Mongo Task Queue Documentation

Release 0.0.0

Sean Ross-Ross

April 30, 2015

Contents

Contents:

CHAPTER 1

API

1.1 Connection

Created on Aug 2, 2013

@author: sean

class mtq.connection.**MTQConnection**(*db*, *collection_base='mq'*, *qsize=50*, *workersize=5*, *logsize=100*, *extra_lognames=()*)

Base object that you should use to create all other TQ objects

Init

Parameters

- **db** – mongo database
- **collection_base** – base name for collection
- **qsize** – the size of the capped collection of the queue

..sealso: MTQConnection.default, MTQConnection.from_config

classmethod default()

Create an MTQConnection default configuration using mongo from localhost

finished_jobs_collection

The collection to push jobs to

classmethod from_config(*config=None*, *client=None*)

Create an MTQConnection from a config dict,

Parameters

- **config** – configutation dict, with the parameters * DB_HOST * DB * COLLECTION_BASE * COLLECTION_SIZE
- **client** – a pymongo.MongoClient or None

get_job(*job_id*)

retrieve a job

get_worker(*worker_name=None*, *worker_id=None*)

retrieve a worker

job_stream(*job_id*)

Get a file like object for the output of a job

logging_collection
The collection to push log lines to

make_query(*queues*, *tags*, *priority=0*, *processed=False*, *failed=False*, ***query*)
return a mongodb query dict to get the next task in the queue

make_tag_query(*tags*)
Query for tags

new_worker(*queues=()*, *tags=()*, *priority=0*, *silence=False*, *log_worker_output=False*, *poll_interval=3*, *args=None*)
Create a worker object

Parameters

- **queues** – names of queues to pop from (these are OR'd)
- **tags** – jobs *must* have all these tags to be processed by this worker
- **priority** – (not implemented yet)
- **log_worker_output** – if true, log worker output to the db

pop_item(*worker_id*, *queues*, *tags*, *priority=0*, *failed=False*)
Pop an item from the queue

queue(*name='default'*, *tags=()*, *priority=0*)
Create a queue object

Parameters

- **name** – the name of the queue
- **tags** – default tags to give to jobs
- **priority** – (not implemented yet)

queue_collection
The collection to push jobs to

queues
List of existing queues

schedule_collection
The collection to push log lines to

worker_collection
Collection to register workers to

worker_stream(*worker_name=None*, *worker_id=None*)
Get a file like object for the output of a worker

workers
List of existing workers

Returns a WorkerProxy object

1.2 Queue

class mtq.queue.**Queue**(*factory*, *name='default'*, *tags=()*, *priority=0*)
A queue to enqueue an pop tasks

Do not create directly use MTQConnection.queue

all_tags
All the unique tags of jobs in this queue

count
The number of jobs in this queue (filtering by tags too)

enqueue(*func_or_str*, **args*, ***kwargs*)
Creates a job to represent the delayed function call and enqueues it.

Expects the function to call, along with the arguments and keyword arguments.

The function argument *func_or_str* may be a function or a string representing the location of a function

enqueue_call(*func_or_str*, *args=()*, *kwargs=None*, *tags=()*, *priority=None*, *timeout=None*, *mutex=None*)
Creates a job to represent the delayed function call and enqueues it.

It is much like *.enqueue()*, except that it takes the function's args and kwargs as explicit arguments. Any kwargs passed to this function contain options for MQ itself.

is_empty()
The number of jobs in this queue (filtering by tags too)

num_failed
The number of jobs in this queue (filtering by tags too)

pop(*worker_id=None*)
Pop a job off the queue

1.3 Job

Created on Aug 2, 2013

@author: sean

class mtq.job.**Job**(*factory*, *doc*)
A Job is just a convenient datastructure to pass around job (meta) data.

Do not create directly, use MTQConnection.get_job

apply()
Execute this task syncronusly

args
The arguments to call func with

finished()
test if this job has finished

func
a callable function for workers to execute

func_name
The name of the task to execute

id
the identifier for this job

kwargs
The keyword arguments to call func with

qname
The name of the queue that this job is in

set_finished(*failed=False*)
Mark this jog as finished.

Parameters **failed** – if true, this was a failed job

stream()
Get a stream to read log lines from this job

tags
List of tags for this job

1.4 Worker

class mtq.worker.**Worker**(*factory*, *queues=()*, *tags=()*, *priority=0*, *poll_interval=1*, *exception_handler=None*, *log_worker_output=False*, *silence=False*, *extra_lognames=()*)
Should create a worker from MTQConnection.new_worker

num_backlog
number of tasks this worker has to complete

process_job(*job*)
Process a single job in a multiprocessing.Process

register(**args*, ***kwds*)
Internal Contextmanager, register the birth and death of this worker

eg::

with worker.register(): # Work

start_main_loop(*one=False*, *batch=False*, *pop_failed=False*, *fail_fast=False*)
Start the main loop and process jobs

work(*one=False*, *batch=False*, *failed=False*, *fail_fast=False*)
Main work function

Parameters

- **one** – wait for the first job execute and then exit
- **batch** – work until the queue is empty, then exit

class mtq.worker.**WorkerProxy**(*factory*, *doc*)
This is a representation of an actual worker process

finished()
test if this worker is finished

last_check_in
last check in time

num_backlog
number of tasks this worker has to complete

num_processed
number of tasks this worker has completed

1.5 Utils

Created on Aug 1, 2013

@author: sean

mtq.utils.**ensure_capped_collection**(*db*, *collection_name*, *size_mb*)

mtq.utils.**handle_signals**()
Handle signals in multiprocess.Process threads

mtq.utils.**import_string**(*import_name*, *silent=False*)
Imports an object based on a string. This is useful if you want to use import paths as endpoints or something similar. An import path can be specified either in dotted notation (`xml.sax.saxutils.escape`) or with a colon as object delimiter (`xml.sax.saxutils:escape`).

If *silent* is True the return value will be *None* if the import fails.

For better debugging we recommend the new `import_module()` function to be used instead.

Parameters

- **import_name** – the dotted name for the object to import.
- **silent** – if set to *True* import errors are ignored and *None* is returned instead.

Returns imported object

mtq.utils.**setup_logging**(*worker_id*, *job_id*, *silence=False*)
set up logging for worker

CHAPTER 2

Indices and tables

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

m

R

S

T

W