**[python PyMongo 教程](https://geek-docs.com/python/python-tutorial/python-pymongo.html)**

2019-12-05 分类：[Python 教程](https://geek-docs.com/python/python-tutorial) 阅读(5616) 评论(0)

上一篇 [Python PostgreSQL 教程](https://geek-docs.com/python/python-tutorial/python-psycopg2.html) 下一篇 [Python PyMySQL 教程](https://geek-docs.com/python/python-tutorial/python-pymysql.html)

**PyMongo 教程**展示了如何使用 Python 编程 MongoDB。

MongoDB 是 NoSQL 跨平台的面向文档的数据库，它是可用的最受欢迎的数据库之一。 MongoDB 由 MongoDB Inc.开发，并作为免费和开源软件发布。

MongoDB 中的记录是一个文档，它是由字段和值对组成的数据结构。 MongoDB 文档与 JSON 对象相似。 字段的值可以包括其他文档，数组和文档数组。 MongoDB 将文档存储在集合中。 集合类似于关系数据库中的表以及行中的文档。

游标是对查询结果集的引用。 客户可以遍历游标以检索结果。 默认情况下，游标闲置十分钟后会超时。

文章目录

* [1 PyMongo](https://geek-docs.com/python/python-tutorial/python-pymongo.html#PyMongo)
* [2 安装 PyMongo](https://geek-docs.com/python/python-tutorial/python-pymongo.html#_PyMongo)
* [3 创建一个 MongoDB 数据库](https://geek-docs.com/python/python-tutorial/python-pymongo.html#_MongoDB)
* [4 PyMongo 创建集合](https://geek-docs.com/python/python-tutorial/python-pymongo.html#PyMongo-2)
* [5 PyMongo 列出集合](https://geek-docs.com/python/python-tutorial/python-pymongo.html#PyMongo-3)
* [6 PyMongo 删除集合](https://geek-docs.com/python/python-tutorial/python-pymongo.html#PyMongo-4)
* [7 PyMongo 运行命令](https://geek-docs.com/python/python-tutorial/python-pymongo.html#PyMongo-5)
* [8 PyMongo 游标](https://geek-docs.com/python/python-tutorial/python-pymongo.html#PyMongo-6)
* [9 PyMongo 读取所有数据](https://geek-docs.com/python/python-tutorial/python-pymongo.html#PyMongo-7)
* [10 PyMongo 计数文件](https://geek-docs.com/python/python-tutorial/python-pymongo.html#PyMongo-8)
* [11 PyMongo 过滤器](https://geek-docs.com/python/python-tutorial/python-pymongo.html#PyMongo-9)
* [12 PyMongo 投影](https://geek-docs.com/python/python-tutorial/python-pymongo.html#PyMongo-10)
* [13 PyMongo 排序文件](https://geek-docs.com/python/python-tutorial/python-pymongo.html#PyMongo-11)
* [14 PyMongo 聚合](https://geek-docs.com/python/python-tutorial/python-pymongo.html#PyMongo-12)
* [15 PyMongo 限制数据输出](https://geek-docs.com/python/python-tutorial/python-pymongo.html#PyMongo-13)

**PyMongo**

PyMongo 是一个 Python 模块，用于在 Python 中使用 MongoDB。

**安装 PyMongo**

以下命令用于安装 PyMongo。

$ sudo pip install pymongo

Py

我们用pip安装 PyMongo。

**创建一个 MongoDB 数据库**

mongo工具是 MongoDB 的交互式 JavaScript Shell 界面，它为系统管理员提供了一个界面，并为开发人员提供了一种直接测试数据库查询和操作的方法。

$ mongo testdb

MongoDB shell version: 2.6.10

connecting to: testdb

> show dbs

admin (empty)

local 0.078GB

test 0.078GB

testdb 0.078GB

Py

我们创建一个testdb数据库。

**PyMongo 创建集合**

在第一个示例中，我们创建一个新集合。 MongoDB 将文档存储在集合中。 集合类似于关系数据库中的表。

create\_collection.py

#!/usr/bin/python3

from pymongo import MongoClient

cars = [ {'name': 'Audi', 'price': 52642},

{'name': 'Mercedes', 'price': 57127},

{'name': 'Skoda', 'price': 9000},

{'name': 'Volvo', 'price': 29000},

{'name': 'Bentley', 'price': 350000},

{'name': 'Citroen', 'price': 21000},

{'name': 'Hummer', 'price': 41400},

{'name': 'Volkswagen', 'price': 21600} ]

client = MongoClient('mongodb://localhost:27017/')

with client:

db = client.testdb

db.cars.insert\_many(cars)

Py

该示例创建一个新的cars集合。 它包含八个文档。

cars = [ {'name': 'Audi', 'price': 52642},

{'name': 'Mercedes', 'price': 57127},

{'name': 'Skoda', 'price': 9000},

{'name': 'Volvo', 'price': 29000},

{'name': 'Bentley', 'price': 350000},

{'name': 'Citroen', 'price': 21000},

{'name': 'Hummer', 'price': 41400},

{'name': 'Volkswagen', 'price': 21600} ]

Py

该 Python 词典存储了八个要插入到 MongoDB 集合中的记录。

client = MongoClient('mongodb://localhost:27017/')

Py

MongoClient用于与 MongoDB 通信。 我们传递MongoClient主机名和端口号。

db = client.testdb

Py

我们获得了对testdb数据库的引用。

db.cars.insert\_many(cars)

Py

使用insert\_many()方法，我们将八个文档插入到cars集合中，该集合也会自动创建。

> db.cars.find()

{ "\_id" : ObjectId("5b41eb21b9c5d915989d48a8"), "price" : 52642, "name" : "Audi" }

{ "\_id" : ObjectId("5b41eb21b9c5d915989d48a9"), "price" : 57127, "name" : "Mercedes" }

{ "\_id" : ObjectId("5b41eb21b9c5d915989d48aa"), "price" : 9000, "name" : "Skoda" }

{ "\_id" : ObjectId("5b41eb21b9c5d915989d48ab"), "price" : 29000, "name" : "Volvo" }

{ "\_id" : ObjectId("5b41eb21b9c5d915989d48ac"), "price" : 350000, "name" : "Bentley" }

{ "\_id" : ObjectId("5b41eb21b9c5d915989d48ad"), "price" : 21000, "name" : "Citroen" }

{ "\_id" : ObjectId("5b41eb21b9c5d915989d48ae"), "price" : 41400, "name" : "Hummer" }

{ "\_id" : ObjectId("5b41eb21b9c5d915989d48af"), "price" : 21600, "name" : "Volkswagen" }

Py

我们使用mongo工具验证数据。

**PyMongo 列出集合**

使用collection\_names()，我们获得数据库中可用列表的列表。

list\_collections.py

#!/usr/bin/python3

from pymongo import MongoClient

client = MongoClient('mongodb://localhost:27017/')

with client:

db = client.testdb

print(db.collection\_names())

Py

该示例在testdb数据库中打印集合。

**PyMongo 删除集合**

drop()方法从数据库中删除一个集合。

drop\_collection.py

#!/usr/bin/python3

from pymongo import MongoClient

client = MongoClient('mongodb://localhost:27017/')

with client:

db = client.testdb

db.cars.drop()

Py

该示例从testdb数据库中删除cars集合。

**PyMongo 运行命令**

我们可以使用command()向 MongoDB 发出命令。 serverStatus命令返回 MongoDB 服务器的状态。

server\_status.py

#!/usr/bin/python3

from pymongo import MongoClient

from pprint import pprint

client = MongoClient('mongodb://localhost:27017/')

with client:

db = client.testdb

status = db.command("serverStatus")

pprint(status)

Py

该示例显示冗长的服务器状态。

dbstats命令返回反映单个数据库使用状态的统计信息。

db\_stats.py

#!/usr/bin/python3

from pymongo import MongoClient

from pprint import pprint

client = MongoClient('mongodb://localhost:27017/')

with client:

db = client.testdb

print(db.collection\_names())

status = db.command("dbstats")

pprint(status)

Py

该示例打印testdb的数据库统计信息。

**PyMongo 游标**

find 方法返回一个 PyMongo 游标，该游标是对查询结果集的引用。

cursor.py

#!/usr/bin/python3

from pymongo import MongoClient

client = MongoClient('mongodb://localhost:27017/')

with client:

db = client.testdb

cars = db.cars.find()

print(cars.next())

print(cars.next())

print(cars.next())

cars.rewind()

print(cars.next())

print(cars.next())

print(cars.next())

print(list(cars))

Py

在示例中，我们使用游标。

cars = db.cars.find()

Py

find()方法返回一个 PyMongo 游标。

print(cars.next())

Py

使用next()方法，我们从结果集中获取下一个文档。

cars.rewind()

Py

rewind()方法将游标倒回其未评估状态。

print(list(cars))

Py

使用list()方法，我们可以将游标转换为 Python 列表。 它将所有数据加载到内存中。

**PyMongo 读取所有数据**

在下面的示例中，我们从集合中读取所有记录。 我们使用 Python for 循环遍历返回的游标。

all\_cars.py

#!/usr/bin/python3

from pymongo import MongoClient

client = MongoClient('mongodb://localhost:27017/')

with client:

db = client.testdb

cars = db.cars.find()

for car in cars:

print('{0} {1}'.format(car['name'],

car['price']))

Py

该示例从集合中打印所有汽车名称及其价格。

cars = db.cars.find()

Py

find()方法选择集合或视图中的文档，然后将游标返回到所选文档。 游标是对查询结果集的引用。

for car in cars:

print('{0} {1}'.format(car['name'],

car['price']))

Py

使用 Python for 循环，我们遍历结果集。

$ ./all\_cars.py

Audi 52642

Mercedes 57127

Skoda 9000

Volvo 29000

Bentley 350000

Citroen 21000

Hummer 41400

Volkswagen 21600

Py

这是输出。

**PyMongo 计数文件**

使用count()方法检索文档数量。

count\_cars.py

#!/usr/bin/python3

from pymongo import MongoClient

client = MongoClient('mongodb://localhost:27017/')

with client:

db = client.testdb

n\_cars = db.cars.find().count()

print("There are {} cars".format(n\_cars))

Py

该示例使用count()计算集合中的汽车数量。

$ ./count\_cars.py

There are 8 cars

Py

集合中有八辆车。

**PyMongo 过滤器**

find()和find\_one()的第一个参数是一个过滤器。 筛选器是所有文档必须匹配的条件。

filtering.py

#!/usr/bin/python3

from pymongo import MongoClient

client = MongoClient('mongodb://localhost:27017/')

with client:

db = client.testdb

expensive\_cars = db.cars.find({'price': {'$gt': 50000}})

for ecar in expensive\_cars:

print(ecar['name'])

Py

该示例打印价格大于 50000 的汽车的名称。

expensive\_cars = db.cars.find({'price': {'$gt': 50000}})

Py

find()方法的第一个参数是所有返回的记录必须匹配的过滤器。 过滤器使用$gt运算符仅返回昂贵的汽车。

$ ./filtering.py

Audi

Mercedes

Bentley

Py

这是输出。

**PyMongo 投影**

通过投影，我们可以从返回的文档中选择特定字段。 投影在find()方法的第二个参数中传递。

projection.py

#!/usr/bin/python3

from pymongo import MongoClient

client = MongoClient('mongodb://localhost:27017/')

with client:

db = client.testdb

cars = db.cars.find({}, {'\_id': 1, 'name':1})

for car in cars:

print(car)

Py

该示例打印文档的\_id和name字段。

cars = db.cars.find({}, {'\_id': 1, 'name':1})

Py

我们可以指定包含或排除投影，但不能同时指定。

$ ./projection.py

{'name': 'Audi', '\_id': ObjectId('5b41eb21b9c5d915989d48a8')}

{'name': 'Mercedes', '\_id': ObjectId('5b41eb21b9c5d915989d48a9')}

{'name': 'Skoda', '\_id': ObjectId('5b41eb21b9c5d915989d48aa')}

{'name': 'Volvo', '\_id': ObjectId('5b41eb21b9c5d915989d48ab')}

{'name': 'Bentley', '\_id': ObjectId('5b41eb21b9c5d915989d48ac')}

{'name': 'Citroen', '\_id': ObjectId('5b41eb21b9c5d915989d48ad')}

{'name': 'Hummer', '\_id': ObjectId('5b41eb21b9c5d915989d48ae')}

{'name': 'Volkswagen', '\_id': ObjectId('5b41eb21b9c5d915989d48af')}

Py

这是输出。

**PyMongo 排序文件**

我们可以使用sort()对文档进行排序。

sorting.py

#!/usr/bin/python3

from pymongo import MongoClient, DESCENDING

client = MongoClient('mongodb://localhost:27017/')

with client:

db = client.testdb

cars = db.cars.find().sort("price", DESCENDING)

for car in cars:

print('{0} {1}'.format(car['name'],

car['price']))

Py

该示例按价格降序对记录进行排序。

$ ./sorting.py

Bentley 350000

Mercedes 57127

Audi 52642

Hummer 41400

Volvo 29000

Volkswagen 21600

Citroen 21000

Skoda 9000

Py

这是输出。

**PyMongo 聚合**

聚合计算集合中数据的聚合值。

aggregate\_sum.py

#!/usr/bin/python3

from pymongo import MongoClient

client = MongoClient('mongodb://localhost:27017/')

with client:

db = client.testdb

agr = [ {'$group': {'\_id': 1, 'all': { '$sum': '$price' } } } ]

val = list(db.cars.aggregate(agr))

print('The sum of prices is {}'.format(val[0]['all']))

Py

该示例计算所有汽车价格的总和。

agr = [ {'$group': {'\_id': 1, 'all': { '$sum': '$price' } } } ]

Py

$sum运算符计算并返回数值的总和。 $group运算符通过指定的标识符表达式对输入文档进行分组，并将累加器表达式（如果指定）应用于每个组。

val = list(db.cars.aggregate(agr))

Py

aggregate()方法将聚合操作应用于cars集合。

$ ./aggregate\_sum.py

The sum of prices is 581769

Py

所有值的总和是 581769。

我们可以使用$match运算符来选择要汇总的特定汽车。

sum\_two\_cars.py

#!/usr/bin/python3

from pymongo import MongoClient

client = MongoClient('mongodb://localhost:27017/')

with client:

db = client.testdb

agr = [{ '$match': {'$or': [ { 'name': "Audi" }, { 'name': "Volvo" }] }},

{ '$group': {'\_id': 1, 'sum2cars': { '$sum': "$price" } }}]

val = list(db.cars.aggregate(agr))

print('The sum of prices of two cars is {}'.format(val[0]['sum2cars']))

Py

该示例计算奥迪和沃尔沃汽车的价格总和。

agr = [{ '$match': {'$or': [ { 'name': "Audi" }, { 'name': "Volvo" }] }},

{ '$group': {'\_id': 1, 'sum2cars': { '$sum': "$price" } }}]

Py

该表达式使用$match，$or，$group和$sum运算符执行任务。

$ ./sum\_two\_cars.py

The sum of prices of two cars is 81642

Py

两辆车的总价是 81642。

**PyMongo 限制数据输出**

limit查询选项指定要返回的文档数量，skip()选项指定某些文档。

MongoSkipLimit.java

#!/usr/bin/python3

from pymongo import MongoClient

client = MongoClient('mongodb://localhost:27017/')

with client:

db = client.testdb

cars = db.cars.find().skip(2).limit(3)

for car in cars:

print('{0}: {1}'.format(car['name'], car['price']))

Py

该示例从cars集合中读取，跳过了前两个文档，并将输出限制为三个文档。

cars = db.cars.find().skip(2).limit(3)

Py

skip()方法跳过前两个文档，limit()方法将输出限制为三个文档。

$ ./limit\_documents.py

Skoda: 9000

Volvo: 29000

Bentley: 350000

Py

这是示例的输出。

在 PyMongo 教程中，我们使用了 MongoDB 和 Python。

您可能还会对以下教程感兴趣： [PyMySQL 教程](https://geek-docs.com/python/python-tutorial/python-pymysql.html)， [pyDAL 教程](https://geek-docs.com/python/python-tutorial/python-pydal.html)， [Peewee 教程](https://geek-docs.com/python/python-tutorial/python-peewee.html)， [SQLite Python 教程](https://geek-docs.com/python/python-tutorial/python-sqlite.html)， [Openpyxl 教程](https://geek-docs.com/python/python-tutorial/python-openpyxl.html)，[Bottle 教程](https://geek-docs.com/python/python-tutorial/python-bottle.html)， [Python CSV 教程](https://geek-docs.com/python/python-tutorial/python-csv.html)或 [Python 教程](https://geek-docs.com/python/python-tutorial/python-tutorial.html)。