

# **BIBD Mini Project**

## **Topic: Implementation of MongoDB**

### **What is MongoDB?**

- MongoDB is a source-available cross-platform document-oriented database program.
- MongoDB is a NoSQL database.
- MongoDB stores data in flexible, JSON-like documents, meaning fields can vary from document to document and data structure can be changed over time
- A record in MongoDB is a document, which is a data structure composed of field and value pairs. MongoDB documents are similar to JSON objects. The values of fields may include other documents, arrays, and arrays of documents.

### **MongoDB Installation:**

Follow the official site installation guide according to your OS:

[link to download](#)

After installation type below commands to start and stop mongod server.

1. To start service

```
$ sudo systemctl start mongo.service
```

2. To stop service

```
$ sudo systemctl stop mongo.service
```

After starting the server we need to open mongo shell

```
$ mongosh
```

```

Activities  Terminal  Fri Apr 8, 20:34
mongosh mongodb://127.0.0.1:27017/?directConnection=true&serverSelectionTimeoutMS=2000
File Edit View Search Terminal Help
(base)
suraj at suraj-linux in Desktop on master [+!?]
$ sudo systemctl start mongod.service
[sudo] password for suraj:
(base)
suraj at suraj-linux in Desktop on master [+!?]
$ mongosh
Current Mongosh Log ID: 62504dc4b0e2f9a12c3cf340
Connecting to:
Using MongoDB: 4.4.1
Using Mongosh: 1.3.1
For mongosh info see: https://docs.mongodb.com/mongod-shell/

-----
The server generated these startup warnings when booting:
2022-04-08T20:05:54.988+05:30: ***** SERVER RESTARTED *****
2022-04-08T20:05:55.005+05:30: Using the XFS filesystem is strongly recommended with the WiredTiger storage engine. See http://dochub.mongodb.org/core/prodnotes-filesystem
2022-04-08T20:05:57.004+05:30: Access control is not enabled for the database. Read and write access to data and configuration is unrestricted
-----

```

How to create database:

```
$ use <database_name>
```

Eg: use mydbs

## How to show existing databases:

```
$ show dbs
```

Now, Your created database (mydb) is not present in the list. To display a database, you need to insert at least one document into it.

```
$ db.movies.insert({"name":"Interstellar"})
```

```
$ show dbs
```

Now our created database will be shown in the list

```
test> show dbs
admin    41 kB
config  12.3 kB
local   73.7 kB
test>

test> use mydb
switched to db mydb
mydb> show dbs
admin    41 kB
config  12.3 kB
local   73.7 kB
mydb> db.movies.insert({"name":"Coco-2017"})
DeprecationWarning: Collection.insert() is deprecated. Use insertOne, insertMany, or bulkWrite.
{
  acknowledged: true,
  insertedIds: { '0': ObjectId("62504ef14e37a041aa582e4e") }
}
mydb> show dbs
admin    41 kB
config  61.4 kB
local   73.7 kB
mydb    8.19 kB
mydb>
```

The screenshot shows a terminal window with a dark background. The left pane displays the MongoDB command prompt. The right pane shows a blurred background image with some text overlays, including "This Guy Shared Super Way to Earn \$100000 in a day!" and "Learn a language while you browse."

MongoDB db.dropDatabase() command is used to drop a existing database.

\$ db.dropDatabase() //this will drop current database

```
mydbs> show dbs
admin    41 kB
config  94.2 kB
local   73.7 kB
mydbs    41 kB

mydbs> db.dropDatabase()
{ ok: 1, dropped: 'mydbs' }

mydbs> show dbs
admin    41 kB
config  94.2 kB
local   73.7 kB
mydbs>
```

MongoDB db.createCollection(name, options) is used to create collections.

\$ db.createCollection("mycollection")

To display collection use

\$ show collection

```
mydbs> show collections

mydbs> db.createCollection("myCollection")
{ ok: 1 }

mydbs> show collection
MongoshInvalidInputError: [COMMON-10001] 'collection' is not a valid argument for "show".

mydbs> show collections
myCollection

mydbs>
```

MongoDB's db.collection.drop() is used to drop a collection from the database.

```
$ db.mycollection.drop()
```

```
mydbs> db.mycollection.drop()
true
mydbs> show collections
mydbs> 
```

## Inserting document into collection

### The insertOne() method

If you need to insert only one document into a collection you can use this method.

```
$ db.COLLECTION_NAME.insertOne(document)
```

```
mydbs> use Blogs
switched to db Blogs
Blogs> db.createCollection("POSTS")
{ ok: 1 }
Blogs> db.POSTS.insertOne(
... {
..... "author": "zayn",
..... "post": "new song",
..... "date": "2022-04-01"
..... })
Browserslist: caniuse-lite is outdated. Please run:
  npx browserslist@latest --update-db
  Why you should do it regularly: https://github.com/browserslist/browserslist#browsers-data-updating
{
  acknowledged: true,
  insertedId: ObjectId("62505ce84e37a041aa582e4f")
}
Blogs> 
```

## The insertMany() method

You can insert multiple documents using the insertMany() method. To this method you need to pass an array of documents.

```
$ db.COLLECTION_NAME.insertMany([document1, document2, ... ])
```

```
Blogs> db.POSTS.insertMany(
... [
... { "author": "harry",
...   "post": "new concert coming soon",
...   "date": "2022-04-02"
... },
... { "author": "louis",
...   "post": "welcome to my new show",
...   "date": "2022-04-03"
... }
... ]
{
  acknowledged: true,
  insertedIds: {
    '0': ObjectId("62505fc54e37a041aa582e50"),
    '1': ObjectId("62505fc54e37a041aa582e51")
  }
}
Blogs> 
```

## To display the inserted document into the collection use find() method

```
$ db.COLLECTION_NAME.find().pretty()
```

// pretty() method is used to display data in a neat way.

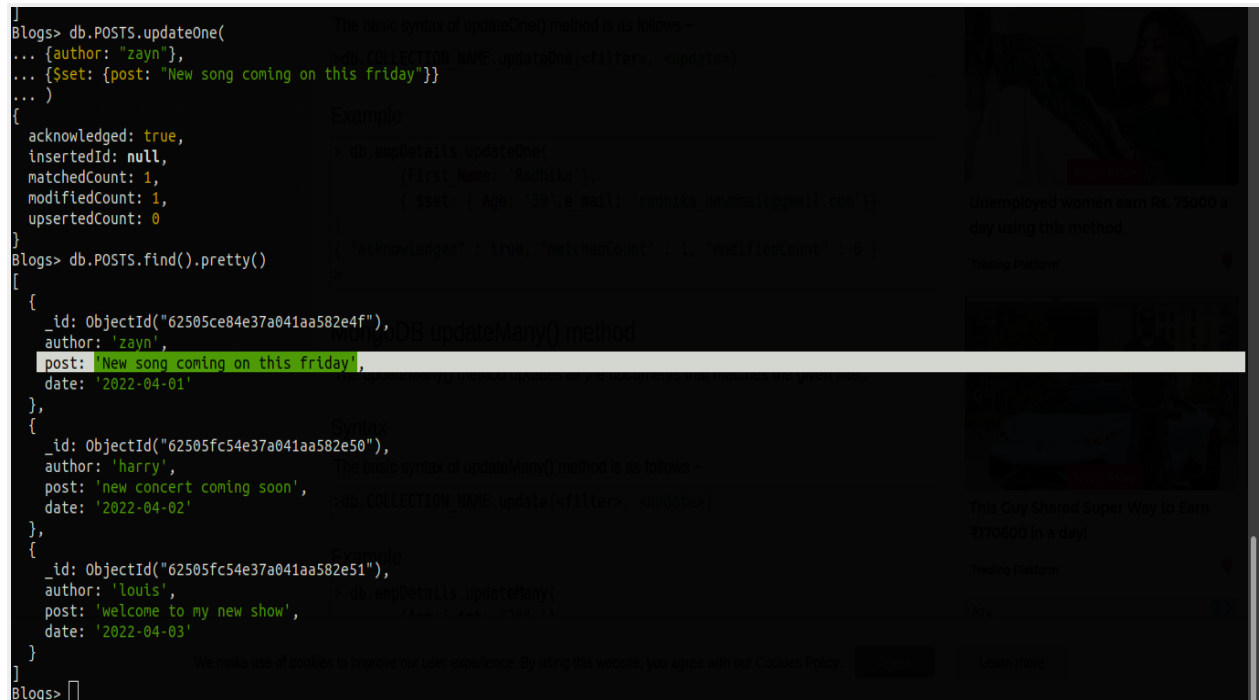
```
Blogs> db.POSTS.find().pretty()
Following example retrieves all the documents from the collection named mycol and
arranges them in an easy-to-read format.
[
  {
    "_id": ObjectId("62505ce84e37a041aa582e4f"),
    "author": "zayn",
    "post": "new song",
    "date": "2022-04-01"
  },
  {
    "_id": ObjectId("62505fc54e37a041aa582e50"),
    "author": "harry",
    "post": "new concert coming soon",
    "date": "2022-04-02"
  },
  {
    "_id": ObjectId("62505fc54e37a041aa582e51"),
    "author": "louis",
    "post": "welcome to my new show",
    "date": "2022-04-03"
  }
]
Blogs> 
```

## Updating the document

### MongoDB updateOne() method

This method updates a single document which matches the given filter.

```
$ db.COLLECTION_NAME.updateOne(<filter>, <update>)
```



```

]
Bloggs> db.POSTS.updateOne(
... {author: "zayn"},
... {$set: {post: "New song coming on this Friday"}}
... )
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
Bloggs> db.POSTS.find().pretty()
[
  {
    _id: ObjectId("62505ce84e37a041aa582e4f"),
    author: 'zayn',
    post: 'New song coming on this Friday',
    date: '2022-04-01'
  },
  {
    _id: ObjectId("62505fc54e37a041aa582e50"),
    author: 'harry',
    post: 'new concert coming soon',
    date: '2022-04-02'
  },
  {
    _id: ObjectId("62505fc54e37a041aa582e51"),
    author: 'louis',
    post: 'welcome to my new show',
    date: '2022-04-03'
  }
]
Bloggs>

```

### MongoDB updateMany() method

The updateMany() method updates all the documents that matches the given filter.

```
$ db.COLLECTION_NAME.updateMany(<filter>, <update>)
```

## Deleting the document

### The remove() Method

MongoDB's `remove()` method is used to remove a document from the collection. `remove()` method accepts two parameters. One is the deletion criteria and second is `justOne` flag.

- deletion criteria – (Optional) deletion criteria according to documents will be removed.
- `justOne` – (Optional) if set to `true` or `1`, then remove only one document.

### Remove Only One

If there are multiple records and you want to delete only the first record, then set `justOne` parameter in `remove()` method.

```
$ db.COLLECTION_NAME.remove(DELETION_CRITERIA,1)
```

```

]
Bloggs> db.POSTS.remove({"author": "Louis"}, 1)
DeprecationWarning: Collection.remove() is deprecated. Use deleteOne, deleteMany, findOneAndDelete, or bulkWrite.
{ acknowledged: true, deletedCount: 1 }
Bloggs> db.POSTS.find().pretty()
  db.COLLECTION_NAME.remove(DELETION_CRITERIA, 1)
[
  {
    _id: ObjectId("62505ce84e37a041aa582e4f"),
    author: 'zayn',
    post: 'New song coming on this friday',
    date: '2022-04-01'
  },
  {
    _id: ObjectId("62505fc54e37a041aa582e50"),
    author: 'harry',
    post: 'new concert coming soon',
    date: '2022-04-02'
  }
]
Bloggs>

```



### Remove All Documents

If you don't specify deletion criteria, then MongoDB will delete whole documents from the collection.

```
$ db.COLLECTION_NAME.remove({})
```

```
]
Blog> db.POSTS.remove({})
{ acknowledged: true, deletedCount: 2 }
Blog> db.POSTS.find().pretty()
Blog> []
```

```
systemctl start mongod
```