Beginner's Guide to MongoDb

Introduction

MongoDB is a popular NoSQL database, and Mongoose is an elegant Object Data Modeling (ODM) library for MongoDB and Node.js. In this tutorial, you'll learn how to set up MongoDB, create a database, configure Mongoose, and save a new user to the database using async/await in a Node.js application. Additionally, you'll use MongoDB Compass, a graphical interface, to interact with your MongoDB database.

Prerequisites

Before you begin, ensure you have the following prerequisites installed on your system:

Node.js and npm (Node Package Manager): You can download and install them from the official website: https://nodejs.org/

MongoDB: Install MongoDB following the official instructions: https://docs.mongodb.com/manual/installation/

MongoDB Compass: Download MongoDB Compass from the official website: https://www.mongodb.com/try/download/compass

Step 1: Start MongoDB

Open your terminal or command prompt.

Start the MongoDB server by running the following command:

```
mongod
```

This command launches the MongoDB server and listens for connections on the default port (27017).

Step 2: Download and Install MongoDB Compass

Download MongoDB Compass from the official website: https://www.mongodb.com/try/download/compass

Follow the installation instructions for your operating system to install MongoDB Compass.

Step 3: Create a MongoDB Database

Open MongoDB Compass after installation.

Click the "New Connection" button to create a new connection to your local MongoDB server. The connection details should already be filled in for a local connection.

Click the "Connect" button to establish a connection.

In the MongoDB Compass interface, you can create a new database by clicking the "Create Database" button. Enter a name for your database (e.g., myappdb) and click the "Create" button.

You've now created a new MongoDB database using MongoDB Compass.

Step 4: Set Up a Node.js Project

- Create a new directory for your Node.js project and navigate to it in your terminal:

```
mkdir mongoose-mongodb-tutorial
cd mongoose-mongodb-tutorial
```

- Initialize a new Node.js project by running the following command and following the prompts:

```
npm init
```

- Install the `mongoose` package, which allows you to work with MongoDB using JavaScript:

```
npm install mongoose
```

Step 5: Create a User Model

In your project directory, create a new JavaScript file (e.g., user.js) to define the User model using Mongoose:

```
// user.js
const mongoose = require('mongoose');

// Define the User schema
const userSchema = new mongoose.Schema({
  name: String,
  email: String,
  age: Number,
});

// Create the User model
const User = mongoose.model('User', userSchema);

module.exports = User;
```

Step 6: Create and Save a User

In this step, you will create a new user instance, save it to the MongoDB database using Mongoose's async/await capabilities, and handle any potential errors.

Step 6.1: Create a New User Instance

In this sub-step, we will create a new user instance with sample data.

- Open your project's app.js file, where you're setting up Mongoose and connecting to MongoDB.
- Inside the db.once('open', async () => { ... }) block, add the following code to create a new user instance:

```
// Create a new user instance
const newUser = new User({
  name: 'John Doe',
  email: 'john@example.com',
  age: 30,
});
```

Step 6.2: Save the User to the Database

In this sub-step, we will save the newly created user to the MongoDB database using the async/await syntax.

- Continue inside the db.once('open', async () => { ... }) block and add the following code to save the user to the database:

```
try {
  // Save the user to the database using async/await
  await newUser.save();

  console.log('User saved successfully');
} catch (error) {
  console.error('Error saving user:', error);
}
```

- We use a try...catch block to handle any potential errors that may occur during the save operation.
- await newUser.save() is used to save the user instance to the database. The await keyword ensures that the save operation completes before moving to the next line.
- If the user is successfully saved, a success message is logged to the console. If there's an error, it will be caught in the catch block, and an error message will be logged.

Step 6.3: Close the MongoDB Connection

In this sub-step, we will close the MongoDB connection after completing the database operation.

- To close the MongoDB connection after saving the user (inside the db.once('open', async () => { ... }) block), add the following code in the finally block:

```
// Close the MongoDB connection
mongoose.connection.close();
```

This ensures that the MongoDB connection is properly closed, whether the save operation succeeds or an error occurs.

Completed code for reference

```
// app.js
const mongoose = require('mongoose');
const User = require('./user'); // Replace './user' with the actual path to your Us

// Connect to MongoDB
mongoose.connect('mongodb://localhost:27017/your-database-name', {
  useNewUrlParser: true,
  useUnifiedTopology: true,
});

const db = mongoose.connection;

db.on('error', (error) => {
  console.error('MongoDB connection error:', error);
});
```

```
db.once('open', async () => {
  console.log('Connected to MongoDB');

  try {
    // Create a new user instance
    const newUser = new User({
      name: 'John Doe',
      email: 'john@example.com',
      age: 30,
    });

    // Save the user to the database using async/await
    await newUser.save();

    console.log('User saved successfully');
  } catch (error) {
    console.error('Error saving user:', error);
  } finally {
    // Close the MongoDB connection
    mongoose.connection.close();
  }
});
```

Make sure to replace 'mongodb://localhost:27017/your-database-name' with the actual MongoDB connection URL and database name.

Step 7: Use MongoDB Compass

- Open MongoDB Compass.
- Connect to your local MongoDB server by clicking the connection that you previously created.
- In the MongoDB Compass interface, you can explore your database, view collections, and query data.
- To view the saved user data, select your database from the left sidebar, then select the "Users" collection (created automatically based on the model name). You should see the user you added in Step 6.

Conclusion

In this tutorial, you've learned how to set up MongoDB, create a MongoDB database, configure Mongoose, and save a new user to the database using async/await in a Node.js application. Additionally, you've used MongoDB Compass to interact with your MongoDB database graphically. This combination of tools allows you to work efficiently with MongoDB and build powerful Node.js applications.

With this knowledge, you can continue to develop your MongoDB-backed applications, define additional models, and perform various CRUD operations using Mongoose and MongoDB Compass.