MongoDB find document examples

By Lokesh Gupta | Filed Under: MongoDB

Learn to **find documents in MongoDB**. This **mongodb find document tutorial** covers a number of ways to **query a single document** or **find multiple documents** based on same condition as we do in SQL using WHERE CLAUSE.

Table of Contents

1) Select all documents from a collection
2) Select first document from a collection
3) Select single document and limited field(s) from a collection
4) Select all documents with unique id from a collection
5) Document ids IN clause example
6) Document ids less than or greater than clause example
7) Document ids NOT IN clause example
8) Documents matching multiple fields example
9) Documents matching field value to some REGEX example

MongoDB find document – test data preparation

For building and running examples used in this tutorial, I have populated 20 employee documents in database with `employeeId` from 1 to 20, and `employeeName` from TestEmployee_1 to TestEmployee_10.

MongoDB insert test data

```
private static void setUpTestData(DBCollection collection){
    for (int i=1; i <= 10; i++) {
        collection.insert(new BasicDBObject().append("employeeId", i).append("employeeName", "TestEmployee_"+i));
    }
}
```

Now run our examples and fetch the data on different scenarios.

1. MongoDB find() – Select all documents from a collection

Select all documents from collection

```
private static void selectAllRecordsFromACollection(DBCollection collection)
{
    DBCursor cursor = collection.find();
    while(cursor.hasNext())
    {
        System.out.println(cursor.next());
    }
}
```

Program Output.

Console

```
{ "_id" : { "$oid" : "538782753641d31b0cad0142"} , "employeeId" : 1 , "employeeName" : "TestEmployee_1"}
{ "_id" : { "$oid" : "538782753641d31b0cad0143"} , "employeeId" : 2 , "employeeName" : "TestEmployee_2"}
{ "_id" : { "$oid" : "538782753641d31b0cad0144"} , "employeeId" : 3 , "employeeName" : "TestEmployee_3"}
{ "_id" : { "$oid" : "538782753641d31b0cad0145"} , "employeeId" : 4 , "employeeName" : "TestEmployee_4"}
```

```
{ "_id" : { "$oid" : "538782753641d31b0cad0146"} , "employeeId" : 5 , "employeeName" : "TestEmployee_5"}
{ "_id" : { "$oid" : "538782753641d31b0cad0147"} , "employeeId" : 6 , "employeeName" : "TestEmployee_6"}
{ "_id" : { "$oid" : "538782753641d31b0cad0148"} , "employeeId" : 7 , "employeeName" : "TestEmployee_7"}
{ "_id" : { "$oid" : "538782753641d31b0cad0149"} , "employeeId" : 8 , "employeeName" : "TestEmployee_8"}
{ "_id" : { "$oid" : "538782753641d31b0cad014a"} , "employeeId" : 9 , "employeeName" : "TestEmployee_9"}
{ "_id" : { "$oid" : "538782753641d31b0cad014b"} , "employeeId" : 10 , "employeeName" : "TestEmployee_10"}
```

2. MongoDB findOne() – Select first document from a collection

Select first document from collection

```
private static void selectFirstRecordInCollection(DBCollection collection)
{
    DBObject dbObject = collection.findOne();
    System.out.println(dbObject);
}
```

Program Output.

Console

```
{ "_id" : { "$oid" : "538782a53641ed9125df86c0"} , "employeeId" : 1 , "employeeName" : "TestEmployee_1"}
```

3. MongoDB where clause – Select single document and limited field(s) from a collection

Select document using where clause

```
private static void selectSingleRecordAndFieldByRecordNumber(DBCollection collection)
{
    BasicDBObject whereQuery = new BasicDBObject();
    whereQuery.put("employeeId", 5);
    BasicDBObject fields = new BasicDBObject();
    fields.put("employeeId", 1);

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

Program Output.

```
Console
```

```
{ "_id" : { "$oid" : "538783323641010 41fb2c141"} , "employeeId" : 5}
```

4. MongoDB find by document id

```
Find by document id
```

```
private static void selectAllRecordByRecordNumber(DBCollection collection)
{
    BasicDBObject whereQuery = new BasicDBObject();
    whereQuery.put("employeeId", 5);
    DBCursor cursor = collection.find(whereQuery);
    while(cursor.hasNext()) {
        System.out.println(cursor.next());
    }
}
```

Program Output.

```
Console
```

```
{ "_id" : { "$oid" : "538783623641e9b2da299fa7"} , "employeeId" : 5 , "employeeName" : "TestEmployee_5"}
```

5. MongoDB IN clause example

```
IN clause example
```

```
private static void in_Example(DBCollection collection)
{
    BasicDBObject inQuery = new BasicDBObject();

    List<Integer> list = new ArrayList<Integer>();
    list.add(2);
    list.add(4);
    list.add(5);

    inQuery.put("employeeId", new BasicDBObject("$in", list));

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

Program Output.

```
Console
```

```
{ "_id" : { "$oid" : "5387838d3641024de174c795"} , "employeeId" : 2 , "employeeName" : "TestEmployee_2"}
{ "_id" : { "$oid" : "5387838d3641024de174c797"} , "employeeId" : 4 , "employeeName" : "TestEmployee_4"}
{ "_id" : { "$oid" : "5387838d3641024de174c798"} , "employeeId" : 5 , "employeeName" : "TestEmployee_5"}
```

6. MongoDB – Less than or greater than clause example

Less than or greater than example

```
private static void lessThan_GreaterThan_Example(DBCollection collection)
{
    BasicDBObject getQuery = new BasicDBObject();
    getQuery.put("employeeId", new BasicDBObject("$gt", 2).append("$lt", 5));
    DBCursor cursor = collection.find(getQuery);
    while(cursor.hasNext()) {
        System.out.println(cursor.next());
    }
}
```

Program Output.

Console

```
{ "_id" : { "$oid" : "538783b63641720cd34f98c8"} , "employeeId" : 3 , "employeeName" : "TestEmployee_3"}
{ "_id" : { "$oid" : "538783b63641720cd34f98c9"} , "employeeId" : 4 , "employeeName" : "TestEmployee_4"}
```

7. MongoDb NOT IN clause example

NOT IN clause example

```
private static void negation_Example(DBCollection collection)
{
    BasicDBObject neQuery = new BasicDBObject();
    neQuery.put("employeeId", new BasicDBObject("$ne", 4));
    DBCursor cursor = collection.find(neQuery);
    while(cursor.hasNext()) {
        System.out.println(cursor.next());
    }
}
```

Program Output.

```
Console
```

```
{ "_id" : { "$oid" : "538783db3641d3ca9d400510"} , "employeeId" : 1 , "employeeName" : "TestEmployee_1"}
{ "_id" : { "$oid" : "538783db3641d3ca9d400511"} , "employeeId" : 2 , "employeeName" : "TestEmployee_2"}
{ "_id" : { "$oid" : "538783db3641d3ca9d400512"} , "employeeId" : 3 , "employeeName" : "TestEmployee_3"}
{ "_id" : { "$oid" : "538783db3641d3ca9d400514"} , "employeeId" : 5 , "employeeName" : "TestEmployee_5"} //4 is
missing
{ "_id" : { "$oid" : "538783db3641d3ca9d400515"} , "employeeId" : 6 , "employeeName" : "TestEmployee_6"}
{ "_id" : { "$oid" : "538783db3641d3ca9d400516"} , "employeeId" : 7 , "employeeName" : "TestEmployee_7"}
{ "_id" : { "$oid" : "538783db3641d3ca9d400517"} , "employeeId" : 8 , "employeeName" : "TestEmployee_8"}
{ "_id" : { "$oid" : "538783db3641d3ca9d400518"} , "employeeId" : 9 , "employeeName" : "TestEmployee_9"}
{ "_id" : { "$oid" : "538783db3641d3ca9d400519"} , "employeeId" : 10 , "employeeName" : "TestEmployee_10"}
```

8. MongoDb find documents matching multiple fields example

```
Find documents matching multiple fields
```

```
private static void andLogicalComparison_Example(DBCollection collection)
{
    BasicDBObject andQuery = new BasicDBObject();
```

```
    List<BasicDBObject> obj = new ArrayList<BasicDBObject>();
    obj.add(new BasicDBObject("employeeId", 2));
    obj.add(new BasicDBObject("employeeName", "TestEmployee_2"));
    andQuery.put("$and", obj);

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

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

Program Output.

```
Console
```

```
{ "$and" : [ { "employeeId" : 2} , { "employeeName" : "TestEmployee_2"}]}
{ "_id" : { "$oid" : "5387840336418a41167caaa4"} , "employeeId" : 2 , "employeeName" : "TestEmployee_2"}
```

9. MongoDb find documents matching REGEX example

```
Find documents matching REGEX
```

```
private static void regex_Example(DBCollection collection) {
    BasicDBObject regexQuery = new BasicDBObject();
    regexQuery.put("employeeName",
        new BasicDBObject("$regex", "TestEmployee_[3]")
        .append("$options", "i"));

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

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

Program Output.

```
Console
```

```
{ "employeeName" : { "$regex" : "TestEmployee_[3]" , "$options" : "i"}}
{ "_id" : { "$oid" : "538784213641917ce7068c57"} , "employeeId" : 3 , "employeeName" : "TestEmployee_3"}
```

10. All mondodb find query examples

```
MongoDBSelectExample.java
```

```
package examples.mongodb.crud;

import java.net.UnknownHostException;
import java.util.ArrayList;
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.MongoClient;
import com.mongodb.WriteResult;

public class MongoDBSelectExample {
```

```
    private static void setUpTestData(DBCollection collection){
        for (int i=1; i <= 10; i++) {
            collection.insert(new BasicDBObject().append("employeeId", i).append("employeeName",
"TestEmployee_"+i));
        }
    }

    public static void main(String[] args) throws UnknownHostException
    {
        MongoClient mongo = new MongoClient("localhost", 27017);
        DB db = mongo.getDB("howtodoinjava");
        DBCollection collection = db.getCollection("users");

        //Delete All documents before running example again
        WriteResult result = collection.remove(new BasicDBObject());
        System.out.println(result.toString());

        //Set up test data
        setUpTestData(collection);

        //Select all document from a collection
        selectAllRecordsFromACollection(collection);

        //Select first document in collection
        selectFirstRecordInCollection(collection);

        //Select single document and single field based on record number
        selectSingleRecordAndFieldByRecordNumber(collection);

        //Select all documents where record number = n
        selectAllRecordByRecordNumber(collection);

        //In example
        in_Example(collection);

        //Less than OR greater than example
```

```
        lessThan_GreaterThan_Example(collection);

        //Select document where record number != n
        negation_Example(collection);

        //And logical comparison query example
        andLogicalComparison_Example(collection);

        //Select documents based on regex match LIKE example
        regex_Example(collection);

    }

    private static void selectFirstRecordInCollection(DBCollection collection) {
        DBObject dbObject = collection.findOne();
        System.out.println(dbObject);
    }

    private static void selectAllRecordsFromACollection(DBCollection collection) {
        DBCursor cursor = collection.find();
        while(cursor.hasNext()) {
            System.out.println(cursor.next());
        }
    }

    private static void selectSingleRecordAndFieldByRecordNumber(DBCollection collection) {
        BasicDBObject whereQuery = new BasicDBObject();
        whereQuery.put("employeeId", 5);
        BasicDBObject fields = new BasicDBObject();
        fields.put("employeeId", 1);

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

```
    private static void selectAllRecordByRecordNumber(DBCollection collection) {
        BasicDBObject whereQuery = new BasicDBObject();
        whereQuery.put("employeeId", 5);
        DBCursor cursor = collection.find(whereQuery);
        while(cursor.hasNext()) {
            System.out.println(cursor.next());
        }
    }

    private static void in_Example(DBCollection collection) {
        BasicDBObject inQuery = new BasicDBObject();
        List<Integer> list = new ArrayList<Integer>();
        list.add(2);
        list.add(4);
        list.add(5);
        inQuery.put("employeeId", new BasicDBObject("$in", list));
        DBCursor cursor = collection.find(inQuery);
        while(cursor.hasNext()) {
            System.out.println(cursor.next());
        }
    }

    private static void lessThan_GreaterThan_Example(
            DBCollection collection) {
        BasicDBObject gtQuery = new BasicDBObject();
        gtQuery.put("employeeId", new BasicDBObject("$gt", 2).append("$lt", 5));
        DBCursor cursor = collection.find(gtQuery);
        while(cursor.hasNext()) {
            System.out.println(cursor.next());
        }
    }

    private static void negation_Example(DBCollection collection) {
        BasicDBObject neQuery = new BasicDBObject();
        neQuery.put("employeeId", new BasicDBObject("$ne", 4));
        DBCursor cursor = collection.find(neQuery);
        while(cursor.hasNext()) {
```

```
            System.out.println(cursor.next());
        }
    }

    private static void andLogicalComparison_Example(DBCollection collection) {
        BasicDBObject andQuery = new BasicDBObject();
        List<BasicDBObject> obj = new ArrayList<BasicDBObject>();
        obj.add(new BasicDBObject("employeeId", 2));
        obj.add(new BasicDBObject("employeeName", "TestEmployee_2"));
        andQuery.put("$and", obj);

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

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

    private static void regex_Example(DBCollection collection) {
        BasicDBObject regexQuery = new BasicDBObject();
        regexQuery.put("employeeName",
            new BasicDBObject("$regex", "TestEmployee_[3]")
            .append("$options", "i"));

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

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

That's all for different techniques to **fetch ducument from MongoDB**.

Happy Learning !!

References:

MongoDB find() docs

About Lokesh Gupta

Founded HowToDoInJava.com in late 2012. I love computers, programming and solving problems everyday. A family guy with fun loving nature. You can find me on Facebook, Twitter and Google Plus.

Feedback, Discussion and Comments

Chandrakanth

July 9, 2018

Hi
I'm trying to find an object from different documents in a single collection. How to write it in Spring Boot

Reply

Laxmi

August 11, 2017

I have two entity classes/entities like below.

```
@Document(collectname="book")
public class Book{
    @Id
     private  String id;
     private  String booName;
     //setter and getter
}

@Document(collection="book")
public class Author{
    @Id
     private  String id;
     private  String authorName;
     List<Book> writtenByAuthor;
     //setter and getter
}
```

Both are pointing to same collection.
Now i want to get the data and set it to two different dto’s

Reply

Shekhar

February 24, 2017

Hi..Very awesome site of MongoDb and clear example. Kindly tell me any book which I can follow to learn MongoDb very in depth like you know …Thanks in advance

Reply

gla

August 16, 2016

bonsoir, svp aidez moi à resoudre ce probleme: Dans ma Base de Données (MongoDB) j'ai deux tables. La premiere table est celle de Country et la seconde table est celle de Languages.Dans la table Country j'ai inseré comme exemple (Canada:English ,Frensh et France:Frensh) et dans la table Languages j'ai inseré seulement comme exemple(English et Frensh).Ma question est que comment creer une classe qui va me permettre de selectionner dans la table Language comme exemple (Frensh) et me donne la liste de tous les Country qui parle le (Frensh tel que Canada et France qui se trouve la table Country)?

Reply

Shashi Ranjan

June 27, 2016

Can you put a GROUPBY clause example?

Reply

Barani

June 23, 2016

Please suggest me how to use $in with case insensitive to find document

Reply

komal

April 6, 2016

I want to find a specific document that i inserted before and want to append a arrayList in same document .how can i do in java 8

Reply

komal

April 6, 2016

How can i find a pre- inserted document and that time also want to append a new document In same document using java 8 with spark.

Reply

Drashti

July 24, 2015

```
DBCursor cursor = db.getCollection("DeptDetail").find();
    while(cursor.hasNext()) {
        System.out.println(cursor.next());
    }
```

So I have try other code but this doesn't show any output or error.

```
FindIterable<Document> iterable = db.getCollection("Restaurant").find();
        iterable.forEach(new Block<Document>() {
            public void apply(final Document document) {
                System.out.println("OUTPUT IS " +document);
            }
        });
```

Reply

rashmi

May 27, 2015

Hi
When I am trying to getDatabasesNames() from a mongoClient, I am getting java.net.SocketException: Connection reset Exception. Can anyone please help to solve this

Thanks

Reply

Lakshmi Kiran

March 24, 2015

how can we use Map-Reduce function of mongodb in springs, i'm not using BasicDBObject

Reply

Lakshmi Kiran

March 24, 2015

please tell me how we can get data from two collections in mongodb using spring data with one query

Reply

Akshay

June 3, 2014

Thanks for your wonderful post.
I have one query
DB db = mongo.getDB(“howtodoinjava”);
DBCollection collection = db.getCollection(“users”);

Here we are getting the ‘howtodoinjava’ and ‘users’. But where are we actually setting them.
Might look like a dumb question. But still need help ☺

Thanks again. Keep Posting!!!

Reply

Lokesh

June 3, 2014

Collections in MongoDB are created lazily as soon as they are written to. Once you insert a document, the collection will automatically be created.

Reply

Marcos Lerin

May 30, 2014

Nice post Lokesh, I have seen that annotations can be used to map POJO's and documents from collections in MongoDB (in Spring), would be great if you take a look into it and share your experience!

Reply

Ask Questions & Share Feedback

Your email address will not be published. Required fields are marked *

Comment

*Want to Post Code Snippets or XML content? Please use [java] ... [/java] tags otherwise code may not appear partially or even fully. e.g.

```
[java]
public static void main (String[] args) {
...
}
[/java]
```

Name *

Email *

Website

POST COMMENT

Search Tutorials

Type and Press ENTER...

MongoDB Tutorial

MongoDB – Introduction

MongoDB – Installation

MongoDB – Get/Save Image GridFS

MongoDB – Insert Document

MongoDB – Query Document

Popular Tutorials

Java 8 Tutorial

Core Java Tutorial

Java Collections

Java Concurrency

Spring Boot Tutorial

Spring AOP Tutorial

Spring MVC Tutorial

Spring Security Tutorial

Hibernate Tutorial

Jersey Tutorial

Maven Tutorial

Log4j Tutorial

Regex Tutorial

Meta Links

Advertise

Contact Us