Mongoose

Mongoose представляє спеціальну ODM-бібліотеку (Object Data Modelling) для роботи з MongoDB, яка дозволяє зіставляти об'єкти класів та документи колекцій із бази даних. Грубо кажучи, Mongoose працює подібно до інструментів ORM. Офіційний сайт бібліотеки, де можна переглянути всю необхідну документацію: http://mongoosejs.com

Для роботи з Mongoose продовжимо роботу з проектом з минулих тем і спочатку встановимо бібліотеку за допомогою команди:

```
npm install mongoose --save
```

Далі визначимо наступний код у файлі програми **app.js** :

```
const mongoose = require("mongoose");
const Schema = mongoose.Schema;

// установка схемы
const userScheme = new Schema({
    name: String,
    age: Number
});
// определяем модель User
const User = mongoose.model("User", userScheme);
// создаем объект модели User
const user = new User({ name: "Bill", age: 41});

async function main() {
    // подключемся к базе данных
    await mongoose.connect("mongodb://127.0.0.1:27017/usersdb");

    // сохраняем модель user в базу данных
    await user.save();
    console.log("Сохранен объект", user);

    // отключаемся от базы данных
```

```
    await mongoose.disconnect();
}
// запускаем подключение и взаимодействие с базой данных
main().catch(console.log);
```

Тут передусім нам треба підключити mongoose:

```
const mongoose = require("mongoose");
```

Данные, которые используются в Mongoose, описываются определенной схемой. Например, в прошлых темах мы сохраняли в базу данных объекты с двумя свойствами `name` и `age`. Поэтому описываем здесь следующую схему:

```
const Schema = mongoose.Schema;

// установка схемы
const userScheme = new Schema({
    name: String,
    age: Number
});
```

Схема содержит метаданные объектов. В частности, здесь устанавливаем, какие свойства будет иметь объект и какой у них будет тип данных. То есть это схема, которая описывает объект пользователя.

Затем, используя эту схему, создаем модель пользователя:

```
const User = mongoose.model("User", userScheme);
```

Первый параметр в методе **mongoose.model** указывает на название модели. Mongoose затем будет автоматически искать в базе данных коллекцию, название которой соответствует названию модели во множественном числе. Например, в данном случае название модели "User". Во множественном числе в соответствии с правилами английского языка это "users". Поэтому при работе с данными модели User (добавлении, удалении, редактировании и получении объектов) mongoose будет обращаться к коллекции "users". Если такая коллекция есть в бд, то с ней будет идти взаимодействие. Если такой коллекции в базе данных нет, то она будет создана автоматически.

Второй параметр функции `mongoose.model` - собственно схема.

Далее мы можем создавать объекты этой модели:

```
const user = new User({
    name: "Bill",
    age: 41
});
```

Основная логика сосредоточена в асинхронной функции main. И чтобы работать с бд MongoDB, необходимо к ней подключиться. Для подключения у объекта mongoose вызывается метод **mongoose.connect()**, в который передается адрес базы данных на сервере mongo:

```
await mongoose.connect("mongodb://127.0.0.1:27017/usersdb");
```

Этот метод возвращает объект Promise, поэтому при вызове метода в асинхронном методе можно применить оператор `await`.

Затем у объекта вызывается метод **save**. Этот метод определен для всех создаваемых моделей, он сохраняет текущий объект в базу данных:

```
await user.save();
console.log("Сохранен объект", user);
```

С помощью метода **mongoose.disconnect();** происходит отключение от бд.

Так как метод save возвращает promise, то есть и другая форма сохранения объекта:

```
await mongoose.disconnect();
});
```

Запустим приложение и выполним добавление объекта:

```
Сохранен объект {
  name: 'Bill',
  age: 41,
  _id: new ObjectId("6377c17b71c0bd75cec4d488"),
  __v: 0
}
```

Назад **Зміст** **Вперед**