

How to Connect Nodejs with MongoDB

We are using WebStorm IDE for this article.

To connect nodejs with mongodb run following command in Terminal

[npm install mongodb](#)

Creating a Database

To create a database in MongoDB, start by creating a MongoClient object, then specify a connection URL with the correct ip address and the name of the database you want to create.

MongoDB will create the database if it does not exist, and make a connection to it.

```
var MongoClient = require('mongodb').MongoClient;
var url = "mongodb://localhost:27017/mydb";
```

```
MongoClient.connect(url, function(err, db) {
  if (err) throw err;
  console.log("Database created!");
  db.close();
});
```

Creating a Collection

To create a collection in MongoDB, use the createCollection() method:

```
var MongoClient = require('mongodb').MongoClient;
var url = "mongodb://localhost:27017/";
```

```
MongoClient.connect(url, function(err, db) {
  if (err) throw err;
  var dbo = db.db("mydb");
  dbo.createCollection("customers", function(err, res) {
    if (err) throw err;
    console.log("Collection created!");
    db.close();
  });
});
```

Insert Into Collection

To insert a record, or document as it is called in MongoDB, into a collection, we use the insertOne() method.

A document in MongoDB is the same as a record in MySQL

The first parameter of the insertOne() method is an object containing the name(s) and value(s) of each field in the document you want to insert.

It also takes a callback function where you can work with any errors, or the result of the insertion:

```

var MongoClient = require('mongodb').MongoClient;
var url = "mongodb://localhost:27017/";

MongoClient.connect(url, function(err, db) {
  if (err) throw err;
  var dbo = db.db("mydb");
  var myobj = { name: "Company Inc", address: "Highway 37" };
  dbo.collection("customers").insertOne(myobj, function(err, res) {
    if (err) throw err;
    console.log("1 document inserted");
    db.close();
  });
});

```

Find One

To select data from a collection in MongoDB, we can use the `findOne()` method.

The `findOne()` method returns the first occurrence in the selection.

The first parameter of the `findOne()` method is a query object. In this example we use an empty query object, which selects all documents in a collection (but returns only the first document).

```

var MongoClient = require('mongodb').MongoClient;
var url = "mongodb://localhost:27017/";

MongoClient.connect(url, function(err, db) {
  if (err) throw err;
  var dbo = db.db("mydb");
  dbo.collection("customers").findOne({}, function(err, result) {
    if (err) throw err;
    console.log(result.name);
    db.close();
  });
});

```

Filter the Result

When finding documents in a collection, you can filter the result by using a query object.

The first argument of the `find()` method is a query object, and is used to limit the search.

```

var MongoClient = require('mongodb').MongoClient;
var url = "mongodb://localhost:27017/";

MongoClient.connect(url, function(err, db) {
  if (err) throw err;
  var dbo = db.db("mydb");
  var query = { address: "Park Lane 38" };
  dbo.collection("customers").find(query).toArray(function(err, result)
{

```

```
    if (err) throw err;
    console.log(result);
    db.close();
  });
});
```

Sort the Result

Use the sort() method to sort the result in ascending or descending order.

The sort() method takes one parameter, an object defining the sorting order.

```
var MongoClient = require('mongodb').MongoClient;
var url = "mongodb://localhost:27017/";
```

```
MongoClient.connect(url, function(err, db) {
  if (err) throw err;
  var dbo = db.db("mydb");
  var mysort = { name: 1 };
  dbo.collection("customers").find().sort(mysort).toArray(function(err,
result) {
    if (err) throw err;
    console.log(result);
    db.close();
  });
});
```

Delete Document

To delete a record, or document as it is called in MongoDB, we use the deleteOne() method.

The first parameter of the deleteOne() method is a query object defining which document to delete.

```
var MongoClient = require('mongodb').MongoClient;
var url = "mongodb://localhost:27017/";
```

```
MongoClient.connect(url, function(err, db) {
  if (err) throw err;
  var dbo = db.db("mydb");
  var myquery = { address: 'Mountain 21' };
  dbo.collection("customers").deleteOne(myquery, function(err, obj) {
    if (err) throw err;
    console.log("1 document deleted");
    db.close();
  });
});
```

Drop Collection

You can delete a table, or collection as it is called in MongoDB, by using the drop() method.

The drop() method takes a callback function containing the error object and the result parameter which returns true if the collection was dropped successfully, otherwise it returns false.

```
var MongoClient = require('mongodb').MongoClient;
var url = "mongodb://localhost:27017/";

MongoClient.connect(url, function(err, db) {
  if (err) throw err;
  var dbo = db.db("mydb");
  dbo.collection("customers").drop(function(err, delOK) {
    if (err) throw err;
    if (delOK) console.log("Collection deleted");
    db.close();
  });
});
```

Update Document

You can update a record, or document as it is called in MongoDB, by using the updateOne() method.

The first parameter of the updateOne() method is a query object defining which document to update.

```
var MongoClient = require('mongodb').MongoClient;
var url = "mongodb://127.0.0.1:27017/";

MongoClient.connect(url, function(err, db) {
  if (err) throw err;
  var dbo = db.db("mydb");
  var myquery = { address: "Valley 345" };
  var newvalues = { $set: {name: "Mickey", address: "Canyon 123" } };
  dbo.collection("customers").updateOne(myquery, newvalues, function(err, res) {
    if (err) throw err;
    console.log("1 document updated");
    db.close();
  });
});
```
