aioredis_models

Release 1.0.0

QCRI Software Group

Apr 04, 2021

CONTENTS:

1 aioredis_models **1**
1.1 aioredis_models package 1

2 Indices and tables **9**

Python Module Index **11**

Index **13**

CHAPTER

ONE

AIOREDIS_MODELS

1.1 aioredis_models package

1.1.1 Submodules

1.1.2 aioredis_models.redis_double_hash module

This module contains the following classes: - RedisDoubleHash: Represents a two-way hash map stored in Redis.

class aioredis_models.redis_double_hash.**RedisDoubleHash**(*redis: aioredis.commands.Redis*, *key: str*, *inverse_key: str*)

Bases: object

Represents a two-way hash map stored in Redis. Each field can be associated with multiple values and each value can be associated with multiple fields. The structure allows for getting all the values for a field and all the fields for a given value. The values are thus referred to as inverted fields.

async delete()
Deletes all mappings from both sides of the hash map.

async fields() → Set
Gets all the fields in the hash map.

Returns The fields in the hash map.

Return type Set

async fields_inverted() → Set
Gets all the inverted fields (values) in the hash map.

Returns The inverted fields (values) in the hash map.

Return type Set

async get(*field: str*) → Set
Gets the inverted fields associated with the given field.

Parameters **field** (*str*) – The field to get.

Returns The set of all inverted fields associated with the given field.

Return type Set

async get_inverted(*field: str*) → Set
Gets the fields associated with the given inverted field (value).

Parameters **field** (*str*) – The inverted field to get.

Returns The set of all fields associated with the given inverted field.

Return type Set

async remove (*field: str*)
Removes the given field from both sides of the hash map.

Parameters **field** (*str*) – The field to remove.

async remove_inverted (*field: str*)
Removes the given inverted field from both sides of the hash map.

Parameters **field** (*str*) – The inverted field to remove.

async set (*field: str*, *value: str*)
Associates the given value with the given field.

Parameters

- **field** (*str*) – The name of the field.
- **value** (*str*) – The value to associate.

async set_inverted (*field: str*, *value: str*)
Associates the given value with the inverted field.

Parameters

- **field** (*str*) – The name of the inverted field.
- **value** (*str*) – The value to associate.

async unset (*field: str*, *value: str*)
Dissociates the given value with the given field.

Parameters

- **field** (*str*) – The name of the field.
- **value** (*str*) – The name of the value to dissociate.

1.1.3 aioredis_models.redis_hash module

This module contains the following classes: - RedisHash: Represents a hash map stored in Redis.

class aioredis_models.redis_hash.**RedisHash** (*redis: aioredis.commands.Redis*, *key: str*)
Bases: *aioredis_models.redis_key.RedisKey*

Represents a hash map stored in Redis.

enumerate (*field_pattern: Optional[str] = None*, *batch_size: Optional[int] = None*, *encoding: str = 'utf-8'*) → AsyncIterator[Any]
Enumerates over the items of the hash using HSCAN command.

Parameters

- **field_pattern** (*str, optional*) – A string to filter fields with, if needed. Defaults to None.
- **batch_size** (*int, optional*) – The maximum number of items to get with each scan. Defaults to None.

Returns An iterator that can be used to iterate over the result.

Return type AsyncIterator[Any]

async field_exists(*field: str*) → Awaitable[bool]
Checks whether the given field exists.

Parameters **field** (*str*) – The field to check.

Returns Whether the field exists or not.

Return type Awaitable[bool]

async field_length(*field: str*) → Awaitable[int]
Gets the length of the given field.

Parameters **field** (*str*) – The field to check.

Returns The length of the given field.

Return type Awaitable[int]

async fields(*encoding='utf-8'*) → Awaitable[Set]
Gets all the fields in the hash map.

Parameters **encoding** (*str, optional*) – The encoding to use for decoding the field keys. Defaults to 'utf-8'.

Returns The set of fields in the hash map.

Return type Awaitable[Set]

async get(*field: str*, *encoding='utf-8'*) → Awaitable[Any]
Gets the value of the given field in the hash map.

Parameters

- **field** (*str*) – The field to get.
- **encoding** (*str, optional*) – The encoding to use for decoding the values. Defaults to 'utf-8'.

Returns The value of the field.

Return type Awaitable[Any]

async get_all(*encoding='utf-8'*) → Awaitable[dict]
Gets the entire hash map.

Parameters **encoding** (*str, optional*) – The encoding to use for decoding the keys and values. Defaults to 'utf-8'.

Returns The hash map.

Return type Awaitable[dict]

async length() → Awaitable[int]
Gets the length of the hash map.

Returns The length of the hash map.

Return type Awaitable[int]

async remove(*field: str*) → Awaitable[int]
Removes the given field from the hash map.

Parameters **field** (*str*) – The field to remove.

Returns The number of field removed from the hash map.

Return type Awaitable[int]

async set(*field: str*, *value: str*)
Set the value of the given field.

Parameters

- **field** (*str*) – The field whose value is to be set.
- **value** (*str*) – The value to set for the given field.

async set_all(*values: dict*)
Sets the entire hash map to the given *dict*.

Parameters **values** (*dict*) – A *dict* containing the key/value map to set.

1.1.4 aioredis_models.redis_key module

This module contains the following classes: - RedisKey: represents a generic Redis key.

class aioredis_models.redis_key.**RedisKey**(*redis: aioredis.commands.Redis*, *key: str*)
Bases: object

Represents a Redis key of any type. Acts as the class for all data structures.

async delete()
Deletes the key from Redis.

async exists() → bool
Checks if the key exists in Redis or not.

Returns A flag indicating whether the key exists or not.

Return type bool

1.1.5 aioredis_models.redis_list module

This module contains the following classes: - RedisList: Represents a list stored in Redis.

class aioredis_models.redis_list.**RedisList**(*redis: aioredis.commands.Redis*, *key: str*)
Bases: *aioredis_models.redis_key.RedisKey*

Represents a list store in Redis.

enumerate(*batch_size: int = 0*, *encoding='utf-8'*) → AsyncIterator[Any]
Enumerates the items of this list in batches.

Parameters

- **batch_size** (*int, optional*) – The number of items to get in each batch. A value of 0 or None indicates a batch size equal to the full length of the list. Defaults to 0.
- **encoding** (*str, optional*) – The encoding to use for the items. Defaults to 'utf-8'.

Returns An iterator that can be used to iterate over the result.

Return type AsyncIterator[Any]

async find_index(*value: Any*, *start: int = 0*, *stop: int = - 1*, *encoding='utf-8'*) → int
Finds the index of the given value, if any.

Parameters

- **value** (*Any*) – The value to look for.
- **start** (*int, optional*) – The index to start looking from. Defaults to 0.

- **stop** (*int, optional*) – The index to stop looking at. Negative indices are offsets from the length of the sequence. Defaults to -1.
- **encoding** (*str, optional*) – The encoding to use for decoding values. Defaults to 'utf-8'.

Returns The index of the provided value. *None* if not found.

Return type int

async get_range(*start: int = 0*, *stop: int = - 1*, *encoding='utf-8'*) → Awaitable[List]
Gets the given sub-sequence of the list.

Parameters

- **start** (*int, optional*) – The start index of the range get. Defaults to 0.
- **stop** (*int, optional*) – The stop index of the range to get. Negative indices are offsets from the length of the sequence. Defaults to -1.
- **encoding** (*str, optional*) – The encoding to use for decoding the values. Defaults to 'utf-8'.

Returns The retrieved range as a list.

Return type List

async length() → Awaitable[int]
Gets the length of the list.

Returns The length of the list.

Return type int

async move(*destination_key: str*, *block: bool = False*, *timeout_seconds: int = 0*, *encoding='utf-8'*) → Awaitable[Any]
Moves a value from the end of one list to the beginning of another.

Parameters

- **destination_key** (*str*) – The key of the list to move popped item to.
- **block** (*bool, optional*) – Whether to block until an item is available to pop. Defaults to *False*.
- **timeout_seconds** (*int, optional*) – The amount of time in seconds to wait before giving up. Defaults to 0.
- **encoding** (*str, optional*) – The encoding to use for decoding the popped value. Defaults to 'utf-8'.

Returns The value popped from the list, if any.

Return type Any

async pop(*reverse: bool = False*, *block: bool = False*, *timeout_seconds: int = 0*, *encoding='utf-8'*) → Awaitable[Any]
Pops a value from the list.

Parameters

- **reverse** (*bool, optional*) – Whether to pop the value from the end of the list. Defaults to *False*.
- **block** (*bool, optional*) – Whether to block until an item is available to pop. Defaults to *False*.

- **timeout_seconds** (*int, optional*) – The amount of time in seconds to wait before giving up. Defaults to 0, which indicates no timeout.
- **encoding** (*str, optional*) – The encoding to use for decoding the popped value. Defaults to 'utf-8'.

Returns The value popped from the list, if any.

Return type Any

async push (**value: Tuple*, *reverse: bool = False*) → Awaitable[int]
Pushes the given values into the list.

Parameters

- **value** (*Tuple*) – The values to push into the list.
- **reverse** (*bool, optional*) – Whether to push the values at the end of the list. Defaults to *False*.

Returns The length of the list after the push operation.

Return type int

async remove (*value: str*, *count: int = 0*) → Awaitable[int]
Removes occurrences of the given value from the list.

Parameters

- **value** (*str*) – The value to remove.
- **count** (*int, optional*) – The number of occurrences to remove. Defaults to 0, which removes all.

Returns The number of items that were removed.

Return type int

async requeue (*block: bool = False*, *timeout_seconds: int = 0*, *encoding='utf-8'*) → Awaitable[Any]
Removes a value from the beginning of the list and pushes it to the end of the same list.

Parameters

- **block** (*bool, optional*) – Whether to block until an item is available. Defaults to *False*.
- **timeout_seconds** (*int, optional*) – The amount of time to wait before giving up. Defaults to 0.
- **encoding** (*str, optional*) – The encoding to use for decoding the popped value. Defaults to 'utf-8'.

Returns The value popped from the list, if any.

Return type Any

1.1.6 aioredis_models.redis_set module

This module contains the following classes: - RedisSet: Represents a set stored in Redis.

class aioredis_models.redis_set.**RedisSet**(*redis: aioredis.commands.Redis*, *key: str*)

Bases: *aioredis_models.redis_key.RedisKey*

Represents a set stored in Redis.

async add(*value: str*) → int

Adds an item to the set.

Parameters **value** (*str*) – The item to add.

Returns The number of items that were added to the set.

Return type int

async get_all(*encoding='utf-8'*) → Set

Gets all the members of the set.

Parameters **encoding** (*str, optional*) – The encoding to use when decoding set members. Defaults to 'utf-8'.

Returns The members of the set.

Return type Set

async remove(*value: str*) → int

Removes an item from the set.

Parameters **value** (*str*) – The item to remove.

Returns The number of elements that were removed from the set.

Return type int

async size() → int

Gets the size of the set.

Returns The size of the set.

Return type int

1.1.7 aioredis_models.redis_string module

This module contains the following classes: - RedisString: Represents a string stored in Redis.

class aioredis_models.redis_string.**RedisString**(*redis: aioredis.commands.Redis*, *key: str*)

Bases: *aioredis_models.redis_key.RedisKey*

Represents a string stored in Redis.

async get(*encoding='utf-8'*) → str

Gets the stored value of the string.

Returns The value of the string.

Return type str

async length() → int

Gets the length of the string.

Returns The length of the string.

Return type int

async set(*value: Union[str, int, float]*, *timeout_seconds: Optional[Union[int, float]] = None*, *if_exists_equals: Optional[bool] = None*)

Sets the string to a given value.

Parameters

- **value** (*Union[str, int, float]*) – The value to set.
- **timeout_seconds** (*Union[int, float], optional*) – The amount of time in seconds after which the key should expire. Defaults to *None*.
- **if_exists_equals** (*bool, optional*) – If *True*, will only set this value if key already exists; if *False*, will only set this value if key does not already exist; if *None*, the value will always be set regardless of whether it already exists or not. Defaults to *None*.

1.1.8 Module contents

This module contains the following classes: - RedisKey - RedisList - RedisHash - RedisSet - RedisString - RedisDoubleHash

CHAPTER

TWO

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

a

aioredis_models, 8
aioredis_models.redis_double_hash, 1
aioredis_models.redis_hash, 2
aioredis_models.redis_key, 4
aioredis_models.redis_list, 4
aioredis_models.redis_set, 7
aioredis_models.redis_string, 7

INDEX

A

add() (*aioredis_models.redis_set.RedisSet method*), 7
aioredis_models
    module, 8
aioredis_models.redis_double_hash
    module, 1
aioredis_models.redis_hash
    module, 2
aioredis_models.redis_key
    module, 4
aioredis_models.redis_list
    module, 4
aioredis_models.redis_set
    module, 7
aioredis_models.redis_string
    module, 7

D

delete() (*aioredis_models.redis_double_hash.RedisDoubleHash method*), 1
delete() (*aioredis_models.redis_key.RedisKey method*), 4

E

enumerate() (*aioredis_models.redis_hash.RedisHash method*), 2
enumerate() (*aioredis_models.redis_list.RedisList method*), 4
exists() (*aioredis_models.redis_key.RedisKey method*), 4

F

field_exists() (*aioredis_models.redis_hash.RedisHash method*), 2
field_length() (*aioredis_models.redis_hash.RedisHash method*), 3
fields() (*aioredis_models.redis_double_hash.RedisDoubleHash method*), 1
fields() (*aioredis_models.redis_hash.RedisHash method*), 3
fields_inverted() (*aioredis_models.redis_double_hash.RedisDoubleHash method*), 1
find_index() (*aioredis_models.redis_list.RedisList method*), 4

G

get() (*aioredis_models.redis_double_hash.RedisDoubleHash method*), 1
get() (*aioredis_models.redis_hash.RedisHash method*), 3
get() (*aioredis_models.redis_string.RedisString method*), 7
get_all() (*aioredis_models.redis_hash.RedisHash method*), 3
get_all() (*aioredis_models.redis_set.RedisSet method*), 7
get_inverted() (*aioredis_models.redis_double_hash.RedisDoubleHash method*), 1
get_range() (*aioredis_models.redis_list.RedisList method*), 5

L

length() (*aioredis_models.redis_hash.RedisHash method*), 3
length() (*aioredis_models.redis_list.RedisList method*), 5
length() (*aioredis_models.redis_string.RedisString method*), 7

M

module
    aioredis_models, 8
    aioredis_models.redis_double_hash, 1
    aioredis_models.redis_hash, 2
    aioredis_models.redis_key, 4
    aioredis_models.redis_list, 4
    aioredis_models.redis_set, 7
    aioredis_models.redis_string, 7
move() (*aioredis_models.redis_list.RedisList method*), 5

P

pop() (*aioredis_models.redis_list.RedisList method*), 5
push() (*aioredis_models.redis_list.RedisList method*), 6

R

RedisDoubleHash (*class in aioredis_models.redis_double_hash*), 1
RedisHash (*class in aioredis_models.redis_hash*), 2
RedisKey (*class in aioredis_models.redis_key*), 4
RedisList (*class in aioredis_models.redis_list*), 4
RedisSet (*class in aioredis_models.redis_set*), 7
RedisString (*class in aioredis_models.redis_string*), 7
remove() (*aioredis_models.redis_double_hash.RedisDoubleHash method*), 2
remove() (*aioredis_models.redis_hash.RedisHash method*), 3
remove() (*aioredis_models.redis_list.RedisList method*), 6
remove() (*aioredis_models.redis_set.RedisSet method*), 7
remove_inverted() (*aioredis_models.redis_double_hash.RedisDoubleHash method*), 2
requeue() (*aioredis_models.redis_list.RedisList method*), 6

S

set() (*aioredis_models.redis_double_hash.RedisDoubleHash method*), 2
set() (*aioredis_models.redis_hash.RedisHash method*), 3
set() (*aioredis_models.redis_string.RedisString method*), 8
set_all() (*aioredis_models.redis_hash.RedisHash method*), 4
set_inverted() (*aioredis_models.redis_double_hash.RedisDoubleHash method*), 2
size() (*aioredis_models.redis_set.RedisSet method*), 7

U

unset() (*aioredis_models.redis_double_hash.RedisDoubleHash method*), 2