

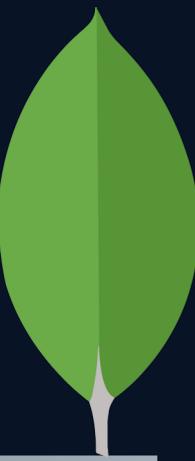


MONGOOSE

By Darshan Vasani
& Krupali Desai



Installation



```
1 npm init -y
2 npm install express mongoose
```



Connect To MongoDB With localhost



• • •

Connect MongoDB

```
// Require Module
const express = require('express');
const app = express();
const mongoose = require('mongoose');

mongoose.connect("mongodb://0.0.0.0:27017/dp", { useNewUrlParser: true, useUnifiedTopology: true })
  .then(() => {
    console.log("MongoDB Is Connected");
    app.listen(5000, () => {
      console.log("Server Is Running On Port 5000");
    });
  })
  .catch((error) => {
    console.log("MongoDB Is Not Connected");
    console.error(error);
  });
}
```





Schema With Validation



```
1  //Schema
2  // Validation Must Be In Schema Only
3  // Schema Define Structure of Document
4  const playlistSchema=new mongoose.Schema({
5      name:{
6          type:String,
7          unique:true,
8          required:true,
9          uppercase:true,
10         minlength:[2, "Length Should be more than 2"] //Custom Error
11     }
12     // properties: [Object],
13     // kind: 'minLength',
14     // path: 'name',
15     // value: 'A',
16     // reason: undefined,
17     // [Symbol(mongoose:validatorError)]: true
18   }
19 },
20 // _message: 'Playlist validation failed'
21 //}
22
23
24 },
25 ctype: {type:String
26     // enum:["frontend", "backend", "database"]
27     // ctype only frontend , backend , database
28 },
29 videos:{type:Number,
30     validate(value){
31         if(value<0){
32             throw new Error("Number Must Be Positive"); //User Define Validator -> Custom Validation
33             // videos: ValidatorError: Number Must Be Positive
34             // properties: [Object],
35             // kind: 'user defined',
36             // path: 'videos',
37             // value: -5,
38             // reason: Error: Number Must Be Positive
39         }
40     }
41 },
42 },
43 author:String,
44 email:{
45     type: String,
46     required:true,
47     unique:true,
48     // NPM Validator -> User Define Validation -> When validate is used their is always User Define Validation
49     validate(value){
50         if(!validator.isEmail(value)){
51             throw new Error("Email Is Invalid");
52         }
53     }
54 },
55 active:Boolean,
56 date:{
57     type:Date,
58     default:Date.now
59 }
60 })
```





Create



```
1 // Create Document
2 const Playlist = new mongoose.model("Playlist",playlistSchema);
3
4 const createDocument = async () =>{
5     //Create Document Or Insert
6     try{
7         const reactPlaylist = new Playlist({
8             name:"darshan",
9             ctype: "Front End",
10            videos:78,
11            author:"CryptoMinds",
12            active:true,
13        })
14
15        const nodePlaylist = new Playlist({
16            name:"rutvika",
17            ctype: "node",
18            videos:18,
19            author:"CryptoMinds",
20            active:true,
21        })
22
23        const mongPlaylist = new Playlist({
24            name:"ankita",
25            ctype: "mongo",
26            videos:50,
27            author:"CryptoMinds",
28            active:true,
29        })
30
31        const reduxPlaylist = new Playlist({
32            name:"krupali",
33            ctype: "redux",
34            videos:28,
35            author:"CryptoMinds",
36            active:true,
37        })
38
39        const nextPlaylist = new Playlist({
40            name:"harshil",
41            ctype: "next",
42            videos:38,
43            author:"CryptoMinds",
44            active:true,
45        })
46        // Always use try and catch with async function
47
48        // const result = await reactPlaylist.save(); For One Document
49        const result = await Playlist.insertMany([nextPlaylist,reduxPlaylist,mongPlaylist,nodePlaylist,reactPlaylist]);
50        console.log(result);
51    }catch(error){
52        console.log(error);
53    }
54
55 }
56 //Promise <<< Async [Better]
57 //Here I also done with Promise But it's too Difficult
58 createDocument();
59
```





Update



```
1 // Update
2 //Directly From MongoDB Compass
3 // Through Mongoose
4 const updateDocument = async (_id) => {
5   try {
6     // Update the document
7     // const result = await Playlist.updateOne({ _id }, { => Show Updated Document Count
8     const result = await Playlist.findByIdAndUpdate({ _id }, { // Show Updated Document
9       $set: {
10         name: "Dpvasan"
11       }
12     });
13
14     console.log(result);
15   } catch (err) {
16     console.log(err);
17   }
18 };
19
20 // Call the function with a valid _id
21 updateDocument("647d826e4e60d7f149677a19");
```



Delete



```
1 // Delete Document
2 const deleteDocument = async (_id) => {
3   try {
4     //const result = await Playlist.findByIdAndDelete({ _id }); => Show Deleted Document
5     const result = await Playlist.deleteOne({ _id }); //=>{ acknowledged: true, deletedCount: 0 } =>Count
6
7     console.log(result);
8   } catch (err) {
9     console.log(err);
10   }
11 };
12
13 // Call the function with a valid _id
14 deleteDocument("647d826e4e60d7f149677a19");
```



Read



```
1      // Read
2 const getDocument = async () =>{
3     const result = await Playlist.find();
4     console.log(result);
5 }
6
7
8 // Specific type
9 const getDocument = async () =>{
10    const result = await Playlist
11      .find({name:"darshan"})
12      .select({name:1})
13      .limit(1);
14    getDocument();
```



Comparison Query Operator



```
1 //Comparison Query Operator
2 //Greater Than 50;
3
4 const getDocument = async () =>{
5     const result = await Playlist
6     .find({videos:{$gt:50}});
7
8
9 //ctype:"Front End", "mongo"
10 const getDocument = async () =>{
11     const result = await Playlist
12     .find({ctype:{'$in':["Front End", "mongo"]}});
13
14
15
16 //Logical Operator With Counting
17 const getDocument = async () =>{
18     const result = await Playlist
19     .find({ $or: [ { ctype:"mongo"}, { author:"CryptoMinds"}] })
20     .count();
21
22
23 //Sorting
24 const getDocument = async () => {
25     const result = await Playlist
26     .find()
27     .sort({name:1});
28     console.log(result);
```



Official Documentation



mongoDB <https://www.mongodb.com/docs/manual/>

Mongoose

<https://mongoosejs.com/docs/guides.html>





Meet The Team



Darshan Vasani

Cofounder & CEO



[darshan-vasani-3299ba245/](https://www.linkedin.com/in/darshan-vasani-3299ba245/)



[dp_vasani56](https://www.instagram.com/dp_vasani56)



<https://dpvasani-57.netlify.app/>



Krupali Desai

Cofounder



[krupali-desai-17269a235](https://www.linkedin.com/in/krupali-desai-17269a235/)



[k_rdesai03](https://www.instagram.com/k_rdesai03)



<https://dkrupali-56.netlify.app>



Code Available Here
[dpvasani/MongoDB Tutorial](https://github.com/dpvasani/MongoDB-Tutorial)