Java MongoDB : Query document

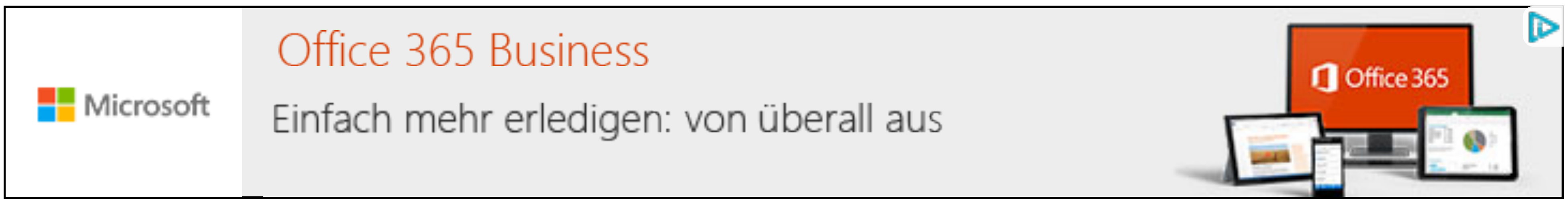
By mkyong (http://www.mkyong.com/author/mkyong/) | May 10, 2011 | Updated : April 2, 2013 | Viewed : 117,351 times

In this tutorial, we show you few common ways to get or query document from collection.

Test Data

Insert 5 dummy documents for testing.

Java
```
{ "_id" : { "$oid" : "id"} , "number" : 1 , "name" : "mkyong-1"}
{ "_id" : { "$oid" : "id"} , "number" : 2 , "name" : "mkyong-2"}
{ "_id" : { "$oid" : "id"} , "number" : 3 , "name" : "mkyong-3"}
{ "_id" : { "$oid" : "id"} , "number" : 4 , "name" : "mkyong-4"}
{ "_id" : { "$oid" : "id"} , "number" : 5 , "name" : "mkyong-5"}
```



1. Find() examples

1.1 Get first matched document only.

Java
```
    DBObject doc = collection.findOne();
    System.out.println(dbObject);
```

Output

Bash
```
    { "_id" : { "$oid" : "id"} , "number" : 1 , "name" : "mkyong-1"}
```

1.2 Get all matched documents.

Java
```
    DBCursor cursor = collection.find();
    while(cursor.hasNext()) {
        System.out.println(cursor.next());
    }
```

Output

Bash

```
{ "_id" : { "$oid" : "id"} , "number" : 1 , "name" : "mkyong-1"}
{ "_id" : { "$oid" : "id"} , "number" : 2 , "name" : "mkyong-2"}
{ "_id" : { "$oid" : "id"} , "number" : 3 , "name" : "mkyong-3"}
{ "_id" : { "$oid" : "id"} , "number" : 4 , "name" : "mkyong-4"}
{ "_id" : { "$oid" : "id"} , "number" : 5 , "name" : "mkyong-5"}
```

1.3 Get single field from matched document.

Java

```
    BasicDBObject allQuery = new BasicDBObject();
    BasicDBObject fields = new BasicDBObject();
    fields.put("name", 1);

    DBCursor cursor = collection.find(allQuery, fields);
    while (cursor.hasNext()) {
        System.out.println(cursor.next());
    }
```

Output

Bash

```
{ "_id" : { "$oid" : "id"} , "name" : "mkyong-1"}
{ "_id" : { "$oid" : "id"} , "name" : "mkyong-2"}
{ "_id" : { "$oid" : "id"} , "name" : "mkyong-3"}
{ "_id" : { "$oid" : "id"} , "name" : "mkyong-4"}
{ "_id" : { "$oid" : "id"} , "name" : "mkyong-5"}
```

2. Find() and Comparison

2.1 Get all documents where `number = 5`.

Java

```
    BasicDBObject whereQuery = new BasicDBObject();
    whereQuery.put("number", 5);
    DBCursor cursor = collection.find(whereQuery);
    while(cursor.hasNext()) {
        System.out.println(cursor.next());
    }
```

Output

Bash

```
{ "_id" : { "$oid" : "id"} , "number" : 5 , "name" : "mkyong-5"}
```

2.2 `$in` example – Get documents where `number in 2, 4 and 5`.

Java

```java
    BasicDBObject inQuery = new BasicDBObject();
    List<Integer> list = new ArrayList<Integer>();
    list.add(2);
    list.add(4);
    list.add(5);
    inQuery.put("number", new BasicDBObject("$in", list));
    DBCursor cursor = collection.find(inQuery);
    while(cursor.hasNext()) {
        System.out.println(cursor.next());
    }
```

Output

Bash
```bash
{ "_id" : { "$oid" : "id"} , "number" : 2 , "name" : "mkyong-2"}
{ "_id" : { "$oid" : "id"} , "number" : 4 , "name" : "mkyong-4"}
{ "_id" : { "$oid" : "id"} , "number" : 5 , "name" : "mkyong-5"}
```

2.3 `$gt $lt` example – Get documents where `5 > number > 2` .

Java
```java
    BasicDBObject gtQuery = new BasicDBObject();
    gtQuery.put("number", new BasicDBObject("$gt", 2).append("$lt", 5));
    DBCursor cursor = collection.find(gtQuery);
    while(cursor.hasNext()) {
        System.out.println(cursor.next());
    }
```

Output

Bash
```bash
{ "_id" : { "$oid" : "id"} , "number" : 3 , "name" : "mkyong-3"}
{ "_id" : { "$oid" : "id"} , "number" : 4 , "name" : "mkyong-4"}
```

2.4 `$ne` example – Get documents where `number != 4` .

Java
```java
    BasicDBObject neQuery = new BasicDBObject();
    neQuery.put("number", new BasicDBObject("$ne", 4));
    DBCursor cursor = collection.find(neQuery);
    while(cursor.hasNext()) {
        System.out.println(cursor.next());
    }
```

Output

Bash
```bash
{ "_id" : { "$oid" : "id"} , "number" : 1 , "name" : "mkyong-1"}
{ "_id" : { "$oid" : "id"} , "number" : 2 , "name" : "mkyong-2"}
{ "_id" : { "$oid" : "id"} , "number" : 3 , "name" : "mkyong-3"}
{ "_id" : { "$oid" : "id"} , "number" : 5 , "name" : "mkyong-5"}
```

3. find() and Logical

3.1 `$and` example – get documents where `number = 2 and name = 'mkyong-2'`.

```java
    BasicDBObject andQuery = new BasicDBObject();
    List<BasicDBObject> obj = new ArrayList<BasicDBObject>();
    obj.add(new BasicDBObject("number", 2));
    obj.add(new BasicDBObject("name", "mkyong-2"));
    andQuery.put("$and", obj);

    System.out.println(andQuery.toString());

    DBCursor cursor = collection.find(andQuery);
    while (cursor.hasNext()) {
        System.out.println(cursor.next());
    }
```

Output

```bash
{ "$and" : [ { "number" : 2} , { "name" : "mkyong-2"}]}

{ "_id" : { "$oid" : "id"} , "number" : 2 , "name" : "mkyong-2"}
```

4. find() and Regex

Find document with regular expression pattern.

4.1 `$regex` example – get documents where `name like pattern 'Mky.*-[1-3]', case insensitive`.

```java
    BasicDBObject regexQuery = new BasicDBObject();
    regexQuery.put("name",
        new BasicDBObject("$regex", "Mky.*-[1-3]")
        .append("$options", "i"));

    System.out.println(regexQuery.toString());

    DBCursor cursor = collection.find(regexQuery);
    while (cursor.hasNext()) {
        System.out.println(cursor.next());
    }
```

Output

```bash
{ "name" : { "$regex" : "Mky.*-[1-3]" , "$options" : "i"}}

{ "_id" : { "$oid" : "515ad59e3004c89329c7b259"} , "number" : 1 , "name" : "mkyong-1"}
{ "_id" : { "$oid" : "515ad59e3004c89329c7b25a"} , "number" : 2 , "name" : "mkyong-2"}
{ "_id" : { "$oid" : "515ad59e3004c89329c7b25b"} , "number" : 3 , "name" : "mkyong-3"}
```

There are more...
Read this MongoDB operator documentation (http://docs.mongodb.org/manual/reference/operators/) for complete set of query operators supported in MongoDB.

5. Full Example

Java

```java
package com.mkyong.core;

import java.net.UnknownHostException;
import java.util.ArrayList;
import java.util.Calendar;
import java.util.List;

import com.mongodb.BasicDBObject;
import com.mongodb.DB;
import com.mongodb.DBCollection;
import com.mongodb.DBCursor;
import com.mongodb.DBObject;
import com.mongodb.Mongo;
import com.mongodb.MongoException;

/**
 * Java MongoDB : Query document
 *
 * @author mkyong
 *
 */
public class QueryApp {

    public static void insertDummyDocuments(DBCollection collection) {

        List<DBObject> list = new ArrayList<DBObject>();

        Calendar cal = Calendar.getInstance();

        for (int i = 1; i <= 5; i++) {

            BasicDBObject data = new BasicDBObject();
            data.append("number", i);
            data.append("name", "mkyong-" + i);
            // data.append("date", cal.getTime());

            // +1 day
            cal.add(Calendar.DATE, 1);

            list.add(data);

        }

        collection.insert(list);

    }

    public static void main(String[] args) {

    try {

      Mongo mongo = new Mongo("localhost", 27017);
      DB db = mongo.getDB("yourdb");
```

```
        // get a single collection
        DBCollection collection = db.getCollection("dummyColl");

        insertDummyDocuments(collection);

        System.out.println("1. Find first matched document");
        DBObject dbObject = collection.findOne();
        System.out.println(dbObject);

        System.out.println("\n1. Find all matched documents");
        DBCursor cursor = collection.find();
        while (cursor.hasNext()) {
          System.out.println(cursor.next());
        }

        System.out.println("\n1. Get 'name' field only");
        BasicDBObject allQuery = new BasicDBObject();
        BasicDBObject fields = new BasicDBObject();
        fields.put("name", 1);

        DBCursor cursor2 = collection.find(allQuery, fields);
        while (cursor2.hasNext()) {
          System.out.println(cursor2.next());
        }

        System.out.println("\n2. Find where number = 5");
        BasicDBObject whereQuery = new BasicDBObject();
        whereQuery.put("number", 5);
        DBCursor cursor3 = collection.find(whereQuery);
        while (cursor3.hasNext()) {
          System.out.println(cursor3.next());
        }

        System.out.println("\n2. Find where number in 2,4 and 5");
        BasicDBObject inQuery = new BasicDBObject();
        List<Integer> list = new ArrayList<Integer>();
        list.add(2);
        list.add(4);
        list.add(5);
        inQuery.put("number", new BasicDBObject("$in", list));
        DBCursor cursor4 = collection.find(inQuery);
        while (cursor4.hasNext()) {
          System.out.println(cursor4.next());
        }

        System.out.println("\n2. Find where 5 > number > 2");
        BasicDBObject gtQuery = new BasicDBObject();
        gtQuery.put("number", new BasicDBObject("$gt", 2).append("$lt", 5));
        DBCursor cursor5 = collection.find(gtQuery);
        while (cursor5.hasNext()) {
          System.out.println(cursor5.next());
        }

        System.out.println("\n2. Find where number != 4");
        BasicDBObject neQuery = new BasicDBObject();
        neQuery.put("number", new BasicDBObject("$ne", 4));
        DBCursor cursor6 = collection.find(neQuery);
```

```
      while (cursor6.hasNext()) {
        System.out.println(cursor6.next());
      }

      System.out.println("\n3. Find when number = 2 and name = 'mkyong-2' example");
      BasicDBObject andQuery = new BasicDBObject();

      List<BasicDBObject> obj = new ArrayList<BasicDBObject>();
      obj.add(new BasicDBObject("number", 2));
      obj.add(new BasicDBObject("name", "mkyong-2"));
      andQuery.put("$and", obj);

      System.out.println(andQuery.toString());

      DBCursor cursor7 = collection.find(andQuery);
      while (cursor7.hasNext()) {
        System.out.println(cursor7.next());
      }

      System.out.println("\n4. Find where name = 'Mky.*-[1-3]', case sensitive example");
      BasicDBObject regexQuery = new BasicDBObject();
      regexQuery.put("name",
        new BasicDBObject("$regex", "Mky.*-[1-3]")
                    .append("$options", "i"));

      System.out.println(regexQuery.toString());

      DBCursor cursor8 = collection.find(regexQuery);
      while (cursor8.hasNext()) {
        System.out.println(cursor8.next());
      }

      collection.drop();

      System.out.println("Done");

     } catch (UnknownHostException e) {
        e.printStackTrace();
     } catch (MongoException e) {
        e.printStackTrace();
     }

    }
}
```

Done.

References

1. Query, Update, and Projection Operators Quick Reference (http://docs.mongodb.org/manual/reference/operators/)

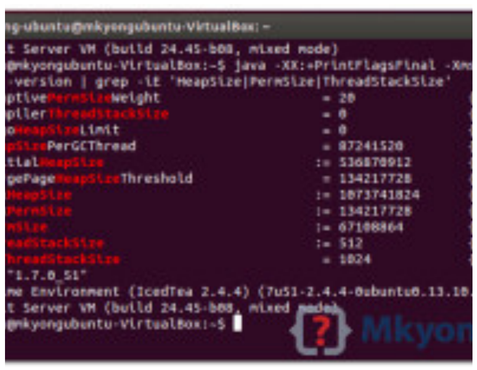
Tags : mongodb (http://www.mkyong.com/tag/mongodb/) query (http://www.mkyong.com/tag/query/)

Share this article on

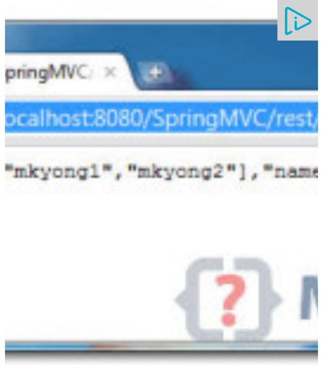
Twitter (https://twitter.com/intent/tweet?text=Java MongoDB : Query document&url=http://www.mkyong.com/mongodb/java-mongodb-query-document/&via=mkyong) Facebook (https://www.facebook.com/sharer/sharer.php?u=http://www.mkyong.com/mongodb/java-mongodb-query-document/) Google+ (https://plus.google.com/share?url=http://www.mkyong.com/mongodb/java-mongodb-query-document/)

Top 20 Java Websites

Spring Data MongoDB – Select fields to return



Find out your Java heap memory size



Spring 3 MVC and example

NEXT ›

Reader also read :

MongoDB : Sort exceeded memory

Spring Data MongoDB – Aggregation

Spring Data MongoDB – Select fields to

MongoDB – Allow remote access

limit of 104857600 bytes (http://www.mkyong.com/mongodb/mongodb-sort-exceeded-memory-limit-of-104857600-bytes/)

Grouping Example (http://www.mkyong.com/mongodb/mongodb-aggregation-grouping-example/)

return (http://www.mkyong.com/mongodb/spring-data-mongodb-select-fields-to-return/)

(http://www.mkyong.com/mongodb/mongodb-allow-remote-access/)

MongoDB – How to remove a field from document (http://www.mkyong.com/mongodb/mongodb-how-to-remove-a-field-from-document/)

About the Author

mkyong

Founder of Mkyong.com (http://mkyong.com) and HostingCompass.com (http://hostingcompass.com), love Java and open source stuff. Follow him on Twitter (https://twitter.com/mkyong), or befriend him on Facebook (http://www.facebook.com/java.tutorial) or Google Plus (https://plus.google.com/110948163568945735692?rel=author). If you like my tutorials, consider make a donation to these charities (http://www.mkyong.com/blog/donate-to-charity/).

Comments

MuleSoft

APIs will change your industry.

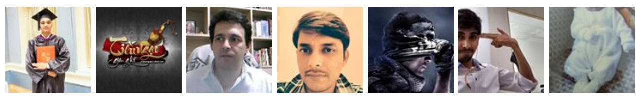
Find out how

Mkyong
48,353 likes

Like Page

Sign Up

Be the first of your friends to like this



Android Getting Started (http://developer.android.com/training/index.html)
Google App Engine – Java (https://cloud.google.com/appengine/docs/java/)
Spring 2.5.x Documentation (http://docs.spring.io/spring/docs/2.5.x/reference/index.html)
Spring 3.2.x Documentation (http://docs.spring.io/spring/docs/3.2.x/spring-framework-reference/html/)
Spring 4.1.x Documentation (http://docs.spring.io/spring/docs/4.1.x/spring-framework-reference/html/)
Java EE 5 Tutorial (http://docs.oracle.com/javaee/5/tutorial/doc/docinfo.html)
Java EE 6 Tutorial (http://docs.oracle.com/javaee/6/tutorial/doc/docinfo.html)
Java EE 7 Tutorial (https://docs.oracle.com/javaee/7/tutorial/index.html)
Java 6 API (http://docs.oracle.com/javase/6/docs/api/overview-summary.html)
Java 7 API (http://docs.oracle.com/javase/7/docs/api/overview-summary.html)
Java 8 API (http://docs.oracle.com/javase/8/docs/api/overview-summary.html)
JSF Home Page (https://javaserverfaces.java.net/)
JSP Home Page (https://jsp.java.net/)

Maven Central Repository (http://search.maven.org/)
Hibernate ORM (http://hibernate.org/orm/)
JAX-WS Home Page (https://jax-ws.java.net/)
JAX-RS Home Page (Jersey) (https://jax-ws.java.net/)

Java Code Geeks (http://www.javacodegeeks.com/)
TestNG Founder (http://beust.com/weblog/)
DZone (https://dzone.com)

Mkyong.com is for Java and J2EE developers, all examples are simple and easy to understand, and well tested in my development environment.

Mkyong.com is created, written by, and maintained by Yong Mook Kim, aka Mkyong. It is built on WordPress (https://wordpress.org/), hosted by Liquid Web (http://mkyong.com/go/liquidweb/), and the caches are served by CloudFlare CDN.
