链滴

python 操作 mongodb

作者：chenruhai

原文链接：https://ld246.com/article/1582990722505

来源网站：链滴

pymongo同步查找MongoDB数据库:

1.导库:

```
import pymongo from pymongo import * import pprint
```

2.创建连接实例:

```
#创建连接实例 #无密码连接方式
 client = pymongo.MongoClient('10.47.29.26', 27030)
# client = MongoClient('mongodb://10.47.29.26:27030/')

#或者这样 # 有密码连接
 client =MongoClient('mongodb://chenruhai:11080226@localhost:27017/dbname')
```

3.获取数据库与集合连接实例对象(没有的话会自动创建)

```
# 获取数据库实例
db = client.zfdb

#或者通过函数方式访问:
client.get_database("zfdb") # db = client['zfdb']

#获取集合
 test = db.test
# test = db.get_collection("test")
```

4.插入文档:

插入一条文档:

```
person = {'name': 'zone1','sex':'boy'}
```

```
person_id = test.insert_one(person).inserted_id
#插入数据之后获取_id
print(person_id)
#insert_one()返回的是InsertOneResult对象，我们可以调用其inserted_id属性获取_id。
```

插入多条文档:

```
# 批量插入
persons = [{'name': 'zone', 'sex': 'boy'}, {'name': 'zone', 'sex': 'boy1'}, {'name':          'qone', 'sex':
'girl'}, {'name': 'qone1', 'sex': 'girl'}]
result = test.insert_many(persons) print(result.inserted_ids)
```

5.删除文档

```
#删除单挑记录，将会删除匹配到的第一条记录:

result1 = test.delete_one({'name': 'zone'}) pprint.pprint(result1)

#批量删除:(删除所有匹配到的满足条件的记录)

result1 = test.delete_many({'sex': 'girl'}) print(result1)
```

6.更新文档:

```
更新单条记录:

res = test.update_one({'name': 'zone'}, {'$set': {'sex': 'boy'}})
print(res.matched_count)


更新多条记录:

test.update_many({'sex': 'boy'}, {'$set': {'sex': 'girl'}})
```

7.查询文档:

```
查找单条记录,返回匹配到的第一条数据:

find_post = test.find_one({"name":"zone1"})
print("查找一条记录的结果：\n",find_post)

查找多条记录,可以添加查找条件:

find_posts = test.find({"sex": "girl"})
 #没有条件将是所有数据,这里得到所有sex为girl的数据
print('\n',list(find_posts))
```

异步操作mongodb数据库----非常适合大量数据操作的情况

注：浏览之前请学习异步编程相关知识

导库：

```
import asyncio from motor.motor_asyncio
import AsyncIOMotorClient from motor
```

```
import * import pprint
#以下是为了防止asyncio存在滞留进程
import nest_asyncio
nest_asyncio.apply()
```

1.创建数据库连接实例对象:

```
# 普通连接
client = AsyncIOMotorClient('10.47.29.26',27030)
# client = AsyncIOMotorClient('mongodb://localhost:27017')

# 副本集连接
# client = AsyncIOMotorClient('mongodb://host1,host2/?replicaSet=my-replicaset-name')

# 密码连接
# client =AsyncIOMotorClient('mongodb://username:password@10.47.29.26:27030/dbname')
```

2.获取数据库和集合连接对象

```
# 获取数据库
db = client.zfdb
# db = client['zfdb']

# 获取 collection
collection = db.test_collection
# collection = db['test']
```

3.插入文档:

增加一条记录:

```
async def do_insert():
    document = {'name': 'zone','sex':'boy'}
    result = await db.test_collection.insert_one(document)
    #需要插入多条文档在上面两行修改代码即可
    print('result %s' % repr(result.inserted_id))

loop = asyncio.get_event_loop()
loop.run_until_complete(do_insert())
```

批量增加记录:

```
async def do_insert():
    result = await db.test_collection.insert_many(
            [~~~~
                {'name': i, 'sex': str(i + 2)}for i in range(20)
            ])
    print('inserted %d docs' % (len(result.inserted_ids),))

loop = asyncio.get_event_loop()
loop.run_until_complete(do_insert())
```

4.查询文档:

查找一条记录:

```
async def do_find_one():
    document = await db.test_collection.find_one({'name': 'zone'})
    pprint.pprint(document)

loop = asyncio.get_event_loop()
loop.run_until_complete(do_find_one())
```

查找多条记录--查找记录可以添加筛选条件:

```
async def do_find():
    cursor = db.test_collection.find({'name': {'$lt': 5}}).sort('i')
    for document in await cursor.to_list(length=100):
        pprint.pprint(document)
loop = asyncio.get_event_loop()
loop.run_until_complete(do_find())
print("\n")

# 添加筛选条件:
# 添加筛选条件---排序、跳过、限制返回结果数
async def do_find():
    cursor = db.test_collection.find({'name': {'$lt': 4}})
    # 在迭代之前修改查询
    cursor.sort('name', -1).skip(1).limit(2)
    #先按照name从大到小排序,跳过一条，限制输出前两条，原本应该输出四条
    async for document in cursor:
        pprint.pprint(document)

loop = asyncio.get_event_loop()
loop.run_until_complete(do_find())
```

5.更新文档:

更新指定文档的字段，不会影响到其他无关的文档的内容:

```
async def do_update():
    coll = db.test_collection
    result = await coll.update_one({'name': 0}, {'$set': {'sex': 'girl'}}) #更新操作
    print('更新条数：  %s ' % result.modified_count)
    new_document = await coll.find_one({'name': 0})
    print('更新结果为：  %s' % pprint.pformat(new_document))

loop = asyncio.get_event_loop()
loop.run_until_complete(do_update())
```

更新多条记录与此类似。

6.删除文档:

删除指定记录:

```
async def do_delete_many():
    coll = db.test_collection
    n = await coll.count_documents({})
```

```
    print('删除前有 %s 条数据' % n)
    result = await db.test_collection.delete_many({'name': {'$gte': 10}}) #删除条件
    print('删除后 %s ' % (await coll.count_documents({})))

loop = asyncio.get_event_loop()
loop.run_until_complete(do_delete_many())
```

7.统计文档

```
async def do_count():
    n = await db.test_collection.count_documents({})
    print('%s 条文档在集合中' % n)
    n = await db.test_collection.count_documents({'name': {'$gt': 8}})
    print('%s 条文档在集合中,当 i > 8时' % n)

loop = asyncio.get_event_loop()
loop.run_until_complete(do_count())
```

8.替换文档:

与更新文档不同，替换文档是将所有内容替换掉，或者是留下某些内容；(包括字段与值)

```
async def do_replace():
    coll = db.test_collection
    old_document = await coll.find_one({'name': 'zone'}) #寻找到目标文档
    print('替换前: %s' % pprint.pformat(old_document))
    _id = old_document['_id'] #获取原文档的_id
    result = await coll.replace_one({'_id': _id}, {'sex': 'hanson boy'}) #替换整条文档,在该文档中,
来的name字段将会消失
    print('replaced %s document' % result.modified_count)
    new_document = await coll.find_one({'_id': _id})
    print('替换后是: %s' % pprint.pformat(new_document))

 loop = asyncio.get_event_loop()
 loop.run_until_complete(do_replace())
```