KB금융그룹 | 국민의 평생 금융파트너

몽고디비와 자바 연결

2025년 상반기 K-디지털 트레이닝

데이터 매핑을 어떻게 할거냐

pojo==vo

MongoDB Java 연동

[KB] IT's Your Life

몽고 디비를 연결하는 2가지 방법

몽고디비는 기본적으로 json이다.

자바로 갖고 와서 해당 데이터를 어떻게 관리할까.

1) 맵으로 관리. 키와 밸류 쌍으로.
도큐먼트 하나당 맵 하나 매핑
조금 불편함. 4장.

4장은 1번

5장은 2번

1보다는2가 편하다

2) 전용 vo객체로 매핑하는 것.
jdbc에서는 테이블을 그렇게 표현했지.
처리하기 쉽다는 장점 5장.

KB 국민은행

MongoDB Java 연동

프로젝트 생성

- Name: java_ex
- Location: C:₩KB_Fullstack₩06_MongoDB
- Build System: Gradle
- Gradle DSL: Groovy

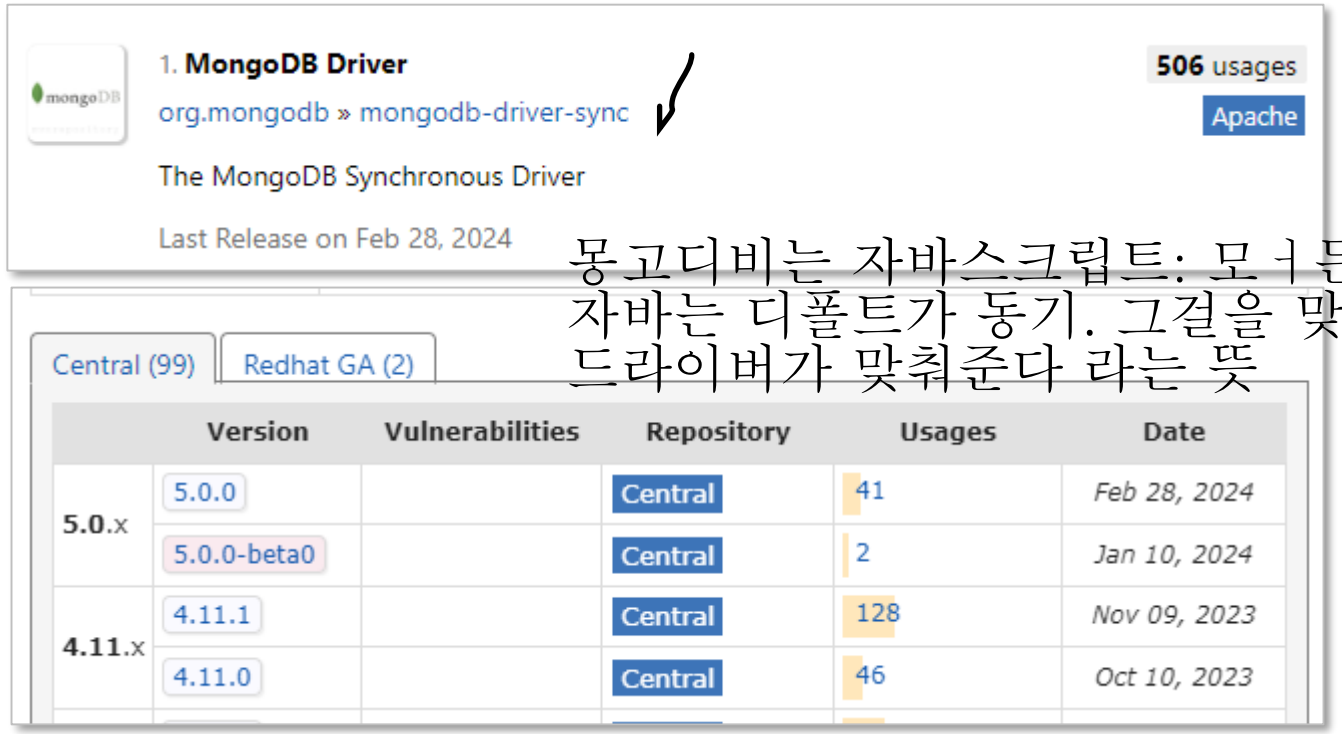
- Advanced Settings
  - GroupId: org.scoula
  - ArtifactId : java_ex

MongoDB Java 연동

MongoDB Java 드라이버

- mvnrepository.com
  - mongodb java 검색

디비별로 드라이버가 필요한건 알죠?



검색하세요;

몽고디비는 자바스크립트: 모ㅣ든 입출력 비동기로 단ㄴ일쓰레드
자바는 디폴트가 동기. 그걸을 맞춰주는의미해서 -sync
드라이버가 맞춰준다 라는 뜻

groupid artifactid version

```
implementation 'org.mongodb:mongodb-driver-sync:5.0.0'
```

MongoDB Java 연동

build.gradle

```
dependencies {
    implementation 'ch.qos.logback:logback-classic:1.2.11'
    implementation 'org.mongodb:mongodb-driver-sync:5.0.0'
    …
}
```

해당 드라이버는 로그시스템에 로그를 남기는 형식이다.

로그에다가 남기니 오류가 나도 로그 시스템에다가 출력한다.

로그 시스템은 스프링때도 나온다

gradle 동기화 잊지말고

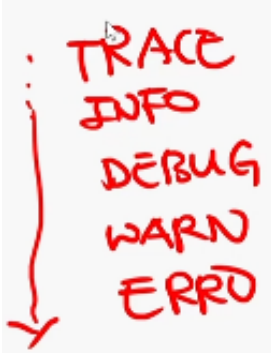
MongoDB Java 연동

resources/logback.xml

로그시스템을 사용하기 위해선 설정을 해야되용

자바코드를 제외한 나머지 : 설정 파일도 포함되어있으니

```
<configuration>
    <appender name="CONSOLE"
              class="ch.qos.logback.core.ConsoleAppender">
        <encoder>
            <pattern>
                %-4relative [%thread] %-5level %logger{30} - %msg%n
            </pattern>
        </encoder>
    </appender>
    <logger name="org.mongodb.driver.connection" level="INFO" additivity="true"/>
    <root level="INFO">
        <appender-ref ref="CONSOLE" />
    </root>
</configuration>
```



로그 메세지 패턴

쓰레드 쓰레드레벨 누가 진짜메세지

메세지 성격에 따른 레벨

어디로 출력할거냐

출력방향콘솔로. 파일도 ㄱㄴ 디비나 네트워크로도 ㄱㄴ

레벨에따라 출력 범위를 지정할 수있다.
개발시 info레벨 아래

확장한 마크업언어.
태그를 개발자가 정의할 수있음
반드시 지켜야하는 규칙은 well-formed

xml도 최소로 따라야하는 웰폼드가 있다.
규칙이 엄격함. 대소문자 구분하고

설정을 위한 파일 형식 2가지
대표적인거

.properties DB정보들
심플하지만 구조적이지 않아요

복자한 설정. 구조는 xml사용.

스프링부트등등은
yml을 사용
properties를 계층적으로 키레벨에서 발전시킨것

MongoDB 연결하기

JDBC와 비슷

서버아이피:포트번호

```
String uri = "mongodb://127.0.0.1:27017";
String db = "todo_db";
```

접속 url

디비 명

인증 없이가 디폴트로 해놨으니
인증은 없다

```
try (MongoClient client =MongoClients.create(uri)){
    MongoDatabase database = client.getDatabase(db);
} catch(Exception e) {
    e.printStackTrace();
}
```

몽고디비와 연결되는 client
JDBC에서 connection과 대응

커넥션을 얻었으니
디비를 얻으면 된다

아까본 전역변수 db에 해당한다.

몽고디비는 원래 인증은 없이 진행하는게
수순이냐

디비는 방화벽 안에 있어서 웹서버만 접근 허용.

db까지 얻는 것을
유틸리티로 만들자

몽고디비 config파일 보면 bindip가 있었죠?
그러면 굳이 유저 아이디로 로그인 안해도
안전하게 운용될수있다라는 의미.
물론 인증모드설정하고 유저등록시키고도 할수 있음

MongoDB Java 연동

ConnectionTest.java

그전에 테스트함 해보고잉

```
package sec01;

import com.mongodb.client.MongoClient;
import com.mongodb.client.MongoClients;
import com.mongodb.client.MongoDatabase;

public class ConnectionTest {

    public static void main(String[] args) {
        String uri = "mongodb://127.0.0.1:27017";
        String db = "todo_db";

        try (MongoClient client =MongoClients.create(uri)){
            MongoDatabase database = client.getDatabase(db);
        } catch(Exception e) {
            e.printStackTrace();
        }
    }
}
```

junit말고 테스트해봐그냥.

번외
파일 구조에서 TEST폴더에서 만드는 것은
junit으로 단위테스트할때 사용하는 폴더다

실패했다면 로그 설정한대로
출력이 안되죠.

실행시 아까 설정한대로 로그 나오죠

쓰레드이름 쓰레드레벨 누가출력한건지 실제 메세지

```
> Task :sec01.ConnectionTest.main()
309  [main] INFO  org.mongodb.driver.client - MongoClient with metadata {"driver": {"
```

MongoDB Java 연동

Database.java

앱패키지 하나 만들고 유틸리티 클래스 만들자

```
package app;

import com.mongodb.ConnectionString;
import com.mongodb.client.MongoClient;
import com.mongodb.client.MongoClients;
import com.mongodb.client.MongoCollection;
import com.mongodb.client.MongoDatabase;
import org.bson.Document;

public class Database {
    static MongoClient client;
    static MongoDatabase database;

    static {
        ConnectionString connectionString = new ConnectionString("mongodb://127.0.0.1:27017");
        client = MongoClients.create(connectionString);
        database = client.getDatabase("todo_db");
    }
```

싱글톤 운영을 위해서
static 필드 2가지 지정

좀더 복잡한 커넥션을 위해서 사용하는 함수.

클라(커넥션)객체 얻고
디비 얻고

이렇게 만들면
유저아이디 패스워드 추후에
추가 가능하다!

Database.java

```
    public static void close() {
        client.close();
    }

    public static MongoDatabase getDatabase() {
        return database;
    }

    public static MongoCollection<Document> getCollection(String colName) {
        MongoCollection<Document> collection = database.getCollection(colName);
        return collection;
    }
}
```

클라이언트 반환

Document객체는
일종의 자바의 맵 확장

도큐먼트라는 객체

컬렉션 얻는 메소드

디비에서 도큐먼트에 대한 콜렉션(테이블) 얻기

MongoDB Java 연동

추가

얻은 컬렉션으로 운용해라

- collection.InsertOne(Document)

```
Document document = new Document();

document.append("title", "MongoDB");
document.append("desc", "MongoDB 공부하기");
document.append("done", false);

InsertOneResult result = collection.insertOne(document);
System.out.println("==> InsertOneResult : " + result.getInsertedId());
```

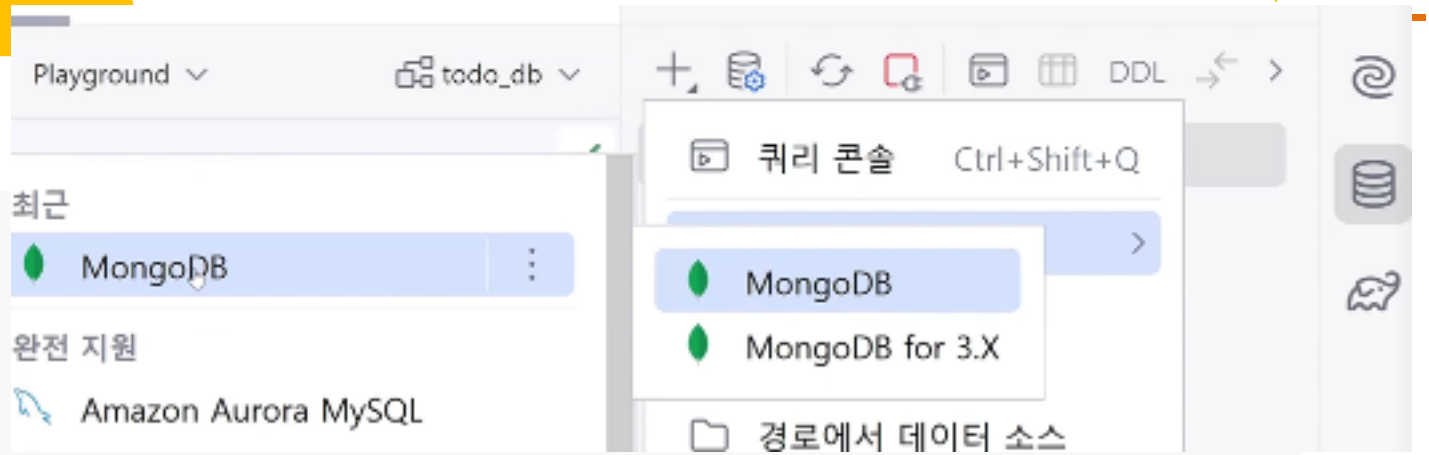
도큐먼트 생성후

append해서 (키와 밸류 상으로)

몇개, 발급된 아이디 정보 포함
등등 들어가있는 타입

MongoDB Java 연동

InsertOneTest.java




```
package sec02;      인서트 테스트 진행

import app.Database;
import com.mongodb.client.MongoCollection;
import com.mongodb.client.result.InsertOneResult;
import org.bson.Document;

public class InsertOneTest {
    public static void main(String[] args) {
        MongoCollection<Document> collection = Database.getCollection("todo");

        Document document = new Document();
        document.append("title", "MongoDB");
        document.append("desc", "MongoDB 공부하기");
        document.append("done", false);

        InsertOneResult result = collection.insertOne(document);
        System.out.println("==> InsertOneResult : " + result.getInsertedId());
        Database.close();
    }
}
```

제대로 됐는지는
studio 3t 열어서 확인해봐

mysql워크벤치에 기능을
인텔리 제이에서 햇듯이

몽고디비(3t)의 기능을
인텔리 제이에서 연결ㄹ하여
할수있다`

MongoDB Java 연동

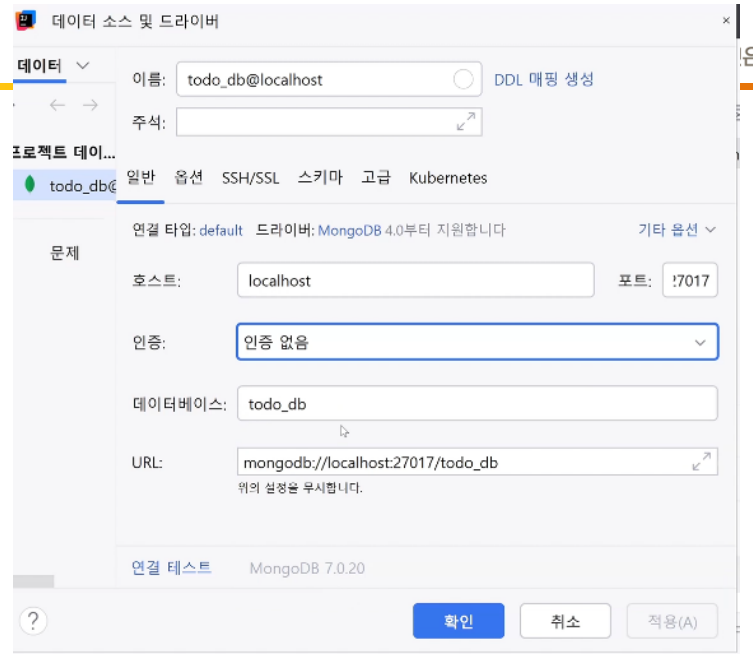
- **추가**
  - collection.insertMany(List)

도큐먼트 많이 만들어서
리스트로 구성하고 넘기면돼

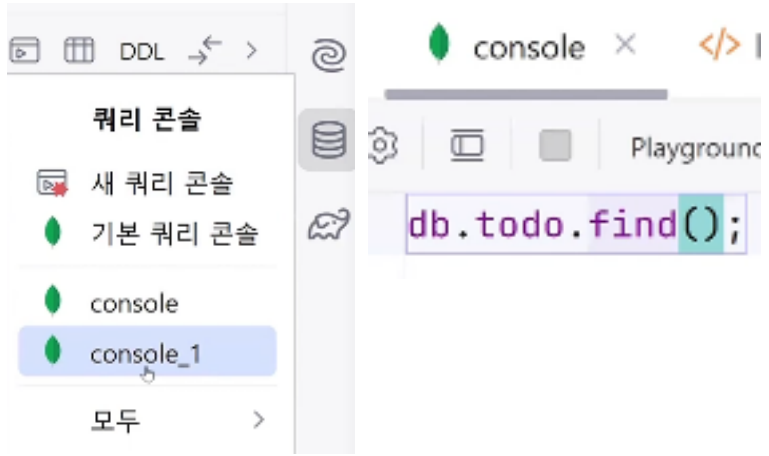
```
List<Document> insertList = new ArrayList<>();
Document document1 = new Document();
Document document2 = new Document();

document1.append("title", "Dune2 영화보기");
document1.append("desc", "이번 주말 IMAX로 Dune2 영화보기");
document1.append("done", false);

document2.append("title", "Java MongoDB 연동");
document2.append("desc", "Java로 MongoDB 연동 프로그래밍 연습하기");
document2.append("done", false);
insertList.add(document1);
insertList.add(document2);
```

복수형 결과 객체

```
InsertManyResult result = collection.insertMany(insertList);
System.out.println("==> InsertManyResult : " + result.getInsertedIds());
```



mysql연동시에는 .ㄴsql파일을 만들어 쿼리문을 실행했지만 몽고는 켜진 콘솔에서 확인하자



InsertManyTest.java

```
package sec02;

import app.Database;
import com.mongodb.client.MongoCollection;
import com.mongodb.client.result.InsertManyResult;
import org.bson.Document;

import java.util.ArrayList;
import java.util.List;

public class InsertManyTest {
    public static void main(String[] args) {
        MongoCollection<Document> collection = Database.getCollection("todo");

        List<Document> insertList = new ArrayList<>();

        Document document1 = new Document();
        Document document2 = new Document();
```

InsertManyTest.java

```
        document1.append("title", "Dune2 영화보기");
        document1.append("desc", "이번 주말 IMAX로 Dune2 영화보기");
        document1.append("done", false);

        document2.append("title", "Java MongoDB 연동");
        document2.append("desc", "Java로 MongoDB 연동 프로그래밍 연습하기");
        document2.append("done", false);

        insertList.add(document1);
        insertList.add(document2);

        InsertManyResult result = collection.insertMany(insertList);
        System.out.println("==> InsertManyResult : " + result.getInsertedIds());

        Database.close();
    }
}
```

InsertManyTest2.java

```
package sec02;

import app.Database;
import com.mongodb.client.MongoCollection;
import com.mongodb.client.result.InsertManyResult;
import org.bson.Document;
…

public class InsertMany2Test {
    public static void main(String[] args) {
        MongoCollection<Document> collection = Database.getCollection("users");

        List<Document> insertList = new ArrayList<>();
        for(int i = 10; i < 21; i++) {
            Document document = new Document();
            document.append("name", "user_" + i);
            document.append("age", i);
            document.append("created", new Date() );
            insertList.add(document);
        }
```

안만들어도
컬렉션 만들어진다

InsertManyTest2.java

```
        InsertManyResult result = collection.insertMany(insertList);
        System.out.println("==> InsertManyResult : " + result.getInsertedIds());

        Database.close();
    }
}
```

MongoDB Java 연동

비교연산자

제일 낯선 파트.

자바에서도 또 달라지는 낯선파트다.
다 메소드로 해ㅓ야한다

이름	설명
eq	지정된 값과 같은 값을 찾습니다.
ne	지정된 값과 같지 않은 모든 값과 일치합니다.
lt	지정된 값보다 작은 값과 일치합니다.
lte	지정된 값보다 작거나 같은 값을 찾습니다.
gt	지정된 값보다 큰 값을 찾습니다.
gte	지정된 값보다 크거나 같은 값과 일치합니다.
in	배열에 지정된 값 중 하나와 일치합니다.
nin	배열에 지정된 값과 일치하지 않습니다.

- static 메서드 임포트로 간소화해서 사용
  - import static com.mongodb.client.model.Filters.eq;

원래 스태틱 매서드 형식인
클래스.매서드명() 해야하는데

이렇게 하면 그냥 메서드명()가능

MongoDB Java 연동

조회

단순 조회

```
FindIterable<Document> doc = collection.find();

Iterator itr = doc.iterator();
while (itr.hasNext()) {
    System.out.println("==> findResultRow : " + itr.next());
}
```

id 조회

필드명 과 밸류

```
String id = "...";
Bson query = eq("_id", new ObjectId(id));
```

원래는 조건 문서를 써야하지만
자바에서는 조건 객체를 만들어서 넣어야한다

조건 메서드

```
Document doc = collection.find(query).first();
System.out.println("==> findByIdResult : " + doc);
```

FindTest.java

```
package sec03;

import app.Database;
import com.mongodb.client.FindIterable;
import com.mongodb.client.MongoCollection;
import org.bson.Document;

import java.util.Iterator;

public class FindTest {
    public static void main(String[] args) {
        MongoCollection<Document> collection = Database.getCollection("todo");

        FindIterable<Document> doc = collection.find();

        Iterator itr = doc.iterator();
        while (itr.hasNext()) {
            System.out.println("==> findResultRow : "+itr.next());
        }

        Database.close();
    }
}
```

FindOneTest.java

```
package sec03;
…
import org.bson.conversions.Bson;
import org.bson.types.ObjectId;

import static com.mongodb.client.model.Filters.eq;

public class FindOneTest {
    public static void main(String[] args) {
        MongoCollection<Document> collection = Database.getCollection("todo");

        String id = "666a6296f4fe57189cd03eea";
        Bson query = eq("_id", new ObjectId(id));

        Document doc = collection.find(query).first();
        System.out.println("==> findByIdResult : " + doc);
        Database.close();
    }
}
```

findOne( ) 대응하는 코드

MongoDB Java 연동

수정

- updateOne(쿼리, 수정내용)

```
Bson query = eq("_id", new ObjectId(id));
```

대상 조건 객체 만들기

```
Bson updates = Updates.combine(
        Updates.set("name", "modify name"),
        Updates.currentTimestamp("lastUpdated")
);
```

수정사항 객체 만들기 $set:{ .....} 에 대응하는 것

필드수정 문장

대상, 수정사항 객체 만들어

```
UpdateResult result = collection.updateOne(query, updates);
System.out.println("==> UpdateResult : " + result.getModifiedCount());
```

UpdateOneTest.java

```
package sec04;
…

public class UpdateOne {
    public static void main(String[] args) {
        MongoCollection<Document> collection = Database.getCollection("users");

        String id = "666a6b65a25a4c74fddfedc2";
        Bson query = eq("_id", new ObjectId(id));

        Bson updates = Updates.combine(
                Updates.set("name", "modify name"),
                Updates.currentTimestamp("lastUpdated"));

        UpdateResult result = collection.updateOne(query, updates);
        System.out.println("==> UpdateResult : " + result.getModifiedCount());

        Database.close();
    }
}
```

MongoDB Java 연동

수정

- updateMany(쿼리, 수정내용)

```
Bson query = gt("age", age);

Bson updates = Updates.combine(
        Updates.set("name", "modify name"),
        Updates.currentTimestamp("lastUpdated"));

UpdateResult result = collection.updateMany(query, updates);
System.out.println("==> UpdateManyResult : " + result.getModifiedCount());
```

UpdateManyTest.java

```
package sec04;
…
import static com.mongodb.client.model.Filters.gt;

public class UpdateMany {
    public static void main(String[] args) {
        MongoCollection<Document> collection = Database.getCollection("users");

        int age = 16;
        Bson query = gt("age", age);

        Bson updates = Updates.combine(
                Updates.set("name", "modify name"),
                Updates.currentTimestamp("lastUpdated"));

        UpdateResult result = collection.updateMany(query, updates);
        System.out.println("==> UpdateManyResult : " + result.getModifiedCount());

        Database.close();
    }
}
```

MongoDB Java 연동

삭제

- deleteOne(쿼리)

```
Bson query = eq("_id", new ObjectId(id));

DeleteResult result = collection.deleteOne(query);
System.out.println("➔ DeleteResult : " + result.getDeletedCount());
```

- deleteMany(쿼리)

```
Bson query = gt("age", age);

DeleteResult result = collection.deleteMany(query);
System.out.println("==> DeleteManyResult : " + result.getDeletedCount());
```

DeleteOneTest.java

```
package sec05;

…

public class DeleteOneTest {
    public static void main(String[] args) {
        MongoCollection<Document> collection = Database.getCollection("users");

        String id = "666a6b65a25a4c74fddfedc2";
        Bson query = eq("_id", new ObjectId(id));

        DeleteResult result = collection.deleteOne(query);
        System.out.println("==> DeleteResult : " + result.getDeletedCount());

        Database.close();

    }
}
```

DeleteManyTest.java

```
package sec05;

…

public class DeleteManyTest {
    public static void main(String[] args) {
        MongoCollection<Document> collection = Database.getCollection("users");

        int age = 15;
        Bson query = gt("age", age);

        DeleteResult result = collection.deleteMany(query);
        System.out.println("==> DeleteManyResult : " + result.getDeletedCount());


        Database.close();

    }
}
```

복잡하고 쓰기 어려운데?
객체 지향적인 자바스타일로 하면 좋을텐데

지금까지 1번 방법 이었고
이제 자바스타일인 2번 방법을 쓰자.

VO를 이용한
POJO 방식 ㄱㄱ