Caesium Documentation

Release 0.2

Christopher D. Hunter

July 03, 2016

Contents

Contents:

CHAPTER 1

caesium package

1.1 caesium.document module

class caesium.document.**AsyncRevisionStackManager**(*settings*)
Bases: object

Find revisions for any document type and action the revision

publish(**args*, ***kwargs*)
Iterate over the scheduler collections and apply any actions found

publish_for_collection(**args*, ***kwargs*)
Run the publishing operations for a given collection

Parameters **collection_name** (*str*) –

set_all_revisions_to_in_process(**args*, ***kwargs*)
Set all revisions found to in process, so that other threads will not pick them up.

Parameters **ids** (*list*) –

class caesium.document.**AsyncSchedulableDocumentRevisionStack**(*collection_name*, *settings*, *collection_schema=None*, *master_id=None*)
Bases: object

This class manages a stack of revisions for a given document in a given collection

DELETE_ACTION = 'delete'

INSERT_ACTION = 'insert'

SCHEMA = {'required': ['toa', 'processed', 'collection', 'master_id', 'action', 'patch'], 'type': 'object', 'properties': {'mast

UPDATE_ACTION = 'update'

list(**args*, ***kwargs*)
Return all revisions for this stack

Parameters

- **toa** (*int*) – The time of action as a UTC timestamp
- **show_history** (*bool*) – Whether to show historical revisions

peek(**args*, ***kwargs*)
Return the top object on the stack for this ID

Returns The next revision

Return type dict

pop(**args*, ***kwargs*)
Pop the top revision off the stack back onto the collection at the given id. This method applies the action.

Note: This assumes you don't have two revisions scheduled closer than a single scheduling cycle.

preview(**args*, ***kwargs*)
Get an ephemeral preview of a revision with all revisions applied between it and the current state

Parameters **revision_id** (*str*) – The ID of the revision state you want to preview the master id at.

Returns A snapshot of a future state of the object

Return type dict

push(**args*, ***kwargs*)
Push a change on to the revision stack for this ObjectId. Pushing onto the stack is how you get revisions to be staged and scheduled for some future time.

Parameters

- **patch** (*dict*) – None Denotes Delete
- **toa** (*int*) – Time of action
- **meta** (*dict*) – The meta data for this action

class caesium.document.**BSONEncoder**(*skipkeys=False*, *ensure_ascii=True*, *check_circular=True*, *allow_nan=True*, *sort_keys=False*, *indent=None*, *separators=None*, *encoding='utf-8'*, *default=None*)
Bases: json.encoder.JSONEncoder

BSONEncorder is used to transform certain value types to a more desirable format

default(*obj*, ***kwargs*)
Handles the adapting of special types from mongo

class caesium.document.**BaseAsyncMotorDocument**(*collection_name*, *settings*, *schema=None*, *scheduleable=False*)
Bases: object

Concrete abstract class for a mongo collection and document interface

This class simplifies the use of the motor library, encoding/decoding special types, etc

create_index(**args*, ***kwargs*)
Create an index on a given attribute

Parameters

- **index** (*str*) – Attribute to set index on
- **index_type** (*str*) – See PyMongo index types for further information, defaults to GEO2D index.

delete(**args*, ***kwargs*)
Delete a document or create a DELETE revision

Parameters **_id** (*str*) – The ID of the document to be deleted

Returns JSON Mongo client response including the "n" key to show number of objects effected

find(**args*, ***kwargs*)
Find a document by any criteria

Parameters

- **query** (*dict*) – The query to perform
- **orderby** (*str*) – The attribute to order results by
- **order_by_direction** (*int*) – 1 or -1
- **page** (*int*) – The page to return
- **limit** (*int*) – Number of results per page

Returns A list of results

Return type list

find_one(**args*, ***kwargs*)
Find one wrapper with conversion to dictionary

Parameters **query** (*dict*) – A Mongo query

find_one_by_id(**args*, ***kwargs*)
Find a single document by id

Parameters **_id** (*str*) – BSON string repreentation of the Id

Returns a signle object

Return type dict

insert(**args*, ***kwargs*)
Create a document

Parameters

- **dct** (*dict*) –
- **toa** (*toa*) – Optional time of action, triggers this to be handled as a future insert action for a new document
- **comment** (*str*) – A comment

Rtype str

Returns string bson id

location_based_search(**args*, ***kwargs*)
Search based on location and other attribute filters

Parameters

- **lng** (*long*) – Longitude parameter
- **lat** (*long*) – Latitude parameter
- **distance** (*int*) – The radius of the query
- **unit** (*str*) – The unit of measure for the query, defaults to miles
- **attribute_map** (*dict*) – Additional attributes to apply to the location bases query
- **page** (*int*) – The page to return
- **limit** (*int*) – Number of results per page

Returns List of objects

Return type list

patch(**args*, ***kwargs*)
Update an existing document via a $set query, this will apply only these attributes.

Parameters

- **predicate_value** – The value of the predicate
- **attrs** (*dict*) – The dictionary to apply to this object
- **predicate_attribute** (*str*) – The attribute to query for to find the object to set this data ond

Returns JSON Mongo client response including the "n" key to show number of objects effected

t

update(**args*, ***kwargs*)
Update an existing document

Parameters

- **predicate_value** – The value of the predicate
- **dct** (*dict*) – The dictionary to update with
- **upsert** (*bool*) – Whether this is an upsert action
- **attribute** (*str*) – The attribute to query for to find the object to set this data ond

Returns JSON Mongo client response including the "n" key to show number of objects effected

upsert(**args*, ***kwargs*)
Update or Insert a new document

Parameters

- **_id** (*str*) – The document id
- **dct** (*dict*) – The dictionary to set on the document
- **attribute** (*str*) – The attribute to query for to find the object to set this data on

Returns JSON Mongo client response including the "n" key to show number of objects effected

exception caesium.document.**DocumentRevisionDeleteFailed**
Bases: exceptions.Exception

Occurs when the async delete process fails

exception caesium.document.**DocumentRevisionInsertFailed**
Bases: exceptions.Exception

Occurs when the revisioned document insert fails

exception caesium.document.**NoRevisionsAvailable**
Bases: exceptions.Exception

No Revisions Available

exception caesium.document.**RevisionActionNotValid**
Bases: exceptions.Exception

Invalid revision type

exception caesium.document.**RevisionNotFound**
Bases: exceptions.Exception

Revision was not found

exception caesium.document.**RevisionNotFoundException**
Bases: exceptions.Exception

exception caesium.document.**RevisionUpdateFailed**
Bases: exceptions.Exception

Occurs whena revision update cannot be applied

1.2 caesium.handler module

class caesium.handler.**BaseBulkScheduleableUpdateHandler**(*application*, *request*, ***kwargs*)
Bases: caesium.handler.BaseHandler

Bulk update objects by id and patch

delete(**args*, ***kwargs*)
Update many objects with a single toa

Parameters **bulk_id** (*str*) – The bulk id for the job you want to delete

initialize()

put(**args*, ***kwargs*)
Update many objects with a single PUT.

Example Request:

```
{
    "ids": ["52b0ede98ac752b358b1bd69", "52b0ede98ac752b358b1bd70"],
    "patch": {
        "foo": "bar"
    }
}
```

class caesium.handler.**BaseHandler**(*application*, *request*, ***kwargs*)
Bases: tornado.web.RequestHandler

A class to collect common handler methods that can be useful in your individual implementation, this includes functions for working with query strings and Motor/Mongo type documents

arg_as_array(*arg*, *split_char='|'*)
Turns an argument into an array, split by the splitChar

Parameters

- **arg** (*str*) – The name of the query param you want to turn into an array based on the value
- **split_char** (*str*) – The character the value should be split on.

Returns A list of values

Return type list

get_arg_value_as_type(*key*, *default=None*, *convert_int=False*)
Allow users to pass through truthy type values like true, yes, no and get to a typed variable in your code

Parameters **val** (*str*) – The string reprensentation of the value you want to convert

Returns adapted value

Return type dynamic

get_current_user()
Gets the current user from the secure cookie store

Returns user name for logged in user

Return type str

get_dict_of_all_args()
Generates a dictionary from a handler paths query string and returns it

Returns Dictionary of all key/values in arguments list

Return type dict

get_json_argument(*name*, *default=None*)
Find and return the argument with key 'name' from JSON request data. Similar to Tornado's get_argument() method.

Parameters

- **name** (*str*) – The name of the json key you want to get the value for
- **default** (*bool*) – The default value if nothing is found

Returns value of the argument name request

get_mongo_query_from_arguments(*reserved_attributes=[]*)
Generate a mongo query from the given URL query parameters, handles OR query via multiples

Parameters **reserved_attributes** (*list*) – A list of attributes you want to exclude from this particular query

Returns dict

group_objects_by(*list*, *attr*, *valueLabel='value'*, *childrenLabel='children'*)
Generates a group object based on the attribute value on of the given attr value that is passed in.

Parameters

- **list** (*list*) – A list of dictionary objects
- **attr** (*str*) – The attribute that the dictionaries should be sorted upon
- **valueLabel** (*str*) – What to call the key of the field we're sorting upon
- **childrenLabel** (*str*) – What to call the list of child objects on the group object

Returns list of grouped objects by a given attribute

Return type list

initialize()

json_obj_to_cursor(*json*)
(Deprecated) Converts a JSON object to a mongo db cursor

Parameters **json** (*str*) – A json string

Returns dictionary with ObjectId type

Return type dict

list_cursor_to_json(*cursor*)
Convenience method for converting a mongokit or pymongo list cursor into a JSON object for return :param Cursor cursor: A motor client database cursor

load_json()
Load JSON from the request body and store them in self.request.arguments, like Tornado does by default for POSTed form parameters.

If JSON cannot be decoded

Raises ValueError JSON Could not be decoded

obj_cursor_to_json(*cursor*)
Handle conversion of pymongo cursor into a JSON object formatted for UI consumption

Parameters **cursor** (*Cursor*) – A motor client database cursor

raise_error(*status=500*, *message='Generic server error. Out of luck...'*)
Sets an error status and returns a message to the user in JSON format

Parameters

- **status** (*int*) – The status code to use
- **message** (*str*) – The message to return in the JSON response

return_resource(*resource*, *status=200*, *statusMessage='OK'*)
Return a resource response

Parameters

- **resource** (*str*) – The JSON String representation of a resource response
- **status** (*int*) – Status code to use
- **statusMessage** (*str*) – The message to use in the error response

unauthorized(*message='Unauthorized request, please login first'*)
Standard Unauthorized response

Parameters **message** (*str*) – The Message to use in the error response

write_hyper_response(*links=[]*, *meta={}*, *entity_name=None*, *entity=None*, *notifications=[]*, *actions=[]*)
Writes a hyper media response object

Parameters

- **links** (*list*) – A list of links to the resources
- **meta** (*dict*) – The meta data for this response
- **entity_name** (*str*) – The entity name
- **entity** (*object*) – The Entity itself
- **notifications** (*list*) – List of notifications
- **actions** (*list*) – List of actions

class caesium.handler.**BaseMotorSearch**(*application*, *request*, ***kwargs*)
Bases: caesium.handler.BaseHandler

Handles searching of the stores endpoint

get(**args*, ***kwargs*)
Standard search end point for a resource of any type, override this get method as necessary in any specifc sub class. This is mostly here as a convenience for basic querying functionality on attribute

example URL:

```
foo?attr1=foo&attr2=true
```

will create a query of:

```
{
    "attr1": "foo",
    "attr2": true
}
```

initialize()
Initializer for the Search Handler

class caesium.handler.**BaseRestfulMotorHandler**(*application*, *request*, ***kwargs*)
Bases: caesium.handler.BaseHandler

Handles the restful endpoints for a mongo document resource, also has some concerns on how to handle document revision scheduling.

delete(**args*, ***kwargs*)
Delete a resource by bson id :raises: 404 Not Found :raises: 400 Bad request :raises: 500 Server Error

get(**args*, ***kwargs*)
Get an by object by unique identifier

Id string id the bson id of an object

Return type JSON

initialize()
Initialize the base handler

post(**args*, ***kwargs*)
Create a new object resource

Json Object to create

Returns json string representation

Return type JSON

put(**args*, ***kwargs*)
Update a resource by bson ObjectId

Returns json string representation

Return type JSON

class caesium.handler.**BaseRevisionList**(*application*, *request*, ***kwargs*)
Bases: caesium.handler.BaseRestfulMotorHandler

get(**args*, ***kwargs*)
Get a list of revisions by master ID

Parameters **master_id** –

Returns

initialize()
Initializer for the Search Handler

class caesium.handler.**RevisionHandler**(*application*, *request*, ***kwargs*)
Bases: caesium.handler.BaseRestfulMotorHandler

delete(**args*, ***kwargs*)
Delete a revision by ID

Parameters **id** – BSON id

Returns

get(**args*, ***kwargs*)
Get revision based on the stack preview algorithm

Parameters **id** – BSON id

Returns JSON

initialize()
Initializer for the Search Handler

post(**args*, ***kwargs*)
Create a revision manually without the stack

Parameters **id** – BSON id

Returns JSON

put(**args*, ***kwargs*)
Update a revision by ID

Parameters **id** – BSON id

Returns

CHAPTER 2

Installation

To install via pip:

```
pip install caesium
```

CHAPTER 3

Example Application

There is a quickstart project you can clone and install easily: CaesiumQuickstart

A simple application that utilizes Caesium for comment content:

```
import tornado.ioloop
import tornado.web
from tornado.options import options
import tornado.httpserver
from caesium.handler import BaseRestfulMotorHandler
from caesium.document import BaseAsyncMotorDocument, AsyncRevisionStackManager
from settings import settings
import logging

class CommentHandler(BaseRestfulMotorHandler):

    def initialize(self):
        self.object_name = "comment"
        self.client = BaseAsyncMotorDocument(self.object_name, self.settings)

url_patterns = [
    (r"/comment", CommentHandler),
    (r"/comment/([0-9a-zA-Z]+)", CommentHandler),
]

class App(tornado.web.Application):

    def __init__(self):
        """App wrapper constructor, global objects within our Tornado platform should be managed here
        self.logger = logging.getLogger(self.__class__.__name__)
        tornado.web.Application.__init__(self, url_patterns, **settings)

        #Document publisher, this allows for patches to be applied
        document_publisher = tornado.ioloop.PeriodicCallback(AsyncRevisionStackManager(settings).publ
                                                             settings['scheduler']["timeout_in_millise
                                                             io_loop=tornado.ioloop.IOLoop.current()
        )

        document_publisher.start()

application = App()

if __name__ == "__main__":
```

```
    logger = logging.getLogger()
    http_server = tornado.httpserver.HTTPServer(application, xheaders=True)
    http_server.listen(options.port)

    try:
        tornado.ioloop.IOLoop.instance().start()
    except KeyboardInterrupt:
        logger.info("\nStopping server on port %s" % options.port)
```

CHAPTER 4

Sample settings.py

Here is an example settings.py file to go along with the above app.py:

```
import tornado
import logging, logging.config
import tornado.template
from tornado.log import LogFormatter as TornadoLogFormatter
from tornado.options import define, options
import os
import motor

path = lambda root,*a: os.path.join(root, *a)

ROOT = os.path.dirname(os.path.abspath(__file__))
CONF_PATH="%s/%s" % (ROOT, "conf")
MEDIA_ROOT = path(ROOT, 'apps/media')
TEMPLATE_ROOT = path(ROOT, 'apps/templates')

define("port", default=8888, help="run on the given port", type=int)
define("config", default=None, help="tornado config file")
define("debug", default=False, help="debug mode")

if options.config:
    tornado.options.parse_config_file(options.config)

tornado.options.parse_command_line()

settings = {}

#Scheduler settings if you choose to use it
settings['scheduler']= {
    "timeout_in_milliseconds": 2000,
    "lazy_migrated_published_by_default": True,
    "collections" : ["comment"]
}

#Static mongo connection settings
settings['mongo'] = {}
settings['mongo']['host'] = "localhost"
settings['mongo']['port'] = 27017
settings['mongo']['db'] = "test"

#Mongo client
settings['db'] = motor.MotorClient("mongodb://%s:%s" % (settings['mongo']['host'], settings['mongo']
```

```
settings['debug'] = options.debug
settings['static_path'] = path(ROOT, 'static/')
settings['cookie_secret'] = "bbb2b20ab0189b93ba0ae55ac571c214185bea9e"
settings['xsrf_cookies'] = False
settings['template_loader'] = tornado.template.Loader(TEMPLATE_ROOT)
settings['session_cookie'] = 'user'
settings['annonymous_user'] = "Annonymous"

LOG_LEVEL = "INFO"

# See: https://docs.python.org/2/library/logging.html
LOGGING_CONFIG = {
    "version": 1,
    "disable_existing_loggers": False,
    "formatters": {
        "simple": {
            "format": "MISL %(asctime)s - %(processName)-10s: %(name)-15s %(levelname)-8s %(message)s
        },
        'tornado': {
                '()': TornadoLogFormatter,
                'fmt': '%(color)s[%(levelname)1.1s %(asctime)s %(name)s.%(funcName)s:%(lineno)d]%(en
                'color': True
        }
    },
    "handlers": {
        "console": {
            "class": "logging.StreamHandler",
            "level": LOG_LEVEL,
            "formatter": "tornado",
            "stream": "ext://sys.stdout"
        }
    },
    "loggers": {
        "transmit": {
            "level": LOG_LEVEL,
            "propagate": False,
            "handlers": ["console"]
        },
    },
    "root": {
        "level": LOG_LEVEL,
        "handlers": ["console"]
    }
}

logging.config.dictConfig(LOGGING_CONFIG)
```

CHAPTER 5

Indices and tables

- *genindex*
- *modindex*
- *search*

C

A

B

C

D

F

G

I