





## MongoDB Tutorial


 MongoDB - Home


 MongoDB - Overview

 MongoDB - Advantages


 MongoDB - Environment


 MongoDB - Data Modeling


 MongoDB - Create Database


 MongoDB - Drop Database

 MongoDB - Create Collection

 MongoDB - Drop Collection


 MongoDB - Data Types

 MongoDB - Insert Document


 MongoDB - Query Document

 MongoDB - Update Document


 MongoDB - Delete Document


 MongoDB - Projection


 MongoDB - Limiting Records


 MongoDB - Sorting Records


 MongoDB - Indexing

 MongoDB - Aggregation


 MongoDB - Replication

 MongoDB - Sharding


 MongoDB - Create Backup

 MongoDB - Deployment

 MongoDB - Java


 MongoDB - PHP


## Advanced MongoDB


 MongoDB Relationships


 MongoDB Database References

 MongoDB Covered Queries


 MongoDB Analyzing Queries

 MongoDB Atomic Operations


 MongoDB Advanced Indexing


 MongoDB Indexing Limitations


 MongoDB ObjectId


 MongoDB Map Reduce


 MongoDB Text Search

 MongoDB Regular Expression

 Working with Rockmongo

 MongoDB GridFS

 MongoDB Capped Collections

 MongoDB Auto-Increment Sequence

- MongoDB - Questions and Answers
- MongoDB - Quick Guide
- MongoDB - Useful Resources
- MongoDB - Discussion

# MongoDB - Java

Previous Page

Next Page

## Installation

Before we start using MongoDB in our Java programs, we need to make sure that we have MongoDB JDBC Driver and Java set up on the machine. You can check Java tutorial for Java installation on your machine. Now, let us check how to set up MongoDB JDBC driver.

- You need to download the jar from the path [Download mongo.jar](#) . Make sure to download latest release of it.
- You need to include the **mongo.jar** into your classpath.

## Connect to database

To connect database, you need to specify database name, if database doesn't exist then mongodb creates it automatically.

Code snippets to connect to database would be as follows –

```
import com.mongodb.MongoClient;
import com.mongodb.MongoException;
import com.mongodb.WriteConcern;

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

import com.mongodb.ServerAddress;
import java.util.Arrays;

public class MongoDBJDBC {

    public static void main( String args[] ) {

        try{

            // To connect to mongodb server
            MongoClient mongoClient = new MongoClient( "localhost" , 27017 );

            // Now connect to your databases
            DB db = mongoClient.getDB( "test" );
            System.out.println("Connect to database successfully");
            boolean auth = db.authenticate(myUserName, myPassword);
            System.out.println("Authentication: "+auth);

        }catch(Exception e){
            System.err.println( e.getClass().getName() + ": " + e.getMessage() );
        }
    }
}
```

Now, let's compile and run above program to create our database test. You can change your path as per your requirement. We are assuming current version of JDBC driver mongo-2.10.1.jar is available in the current path.

```
$javac MongoDBJDBC.java
$java -classpath ".:mongo-2.10.1.jar" MongoDBJDBC
```

Connect to database successfully  
Authentication: true

If you are going to use Windows machine, then you can compile and run your code as follows –

```
$javac MongoDBJDBC.java
$java -classpath ".;mongo-2.10.1.jar" MongoDBJDBC
Connect to database successfully
Authentication: true
```

Value of **auth** will be true, if the user name and password are valid for the selected database.

## Create a collection

To create a collection, **createCollection()** method of **com.mongodb.DB** class is used.

Code snippets to create a collection –

```
import com.mongodb.MongoClient;
import com.mongodb.MongoException;
import com.mongodb.WriteConcern;

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

import com.mongodb.ServerAddress;
import java.util.Arrays;

public class MongoDBJDBC {

    public static void main( String args[] ) {

        try{

            // To connect to mongodb server
            MongoClient mongoClient = new MongoClient( "localhost" , 27017 );

            // Now connect to your databases
            DB db = mongoClient.getDB( "test" );
            System.out.println("Connect to database successfully");

            boolean auth = db.authenticate(myUserName, myPassword);
            System.out.println("Authentication: "+auth);

            DBCollection coll = db.createCollection("mycol");
            System.out.println("Collection created successfully");
        }catch(Exception e){
            System.err.println( e.getClass().getName() + ": " + e.getMessage() );
        }
    }
}
```

When program is compiled and executed, it will produce the following result –

```
Connect to database successfully
Authentication: true
Collection created successfully
```

## Getting/ selecting a collection

To get/select a collection from the database, **getCollection()** method of **com.mongodb.DBCollection** class is used.

Code snippets to get/select a collection –

```
import com.mongodb.MongoClient;
import com.mongodb.MongoException;
import com.mongodb.WriteConcern;

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

import com.mongodb.ServerAddress;
import java.util.Arrays;

public class MongoDBJDBC {

    public static void main( String args[] ) {

        try{

            // To connect to mongodb server
            MongoClient mongoClient = new MongoClient( "localhost" , 27017 );
```

```
// Now connect to your databases
DB db = mongoClient.getDB( "test" );
System.out.println("Connect to database successfully");

boolean auth = db.authenticate(myUserName, myPassword);
System.out.println("Authentication: "+auth);

DBCollection coll = db.createCollection("mycol");
System.out.println("Collection created successfully");

DBCollection coll = db.getCollection("mycol");
System.out.println("Collection mycol selected successfully");
}catch(Exception e){
    System.err.println( e.getClass().getName() + ": " + e.getMessage() );
}
}
```

When program is compiled and executed, it will produce the following result –

```
Connect to database successfully
Authentication: true
Collection created successfully
Collection mycol selected successfully
```

## Insert a document

To insert a document into mongodb, **insert()** method of **com.mongodb.DBCollection** class is used.

Code snippets to insert a documents –

```
import com.mongodb.MongoClient;
import com.mongodb.MongoException;
import com.mongodb.WriteConcern;

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

import com.mongodb.ServerAddress;
import java.util.Arrays;

public class MongoDBJDBC {

    public static void main( String args[] ) {

        try{

            // To connect to mongodb server
            MongoClient mongoClient = new MongoClient( "localhost" , 27017 );

            // Now connect to your databases
            DB db = mongoClient.getDB( "test" );
            System.out.println("Connect to database successfully");

            boolean auth = db.authenticate(myUserName, myPassword);
            System.out.println("Authentication: "+auth);
            DBCollection coll = db.getCollection("mycol");
            System.out.println("Collection mycol selected successfully");

            BasicDBObject doc = new BasicDBObject("title", "MongoDB").
                append("description", "database").
                append("likes", 100).
                append("url", "http://www.tutorialspoint.com/mongodb/").
                append("by", "tutorials point");

            coll.insert(doc);
            System.out.println("Document inserted successfully");
        }catch(Exception e){
            System.err.println( e.getClass().getName() + ": " + e.getMessage() );
        }
    }
}
```

When program is compiled and executed, it will produce the following result –

```
Connect to database successfully
Authentication: true
Collection mycol selected successfully
Document inserted successfully
```

## Retrieve all documents

To select all documents from the collection, **find()** method of **com.mongodb.DBCollection** class is used. This method returns a cursor, so



you need to iterate this cursor.

Code snippets to select all documents –

```
import com.mongodb.MongoClient;
import com.mongodb.MongoException;
import com.mongodb.WriteConcern;

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

import com.mongodb.ServerAddress;
import java.util.Arrays;

public class MongoDBJDBC {

    public static void main( String args[] ) {

        try{

            // To connect to mongodb server
            MongoClient mongoClient = new MongoClient( "localhost" , 27017 );

            // Now connect to your databases
            DB db = mongoClient.getDB( "test" );
            System.out.println("Connect to database successfully");

            boolean auth = db.authenticate(myUserName, myPassword);
            System.out.println("Authentication: "+auth);

            DBCollection coll = db.getCollection("mycol");
            System.out.println("Collection mycol selected successfully");

            DBCursor cursor = coll.find();
            int i = 1;

            while (cursor.hasNext()) {
                System.out.println("Inserted Document: "+i);
                System.out.println(cursor.next());
                i++;
            }

        }catch(Exception e){
            System.err.println( e.getClass().getName() + ": " + e.getMessage() );
        }
    }
}
```

When program is compiled and executed, it will produce the following result –

```
Connect to database successfully
Authentication: true
Collection mycol selected successfully
Inserted Document: 1
{
  "_id" : ObjectId(7df78ad8902c),
  "title": "MongoDB",
  "description": "database",
  "likes": 100,
  "url": "http://www.tutorialspoint.com/mongodb/",
  "by": "tutorials point"
}
```

## Update document

To update document from the collection, **update()** method of **com.mongodb.DBCollection** class is used.

Code snippets to select first document –

```
import com.mongodb.MongoClient;
import com.mongodb.MongoException;
import com.mongodb.WriteConcern;

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

import com.mongodb.ServerAddress;
import java.util.Arrays;

public class MongoDBJDBC {
```

```

public static void main( String args[] ) {

    try{

        // To connect to mongodb server
        MongoClient mongoClient = new MongoClient( "localhost" , 27017 );

        // Now connect to your databases
        DB db = mongoClient.getDB( "test" );
        System.out.println("Connect to database successfully");

        boolean auth = db.authenticate(myUserName, myPassword);
        System.out.println("Authentication: "+auth);

        DBCollection coll = db.getCollection("mycol");
        System.out.println("Collection mycol selected successfully");

        DBCursor cursor = coll.find();

        while (cursor.hasNext()) {
            DBObject updateDocument = cursor.next();
            updateDocument.put("likes","200")
            coll.update(updateDocument);
        }

        System.out.println("Document updated successfully");
        cursor = coll.find();

        int i = 1;

        while (cursor.hasNext()) {
            System.out.println("Updated Document: "+i);
            System.out.println(cursor.next());
            i++;
        }

    }catch(Exception e){
        System.err.println( e.getClass().getName() + ": " + e.getMessage() );
    }
}
}

```

When program is compiled and executed, it will produce the following result –

```

Connect to database successfully
Authentication: true
Collection mycol selected successfully
Document updated successfully
Updated Document: 1
{
  "_id" : ObjectId(7df78ad8902c),
  "title": "MongoDB",
  "description": "database",
  "likes": 100,
  "url": "http://www.tutorialspoint.com/mongodb/",
  "by": "tutorials point"
}

```

## Delete first document

To delete first document from the collection, you need to first select the documents using **findOne()** method and then **remove** method of **com.mongodb.DBCollection** class.

Code snippets to delete first document –

```

import com.mongodb.MongoClient;
import com.mongodb.MongoException;
import com.mongodb.WriteConcern;

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

import com.mongodb.ServerAddress;
import java.util.Arrays;

public class MongoDBJDBC {

    public static void main( String args[] ) {

        try{

            // To connect to mongodb server
            MongoClient mongoClient = new MongoClient( "localhost" , 27017 );

```

```
// Now connect to your databases
DB db = mongoClient.getDB( "test" );
System.out.println("Connect to database successfully");

boolean auth = db.authenticate(myUserName, myPassword);
System.out.println("Authentication: "+auth);

DBCollection coll = db.getCollection("mycol");
System.out.println("Collection mycol selected successfully");

DBObject myDoc = coll.findOne();
coll.remove(myDoc);
DBCursor cursor = coll.find();
int i = 1;

while (cursor.hasNext()) {
    System.out.println("Inserted Document: "+i);
    System.out.println(cursor.next());
    i++;
}

System.out.println("Document deleted successfully");
}catch(Exception e){
    System.err.println( e.getClass().getName() + ": " + e.getMessage() );
}
}
```

When program is compiled and executed, it will produce the following result –

```
Connect to database successfully
Authentication: true
Collection mycol selected successfully
Document deleted successfully
```

Remaining mongodb methods **save(), limit(), skip(), sort()** etc works same as explained in subsequent tutorial.

Advertisements



Write for us | FAQ's | Helping | Contact

© Copyright 2016. All Rights Reserved.

Enter email for newsletter

go