Mambaex Documentation

Release 0.0.1-alpha.3

Prashant Farkya

Jul 06, 2021

Contents

CHAPTER 1

Installation

You can install it with following way:

```
pip install mambaex
```

or:

```
pipenv install mambaex
```

CHAPTER 2

API Documentation will go here

2.1 mambaex http server library

Mambaex is an HTTP server library, written in Python, for creating a simplified webservice in ease manner.

Basic GET usage:

```
>>> from mambaex import MambaexApps
>>> r = MambaexApps.getOrCreateApp('<nameOfServer>')
```

The other methods are supported - see *mambaex.api*. Full documentation is at <https://mambaex.readthedocs.io>.

copyright

(c) 2020 by Prashant Farkya.

license MIT License, see LICENSE for more details.

class mambaex.**MambaexApps**

Bases: object

A class to maintain the multiple port listner servers

classmethod getOrCreateApp(*name=''*)

Create and Returns a server intance

Parameters **name** (*string*) – Name of the server

Returns server instance of it.

Return type *MambaexApp*

Example

```
>>> xyzServer = MambaexApps.getOrCreateApp('XYZ')
```

class mambaex.mambaexApp.**MambaexApp**(*name*)

Bases: http.server.BaseHTTPRequestHandler

A class for handling server instance

MessageClass
alias of `http.client.HTTPMessage`

address_string()
Return the client address.

date_time_string(*timestamp=None*)
Return the current date and time formatted for a message header.

default_request_version = 'HTTP/0.9'

delete(*path*, *callback*)
Add the callback function defination to the stack of the DELETE method of given path

Parameters

- **path** (*string*) – A path string/regex string to match
- **callback** (*function*) – A function to callback

Returns app itself for chaining avability

Return type *MambaexApp*

disable_nagle_algorithm = False

do_DELETE()

do_GET()

do_PATCH()

do_POST()

do_PUT()

end_headers()
Send the blank line ending the MIME headers.

error_content_type = 'text/html;charset=utf-8'

error_message_format = '<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"\n "http://ww

finish()

flush_headers()

get(*path*, *callback*)
Add the callback function defination to the stack of the GET method of given path

Parameters

- **path** (*string*) – A path string/regex string to match
- **callback** (*function*) – A function to callback

Returns app itself for chaining avability

Return type *MambaexApp*

handle()
Handle multiple requests if necessary.

handle_expect_100()
Decide what to do with an "Expect: 100-continue" header.

If the client is expecting a 100 Continue response, we must respond with either a 100 Continue or a final response before waiting for the request body. The default is to always respond with a 100 Continue. You can behave differently (for example, reject unauthorized requests) by overriding this method.

This method should either return True (possibly after sending a 100 Continue response) or send an error response and return False.

handle_one_request()
Handle a single HTTP request.

You normally don't need to override this method; see the class __doc__ string for information on how to handle specific HTTP commands such as GET and POST.

listen(*port*)
Start listening to port

Parameters **port** (*int*) – A port at which it starts listening

Returns app itself for chaining avability

Return type *MambaexApp*

log_date_time_string()
Return the current time formatted for logging.

log_error(*format*, **args*)
Log an error.

This is called when a request cannot be fulfilled. By default it passes the message on to log_message().

Arguments are the same as for log_message().

XXX This should go to the separate error log.

log_message(*format*, **args*)
Log an arbitrary message.

This is used by all other logging functions. Override it if you have specific logging wishes.

The first argument, FORMAT, is a format string for the message to be logged. If the format string contains any % escapes requiring parameters, they should be specified as subsequent arguments (it's just like printf!).

The client ip and current date/time are prefixed to every message.

log_request(*code='-'*, *size='-'*)
Log an accepted request.

This is called by send_response().

monthname = [None, 'Jan', 'Feb', 'Mar', 'Apr', 'May', 'Jun', 'Jul', 'Aug', 'Sep', 'Oct

parse_request()
Parse a request (internal).

The request should be stored in self.raw_requestline; the results are in self.command, self.path, self.request_version and self.headers.

Return True for success, False for failure; on failure, any relevant error response has already been sent back.

patch(*path*, *callback*)
Add the callback function defination to the stack of the PATCH method of given path

Parameters

- **path** (*string*) – A path string/regex string to match
- **callback** (*function*) – A function to callback

Returns app itself for chaining avability

Return type *MambaexApp*

post(*path*, *callback*)
Add the callback function defination to the stack of the POST method of given path

Parameters

- **path** (*string*) – A path string/regex string to match
- **callback** (*function*) – A function to callback

Returns app itself for chaining avability

Return type *MambaexApp*

protocol_version = 'HTTP/1.0'

put(*path*, *callback*)
Add the callback function defination to the stack of the PUT method of given path :param string path: A path string/regex string to match :param function callback: A function to callback :return: app itself for chaining avability :rtype: *MambaexApp*

rbufsize = -1

responses = {<HTTPStatus.CONTINUE: 100>: ('Continue', 'Request received, please conti

send_error(*code*, *message=None*, *explain=None*)
Send and log an error reply.

Arguments are * code: an HTTP error code

3 digits

- **message: a simple optional 1 line reason phrase.** *(HTAB / SP / VCHAR / %x80-FF) defaults to short entry matching the response code
- **explain: a detailed message defaults to the long entry** matching the response code.

This sends an error response (so it must be called before any output has been generated), logs the error, and finally sends a piece of HTML explaining the error to the user.

send_header(*keyword*, *value*)
Send a MIME header to the headers buffer.

send_response(*code*, *message=None*)
Add the response header to the headers buffer and log the response code.

Also send two standard headers with the server software version and the current date.

send_response_only(*code*, *message=None*)
Send the response header only.

server_version = 'BaseHTTP/0.6'

setup()

stop()
Use to stop listening

Returns app itself for chaining avability

Return type *MambaexApp*

sys_version = 'Python/3.7.9'

timeout = None

use(*path*, *callback*)
Add the callback function defination to the stack of the ALL method of given path

Parameters

- **path** (*string*) – A path string/regex string to match
- **callback** (*function*) – A function to callback

Returns app itself for chaining avability

Return type *MambaexApp*

version_string()
Return the server software version string.

wbufsize = 0

weekdayname = ['Mon', 'Tue', 'Wed', 'Thu', 'Fri', 'Sat', 'Sun']

CHAPTER 3

Release and Version History

CHAPTER 4

Release History

4.1 Dev

Title

- [List of changes]

4.2 0.0.1-alpha.3 (16 Feb 2020)

Improvements

- Added Example to use it.
- Added new classes to start developing first phase work

Note

- Not a usable library yet still in initial development phase to make it minimal project requirement

4.3 Next

- Automate the History tracking
- Adding Contribution guide
- Adding new classes for minimal functionality support

Python Module Index

m

Index

P

R

S

T

U

V

W