Mongo DB & Python

pymongo driver

- Mongo DB 사용
  - ✓ pymongo 패키지 설치

    pip install pymongo
  - ✓ 서버 연결

    con = pymongo.MongoClient("서버 IP", 포트번호) # 포트번호는 27017 이면 생략 가능
  - ✓ 데이터베이스 연결

    db = con.데이터베이스이름

    #없으면 생성됨
  - ✓ 컬렉션 연결

    collection = db.collection 이름

    #없으면 생성됨

pymongo driver

- 데이터 삽입
  - ✓ insert_one, insert_many 함수 이용
    - Insert_one 함수의 경우 데이터를 삽입하고 리턴된 결과의 inserted_id 속성을 호출하면 삽입된 ObjectId 확인 가능
  - ✓ 데이터 개수 확인: count_documents({}) 함수 이용

pymongo driver

❖ 데이터 삽입

```
from pymongo import MongoClient
#데이터베이스 연결
conn = MongoClient('127.0.0.1')

#데이터베이스 설정
db = conn.adam

#컬렉션 설정
collect = db.users

#document 생성 : {'key':'value'}
doc1 = {'empno':'10001', 'name':'kim', 'phone':'010-111-111', 'age':35}
doc2 = {'empno':'10002', 'name':'lee', 'age':45}
doc3 = {'empno':'10003', 'name':'park', 'phone':'010-222-222', 'age':25}

doc4 = {'empno':'10004', 'name':'choi', 'age':42}
doc5 = {'empno':'10005', 'name':'yun', 'phone':'010-222-222', 'age':37}
```

pymongo driver

- 데이터 삽입

```
collect.insert_one(doc1)
collect.insert_one(doc2)
collect.insert_one(doc3)

collect.insert_many([doc4, doc5])

print(collect.count_documents({}))

collect.insert_many(
    [
        {'num': i} for i in range(10)
    ]
)
```

pymongo driver

- 데이터 검색
  - ✓ find_one 은 하나의 데이터만 리턴
  - ✓ find()는 모든 데이터를 리턴

pymongo driver

- 데이터 검색
  - ✓ 컬렉션의 전체 데이터 조회

```
from pymongo import MongoClient
#데이터베이스 연결
conn = MongoClient('127.0.0.1')

#데이터베이스 설정
db = conn.adam

#컬렉션 설정
collect = db.users

# 전체 문서 조회
result = collect.find()

#print(result)
# 조회 문서 출력
for r in result :
    print(r)
```

pymongo driver

- 데이터 검색
  - ✓ 컬렉션의 전체 데이터 조회

```
# 조건 검색
print('조건 검색')
result2 = collect.find({ 'age':{'$gte':30} }) #크거나 같다
for r in result2 :
    print(r)
print()

result2 = collect.find({ 'age':{'$gte':30} }).sort("age")
for r in result2 :
    print(r)
```

pymongo driver

- 데이터 수정
  - ✓ update_one() : 가장 먼저 검색되는 한 Document만 수정
  - ✓ update_many() : 조건에 맞는 모든 Document 수정

pymongo driver

❖ 데이터 수정

```
from pymongo import MongoClient
#데이터베이스 연결
conn = MongoClient('127.0.0.1')

#데이터베이스 설정
db = conn.adam

#컬렉션 설정
collect = db.users

collect.update_one(
    { "empno" : "10001" },
    { "$set" :
        { "name" : "kkk" }
    }
)
```

pymongo driver

- 데이터 수정

```
collect.update_many(
    { 'age':{'$gte':30} },
    { "$set" :
        { "name" : "park" }
    }
)

result = collect.find()
# 조회 문서 출력
for r in result :
    print(r)
```

pymongo driver

- 데이터 삭제
  - ✓ delete_one() 과 delete_many()
  - ✓ delete_one() : 가장 먼저 검색되는 한 Document만 삭제
  - ✓ delete_many() : 조건에 맞는 모든 Document 삭제

pymongo driver

❖ 데이터 삭제

```
from pymongo import MongoClient
#데이터베이스 연결
conn = MongoClient('127.0.0.1')

#데이터베이스 설정
db = conn.adam

#컬렉션 설정
collect = db.users

collect.delete_many( {"age": { "$gte": 30 } } )

# 전체 문서 조회
result = collect.find()

# 조회 문서 출력
for r in result :
    print(r)
```

ODM

❖ 개요

- ✓ 객체 ↔ 문서(Document) 매핑 도구
- ✓ 관계형 DB의 ORM(Object Relational Mapper) 과 비슷하지만 NoSQL 문서형 데이터베이스(주로 MongoDB)가 대상
- ✓ MongoDB 같은 문서형 DB는 JSON 형태의 데이터를 다루는데 ODM을 쓰면 이 문서를 Python 클래스(객체) 처럼 다룰 수 있음
- ✓ SQL 없이도 딕셔너리 대신 객체를 이용해서 작업 가능
- ✓ 종류로는 MongoEngine, Beanie(FastAPI와 궁합 좋음), ODMantic 등이 있음
- ✓ ODM vs ORM 차이

구분	ODM	ORM
대상 DB	MongoDB 등 문서형 DB	MySQL, PostgreSQL
데이터 구조	JSON / Document	Table / Row
Python 도구	MongoEngine, Beanie	SQLAlchemy, Django ORM

ODM

- MongoEngine(mongoengine 패키지 사용)

```
from mongoengine import connect

connect(
    db="test_db",
    host="localhost",
    port=27017
)
```

ODM

❖ MongoEngine(mongoengine 패키지 사용)

```
from mongoengine import Document, StringField, IntField, EmailField, DateTimeField
from datetime import datetime

class User(Document):
    name = StringField(required=True, max_length=50)
    age = IntField(min_value=0)
    email = EmailField(unique=True)
    created_at = DateTimeField(default=datetime.utcnow)

    meta = {
        "collection": "users"
    }
```

ODM

- MongoEngine(mongoengine 패키지 사용)

```
user = User(
    name="박문석",
    age=55,
    email="ggangpae1@gmail.com"
)
user.save()
```

ODM

- MongoEngine(mongoengine 패키지 사용)

```
# 전체 조회
users = User.objects()
# 조건 조회
user = User.objects(name="박문석").first()
# 필터 + 정렬
users = User.objects(age__gte=20).order_by("-age")
```

ODM

- MongoEngine(mongoengine 패키지 사용)

```
# 객체 수정
user = User.objects(name="박문석").first()
user.age = 56
user.save()

# QuerySet 업데이트
User.objects(name="박문석").update_one(set__age=56)
```

ODM

- MongoEngine(mongoengine 패키지 사용)

```
# 하나 삭제
user = User.objects(name="박문석").first()
user.delete()

# 여러 개 삭제
User.objects(age__lt=18).delete()
```

ODM

- Beanie
  - ✓ Beanie는 Python의 비동기 드라이버인 Motor와 데이터 검증 라이브러리인 Pydantic을 기반으로 한 비동기 ODM(Object-Document Mapper)
  - ✓ 가장 현대적이고 인기 있는 방식인 FastAPI 스타일의 모델 정의와 비동기 처리를 지원하는 것이 특징
  - ✓ 패키지 설치: pip install beanie motor

ODM

- Beanie
  - ✓ 샘플 코드

```
import asyncio
from typing import Optional
from motor.motor_asyncio import AsyncIOMotorClient
from beanie import Document, init_beanie
from pydantic import Field

# 1. 문서 모델 정의 (Pydantic 기반)
class Product(Document):
    name: str
    price: float
    category: str
    description: Optional[str] = None

    class Settings:
        name = "products"  # MongoDB 컬렉션 이름 지정
```

ODM

- Beanie
  - ✓ 샘플 코드

```
async def run_example():
    # 2. MongoDB 클라이언트 생성 및 데이터베이스 연결
    client = AsyncIOMotorClient("mongodb://localhost:27017")
    db = client.test_database

    # 3. Beanie 초기화 (모델 등록)
    await init_beanie(database=db, document_models=[Product])

    # --- CRUD 작업 시작 ---

    # [C] 데이터 생성 (Create)
    new_product = Product(name="Gaming Mouse", price=45000,
category="Electronics")
    await new_product.insert()
    print(f"생성된 상품 ID: {new_product.id}")
```

ODM

- Beanie
  - ✓ 샘플 코드

```
# [R] 데이터 조회 (Read)
# 특정 이름으로 하나 찾기
product = await Product.find_one(Product.name == "Gaming Mouse")
if product:
    print(f"조회된 가격: {product.price}")

# [U] 데이터 수정 (Update)
if product:
    await product.set({Product.price: 39000})
    print("가격이 업데이트되었습니다.")

# [D] 데이터 삭제 (Delete)
await product.delete()
print("상품이 삭제되었습니다.")
```

ODM

- Beanie
  - ✓ 샘플 코드

```
# 비동기 루프 실행
if __name__ == "__main__":
    asyncio.run(run_example())
```