**Data Access in Node.js**

Node.js supports all kinds of databases no matter if it is a relational database or NoSQL database. However, NoSQL databases like Mongo DB are the best fit with Node.js.

The following table lists important relational databases and respective drivers.

| Relational Databases | Driver | NPM Command |
| --- | --- | --- |
| MS SQL Server | [mssql](https://www.npmjs.com/package/mssql) | npm install mssql |
| Oracle | [oracledb](https://www.npmjs.com/package/oracledb) | npm install oracledb |
| MySQL | [MySQL](https://www.npmjs.com/package/mysql) | npm install mysql |
| PostgreSQL | [pg](https://www.npmjs.com/package/pg) | npm install pg |
| SQLite | [node-sqlite3](https://github.com/mapbox/node-sqlite3) | npm install node-sqlite |

The following table lists important NoSQL databases and respective drives.

| NoSQL Databases | Driver | NPM Command |
| --- | --- | --- |
| MongoDB | [mongodb](https://www.npmjs.com/package/mongodb) | npm install mongodb |
| Cassandra | [cassandra-driver](https://www.npmjs.com/package/cassandra-driver) | npm install cassandra-driver |
| LevelDB | [leveldb](https://www.npmjs.com/package/leveldb) | npm install level levelup leveldown |
| RavenDB | [ravendb](https://www.npmjs.com/package/ravendb) | npm install ravendb |
| Neo4j | [neo4j](https://www.npmjs.com/package/neo4j) | npm install neo4j |
| Redis | [redis](https://npmjs.org/package/redis) | npm install redis |
| CouchDB | [nano](https://www.npmjs.com/package/nano) | npm install nano |

 Note:

The above database list is not limited. There are many other databases and drivers available to be used with Node.js. Also, there are many drivers available for each database. So, choose a driver carefully based on your need.

# Access SQL Server in Node.js

# In order to access MS SQL database, we need to install drivers for it. There are many drivers available for SQL server in NPM. We will use mssql driver here.

# *npm install mssql*

# Now, create server.js and write the following code.

var express = require('express');

var app = express();

app.get('/', function (req, res) {

var sql = require("mssql");

// config for your database

var config = {

user: 'sa',

password: 'mypassword',

server: 'localhost',

database: 'SchoolDB'

};

// connect to your database

sql.connect(config, function (err) {

if (err) console.log(err);

// create Request object

var request = new sql.Request();

// query to the database and get the records

request.query('select \* from Student', function (err, recordset) {

if (err) console.log(err)

// send records as a response

res.send(recordset);

});

});

});

var server = app.listen(5000, function () {

console.log('Server is running..');

});

# Access MongoDB in Node.js - In order to access MongoDB database, we need to install MongoDB drivers.

# npm install mongodb –save

# start the MongoDB server using the following command. (Assuming that your MongoDB database is at C:\MyNodeJSConsoleApp\MyMongoDB folder.)

# mongod -dbpath C:\MyNodeJSConsoleApp\MyMongoDB

## **Connecting MongoDB**

var MongoClient = require('mongodb').MongoClient;

// Connect to the db.

MongoClient.connect("mongodb://localhost:27017/MyDb",function (err, db) {

if(err) throw err;

//Write database Insert/Update/Query code here...

});

## **Insert Documents**

var MongoClient = require('mongodb').MongoClient;

// Connect to the db

MongoClient.connect("mongodb://localhost:27017/MyDb",function (err, db) {

db.collection('Persons', function (err, collection) {

collection.insert({id: 1, firstName: 'Steve', lastName: 'Jobs' });

collection.insert({ id: 2, firstName: 'Bill', lastName: 'Gates' });

collection.insert({ id: 3, firstName: 'James', lastName: 'Bond' });

db.collection('Persons').count(function (err, count) {

if (err) throw err;

console.log('Total Rows: ' + count);

});

});

});

Running the above example displays the following result.

 Node app.js

## **Update/Delete Documents**

var MongoClient = require('mongodb').MongoClient;

// Connect to the db

MongoClient.connect("mongodb://localhost:27017/MyDb",function (err, db) {

db.collection('Persons', function (err, collection) {

collection.update({id:1},{$set:{firstName:'James',lastName: 'Gosling'} }, {w:1},

function(err, result){

if(err) throw err;

console.log('Document Updated Successfully');

});

collection.remove({id:2}, {w:1}, function(err, result) {

if(err) throw err;

console.log('Document Removed Successfully');

});

});

});

## **Query Database**

var MongoClient = require('mongodb').MongoClient;

// Connect to the db

MongoClient.connect("mongodb://localhost:27017/MyDb",function(err,db) {

db.collection('Persons', function (err, collection) {

collection.find().toArray(function(err, items) {

if(err) throw err;

console.log(items);

});

});

});

## **Mongoose**

Mongoose is a very popular ODM for MongoDB in Node.js. Mongoose provides a straight-forward, schema-based solution to model your application data. It includes built-in type casting, validation, query building, business logic hooks and more.