MongoDB ODM Mongoose

Andrés Muñoz

Alejandro Pozo

Enrique Barra

Contenidos

- Introducción
- Crear Conexión
- Schema
  - Crear Schema
  - Generar Modelo
- Insert
  - Crear Documento
  - Insertar un Documento
  - Insertar varios documentos
- Find
  - Recorrer una Colección
  - Buscar registros en concreto
- Update
  - Actualizar documentos
- Delete
  - Borrar Documentos
- Demo

Introducción

- Para conectar con una base de datos MongoDB desde un programa hecho en Node hace falta un driver, librería o un driver ODM que entienda el protocolo de la base de datos y se comunique con ella.
- Y así me permita escribir sentencias de mi lenguaje de programación y no tener que escribir queries de la base de datos en concreto y desarrollarme el protocolo concreto
- En este caso se el ODM que vamos a utilizar se llama Mongoose

Introducción

- Versiones :
  - Una síncrona, en la que el programa espera a que la bbdd devuelva los resultados
  - Una asíncrona, en la que el programa sigue haciendo cosas y hay un callback que se ejecuta cuando la bbdd devuelve los resultados. Esta es más eficiente pero a la vez más compleja de programar.
- Utilizaremos la asíncrona

- Necesitaremos instalar la dependencia con npm:
  - npm install mongoose

- Documentación y más: https://mongoosejs.com/docs/index.html

Crear una conexión

- Arrancar previamente la base de datos

```
const mongoose = require('mongoose');
mongoose.connect('mongodb://localhost/test',
{useNewUrlParser: true});
```

- Obtenemos información de si la conexión se ha establecido correctamente

```
const db = mongoose.connection; db.on('error',
console.error.bind(console, 'connection error:'));
db.once('open', function() { // we're connected! });
```

JavaDoc: https://mongoosejs.com/docs/connections.html

Schema

Crear el Schema

- En mongoose siempre se parte de un Schema:

```
{
  "name" : "MongoDB",
  "type" : "database",
  "count" : 1,
  "versions": [ "v3.2", "v3.0", "v2.6" ],
  "info" : { x : 203, y : 102 }
}
```

- Creamos el Schema con la estructura del anterior JSON

```
const DatabaseSchema = new mongoose.Schema(
{ name : String,
  type : String,
  count : Number,
  versions : Array,
  info : { x : Number, y : Number }
});
```

Generar el Modelo

- Definimos el modelo para usarlo posteriormente para la operaciones CRUD

```
const Database = mongoose.model('Database', DatabaseSchema);
```

Insert

Crear (la C del CRUD)

Crear el Documento

- Json con info del documento a insertar

```
{
  "name" : "MongoDB",
  "type" : "database",
  "count" : 1,
  "versions": [ "v3.2", "v3.0", "v2.6" ],
  "info" : { x : 203, y : 102 }
}
```

- Para crear documentos se usa el modelo previamente creado:

```
var newDoc = new Database(
{ "name" : "MongoDB",
  "type" : "database",
  "count" : 1,
  "versions": [ "v3.2", "v3.0", "v2.6" ],
  "info" : { x : 203, y : 102 }
});
```

Insertar un Documento

- Definimos una función asíncrona para insertar el documento definido

```
async function insert(newDoc) {
   let result= await newDoc.save();
   return result;
}
```

- Como siempre, si no tiene campo _id MongoDB lo asignará automáticamente

Insertar varios Documentos

- Si se tienen varios documentos:

```
var miltiDoc = [
        { "name" : "CouchDB",
          "type" : "database",
          "count" : 2,
          "versions" : [ "v0.1", "v0.2", "v0.6" ],
          "info" : { x : 800, y : 366 }
        },
        { "name" : "Mysql",
          "type" : "database",
          "count" : 1,
          "versions" : [ "v3", "v4", "v5" ],
          "info" : { x : 10, y : 14 }
        }
];
```

- Se los añade usando Model.insertMany():

```
async function insertarMulti(multiDoc) {
  let result= await Database.insertMany(multiDoc);
  return result;
}
```

Find

Buscar - Read (la R del CRUD)

Recorrer una colección

- El método Model.find() devuelve una lista con todos los documentos de esa collection

```
async function buscar() {
  let result= await Database.find();
  return result;
}
```

Buscar registros en concreto

- Se utiliza find() pero pasándole filtros.
- Información sobre los filtros: https://docs.mongodb.com/manual/reference/operator/query/
- Ejemplos:
  - Buscar el que tenga nombre MongoDB e imprimirlo

```
async function buscarPorCampo() {
  let result= await Database.find(name:"MongoDB");
  console.log(result);
  return result;
}
```

Update

Actualizar - Update (la U del CRUD)

Actualizar documentos

- Buscar el registro por campo que queremos y luego indicarle los campos que vamos a actualizar

```
async function actualizar() {
  let result= await Database.findOneAndUpdate(
  {_id: "MongoDB"}, {count: 2}, { new: true });
  return result;
}
```

Delete

Borrar - Delete (la D del CRUD)

Borrar documentos

- Borrar uno:

```
async function eliminar() {
  let result = await Database.deleteOne({name: "MongoDB"});
  return result;
}
```

- Borrar varios:

```
async function eliminarVarios() {
   let result = await Database.deleteMany({count:3});
}
```

Demo

MongoDB ODM Mongoose

Andrés Muñoz
Alejandro Pozo
Enrique Barra