파이썬에서 mongoDB 연동하기 - Pymongo

파이썬 3.8.2 기준으로 작성된 문서입니다.

파이썬으로 mongoDB를 제어하는 방법을 정리했습니다.

DB 연결 방법, insert, delete, update, query(=find)문을 사용하는 방법을 정리했습니다.

1. Pymongo 패키지 설치

파이썬에서 mongoDB를 제어하기 위해 먼저 pymongo 패키지를 설치합니다.

```
python -m pip install pymongo
```

2. DB 연결

```
from pymongo import MongoClient
from pymongo.cursor import CursorType

host = "localhost"
port = "27017"
mongo = MongoClient(host, int(port))
print(mongo)
```

3. Insert문

insertOne(), insertMany()를 실행하는 코드를 아래와 같이 insert_item_one, insert_item_many로 구현했습니다.

(mongoClient, data, db, collection)을 parameter로 넘겨주면 됩니다.

```
def insert_item_one(mongo, data, db_name=None, collection_name=None):
    result = mongo[db_name][collection_name].insert_one(data).inserted_id
    return result

def insert_item_many(mongo, datas, db_name=None, collection_name=None):
    result = mongo[db_name][collection_name].insert_many(datas).inserted_ids
    return result

host = "localhost"
port = "27017"
mongo = MongoClient(host, int(port))
#print(mongo)

insert_item_one(mongo, {"text": "Hello Python"}, "test", "test")
```

4. Find문

Find는 SQL에서 SELECT와 동일한 기능입니다.

findOne(), find()를 실행하는 코드를 아래와 같이 find_item_one, find_item으로 구현했습니다.

(mongoClient, condition, db, collection)을 parameter로 넘겨주면 됩니다.

```
def find_item_one(mongo, condition=None, db_name=None, collection_name=None):
    result = mongo[db_name][collection_name].find_one(condition, {"_id": False})
    return result

def find_item(mongo, condition=None, db_name=None, collection_name=None):
    result = mongo[db_name][collection_name].find(condition, {"_id": False},
no_cursor_timeout=True, cursor_type=CursorType.EXHAUST)
    return result

host = "localhost"
port = "27017"
mongo = MongoClient(host, int(port))

cursor = find_item(mongo, None, "test", "test")
for list in cursor:
    print(list["text"])
```

5. Delete문

deleteOne(), deleteMany()를 실행하는 코드를 아래와 같이 delete_item_one, delete_item_many로 구현했습니다.

```
def delete_item_one(mongo, condition=None, db_name=None, collection_name=None):
    result = mongo[db_name][collection_name].delete_one(condition)
    return result

def delete_item_many(mongo, condition=None, db_name=None, collection_name=None):
    result = mongo[db_name][collection_name].delete_many(condition)
    return result

host = "localhost"
port = "27017"
mongo = MongoClient(host, int(port))

delete_item_one(mongo, {"text": "Hello Python"}, "test", "test")
```

db의 collection에 있는 모든 데이터를 삭제하고 싶다면 아래와 같이 짜면 됩니다.

```
delete_item_many(mongo, {}, "test", "test")
```

6. Update문

updateOne(), updateMany()를 실행하는 코드를 아래와 같이 update_item_one, update_item_many 로 구현했습니다.

```
def update_item_one(mongo, condition=None, update_value=None, db_name=None,
collection_name=None):
    result = mongo[db_name][collection_name].update_one(filter=condition,
update=update_value)
    return result

def update_item_many(mongo, condition=None, update_value=None, db_name=None,
collection_name=None):
    result = mongo[db_name][collection_name].update_many(filter=condition,
update=update_value)
    return result

host = "localhost"
port = "27017"
mongo = MongoClient(host, int(port))

update_item_one(mongo, {"text": "Hello Python"}, { "$set": {"text": "Hello
Kotlin"}}, "test", "test")
```

```
> db.test.find()
{ "_id" : ObjectId("5e9c5b6642600d4ceedfc882"), "text" : "Hello World" }
{ "_id" : ObjectId("5e9c5b6642600d4ceedfc883"), "text" : "Hello Python" }
{ "_id" : ObjectId("5e9c5b6642600d4ceedfc884"), "text" : "Hello JAVA" }
{ "_id" : ObjectId("5e9c5b6642600d4ceedfc885"), "text" : "Python World" }
> db.test.find()
{ "_id" : ObjectId("5e9c5b6642600d4ceedfc882"), "text" : "Hello World" }
{ "_id" : ObjectId("5e9c5b6642600d4ceedfc883"), "text" : "Hello Kotlin" }
{ "_id" : ObjectId("5e9c5b6642600d4ceedfc884"), "text" : "Hello JAVA" }
{ "_id" : ObjectId("5e9c5b6642600d4ceedfc885"), "text" : "Python World" }
> ^C
```

결과

7. Text Search

```
def text_search(mongo, text=None, db_name=None, collection_name=None):
    result = mongo[db_name][collection_name].find({"$text": {"$search": text}})
    return result

host = "localhost"
port = "27017"
mongo = MongoClient(host, int(port))

cursor = text_search(mongo, "Hello", "test", "test")
for list in cursor:
    print(list)
```

+추가

이런 db handler 코드는 한번 클래스로 만들어 놓으면 사용하기도 편하고 재사용하기 좋습니다.

제 DBHandler 클래스의 전체 코드를 공유하겠습니다.

```
class DBHandler:
    def __init__(self):
        host = "localhost"
        port = "27017"
        self.client = MongoClient(host, int(port))

    def insert_item_one(self, data, db_name=None, collection_name=None):
        result = self.client[db_name]
[collection_name].insert_one(data).inserted_id
        return result

    def insert_item_many(self, datas, db_name=None, collection_name=None):
        result = self.client[db_name]
[collection_name].insert_many(datas).inserted_ids
        return result

    def find_item_one(self, condition=None, db_name=None, collection_name=None):
        result = self.client[db_name][collection_name].find_one(condition,
{"_id": False})
        return result

    def find_item(self, condition=None, db_name=None, collection_name=None):
        result = self.client[db_name][collection_name].find(condition, {"_id":
False}, no_cursor_timeout=True, cursor_type=CursorType.EXHAUST)
        return result

    def delete_item_one(self, condition=None, db_name=None,
collection_name=None):
        result = self.client[db_name][collection_name].delete_one(condition)
        return result

    def delete_item_many(self, condition=None, db_name=None,
collection_name=None):
        result = self.client[db_name][collection_name].delete_many(condition)
        return result

    def update_item_one(self, condition=None, update_value=None, db_name=None,
collection_name=None):
        result = self.client[db_name]
[collection_name].update_one(filter=condition, update=update_value)
        return result

    def update_item_many(self, condition=None, update_value=None, db_name=None,
collection_name=None):
        result = self.client[db_name]
[collection_name].update_many(filter=condition, update=update_value)
        return result

    def text_search(self, text=None, db_name=None, collection_name=None):
```

```
        result = self.client[db_name][collection_name].find({"$text": {"$search": text}})
        return result
```

아래와 같이 사용할 수 있습니다.

```
mongo = DBHandler()
mongo.find_item_one(None, "test", "test")
```

Python에서 mongoDB를 제어하는 방법을 간단하게 정리해봤습니다.

mongoDB를 사용해 보신 분들이 볼거라고 생각해서 설명 없이 코드만 나열했는데, 포스팅을 다 보고나니 mongoDB를 접한지 얼마 안된 분에게는 불친절한 글이네요. 시간을 내서 mongoDB 사용방법도 정리해보겠습니다.