Working with MongoDB and Node JS

A Day in the Life of a MEAN Stack Developer

In this sprint, Joe has to develop the frontend of the application using an Angular framework. Since he was recently trained in working on this framework, he has been chosen to complete the database integration with the application and he agrees to complete it by next week.

In this lesson, we will learn how to solve this real-world scenario to help Joe effectively complete his task.

Learning Objectives

By the end of this lesson, you will be able to:

- Establish connection between Node JS and MongoDB
- Implement basic operations with MongoDB
- Run Node JS with Mongoose

MongoDB with Node JS

Node JS and NoSQL Databases

Two of the major NoSQL databases are MySql and MongoDB which use mysql and mongoose modules respectively for database connectivity with different technologies.These modules help you to:

Manage connection pooling

Create and close database connection

Execute different database queries

Perform data manipulation language operations

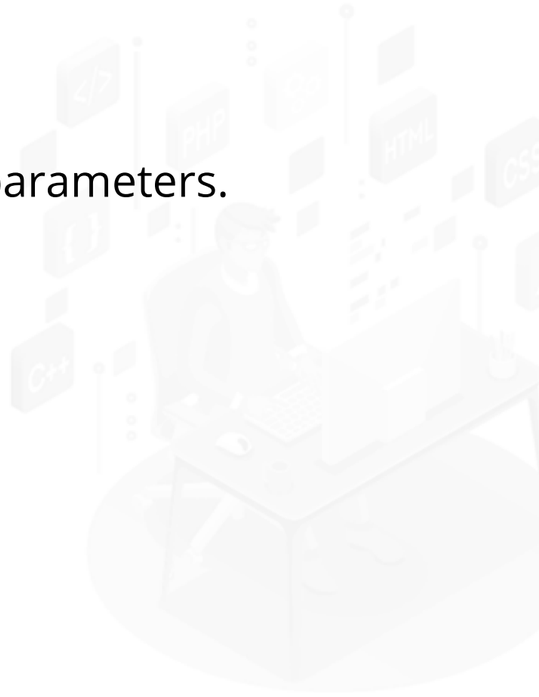
Setting Up MongoDB with Node JS

To set up MongoDB with our Node JS program, let us assume that we already have the database table created in your MongoDB database storage.

- Make sure that npm module is installed. If not, run the command given below:
  Npm install mongodb

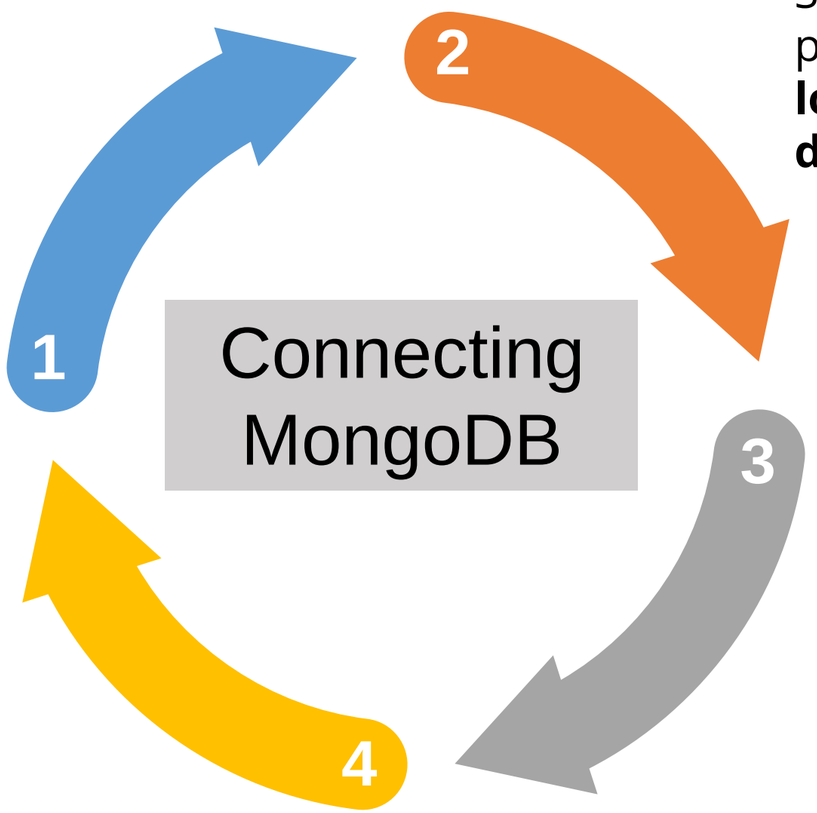
- Write the given script in your **.js** file for MongoDB connection and provide necessary parameters.

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

var url = 'mongodb://localhost/<databaseName>

MongoClient.connect(url, function(err, db){
        console.log("Connected);
});
```



MongoDB Connection Script Description

First statement includes the addition of **mongoose** module with the help of **require** method.



Second statement includes 3 parameters: **mongodb, localhost, and databaseName**

Third statement establishes connection with the database taking url and function to handle exception.

Fourth statement in the function includes a statement to check if the connection is successfully established.

Querying Data from MongoDB

Data can be retrieved from the database using MongoDB driver by creating cursor for the records and fetching data accordingly from each record.

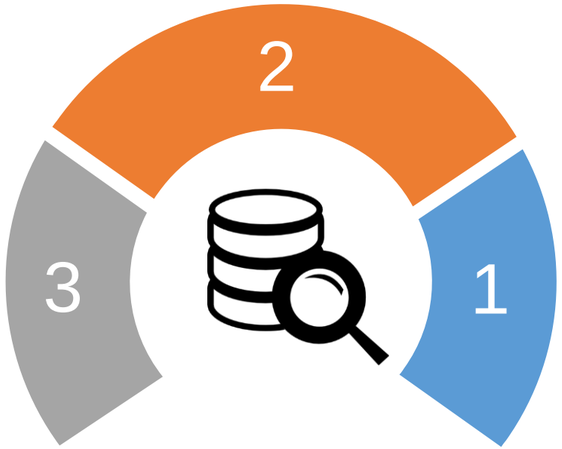
Below script can be used to fetch all records data from MongoDB database.

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

var url = 'mongodb://localhost/EmployeeDB';

MongoClient.connect(url, function(err, db)
{
        var cursor = db.collection('Employee').find();
        cursor.each(function(err, doc)
        {
                console.log(doc);
        });
});
```

Querying Script Description

Second statement iterates the cursor through every record to fetch the required data.

Third statement prints the records in the console using **console.log()** method.



First statement in connection method is used to create cursor to point the record that is to be retrieved

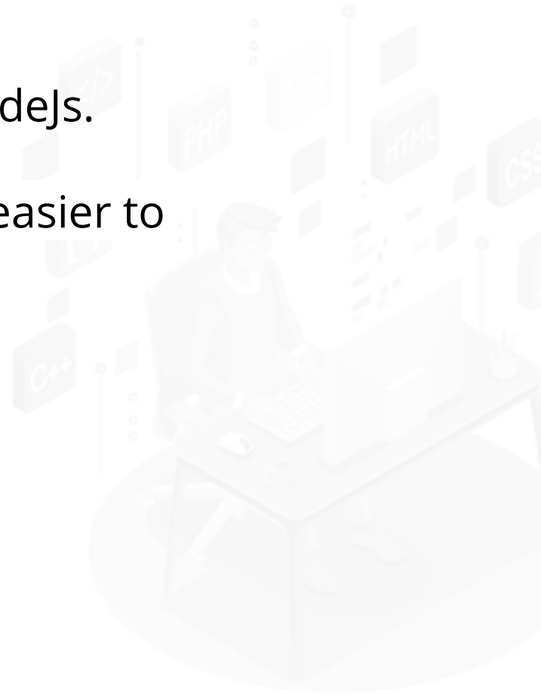
Note: One can also retrieve specific record from the database. To do so, **find()** method can be used along with condition parameters.

Below is the script for the same:

```
var cursor=db.collection('Employee').find({EmployeeName: “Ravi"});
```

Mongoose Node Module

- Mongoose is an Object Document Mapper (ODM). It allows to define objects with strongly-typed schema that is mapped to a MongoDB document.
- Mongoose is used to easily perform the integration between MongoDB and NodeJs.
- Mongoose.js also has utilities to simplify Node's callback patterns that make it easier to work with than the standard driver alone.
- To install mongoose on system, type *np install mongoose.*



MongoDB NodeJs Module

- The MongoDB NodeJs driver provides callback-based and promise-based interactions with MongoDB.

- This driver provides support for the shared CRUD API specification and the Server Discovery And Monitoring Specification (SDAM).

Perform CRUD and Insert Operations

Duration: 60 min.

Problem Statement:
You are given a project to implement CRUD operations with MongoDB database in Node JS.

ASSISTED PRACTICE

Assisted Practice: Guidelines to Perform CRUD Operations

1. Create a new NodeJS Project.
2. Install express-handlebars.
3. Create controller.
4. Create model.
5. Create view.
6. Push the code to GitHub repositories.

Using DB Cursor to Retrieve Documents

Duration: 45 min.

Problem Statement:
You are given a project to retrieve documents using DB cursor.

ASSISTED PRACTICE

Assisted Practice: Guidelines to Retrieve Documents

1. Create a new NodeJS project.
2. Install express-handlebars.
3. Create index.js file.
4. Push the code to GitHub repositories.

Key Takeaways

- There are two types of noSQL databases: MongoDB and mysql.
- MongoDB module is added using **require()** method to perform various database operations in MongoDB using Node JS.
- DB cursor is a more efficient way of working with records in MongoDB.

Connecting MongoDB with the Website

Duration: 30 min.

Problem Statement:

As a Full Stack Developer, you have to create a database connection with MongoDB using ExpressJS.

LESSON-END PROJECT

Before the Next Class

You should know:

- Advanced Node.js
- Express.js and Socket.io
- Operations performed on MongoDB
- MongoDB Aggregation
- Integration of MongoDB with Node.js

Chatlog

Duration: 240 min.

Project Objective:

As a MEAN Stack Developer, you have to develop an online chat portal and store the messages in the database.

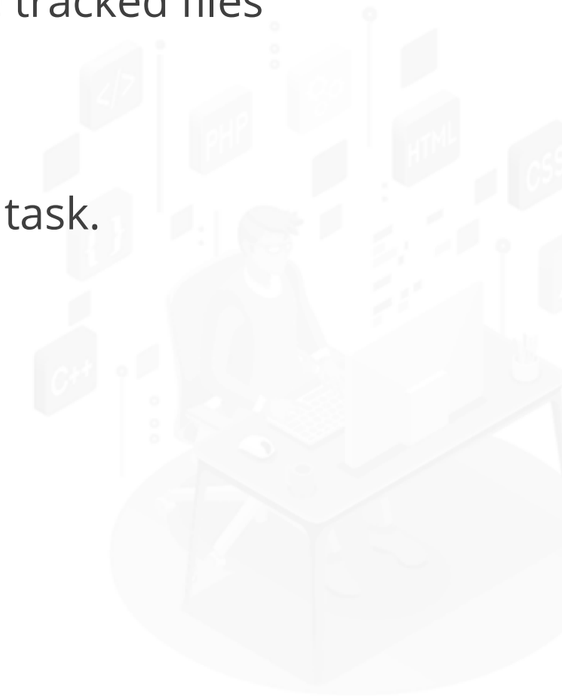
PHASE-END PROJECT

Background of the Project Statement

As the project is progressing, management has asked you to modify the chat application and connect it with a database so that the users can download or store the chat log.

You Are Asked to Do

- Source code should be tracked on GitHub repositories. You need to document the tracked files that are ignored during the final push to the GitHub repository.
- The submission of your GitHub repository link is mandatory in order to track your task.
- The step-by-step process involved in completing this task should be documented.



You must use the following

Visual Studio Code

Express.js

Git and GitHub

Node.js and Socket.io

MongoDB

Specification document