무료 전자 책

배우기

PyMongo

Free unaffiliated eBook created from
Stack Overflow contributors.

#pymongo

You can share this PDF with anyone you feel could benefit from it, downloaded the latest version from: pymongo

It is an unofficial and free PyMongo ebook created for educational purposes. All the content is extracted from Stack Overflow Documentation, which is written by many hardworking individuals at Stack Overflow. It is neither affiliated with Stack Overflow nor official PyMongo.

Use the content presented in this book at your own risk; it is not guaranteed to be correct nor accurate, please send your feedback and corrections to info@zzzprojects.com

1: PyMongo

pymongo , .

pymongo . pymongo .

Examples

pymongo .

- Pip
  - pymongo ,

    ```
    pip install pymongo
    ```

  - pymongo :

    XXX .

    ```
    pip install pymongo==XXX
    ```

  - pymongo :

    ```
    pip install --upgrade pymongo
    ```

- easy_install
  - pymongo ,

    ```
    python -m easy_install pymongo
    ```

  - pymongo :

    ```
    python -m easy_install -U pymongo
    ```

,

PyMongo MongoDB Python .

PyMongo

```
pip install pymongo
```

MongoClient . MongoClient `localhost:27017` MongoDB .

```
from pymongo import MongoClient
client = MongoClient()
```

PyMongo Database MongoDB . .

```
db = client.mydb
```

PyMongo Collection MongoDB . .

```
col = db.mycollection
```

MongoDB .

CRUD

MongoDB BSON . BSON JSON 2 .

```
$ python
>>> from pymongo import MongoClient
>>> client = MongoClient()
>>> col = client.mydb.test
```

`insert_one(document)`

```
>>> result = col.insert_one({'x':1})
>>> result.inserted_id
ObjectId('583c16b9dc32d44b6e93cd9b')
```

`insert_many(documents)`

```
>>> result = col.insert_many([{'x': 2}, {'x': 3}])
>>> result.inserted_ids
[ObjectId('583c17e7dc32d44b6e93cd9c'), ObjectId('583c17e7dc32d44b6e93cd9d')]
```

`replace_one(filter, replacement, upsert=False)` . `upsert=True` `upsert=True`

```
>>> result = col.replace_one({'x': 1}, {'y': 1})
>>> result.matched_count
1
>>> result.modified_count
1
```

`update_one(filter, update, upsert=False)`

```
>>> result = col.update_one({'x': 1}, {'x': 3})
```

`update_many(filter, update, upsert=False)` `update_many(filter, update, upsert=False)`

```
>>> result = col.update_many({'x': 1}, {'x': 3})
```

`find(filter=None, projection=None, skip=0, limit=0, no_cursor_timeout=False)` . *filter* .

```
>>> result = col.find({'x': 1})
```

`find_one(filter=None)`

```
>>> result = col.find_one()
```

```
query={'x':1}
projection={'_id':0, 'x':1} # show x but not show _id
result=col.find(query,projection)
```

`delete_one(filter)`

```
>>> result = col.delete_one({'x': 1})
>>> result.deleted_count
1
```

`delete_many(filter)`

```
>>> result = col.delete_many({'x': 1})
>>> result.deleted_count
3
```

PyMongo `find_one_and_delete()` , `find_one_and_update()` `find_one_and_replace()` .

PyMongo : https://riptutorial.com/ko/pymongo/topic/2612/pymongo-

2: BSON JSON

MongoDB JSON serialize. date, datetime, objectId, binary, code `json.dumps` `TypeError: not JSON serializable` `TypeError: not JSON serializable` . .

Examples

json_util

json_util json json BSON , `dumps` `loads` .

```
from bson.json_util import loads, dumps
record = db.movies.find_one()
json_str = dumps(record)
record2 = loads(json_str)
```

`record` :

```
{
    "_id" : ObjectId("5692a15524de1e0ce2dfcfa3"),
    "title" : "Toy Story 4",
    "released" : ISODate("2010-06-18T04:00:00Z")
}
```

`json_str` .

```
{
    "_id": {"$oid": "5692a15524de1e0ce2dfcfa3"},
    "title" : "Toy Story 4",
    "released": {"$date": 1276833600000}
}
```

JSONOptions

`JSONOptions` `dumps` . `DEFAULT_JSON_OPTIONS` `STRICT_JSON_OPTIONS` .

```
>>> bson.json_util.DEFAULT_JSON_OPTIONS
    JSONOptions(strict_number_long=False, datetime_representation=0,
     strict_uuid=False, document_class=dict, tz_aware=True,
     uuid_representation=PYTHON_LEGACY, unicode_decode_error_handler='strict',
     tzinfo=<bson.tz_util.FixedOffset object at 0x7fc168a773d0>)
```

.

1. `DEFAULT_JSON_OPTIONS` . `dumps` .

   ```
   from bson.json_util import DEFAULT_JSON_OPTIONS
   DEFAULT_JSON_OPTIONS.datetime_representation = 2
   ```

```
dumps(record)
```

2. `json_options` `dumps` `JSONOptions` .

```
# using strict
dumps(record, json_options=bson.json_util.STRICT_JSON_OPTIONS)

# using a custom set of options
from bson.json_util import JSONOptions
options = JSONOptions() # options is a copy of DEFAULT_JSON_OPTIONS
options.datetime_representation=2
dumps(record, json_options=options)
```

`JSONOptions` `JSONOptions` .

- **strict_number_long** : true Int64 MongoDB Extended JSON Strict NumberLong, `{"$numberLong": "<number>" }` . , int encode. False.
- **datetime_representation** : **datetime.datetime** . 0 => `{"$date": <dateAsMilliseconds>}` 1 => `{"$date": {"$numberLong": "<dateAsMilliseconds>"}}` 2 => `{"$date": "<ISO-8601>"}`
- **strict_uuid** : true uuid.UUID MongoDB Extended JSON Strict Binary . `{"$uuid": "<hex>" }` . False.
- **document_class** : loads () BSON . collections.MutableMapping . dict.
- **uuid_representation** : uuid.UUID BSON . PYTHON_LEGACY.
- **tz_aware** : true MongoDB Extended JSON Strict Date datetime.datetime . . True .
- **tzinfo** : datetime `datetime.tzinfo` . utc.

python-bsonjs

python-bsonjs PyMongo json_util `json_util` .

bsonjs BSON JSON JSON BSON PyMongo json_util 10-15 .

:

- bsonjs `RawBSONDocument`
- LEGACY `{"$date": <dateAsMilliseconds>}` . .

```
pip install python-bsonjs
```

bsonjs `RawBSONDocument` . , `dumps` JSON bson `loads` bson JSON :

```
import pymongo
import bsonjs
from pymongo import MongoClient
from bson.raw_bson import RawBSONDocument

# configure mongo to use the RawBSONDocument representation
db = pymongo.MongoClient(document_class=RawBSONDocument).samples
# convert json to a bson record
json_record = '{"_id": "some id", "title": "Awesome Movie"}'
raw_bson = bsonjs.loads(json_record)
```

```
bson_record = RawBSONDocument(raw_bson)
# insert the record
result = db.movies.insert_one(bson_record)
print(result.acknowledged)

# find some record
bson_record2 = db.movies.find_one()
# convert the record to json
json_record2 = bsonjs.dumps(bson_record2.raw)
print(json_record2)
```

json

mongo json `json` . datetime .

iso id 16 .

```
import pymongo
import json
import datetime
import bson.objectid

def my_handler(x):
    if isinstance(x, datetime.datetime):
        return x.isoformat()
    elif isinstance(x, bson.objectid.ObjectId):
        return str(x)
    else:
        raise TypeError(x)

db = pymongo.MongoClient().samples
record = db.movies.find_one()
# {u'_id': ObjectId('5692a15524de1e0ce2dfcfa3'), u'title': u'Toy Story 4',
#   u'released': datetime.datetime(2010, 6, 18, 4, 0),}

json_record = json.dumps(record, default=my_handler)
# '{"_id": "5692a15524de1e0ce2dfcfa3", "title": "Toy Story 4",
#    "released": "2010-06-18T04:00:00"}'
```

BSON JSON : https://riptutorial.com/ko/pymongo/topic/9348/bson-json--

3: ObjectId

ObjectId pymongo .

Examples

60

60

```
seconds = 60

gen_time = datetime.datetime.today() - datetime.timedelta(seconds=seconds)
dummy_id = ObjectId.from_datetime(gen_time)

db.CollectionName.find({"_id": {"$gte": dummy_id}})
```

datetime UTC

```
seconds = 60

gen_time = datetime.datetime.today() - datetime.timedelta(seconds=seconds)
# converts datetime to UTC
gen_time=datetime.datetime.utcfromtimestamp(gen_time.timestamp())

dummy_id = ObjectId.from_datetime(gen_time)

db.Collection.find({"_id": {"$gte": dummy_id}})
```

ObjectId : https://riptutorial.com/ko/pymongo/topic/9855/objectid-----

S. No		Contributors
1	PyMongo	Community, Himavanth, Kheshav Sewnundun, tim
2	BSON JSON	Derlin
3	ObjectId	Sawan Vaidya