MongoDB and Mongoose

Software Engineering, Tutorial

Antonio Bucchiarone - bucchiarone@fbk.eu

Academic year 2022/2023

Contents of today class

- JSON
- NodeJS and NPM
- MongoDB and Mongoose

> Material: https://github.com/antbucc/IS-22_23

JSON

- JSON stands for *J*ava*S*cript *O*bject *N*otation
- JSON is a text format for storing and transporting data
- JSON is "self-describing" and easy to understand

```
{"name":"John", "age":30, "car":null}
```

- object with 3 properties: name, age, car

JSON Values

In JSON, values must be one of the following data types:

- a string
- a number
- an object
- an array
- a boolean
- null

JSON Examples

```
{"name":"John"}
```

```
{"age":30}
```

```
{"employee":{"name":"John", "age":30, "city":"New York"}}
```

```
{"employees":["John", "Anna", "Peter"]}
```

```
{"sale":true}
```

```
{"middlename":null}
```

JSON.parse()

- A common use of JSON is to exchange data to/from a web server.
- When receiving data from a web server, the data is always a string.
- Parse the data with JSON.parse(), and the data becomes a JavaScript object.

```
{"name":"John", "age":30, "city":"New York"}
```

```
const obj = JSON.parse('{"name":"John", "age":30, "city":"New York"}');
```

- Example 1: Create an Object from a JSON String
- Example 2: Convert a string into a date object.
- Example 3: Access Array Values

NodeJS - https://nodejs.org/

- Node.js is an **open-source** and **cross-platform** JavaScript runtime environment.
- Node.js runs the **V8 JavaScript engine**, the core of Google Chrome, outside of the browser. This allows Node.js to be very performant.
- When Node.js performs an **I/O operation**, like reading from the network, **accessing a database** or the filesystem, instead of blocking the thread and wasting CPU cycles waiting, Node.js will resume the operations when the response comes back.
- This allows Node.js to handle thousands of **concurrent connections** with a **single server** without introducing the burden of managing thread concurrency, which could be a significant source of bugs.

npm - https://www.npmjs.com/

- **npm** with its simple structure helped the ecosystem of Node.js proliferate, and now the npm registry hosts over 1,000,000 open source packages you can freely use.

```
npm install -g npm
```

Checking your version of npm and Node.js

```
node -v
```

```
npm -v
```

NodeJS - Hello World

- An asynchronous event-driven JavaScript runtime,

  `node server.js`

```
const http = require('http');

const hostname = '127.0.0.1';
const port = 3000;

const server = http.createServer((req, res) => {
    res.statusCode = 200;
    res.setHeader('Content-Type', 'text/plain');
    res.end('Hello World');
});

server.listen(port, hostname, () => {
    console.log(`Server running at http://${hostname}:${port}/`);
});
```

MongoDB - mongodb.com

> https://www.mongodb.com/en-us/what-is-mongodb

- MongoDB stores data in flexible, JSON-like documents, meaning fields can vary from document to document and data structure can be changed over time
- The document model maps to the objects in your application code, making data easy to work with
- Ad hoc queries, indexing, and real time aggregation provide powerful ways to access and analyze your data
- MongoDB is a distributed database at its core, so high availability, horizontal scaling, and geographic distribution are built in and easy to use
- MongoDB is free to use.

Getting Started

> https://www.mongodb.com/docs/guides/server/introduction/

- Define Your Data Set
- Start Thinking in JSON
- Identify Candidates for Embedded Data and Model Your Data

```
{ "name": "notebook",
  "qty": 50,
  "rating": [ { "score": 8 }, { "score": 9 } ],
  "size": { "height": 11, "width": 8.5, "unit": "in" },
  "status": "A",
  "tags": [ "college-ruled", "perforated"]
}
```

Get MongoDB

- Install MongoDB locally - www.mongodb.com/try/download/community
  - Tutorial https://www.mongodb.com/docs/guides/server/install/
- Use MongoDB as a service - cloud.mongodb.com
- Develop on codesandbox.io or replit.com

MongoDB as a service - cloud.mongodb.com

- Register on cloud.mongodb.com
- Create a new project
- Build a Database (Free version)
    - Setup username and password used to connect db
- Go to Network Access -> Add IP adress -> Allow Access from Anywhere
- Go back on 'Datbase' Click and click on 'Connect' to get connection details.

> Replace <password> with the password for the admin user. Replace myFirstDatabase with the name of the database that connections will use by default. Ensure any option params are URL encoded.

Mongoose mongoosejs.com

> elegant mongodb object modeling for node.js

> Mongoose provides a straight-forward, **schema-based** solution to model your application data. It includes *built-in type casting, validation, query building, business logic hooks* and more, out of the box.

Get Mongoose

```
$ npm install mongoose
```

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

> https://mongoosejs.com/docs/guide.html

Defining your schema

```
import mongoose from 'mongoose';
const { Schema } = mongoose;

const studentSchema = new Schema({
    name: 'string',
    surname: 'string'
    });
```

Ids - By default, Mongoose adds an _id property to your schemas.

Creating a model

To use our schema definition, we need to convert our **studentSchema** into a **Model** we can work with. To do so, we pass it into mongoose.model(modelName, schema):

```
const Student = mongoose.model('Student', studendSchema);

//new student added to the DB
const student = new Student({ name: 'Antonio', surname: 'Bucchiarone' });
student.save(function (err) {
    if (err) return handleError(err);
    // saved!
    console.log("Student added");
});
```

When you create a new document, a new _id of type ObjectId is created.

inserting large batches of documents

```
Student.insertMany(
    [
        { name: 'mario', surname: 'rossi' },
        { name: 'paolo', surname: 'neri' }],
    function (err) {
        console.log("array of students added");
    });
```

Deleting

- Models have static `deleteOne()` and `deleteMany()` functions for removing all documents matching the given filter.

```
Student.deleteOne({ name: 'Antonio' }, function (err) {
  if (err) return handleError(err);
  // deleted at most one tank document
  console.log("element deleted");
});
```

```
Student.deleteMany({ name: 'Antonio' }, function (err) {
    if (err) return handleError(err);
    // deleted at most one tank document
    console.log("elements deleted");
});
```

Querying

> https://mongoosejs.com/docs/models.html#querying

Finding documents is easy with Mongoose, which supports the rich query syntax of MongoDB. Documents can be retrieved using a model's **find**, **findById**, **findOne**, or **where** static methods.

```
// Find one student whose `name` is 'Antonio', otherwise `null`
Student.findOne({ name: 'Antonio' }).exec();

// using callback
Student.findOne({ name: 'Antonio' }, function (err, student) {
    console.log("surname: "+student.surname);
});

// select only the students name and surname
Student.findOne({ name: 'Antonio' }, 'name surname').exec();
```

Subdocuments versus Nested Paths

> https://mongoosejs.com/docs/subdocs.html#subdocuments-versus-nested-paths

```
// Subdocument
const subdocumentSchema = new mongoose.Schema({
  child: new mongoose.Schema({ name: String, age: Number })
});
const Subdoc = mongoose.model('Subdoc', subdocumentSchema);

// Nested path
const nestedSchema = new mongoose.Schema({
  child: { name: String, age: Number }
});
const Nested = mongoose.model('Nested', nestedSchema);
```

Questions?

bucchiarone@fbk.eu