### mongo

몽고DB를 탐색하거나, 쿼리 결과를 파일로 전송합니다.

#### 문법

mongo PROFILE [database=DB\_NAME] SUBCOMMANDmongo PROFILE [database=DB\_NAME] MONGODB\_METHOD

필수 매개변수

**PROFILE**

몽고DB 접속 프로파일. 웹 콘솔에서 프로파일을 구성할 수 있습니다.

ENT-3.10.2009.0, SNR-3.1.2008.0부터 몽고DB 프로파일이 접속 프로파일에 통합되었습니다.

**SUBCOMMAND**

몽고DB 세션에서 실행할 명령: dbs, cols

**cols**

database=DB\_NAME 옵션으로 지정한 데이터베이스의 컬렉션 목록을 출력합니다. 데이터베이스를 지정하지 않으면 모든 데이터베이스의 컬렉션 목록을 보여줍니다.

**dbs**

몽고DB의 모든 데이터베이스 목록을 출력합니다. 다른 매개변수 없이 단독으로 사용합니다.

**MONGO\_METHOD**

몽고DB 세션에서 실행할 네이티브 메서드. COL\_NAME으로 지정한 컬렉션에 대해 입력 레코드나 표현식 등을 매개변수 인자로 받아 메소드를 실행합니다.

선택 매개변수

**database=DB\_NAME**

cols 명령 또는 MONGO\_METHOD로 지정한 메서드의 대상이 되는 데이터베이스

#### 설명

COMMAND별 출력 필드

dbs 명령의 출력 필드는 다음과 같습니다.

1. **name**: 데이터베이스 이름
2. **disk\_usage**: 디스크 사용량 (바이트)
3. **empty**: 데이터베이스가 비어 있으면 true, 그렇지 않으면 false

cols 명령의 출력 필드는 다음과 같습니다.

1. **name**: 컬렉션 이름
2. **type**: 타입 (collection)
3. **options**: 컬렉션 구성 옵션
4. **info**: 읽기 전용 여부 등 속성 정보
5. **idIndex**: \_id 인덱스 명세

MONGO\_METHOD

mongodb 명령어는 다음과 같은 몽고DB 네이티브 메소드를 지원합니다. [몽고DB 레퍼런스 매뉴얼](https://www.mongodb.com/docs/manual/reference/method/)에서 각 메소드의 사용법을 확인할 수 있습니다.

1. [db.COL\_NAME.find()](https://www.mongodb.com/docs/manual/reference/method/db.collection.find/): 문서 검색
2. [db.COL\_NAME.insert()](https://www.mongodb.com/docs/manual/reference/method/db.collection.insert/): 컬렉션에 문서 삽입
3. [db.COL\_NAME.updateOne()](https://www.mongodb.com/docs/manual/reference/method/db.collection.updateOne/): 단일 문서 수정
4. [db.COL\_NAME.updateMany()](https://www.mongodb.com/docs/manual/reference/method/db.collection.updateMany/): 여러 문서 수정
5. [db.COL\_NAME.deleteOne()](https://www.mongodb.com/docs/manual/reference/method/db.collection.deleteOne/): 단일 문서 삭제
6. [db.COL\_NAME.deleteMany()](https://www.mongodb.com/docs/manual/reference/method/db.collection.deleteMany/): 여러 문서 삭제

find()

db.COL\_NAME.find("FILTER\_EXPR", "PROJECTION\_EXPR")

COL\_NAME로 지정한 컬렉션에서 문서를 조회합니다.

**"FILTER\_EXPR"**

검색 조건(필터) 표현식. JSON 형식으로 작성합니다. [몽고DB 질의 연산자](https://www.mongodb.com/docs/manual/reference/operator/query/)를 사용할 수 있습니다. 매개변수로 null을 지정하면 모든 문서를 조회합니다.

**"PROJECTION\_EXPR"**

검색된 문서에서 가져올 필드 표현식. JSON 형식으로 작성합니다. 이 매개변수를 생략하면 메소드는 검색 조건과 일치하는 문서의 모든 필드를 반환합니다. 자세한 내용은 [Projection](https://www.mongodb.com/docs/manual/reference/method/db.collection.find/)을 참고하십시오.

insert()

db.COL\_NAME.insert()

입력으로 받는 레코드를 COL\_NAME으로 지정한 컬렉션에 문서로 기록합니다.

입력 레코드에 **\_id** 필드 값이 있으면 고유 식별 키(ObjectId)로 사용합니다. 고유 식별 키로 사용할 필드의 값은 모두 고유해야 합니다. 기록할 레코드에 **\_id** 필드 값이 없으면 몽고DB 드라이버가 12바이트의 16진수 고유 식별 키를 부여합니다. 사용 예 4, 5를 참조하십시오.

updateOne()

db.COL\_NAME.updateOne("KEY", "UPDATE")

COL\_NAME으로 지정한 컬렉션에서 입력받은 KEY 필드의 값으로 검색해 가장 먼저 검색된 문서에서 UPDATE 필드의 값을 수정합니다. 사용 예 6을 참조하십시오.

**\_id** 필드를 KEY로 사용할 때 고유식별자가 자동 생성된 12자리 바이너리인 경우, **\_id** 필드의 값을 바이너리 타입으로 설정해야 합니다.

updateMany()

db.COL\_NAME.updateMany("KEY\_LIST", "UPDATE\_LIST")

COL\_NAME으로 지정한 컬렉션에서 입력받은 KEY\_LIST 필드의 값으로 검색해 UPDATE\_LIST 필드의 값을 수정합니다. KEY\_LIST, UPDATE\_LIST는 모두 구분자로 쉼표(,)를 사용합니다.

db.COL\_NAME.updateOne() 메서드와 달리 검색된 모든 문서를 대상으로 수정합니다.

deleteOne()

db.COL\_NAME.deleteOne("KEY")

COL\_NAME으로 지정한 컬렉션에서 입력받은 KEY 필드의 값으로 검색해 가장 먼저 검색된 문서를 삭제합니다. 사용 예 7을 참조하십시오.

**\_id** 필드를 KEY로 사용할 때 고유식별자가 자동 생성된 12자리 바이너리인 경우, **\_id** 필드의 값을 바이너리 타입으로 설정해야 합니다.

deleteMany()

db.COL\_NAME.deleteMany("KEY\_LIST")

COL\_NAME으로 지정한 컬렉션에서 입력받은 KEY\_LIST 필드의 값과 일치하는 모든 문서를 삭제합니다. KEY\_LIST는 구분자로 쉼표(,)를 사용합니다.

#### 사용 예

nventory 컬렉션의 모든 문서 조회

mongo PROFILE db.inventory.find()

restaurants 컬렉션에서 stars 3점 이상의 문서 조회

mongo PROFILE db.restaurants.find("{stars: {$gte: 3}}")

restaurants 컬렉션에서 name, stars 필드만 출력

mongo PROFILE db.restaurants.find(null, "{name: true, stars: true}")

임의의 JSON 문서를 몽고DB restaurants 컬렉션에 입력

json "{ name: Café Con Leche, contact: { phone: 228-555-0149, email: cafeconleche@example.com, location: [-73.92502, 40.8279556] }, stars:3, categories: [Bakery, Coffee, Pastries] }" | mongo PROFILE db.restaurants.insert()

로그프레소 inventory 테이블의 최근 10건을 몽고DB inventory 컬렉션에 입력

table limit=10 inventory | mongo PROFILE db.inventory.insert()

name 필드를 기준으로 restaurants 컬렉션을 검색하고 stars 값을 수정

json "{}" | eval name="Café Con Leche", stars=4 | mongo PROFILE db.restaurants.updateOne("name", "stars")

ObjectId가 5982df1b7098262f64d4ffaf인 문서 1건을 검색하여 삭제

json "{}" | eval \_id = fromhex("5982df1b7098262f64d4ffaf") | mongo stream2 db.restaurants.deleteOne("\_id")