
memcache

An Long

Sep 18, 2022

CONTENTS

1	Installation	3
2	API document	5
3	License	7
	Python Module Index	9
	Index	11

Experimental memcached client library for python. This project is in WIP status, please don't use it in production environment.

Key features:

- Based on memcached's new meta commands;
- Asyncio support;
- Type hints.

INSTALLATION

```
$ pip install memcache
```


API DOCUMENT

```

class memcache.AsyncMemcache(addr: ~typing.Optional[~typing.Union[~typing.Tuple[str, int],
    ~typing.List[~typing.Tuple[str, int]]]] = None, *, pool_size:
    ~typing.Optional[int] = 23, pool_timeout: ~typing.Optional[int] = 1,
    load_func: ~typing.Callable[[~typing.Union[str, bytes], bytes, int],
    ~typing.Any] = <function load>, dump_func:
    ~typing.Callable[[~typing.Union[str, bytes], ~typing.Any],
    ~typing.Tuple[bytes, int]] = <function dump>, username:
    ~typing.Optional[str] = None, password: ~typing.Optional[str] = None)

    async delete(key: Union[bytes, str]) → None

    async execute_meta_command(command: MetaCommand) → MetaResult

    async flush_all() → None

    async get(key: Union[bytes, str]) → Optional[Any]

    async set(key: Union[bytes, str], value: Any, *, expire: Optional[int] = None) → None

class memcache.Memcache(addr: ~typing.Optional[~typing.Union[~typing.Tuple[str, int],
    ~typing.List[~typing.Tuple[str, int]]]] = None, *, pool_size: ~typing.Optional[int] =
    23, pool_timeout: ~typing.Optional[int] = 1, load_func:
    ~typing.Callable[[~typing.Union[str, bytes], bytes, int], ~typing.Any] = <function
    load>, dump_func: ~typing.Callable[[~typing.Union[str, bytes], ~typing.Any],
    ~typing.Tuple[bytes, int]] = <function dump>, username: ~typing.Optional[str] =
    None, password: ~typing.Optional[str] = None)

    delete(key: Union[bytes, str]) → None

    execute_meta_command(command: MetaCommand) → MetaResult

    flush_all() → None

    get(key: Union[bytes, str]) → Optional[Any]

    set(key: Union[bytes, str], value: Any, *, expire: Optional[int] = None) → None

exception memcache.MemcacheError

class memcache.MetaCommand(cm: bytes, key: Union[bytes, str], datalen: Union[int, NoneType] = None, flags:
    Union[List[bytes], NoneType] = None, value: Union[bytes, NoneType] = None)

    cm: bytes

```

datalen: `Optional[int]`

dump_header() \rightarrow `bytes`

flags: `List[bytes]`

key: `bytes`

value: `Optional[bytes]`

class `memcache.MetaResult`(*rc: bytes, datalen: Union[int, NoneType], flags: List[bytes], value: Union[bytes, NoneType]*)

datalen: `Optional[int]`

flags: `List[bytes]`

static load_header(*line: bytes*) \rightarrow *MetaResult*

rc: `bytes`

value: `Optional[bytes]`

CHAPTER THREE

LICENSE

Memcache is distributed by a [MIT license](#).

PYTHON MODULE INDEX

m

memcache, [5](#)

INDEX

A

`AsyncMemcache` (class in *memcache*), 5

C

`cm` (*memcache.MetaCommand* attribute), 5

D

`datalen` (*memcache.MetaCommand* attribute), 5

`datalen` (*memcache.MetaResult* attribute), 6

`delete()` (*memcache.AsyncMemcache* method), 5

`delete()` (*memcache.Memcache* method), 5

`dump_header()` (*memcache.MetaCommand* method), 6

E

`execute_meta_command()` (*memcache.AsyncMemcache* method), 5

`execute_meta_command()` (*memcache.Memcache* method), 5

F

`flags` (*memcache.MetaCommand* attribute), 6

`flags` (*memcache.MetaResult* attribute), 6

`flush_all()` (*memcache.AsyncMemcache* method), 5

`flush_all()` (*memcache.Memcache* method), 5

G

`get()` (*memcache.AsyncMemcache* method), 5

`get()` (*memcache.Memcache* method), 5

K

`key` (*memcache.MetaCommand* attribute), 6

L

`load_header()` (*memcache.MetaResult* static method), 6

M

`memcache`
module, 5

`Memcache` (class in *memcache*), 5

`MemcacheError`, 5

`MetaCommand` (class in *memcache*), 5

`MetaResult` (class in *memcache*), 6

module

memcache, 5

R

`rc` (*memcache.MetaResult* attribute), 6

S

`set()` (*memcache.AsyncMemcache* method), 5

`set()` (*memcache.Memcache* method), 5

V

`value` (*memcache.MetaCommand* attribute), 6

`value` (*memcache.MetaResult* attribute), 6