiohttp.web_request.Request*)

Updates and returns the current status of the processor

async src.processors.views.**processor_stop**(*request: aiohttp.web_request.Request*)

Stop the processor with the given id.

async src.processors.views.**processor_subscribe**(*request: aiohttp.web_request.Request*)

Subscribe to the processor with the given id

async src.processors.views.**processor_unsubscribe**(*request: aiohttp.web_request.Request*)

Unsubscribe to the processor with the given id

async src.processors.views.**processors_clear**(*request: aiohttp.web_request.Request*)

Delete data from all processors that are not running

async src.processors.views.**retrieve_processor_status**(*app*, *processor_instance*)

Retrieve the initialization results from a processor

Will put the results in app['topics'] and return them.

2.2 Submodules

2.3 src.kafka module

async src.kafka.**consume_from_kafka**(*app: aiohttp.web_app.Application*)

asdf

2.4 src.server module

2.5 src.utils module

class src.utils.**RouteTableDefDocs**
Bases: aiohttp.web_routedef.RouteTableDef

src.utils.**dumps**(*obj*, ***, *skipkeys=False*, *ensure_ascii=True*, *check_circular=True*, *allow_nan=True*, *cls=None*, *indent=None*, *separators=None*, *default=<function make_serializable>*, *sort_keys=False*, ***kw*)
A version of json.dumps that uses make serializable recursively to make objects serializable

2.6 src.views module

async src.views.**history**(*request: aiohttp.web_request.Request*)
Get historic data from the given topic

get params: - start: the start timestamp as milliseconds since 00:00:00 Thursday, 1 January 1970 - end: (optinoal) the end timestamp as milliseconds since 00:00:00 Thursday, 1 January 1970

async src.views.**index**(*request: aiohttp.web_request.Request*)
The API index

A standard HTTP request will return a sample page with a simple example of api use. A WebSocket request will initiate a websocket connection making it possible to retrieve measurement and simulation data.

Available endpoints are - /client for information about the clients websocket connections - /datasources/ for measurement data sources - /processors/ for running processors on the data - /blueprints/ for the blueprints used to create processors - /fmus/ for available FMUs (for the fmu blueprint) - /models/ for available models (for the fedem blueprint) - /topics/ for all available data sources (datasources and processors)

async src.views.**models**(*request: aiohttp.web_request.Request*)
List available models for the fedem blueprint

async src.views.**session_endpoint**(*request: aiohttp.web_request.Request*)
Only returns a session cookie

Generates and returns a session cookie.

async src.views.**subscribe**(*request: aiohttp.web_request.Request*)
Subscribe to the given topic

async src.views.**topics**(*request: aiohttp.web_request.Request*)
Lists the available data sources for plotting or processors

Append the id of a topic to get details about only that topic Append the id of a topic and /subscribe to subscribe to a topic Append the id of a topic and /unsubscribe to unsubscribe to a topic Append the id of a topic and /history to get historic data from a topic

async src.views.**topics_detail**(*request: aiohttp.web_request.Request*)
Show a single topic

Append /subscribe to subscribe to the topic Append /unsubscribe to unsubscribe to the topic Append /history to get historic data from a topic

async src.views.**unsubscribe**(*request: aiohttp.web_request.Request*)
Unsubscribe to the given topic

2.7 Module contents

CHAPTER

THREE

BLUEPRINTS PACKAGE

3.1 Submodules

3.2 blueprints.fmu module

A blueprint for running FMUs.

class files.blueprints.fmu.**P**(*fmu='testrig.fmu'*)
The interface between the application and the FMU

start(*start_time*, *time_step_input_ref='-1'*)
Starts the FMU

Parameters

- **start_time** – not used in this blueprint
- **time_step_input_ref** – optional value for custom time_step input

files.blueprints.fmu.**prepare_outputs**(*output_refs*)
Create FMUPy compatible value references and outputs buffer from output_refs

Parameters **output_refs** – list of output indices

Returns tuple with outputs buffer and value reference list

CHAPTER

FOUR

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

f

m

s

INDEX

R

S

T

U

V