Mục tiêu:

1. Hiểu được ODM là gì?
2. Hiểu được vai trò và các tính năng mạnh mẽ của ODM Mongoose.
3. Áp dụng Mongoose vào việc kết nối và tạo model cho ứng dụng Node.js.

ODM là gì?

ODM (Object Data Mapping) là một kỹ thuật hoặc công cụ để ánh xạ giữa các đối tượng trong ứng dụng và các tài liệu trong cơ sở dữ liệu NoSQL. ODM đóng vai trò tương tự như ORM (Object Relational Mapping) nhưng dành cho cơ sở dữ liệu NoSQL thay vì cơ sở dữ liệu quan hệ.

Trong ngữ cảnh các cơ sở dữ liệu NoSQL như MongoDB, các tài liệu không có cấu trúc quan hệ cố định (ví dụ: không có bảng và cột giống như trong cơ sở dữ liệu quan hệ). ODM giúp quản lý việc ánh xạ và tương tác giữa các đối tượng JavaScript (hoặc của ngôn ngữ lập trình bạn đang sử dụng) và các tài liệu trong cơ sở dữ liệu NoSQL.

ODM thực hiện các nhiệm vụ sau:

- **Ánh xạ đối tượng:** Giúp ánh xạ các đối tượng trong mã nguồn của ứng dụng với các tài liệu (document) trong cơ sở dữ liệu NoSQL.
- **Quản lý truy vấn:** Cung cấp các phương thức đơn giản để thực hiện các truy vấn với cơ sở dữ liệu NoSQL (chẳng hạn như thêm, sửa, xóa, tìm kiếm dữ liệu) mà không cần phải viết trực tiếp truy vấn cơ sở dữ liệu gốc.
- **Xác thực dữ liệu:** ODM có thể cung cấp các cơ chế xác thực dữ liệu trước khi lưu chúng vào cơ sở dữ liệu, đảm bảo rằng dữ liệu đáp ứng đúng các yêu cầu nghiệp vụ.
- **Chuyển đổi dữ liệu:** Khi làm việc với cơ sở dữ liệu NoSQL, dữ liệu lưu trữ dưới dạng JSON hoặc BSON. ODM sẽ giúp chuyển đổi dữ liệu từ định dạng của cơ sở dữ liệu sang định dạng mà ứng dụng sử dụng (ví dụ: từ BSON sang đối tượng JavaScript).

Giới thiệu Mongoose

Khi làm việc với MongoDB, Mongoose là một ODM giúp lập mô hình dữ liệu MongoDB trong ứng dụng Node.js. Nó cung cấp các tính năng như định nghĩa schema, xác thực dữ liệu, và quản lý các kết nối đến cơ sở dữ liệu MongoDB.

Mongoose cung cấp các tính năng sau:

- **Định nghĩa schema:** Mongoose cho phép bạn định nghĩa schema cho các tài liệu trong cơ sở dữ liệu MongoDB. Schema giúp xác định cấu trúc dữ liệu, kiểu dữ liệu, và các ràng buộc dữ liệu.
- **Xác thực dữ liệu:** Mongoose cung cấp các cơ chế xác thực dữ liệu trước khi lưu vào cơ sở dữ liệu, giúp đảm bảo rằng dữ liệu đáp ứng đúng các yêu cầu nghiệp vụ.
- **Quản lý kết nối:** Mongoose giúp quản lý kết nối đến cơ sở dữ liệu MongoDB, giúp tạo, duy trì, và đóng kết nối một cách dễ dàng.
- **Thực thi truy vấn:** Mongoose cung cấp các phương thức đơn giản để thực hiện các truy vấn với cơ sở dữ liệu MongoDB mà không cần phải viết trực tiếp truy vấn cơ sở dữ liệu gốc.

- **Chuyển đổi dữ liệu:** Mongoose giúp chuyển đổi dữ liệu giữa định dạng của cơ sở dữ liệu MongoDB và định dạng mà ứng dụng sử dụng.
- **Middleware:** Mongoose hỗ trợ middleware để thực thi các hàm trước hoặc sau các sự kiện như lưu dữ liệu, xóa dữ liệu, hoặc tìm kiếm dữ liệu.
- **Populate:** Mongoose hỗ trợ populate để lấy dữ liệu từ các bảng liên quan trong cơ sở dữ liệu MongoDB.
- **Validation:** Mongoose hỗ trợ xác thực dữ liệu trước khi lưu vào cơ sở dữ liệu.
- **Query builder:** Mongoose hỗ trợ query builder giúp xây dựng các truy vấn phức tạp với cú pháp dễ đọc.
- **Plugins:** Mongoose hỗ trợ plugins để mở rộng tính năng của schema và model.
- **Virtuals:** Mongoose hỗ trợ virtuals để tạo các trường ảo trong schema.
- **Indexes:** Mongoose hỗ trợ indexes để tạo các chỉ mục cho các trường trong schema.

Cài đặt Mongoose và kết nối MongoDB với Mongoose

Bước 1: Cài đặt Mongoose

Để bắt đầu sử dụng Mongoose, bạn cần cài đặt thư viện Mongoose thông qua npm:

```
npm install mongoose
```

Bước 2: Kết nối MongoDB với Mongoose

```
import express from "express";
import mongoose from "mongoose";

const app = express();

app.use(express.json());

const mongoDBUrl = "mongodb://localhost:27017/mydatabase";

mongoose
    .connect(mongoDBUrl, {
        // Thêm các tùy chọn cấu hình kết nối nếu cần
    })
    .then(() => {
        console.log("Kết nối MongoDB thành công!");
    })
    .catch((err) => {
        console.error("Lỗi kết nối MongoDB:", err);
    });

app.post("/users", (req, res) => {
    const user = req.body;
    users.push(user);
    res.json(user);
});
```

```
app.listen(3000, () => {
    console.log("Server is running on port 3000");
});
```

Trong đó:

- `mongoose.connect(mongoDBUrl, options)`: Phương thức này trả về một Promise được sử dụng để kết nối với cơ sở dữ liệu MongoDB. Trong đó, `mongoDBUrl` là URL của cơ sở dữ liệu MongoDB, `options` là các tùy chọn cấu hình cho kết nối.

Tạo Model với Mongoose

Chúng ta đã tạo ra cấu trúc thư mục theo mô hình MVC trong bài học trước. Trong thư mục `models`, chúng ta sẽ tạo các file model để định nghĩa schema và model cho các tài liệu trong cơ sở dữ liệu MongoDB.

Giả sử với một ứng dụng ecommerce, chúng ta có đối tượng product, chúng ta tạo file productModel.js trong models để định nghĩa schema và model cho sản phẩm:

```
import mongoose from "mongoose";

const productSchema = new mongoose.Schema(
    {
        name: {
            type: String,
            required: true,
        },
        price: {
            type: Number,
            required: true,
        },
        description: {
            type: String,
        },
        shortDesscription: {
            type: String,
        },
        images: {
            type: Array,
        },
        category: {
            type: String,
        },
        quantity: {
            type: Number,
        },
        thumbnail: {
            type: String,
        },
    },
```

```
    {
        timestamps: true,
        versionKey: false,
    }
);

export default mongoose.model("Product", productSchema);
```

Như vậy chúng ta đã sử dụng mongoose để kết nối Node.js với cơ sở dữ liệu MongoDB và tạo model cho sản phẩm. Ở phần tiếp theo chúng ta sẽ viết Controller để thực hiện các thao tác CRUD với sản phẩm.

Kiến thức cần nhớ

1. **ODM (Object Data Mapping)** là một kỹ thuật hoặc công cụ để ánh xạ giữa các đối tượng trong ứng dụng và các tài liệu trong cơ sở dữ liệu NoSQL.
2. **Mongoose** là một ODM giúp lập mô hình dữ liệu MongoDB trong ứng dụng Node.js.
3. **Mongoose cung cấp nhiều tính năng mạnh mẽ như:** định nghĩa schema, xác thực dữ liệu, quản lý kết nối, thực thi truy vấn, chuyển đổi dữ liệu, middleware, populate, validation, query builder, plugins, virtuals, indexes...