
MongoKat Documentation

Release 0.1

Pricing Assistant

Nov 14, 2017

Contents

1	Installation	3
2	Code sample	5
3	Migration guide from MongoKit	7
4	API Reference	9
4.1	mongokat.collection.Collection	9
4.2	mongokat.document.Document	11
5	Credits	13

MongoKat is a minimalist MongoDB ORM/ODM, inspired by the “hands off” API of [MongoKit](#). It was created at [Pricing Assistant](#), drawing from our experience managing a large Python codebase.

It differs from MongoKit in a few ways:

- **Less features:** we focus on basic Collection & Document methods.
- **Less magic:** MongoKit’s use of complex Python features like `__mro__` and `__metaclass__` made bugs and memory leaks hard to debug.
- **Cleaner design:** We enforce a separation between collection-level methods (`find`, `aggregate`, ...) and document-level methods (`save`, `reload`, ...)
- **Better performance:** The Cursor class is not wrapped anymore so the overhead of instantiating Documents instead of dicts is now close to zero.
- **Requires pymongo 3.0+**, taking advantage of its new features. To make transition to 3.0 easier (lots of pymongo’s APIs got renamed or deprecated) MongoKat still supports some 2.x-style parameters and method names.
- **Support for simple hooks:** `before_delete`, `after_delete`, `after_save`. Useful for keeping data up-to-date in ElasticSearch for instance, on a best-effort basis (some hooks may be lost under high load when using methods like `update_many`).
- **Support for protected fields** that can’t be updated directly. Useful for making sure developers to use specific methods of a Document.

CHAPTER 1

Installation

You can either clone the [code from GitHub](#) or install it via pip:

```
` pip install mongokat `
```


CHAPTER 2

Code sample

```
# First, declare a Document/Collection pair (a "model"):

from mongokat import Collection, Document

class SampleDocument(Document):

    def my_sum(self):
        return self["a"] + self["b"]

class SampleCollection(Collection):
    document_class = SampleDocument

    def find_by_a(self, a_value):
        return self.find_one({"a": a_value})

# Then use it in your code like this:

from pymongo import MongoClient
client = MongoClient()
Sample = SampleCollection(collection=client.my_db.my_col)

Sample.insert({"a": 1, "b": 2})
Sample.insert({"a": 2, "b": 3})

assert Sample.count() == 2

five = Sample.find_by_a(2)
assert five.my_sum() == 5
```

By the way, this is an actual test!

Migration guide from MongoKit

First you should get familiar with the new `CRUD methods`) from PyMongo 3.0. All of them work as expected in MongoKat.

We have generally tried to limit the changes needed for a migration to the models themselves, while the code actually using them should work without major changes.

Here is a list of things you should be aware of:

- You will have to split your Models into Document and Collection classes. For instance, `find()` belongs to a Collection, whereas `reload()` belongs to a Document.
- Initialization logic is different/cleaner, models are not magically registered everywhere, you have to explicitly instantiate them.
- Structures are not inherited.

4.1 mongokat.collection.Collection

class `mongokat.collection.Collection` (*collection=None, database=None, client=None*)
`mongokat.Collection` wraps a `pymongo.collection.Collection`

document_class

alias of `Document`

structure = None

immutable = False

protected_fields = ()

exists (*query, **args*)

Returns True if the search matches at least one document

count (**args, **kwargs*)

distinct (**args, **kwargs*)

group (**args, **kwargs*)

aggregate (**args, **kwargs*)

find (**args, **kwargs*)

find_one (**args, **kwargs*)

find_by_id (**args, **kwargs*)

find_by_ids (**args, **kwargs*)

find_by_b64id (**args, **kwargs*)

find_by_b64ids (**args, **kwargs*)

list_column (**args, **kwargs*)

Return one field as a list

iter_column (*query=None, field='_id', **kwargs*)
Return one field as an iterator. Beware that if your query returns records where the field is not set, it will raise a KeyError.

find_random (***kwargs*)
return one random document from the collection

one (**args, **kwargs*)

insert (*data, return_object=False*)
Inserts the data as a new document.

bulk_write (**args, **kwargs*)
Hook are not supported for this method!

insert_one (*document, **kwargs*)

insert_many (*documents, **kwargs*)

replace_one (*filter, replacement, **kwargs*)

update_one (*filter, update, **kwargs*)

update_many (*filter, update, **kwargs*)

delete_one (*filter, **kwargs*)

delete_many (*filter, **kwargs*)

find_one_and_delete (**args, **kwargs*)

find_one_and_replace (**args, **kwargs*)

find_one_and_update (**args, **kwargs*)

has_trigger (*event*)
Does this trigger need to run?

trigger (*event, filter=None, update=None, documents=None, ids=None, replacements=None*)
Trigger the after_save hook on documents, if present.

connection

db

save (*to_save, **kwargs*)

update (*spec, document, **kwargs*)

remove (*spec_or_id=None, **kwargs*)

find_and_modify (*query={}, update=None, **kwargs*)

get_from_id (*_id*)

fetch (*spec=None, *args, **kwargs*)
return all document which match the structure of the object *fetch()* takes the same arguments than the the *pymongo.collection.find* method. The query is launch against the db and collection of the object.

fetch_one (**args, **kwargs*)
return one document which match the structure of the object *fetch_one()* takes the same arguments than the the *pymongo.collection.find* method. If multiple documents are found, raise a *MultipleResultsFound* exception. If no document is found, return *None* The query is launch against the db and collection of the object.

4.2 mongokat.document.Document

```
class mongokat.document.Document (doc=None, mongokat_collection=None, fetched_fields=None,
                                   gen_skel=None)
```

mongokat_collection = None

gen_skel = True

b64id

Returns the document's `_id` as a base64-encoded string

ensure_fields (*fields*, *force_refetch=False*)

Makes sure we fetched the fields, and populate them if not.

refetch_fields (*missing_fields*)

Refetches a list of fields from the DB

unset_fields (*fields*)

Removes this list of fields from both the local object and the DB.

reload ()

allow to refresh the document, so after using `update()`, it could reload its value from the database.

Be careful : `reload()` will erase all unsaved values.

If no `_id` is set in the document, a `KeyError` is raised.

delete ()

delete the document from the collection from his `_id`.

save (*force=False*, *uuid=False*, ***kwargs*)

REPLACES the object in DB. This is forbidden with objects from `find()` methods unless `force=True` is given.

save_partial (*data=None*, *allow_protected_fields=False*, ***kwargs*)

Saves just the currently set fields in the database.

generate_skeleton ()

get_size ()

return the size of the underlying bson object

validate ()

We do not support validation yet.

CHAPTER 5

Credits

- [MongoKit](#), for the inspiration and part of the code
- [PyMongo](#)

A

aggregate() (mongokat.collection.Collection method), 9

B

b64id (mongokat.document.Document attribute), 11

bulk_write() (mongokat.collection.Collection method), 10

C

Collection (class in mongokat.collection), 9

connection (mongokat.collection.Collection attribute), 10

count() (mongokat.collection.Collection method), 9

D

db (mongokat.collection.Collection attribute), 10

delete() (mongokat.document.Document method), 11

delete_many() (mongokat.collection.Collection method), 10

delete_one() (mongokat.collection.Collection method), 10

distinct() (mongokat.collection.Collection method), 9

Document (class in mongokat.document), 11

document_class (mongokat.collection.Collection attribute), 9

E

ensure_fields() (mongokat.document.Document method), 11

exists() (mongokat.collection.Collection method), 9

F

fetch() (mongokat.collection.Collection method), 10

fetch_one() (mongokat.collection.Collection method), 10

find() (mongokat.collection.Collection method), 9

find_and_modify() (mongokat.collection.Collection method), 10

find_by_b64id() (mongokat.collection.Collection method), 9

find_by_b64ids() (mongokat.collection.Collection method), 9

find_by_id() (mongokat.collection.Collection method), 9

find_by_ids() (mongokat.collection.Collection method), 9

find_one() (mongokat.collection.Collection method), 9

find_one_and_delete() (mongokat.collection.Collection method), 10

find_one_and_replace() (mongokat.collection.Collection method), 10

find_one_and_update() (mongokat.collection.Collection method), 10

find_random() (mongokat.collection.Collection method), 10

G

gen_skel (mongokat.document.Document attribute), 11

generate_skeleton() (mongokat.document.Document method), 11

get_from_id() (mongokat.collection.Collection method), 10

get_size() (mongokat.document.Document method), 11

group() (mongokat.collection.Collection method), 9

H

has_trigger() (mongokat.collection.Collection method), 10

I

immutable (mongokat.collection.Collection attribute), 9

insert() (mongokat.collection.Collection method), 10

insert_many() (mongokat.collection.Collection method), 10

insert_one() (mongokat.collection.Collection method), 10

iter_column() (mongokat.collection.Collection method), 9

L

list_column() (mongokat.collection.Collection method), 9

M

`mongokat_collection` (`mongokat.document.Document` attribute), 11

O

`one()` (`mongokat.collection.Collection` method), 10

P

`protected_fields` (`mongokat.collection.Collection` attribute), 9

R

`refetch_fields()` (`mongokat.document.Document` method), 11

`reload()` (`mongokat.document.Document` method), 11

`remove()` (`mongokat.collection.Collection` method), 10

`replace_one()` (`mongokat.collection.Collection` method), 10

S

`save()` (`mongokat.collection.Collection` method), 10

`save()` (`mongokat.document.Document` method), 11

`save_partial()` (`mongokat.document.Document` method), 11

`structure` (`mongokat.collection.Collection` attribute), 9

T

`trigger()` (`mongokat.collection.Collection` method), 10

U

`unset_fields()` (`mongokat.document.Document` method), 11

`update()` (`mongokat.collection.Collection` method), 10

`update_many()` (`mongokat.collection.Collection` method), 10

`update_one()` (`mongokat.collection.Collection` method), 10

V

`validate()` (`mongokat.document.Document` method), 11