Node.js

Parte III

Javier Miguel

- CTO & Freelance Developer
- Email: jamg44@gmail.com
- Twitter: @javiermiguelg

Bases de datos

Bases de datos

Node.js, a través de módulos de terceros, se puede conectar casi con cualquier base de datos del mercado.

http://expressjs.com/en/guide/database-integration.html

Bases de datos

Basta con cargar el driver (módulo) adecuado y establecer la conexión.

```
$ npm install mysql
```

```
$ npm install mongoskin
```

MySQL

Bases de datos - MySQL

Por ejemplo con MySQL:

```
var mysql      = require('mysql');
var connection = mysql.createConnection({
   host     : 'localhost',
   user     : 'usuario',
   password : 'pass',
   database : 'cursonode'
});

connection.connect(); // callback opcional

connection.query('SELECT * from agentes', function(err, rows, fields) {
   if (err) throw err;
   console.log(rows);
});
```

ejemplos/db/mysql

Bases de datos

Para refrescar conceptos de SQL:

https://www.sqlteaching.com/

Bases de datos - SQL ORMs

Un ORM (Object Relational Mapping) se encarga principalmente de:

- Convertir objetos en consultas SQL para que puedan ser persistidos en una base de datos relacional.

- Traducir los resultados de una consulta SQL y generar objetos.

Esto nos resultará útil si el diseño de nuestra aplicación es orientado a objetos (OOP).

Bases de datos - SQL ORMs

Un ORM muy usado para bases de datos SQL es sequelize:

http://docs.sequelizejs.com/en/latest/

Sequelize tiene soporte para MySQL, MariaDB, SQLite, PostgreSQL y MSSQL.

Otras buenas alternativas son Knex y Bookshelf.

MongoDB

Bases de datos - MongoDB

MongoDB en una base de datos no relacional sin esquemas, esto significa principalmente que:

- No tenemos JOIN, tendremos que hacerlo nosotros
- Cada registro podría tener una estructura distinta
- Mínimo soporte a transacciones

A la hora de decidir que base de datos usar para una aplicación debemos pensar como vamos a organizar los datos para saber si nos conviene usar una base de datos relacional o no relacional.

Bases de datos - MongoDB

Usar una base de datos como MongoDB puede darnos más rendimiento principalmente por alguna de estas razones:

- No tiene que gestionar transacciones
- No tiene que gestionar relaciones
- No es necesario convertir objetos a tablas y tablas a objetos (Object-relation Impedance Mismatch)

Bases de datos - MongoDB shell basics

Para acceder a la shell usaremos:

```
~/master/cursonode/mongodb-server/bin/mongo
MongoDB shell version: 3.0.4
connecting to: test
>
```

Bases de datos - MongoDB shell basics

```
show dbs
use <dbname>
show collections
show users
db.agentes.find().pretty()
db.agentes.insert({name: "Brown", age: 37})
db.agentes.remove({_id: ObjectId("55ead8899123383864857 0dd")})
db.agentes.update({_id: ObjectId("55eadb4191233838648570de")}, {$set: {age: 38}})
——————————————————————————————————————————— cuidado con el $set! ——

db.coleccion.drop()
db.agentes.createIndex({name:1, age:-1})
db.agentes.getIndexes()
```

Mas operaciones en la referencia rápida a la shell de MongoDB

Bases de datos - MongoDB queries

```
db.agentes.find({ name : 'Smith'})
db.agentes.find({ _id : ObjectId("55eadb41912338386485 70de")})
db.agentes.find({ age: { $gt: 30}}) // $lt, $gte, $lte, ...
db.agentes.find({ age: { $gt: 30, $lt: 40}});
db.agentes.find({ name: { $in: [ 'Jones', 'Brown']}}) //$nin
db.agentes.find({ name: 'Smith', $or: [
  { age: { $lt: 30}},
  { age: 43 } // 'Smith' and ( age < 30 or age = 43)
] })
```

Bases de datos - MongoDB queries

```
// subdocuments
db.agentes.find({ 'producer.company': 'ACME'})

// arrays
db.agentes.find({ bytes: [ 5, 8, 9 ]}) // array exact
db.agentes.find({ bytes: 5}) // array contain
db.agentes.find({ 'bytes.0': 5}) // array position
```

http://docs.mongodb.org/manual/reference/method/db.collection.find/#db.collection.find

http://docs.mongodb.org/manual/tutorial/query-documents/

Bases de datos - MongoDB queries

Ordenar:

```
db.agentes.find().sort({age: -1})
```

Descartar resultados:

```
db.agentes.find().skip(1).limit(1)
db.agentes.findOne({name: 'Brown'}) // igual a limit(1)
```

Contar:

```
db.agentes.find().count() // db.agentes.count()
```

Bases de datos - MongoDB queries

Full Text Search

Crear índice por los campos de texto involucrados:

```
db.agentes.createIndex({title: 'text', lead: 'text', body: 'text'});
```

Para hacer la búsqueda usar:

```
db.agentes.find({$text:{$search:'smith jones'});
```

Bases de datos - MongoDB queries

Full Text Search

Frase exacta:

```
db.agentes.find({$text:{$search:'smith jones "el elegido"'}});
```

Excluir un término:

```
db.agentes.find({$text:{$search:'smith jones -mister'});
```

Más info:

https://docs.mongodb.com/v3.2/text-search/

https://docs.mongodb.com/v3.2/tutorial/specify-language-for-text-index/

Bases de datos - MongoDB transacción

`findAndModify` es una operación atómica, lo que nos dará un pequeño respiro transaccional.

```
db.agentes.findAndModify({
    query: { name: "Brown"},
    update: { $inc: { age: 1}}
})
```

Lo busca y si lo encuentra lo modifica, no permitiendo que otro lo cambie antes de modificarlo.

Bases de datos - MongoDB

Ejemplo de uso MongoDB:

```
$ npm install mongodb
```

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

client.connect('mongodb://localhost:27017/cursonode',
function(err, db) {
  if (err) throw err;
  db.collection('agentes').find({}).toArray(function(err, docs) {
    if (err) throw err;
    console.dir(docs);
    db.close();
  });
});
```

ejemplos/db/mongodb

Mongoose

Mongoose

Mongoose es una herramienta que nos permite persistir objetos en MongoDB, recuperarlos y mantener esquemas de estos fácilmente.

Este tipo de herramientas suelen denominarse ODM (Object Document Mapper).

Mongoose

Instalación como siempre:

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

Mongoose

Conectar a la base de datos:

```
var mongoose = require('mongoose');
var conn = mongoose.connection;

conn.on('error', console.error.bind(console, 'mongodb connection
error:'));
conn.once('open', function() {
    console.info('Connected to mongodb.');
});

mongoose.connect('mongodb://localhost/diccionario');?
```

Mongoose

Crear un modelo:

```
var mongoose = require('mongoose');

var agenteSchema = mongoose.Schema({
    name: String,
    age: Number
});

mongoose.model('Agente', agenteSchema);
```

Mongoose

Guardar un registro:

```
var agente = new Agente({name: 'Smith', age: 43});

agente.save(function (err, agenteCreado) {
    if (err) throw err;
    console.log('Agente ' + agenteCreado.name + ' creado');
});
```

Mongoose

Eliminar registros:

```
Agente.remove({ [filters] }, function(err) {
    if (err) return cb(err);
    cb(null);
});
```

Mongoose

Crear un método estático a un modelo:

```
agenteSchema.statics.deleteAll = function(cb) {
    Agente.remove({}, function(err) {
        if (err) return cb(err);
        cb(null);
    });
};
```

Mongoose

Crear un método de instancia a un modelo:

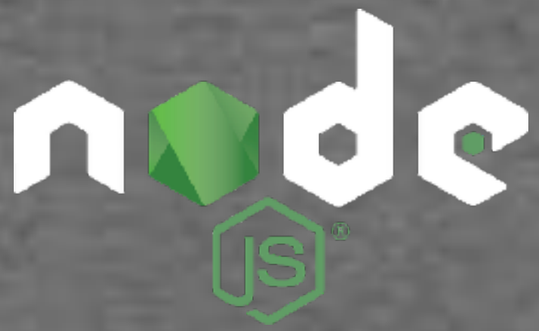
```
agenteSchema.methods.findSimilarAges = function (cb) {
  return this.model('Agente').find({ age: this.age }, cb);
}
```

Mongoose

Listando registros:

```
agenteSchema.statics.list = function(cb) {
  var query = Agente.find({});
  query.sort('name');
  query.skip(500);
  query.limit(100);
  query.select('name age');
  return query.exec(function(err, rows) {
      if (err) { return cb(err);}
      return cb(null, rows);
  });
});
```

ENHORABUENA!



Express.js