QuLab

Release 0.9.4

feihoo87

Jun 09, 2020

CONTENTS

1 Installation 1

2 Usage 3

3 Running Tests 5

4 Reporting Issues 7

5 License 9

6 QuLab API 11
6.1 qulab package 11

7 Indices and tables 17

Python Module Index 19

Index 21

CHAPTER

ONE

INSTALLATION

We encourage installing QuLab via the pip tool (a python package manager):

```
$ python -m pip install QuLab
```

To install from the latest source, you need to clone the GitHub repository on your machine:

```
$ git clone https://github.com/feihoo87/QuLab.git
```

Then dependencies and QuLab can be installed in this way:

```
$ cd QuLab
$ python -m pip install -r requirements.txt
$ python -m pip install -e .
```

CHAPTER

TWO

USAGE

CHAPTER

THREE

RUNNING TESTS

To run tests:

```
$ python -m pip install -r requirements-dev.txt
$ python -m pytest
```

CHAPTER

FOUR

REPORTING ISSUES

Please report all issues on github.

CHAPTER

FIVE

LICENSE

MIT

CHAPTER

SIX

QULAB API

The best place to start is the examples folder before diving into the API.

6.1 qulab package

The best place to start is the examples folder before diving into the API.

6.1.1 qulab.dht package

This package is developed base on [kademlia](https://github.com/bmuller/kademlia)

Kademlia is a Python implementation of the Kademlia protocol which utilizes the asyncio library.

6.1.2 qulab.math package

6.1.3 qulab.storage module

class qulab.storage.memstorage.**ForgetfulStorage**(*ttl=604800*)

Bases: *qulab.storage.memstorage.IStorage*

cull()

get(*key*, *default=None*)
Get given key. If not found, return default.

iter_older_than(*seconds_old*)
Return the an iterator over (key, value) tuples for items older than the given secondsOld.

class qulab.storage.memstorage.**IStorage**

Bases: abc.ABC

Local storage for this node. IStorage implementations of get must return the same type as put in by set

abstract get(*key*, *default=None*)
Get given key. If not found, return default.

abstract iter_older_than(*seconds_old*)
Return the an iterator over (key, value) tuples for items older than the given secondsOld.

6.1.4 qulab.exceptions module

exception qulab.exceptions.**QuLabDHTMalformedMessage**
Bases: *qulab.exceptions.QuLabException*

Message does not contain what is expected.

exception qulab.exceptions.**QuLabException**
Bases: Exception

Base exception.

exception qulab.exceptions.**QuLabRPCError**
Bases: *qulab.exceptions.QuLabException*

RPC base exception.

exception qulab.exceptions.**QuLabRPCServerError**
Bases: *qulab.exceptions.QuLabRPCError*

Server side error.

classmethod make(*exce*)

exception qulab.exceptions.**QuLabRPCTimeout**
Bases: *qulab.exceptions.QuLabRPCError*

Timeout.

6.1.5 qulab.log module

class qulab.log.**BaseHandler**
Bases: logging.Handler

emit(*record*)
Emit a log message.

send_bytes(*bmsg*)

serialize(*record*)
Serialize the record in binary format, and returns it ready for transmission across the socket.

class qulab.log.**RedisHandler**(*conn*, *channel='log'*)
Bases: *qulab.log.BaseHandler*

Publish log by redis

send_bytes(*bmsg*)

class qulab.log.**ZMQHandler**(*socket: zmq.sugar.socket.Socket*)
Bases: *qulab.log.BaseHandler*

Publish log by zmq socket

send_bytes(*bmsg*)

qulab.log.**level**()
Get default log level

6.1.6 qulab.rpc module

class qulab.rpc.**RPCClientMixin**
Bases: *qulab.rpc.RPCMixin*

on_response(*source*, *data*)
Client side.

remoteCall(*addr*, *methodNane*, *args=()*, *kw=None*)

set_timeout(*timeout=10*)

class qulab.rpc.**RPCMixin**
Bases: abc.ABC

cancelPending(*addr*, *msgID*, *cancelRemote*)
Give up when request timeout and try to cancel remote task.

cancelRemoteTask(*addr*, *msgID*)
Try to cancel remote task.

cancelTask(*msgID*)
Cancel the task for msgID.

close()

createPending(*addr*, *msgID*, *timeout=1*, *cancelRemote=True*)
Create a future for request, wait response before timeout.

createTask(*msgID*, *coro*, *timeout=0*)
Create a new task for msgID.

handle(*source*, *data*)
Handle received data.

Should be called whenever received data from outside.

is_admin(*source*, *data*)

abstract property loop
Event loop.

on_cancel(*source*, *data*)

on_ping(*source*, *data*)

on_pong(*source*, *data*)

on_request(*source*, *data*)
Handle request.

Overwrite this method on server.

on_response(*source*, *data*)
Handle response.

Overwrite this method on client.

on_shutdown(*source*, *data*)

property pending

async ping(*addr*, *timeout=1*)

async pong(*addr*)

async request(*address*, *msgID*, *msg*)

async response(*address*, *msgID*, *msg*)

abstract async sendto(*data*, *address*)
Send message to address.

async shutdown(*address*)

start()

stop()

property tasks

class qulab.rpc.**RPCServerMixin**
Bases: *qulab.rpc.RPCMixin*

property executor

abstract getRequestHandler(*methodNane*, *source*, *msgID*)
Get suitable handler for request.

You should implement this method yourself.

async handle_request(*source*, *msgID*, *method*, **args*, ***kw*)
Handle a request from source.

on_request(*source*, *data*)
Received a request from source.

class qulab.rpc.**ZMQClient**(*addr*, *timeout=10*, *loop=None*)
Bases: *qulab.rpc.RPCClientMixin*

property loop
Event loop.

performMethod(*methodNane*, *args*, *kw*)

async ping(*timeout=1*)

async run()

async sendto(*data*, *addr*)
Send message to address.

class qulab.rpc.**ZMQRPCCallable**(*methodNane*, *owner*)
Bases: object

class qulab.rpc.**ZMQServer**(*loop=None*)
Bases: *qulab.rpc.RPCServerMixin*

property executor

getRequestHandler(*methodNane*, ***kw*)
Get suitable handler for request.

You should implement this method yourself.

property loop
Event loop.

property port

async run()

async sendto(*data*, *address*)
Send message to address.

set_module(*mod*)

set_socket(*sock*)

start()

stop()

6.1.7 qulab.serialize module

qulab.serialize.**encode_excepion**(*e: Exception*) → bytes

qulab.serialize.**pack**(*obj: Any*) → bytes
Serialize

qulab.serialize.**packz**(*obj: Any*) → bytes
Serialize and compress.

qulab.serialize.**register**(*cls: type, encode: Callable[[cls], bytes] = <built-in function dumps>, decode: Callable[[bytes], cls] = <built-in function loads>*) → None
Register a serializable type

Parameters

- **cls** – type
- **encode** – Callable translate an object of type *cls* into *bytes* default: pickle.dumps
- **decode** – Callable translate *bytes* to an object of type *cls* default: pickle.loads

qulab.serialize.**unpack**(*buff: bytes*) → Any
Unserialize

qulab.serialize.**unpackz**(*buff: bytes*) → Any
Decompress and unserialize.

6.1.8 qulab.utils module

qulab.utils.**IEEE_488_2_BinBlock**(*datalist*, *dtype='int16'*, *is_big_endian=True*)
IEEE 488.2

Parameters

- **datalist** –
- **dtype** –
- **endian** –

Returns binblock, header , 'header'

qulab.utils.**ShutdownBlocker**(**args*, ***kwds*)

qulab.utils.**acceptArg**(*f*, *name*, *keyword=True*)
Test if argument is acceptable by function.

Parameters

- **f** – callable function
- **name** – str argument name

qulab.utils.**getHostIP**()
ip

qulab.utils.**getHostIPv6**()
ipv6

qulab.utils.**getHostMac**()
mac

qulab.utils.**randomID**()
Generate a random msg ID.

qulab.utils.**retry**(*exception_to_check*, *tries=4*, *delay=0.5*, *backoff=2*, *logger=None*)
Retry calling the decorated function using an exponential backoff. :param exception_to_check: the exception to check.

may be a tuple of exceptions to check

Parameters

- **tries** (*int*) – number of times to try (not retry) before giving up
- **delay** (*float, int*) – initial delay between retries in seconds
- **backoff** (*int*) – backoff multiplier e.g. value of 2 will double the delay each retry
- **logger** (*logging.Logger*) – logger to use. If None, print

CHAPTER

SEVEN

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

q

qulab, 11
qulab.dht, 11
qulab.exceptions, 12
qulab.log, 12
qulab.math, 11
qulab.rpc, 13
qulab.serialize, 15
qulab.storage.memstorage, 11
qulab.utils, 15

INDEX

A

acceptArg() (*in module qulab.utils*), 15

B

BaseHandler (*class in qulab.log*), 12

C

cancelPending() (*qulab.rpc.RPCMixin method*), 13
cancelRemoteTask() (*qulab.rpc.RPCMixin method*), 13
cancelTask() (*qulab.rpc.RPCMixin method*), 13
close() (*qulab.rpc.RPCMixin method*), 13
createPending() (*qulab.rpc.RPCMixin method*), 13
createTask() (*qulab.rpc.RPCMixin method*), 13
cull() (*qulab.storage.memstorage.ForgetfulStorage method*), 11

E

emit() (*qulab.log.BaseHandler method*), 12
encode_excepion() (*in module qulab.serialize*), 15
executor() (*qulab.rpc.RPCServerMixin property*), 14
executor() (*qulab.rpc.ZMQServer property*), 14

F

ForgetfulStorage (*class in qulab.storage.memstorage*), 11

G

get() (*qulab.storage.memstorage.ForgetfulStorage method*), 11
get() (*qulab.storage.memstorage.IStorage method*), 11
getHostIP() (*in module qulab.utils*), 15
getHostIPv6() (*in module qulab.utils*), 15
getHostMac() (*in module qulab.utils*), 16
getRequestHandler() (*qulab.rpc.RPCServerMixin method*), 14
getRequestHandler() (*qulab.rpc.ZMQServer method*), 14

H

handle() (*qulab.rpc.RPCMixin method*), 13
handle_request() (*qulab.rpc.RPCServerMixin method*), 14

I

IEEE_488_2_BinBlock() (*in module qulab.utils*), 15
is_admin() (*qulab.rpc.RPCMixin method*), 13
IStorage (*class in qulab.storage.memstorage*), 11
iter_older_than() (*qulab.storage.memstorage.ForgetfulStorage method*), 11
iter_older_than() (*qulab.storage.memstorage.IStorage method*), 11

L

level() (*in module qulab.log*), 12
loop() (*qulab.rpc.RPCMixin property*), 13
loop() (*qulab.rpc.ZMQClient property*), 14
loop() (*qulab.rpc.ZMQServer property*), 14

M

make() (*qulab.exceptions.QuLabRPCServerError class method*), 12
module
    qulab, 11
    qulab.dht, 11
    qulab.exceptions, 12
    qulab.log, 12
    qulab.math, 11
    qulab.rpc, 13
    qulab.serialize, 15
    qulab.storage.memstorage, 11
    qulab.utils, 15

O

on_cancel() (*qulab.rpc.RPCMixin method*), 13
on_ping() (*qulab.rpc.RPCMixin method*), 13
on_pong() (*qulab.rpc.RPCMixin method*), 13
on_request() (*qulab.rpc.RPCMixin method*), 13
on_request() (*qulab.rpc.RPCServerMixin method*), 14

on_response() (*qulab.rpc.RPCClientMixin method*), 13
on_response() (*qulab.rpc.RPCMixin method*), 13
on_shutdown() (*qulab.rpc.RPCMixin method*), 13

P

pack() (*in module qulab.serialize*), 15
packz() (*in module qulab.serialize*), 15
pending() (*qulab.rpc.RPCMixin property*), 13
performMethod() (*qulab.rpc.ZMQClient method*), 14
ping() (*qulab.rpc.RPCMixin method*), 13
ping() (*qulab.rpc.ZMQClient method*), 14
pong() (*qulab.rpc.RPCMixin method*), 13
port() (*qulab.rpc.ZMQServer property*), 14

Q

qulab
    module, 11
qulab.dht
    module, 11
qulab.exceptions
    module, 12
qulab.log
    module, 12
qulab.math
    module, 11
qulab.rpc
    module, 13
qulab.serialize
    module, 15
qulab.storage.memstorage
    module, 11
qulab.utils
    module, 15
QuLabDHTMalformedMessage, 12
QuLabException, 12
QuLabRPCError, 12
QuLabRPCServerError, 12
QuLabRPCTimeout, 12

R

randomID() (*in module qulab.utils*), 16
RedisHandler (*class in qulab.log*), 12
register() (*in module qulab.serialize*), 15
remoteCall() (*qulab.rpc.RPCClientMixin method*), 13
request() (*qulab.rpc.RPCMixin method*), 13
response() (*qulab.rpc.RPCMixin method*), 13
retry() (*in module qulab.utils*), 16
RPCClientMixin (*class in qulab.rpc*), 13
RPCMixin (*class in qulab.rpc*), 13
RPCServerMixin (*class in qulab.rpc*), 14
run() (*qulab.rpc.ZMQClient method*), 14
run() (*qulab.rpc.ZMQServer method*), 14

S

send_bytes() (*qulab.log.BaseHandler method*), 12
send_bytes() (*qulab.log.RedisHandler method*), 12
send_bytes() (*qulab.log.ZMQHandler method*), 12
sendto() (*qulab.rpc.RPCMixin method*), 14
sendto() (*qulab.rpc.ZMQClient method*), 14
sendto() (*qulab.rpc.ZMQServer method*), 14
serialize() (*qulab.log.BaseHandler method*), 12
set_module() (*qulab.rpc.ZMQServer method*), 14
set_socket() (*qulab.rpc.ZMQServer method*), 15
set_timeout() (*qulab.rpc.RPCClientMixin method*), 13
shutdown() (*qulab.rpc.RPCMixin method*), 14
ShutdownBlocker() (*in module qulab.utils*), 15
start() (*qulab.rpc.RPCMixin method*), 14
start() (*qulab.rpc.ZMQServer method*), 15
stop() (*qulab.rpc.RPCMixin method*), 14
stop() (*qulab.rpc.ZMQServer method*), 15

T

tasks() (*qulab.rpc.RPCMixin property*), 14

U

unpack() (*in module qulab.serialize*), 15
unpackz() (*in module qulab.serialize*), 15

Z

ZMQClient (*class in qulab.rpc*), 14
ZMQHandler (*class in qulab.log*), 12
ZMQRPCCallable (*class in qulab.rpc*), 14
ZMQServer (*class in qulab.rpc*), 14