CS 572 Modern Web Applications

Najeeb Najeeb, PhD (najeeb@miu.edu)

V3.0.0

JavaScript Full Stack Development

- MongoDB
  - NoSQL database (document store)
  - Stores JSON documents
- Express
  - JavaScript web framework
  - On top of Node
- Angular
  - JavaScript UI framework
  - Single Page Applications
- Node
  - JavaScript server-side platform
  - Single threaded, fast and scalable

Roadmap and Outcomes

- Node.js: write asynchronous (non-blocking) code. Understand node platform to start a project.
- Express: setup express and get requests and send back responses. REST API.
- MongoDB: what NoSQL DB looks like. Full API interacting with DB.
- Angular: Investigate Angular and the architecture of an Angular application. Build a single-page application.
- MEAN application: Learn by example. We will create a MEAN Games application.

Why Mongoose

- Create a controller for each document and define constraints in the controller.
  - Too much work and could end up repeating a lot of the same stuff.
  - Errors and inconsistencies.
- Better to have one schema (define it once) and use it for all my documents.
- Mongoose comes to the rescue.
  - Helps us focus on building our application and building the API.
  - Abstracts complexity of using native driver.
  - Provides helper methods to work with DB.
  - We can define the structure of our data in the application (schema).

Mongoose
Do Less Accomplish More

Wholeness

Mongoose is built on top of MongoDB driver; that is why it provides us with all the benefits of using MongoDB driver. By understanding Mongoose and using it properly we not only gain performance benefits, but also the lines of code we need to write are fewer than what is needed to perform the same task using MongoDB driver alone. When you are in tune with the laws of nature your actions become spontaneously correct. By actions being correct the first time, we do not need to spend a lot of time on an issue, we get an issue addressed properly with fewer actions.

Mongoose
Do Less Accomplish More

1. What is Mongoose and how to set it up?
2. How to use Schema with Mongoose?
3. What is GeoSearch, and how to use it?
4. How to perform CRUD operations in Mongoose?

REST API

URL Patterns

PATTERN

- Base URL (www.myapplication.com)
- Actions, depending on the method
- Get all/multiple items
  - GET (/api/items)
- Create a new item
  - POST (/api/items)
- Get single item
  - GET (/api/items/123)
- Update a single item
  - PUT (api/items/123)
- Delete a single item
  - DELETE (api/items/123)

NESTED

- Get all reviews for item (123)
  - GET (/api/items/123/reviews)
- Create a review for item (123)
  - POST (/api/items/123/reviews)
- Get single review (222) for items 123
  - GET (/api/items/123/reviews/222)
- Update a single review
  - PUT (api/items/123/reviews/222)
- Delete a single review
  - DELETE (api/items/123/reviews/222)

Mongoose

What is Mongoose

- A module built on top of MongoDB driver
- Can perform all functionality performed by MongoDB driver
- Enables our application to have a Schema
- Makes it easy to perform GeoLocation Searches
- Enables us to focus on our application and it will take care of dealing with the Database
    - More application features
    - Create REST API
    - Hardening of API

Mongoose

Install
Connect
Disconnect
Terminate
Restart

Use the last application from Lesson02 (app2)

Install Mongoose

```
npm install mongoose
mongoose@6.1.4 node_modules/mongoose
```

Mongoose

Install

Connect

Disconnect

Terminate

Restart

Create file /api/data/db.js

```
const mongoose= require("mongoose");
mongoose.connect(process.env.DB_URL, { useNewUrlParser: true, useUn
ifiedTopology: true });
mongoose.connection.on("connected", function() {
   console.log("Mongoose connected to "+ process.env.DB_NAME);
});
mongoose.connection.on("disconnected", function() {
   console.log("Mongoose disconnected");
});
mongoose.connection.on("error", function(err) {
   console.log("Mongoose connection error "+ err);
});
```

Update app.js to use mongoose

```
require("./api/data/db.js");
```

Add environment variables to .env

```
DB_URL= "mongodb://localhost:27017/meanGames"
DB_NAME= meanGames
```

Mongoose

Install

Connect

Disconnect

Terminate

Restart

Create file /api/data/db.js

```
process.on("SIGINT", function() {
    mongoose.connection.close(function() {
    console.log(process.env.SIGINT_MESSAGE);
    process.exit(0);
});
});
```

Add environment variable to .env

```
SIGINT_MESSAGE= "Mongoose disconnected by app disconnect"
```

Mongoose

Install

Connect

Disconnect

Terminate

Restart

Create file /api/data/db.js

```
process.on("SIGTERM", function() {
    mongoose.connection.close(function() {
    console.log(process.env.SIGTERM_MESSAGE);
    process.exit(0);
});
});
```

Add environment variable to .env

```
SIGTERM_MESSAGE= "Mongoose disconnected by app termination"
```

Mongoose

Install

Connect

Disconnect

Terminate

Restart

Create file /api/data/db.js

```
process.once("SIGUSR2", function() {
    mongoose.connection.close(function() {
    console.log(process.env.SIGUSR2_MESSAGE);
    process.kill(process.pid, "SIGUSR2");
});
});
```

Add environment variable to .env

```
SIGUSR2_MESSAGE= "Mongoose disconnected by
app restart"
```

Mongoose
Install
Connect
Disconnect
Terminate
Restart

New nodemon and Windows not sending SIGUSR2

Add to nodemon.json

```
{
   "signal": "SIGHUP",
   "env": {
      "NODE_ENV": "development"
   }
}
```

Run nodemon as

```
nodemon --inspect
```

Update package.json to use this

```
"dev": "nodemon --inspect",
```

To start the application run

```
npm run dev
```

Mongoose
Do Less Accomplish More

1. What is Mongoose and how to set it up?
2. How to use Schema with Mongoose?
3. What is GeoSearch, and how to use it?
4. How to use CRUD operations with Mongoose?

Mongoose Schemas & Models

Mongoose
Add Schema
Data Validation
Compile Model

Separate schema from connection, what gets exported is a model (even though it is called schema)

Create file /api/data/games-model.js

```
const mongoose= require("mongoose");
const gameSchema= mongoose.Schema({
    title: String,
    year: Number,
    rate: Number
    price : Number,
    minPlayers: Number,
    maxPlayers: Number,
    minAge: Number,
    designers : [String]
});
```

Mongoose

Add Schema

Data Validation

Compile Model

Mandatory fields for a document

Modify file /api/data/games-model.js

```
const mongoose= require("mongoose");
const gameSchema= mongoose.Schema({
    title: {
        type: String,
        required: true
    },
    year: Number,
    rate: {
        type: Number,
        min: 1,
        max: 5,
        "default": 1
    },
    price : Number,
    minPlayers: {
        type: Number,
        min : 1,
        max: 10
    },
    maxPlayers: {
        type: Number,
        min : 1,
        max: 10
    },
    minAge: Number,
    designers : [String]
});
```

Mongoose

Add Schema

Data Validation

Compile Model

Mandatory fields for a document

Modify file /api/data/games-model.js

```
mongoose.model("Game",  gameSchema, "games");
```

Modify db.js to let it know about our model

```
require("./games-model");
```

Schema

Nested Doc

Nested Docs

Geo-Location

A game is normally published by a publisher. The publisher is from a certain country, established at a certain date, also famous for a certain game
Modify file /api/data/games-model.js

```
...
const publisherSchema= new mongoose.Schema({
    name: {
        type: String,
        required: true
    },
    country: String,
    established: Number, //Not a date since we only have year
    location: String
});
const gameSchema = mongoose.Schema({
...
    publisher: publisherSchema
});
...
```

Schema

Nested Doc

Nested Docs

A review is a sub-document. A review is for a game by a user with some rating and description at a certain date.
Modify file /api/data/games-model.js

```
...
const reviewSchema= new mongoose.Schema({
  title: {
    type: String,
    required: true
  },
  rating: {
    type: Number,
    min: 1,
    max: 5,
    required: true
  },
  review: {
    type: String,
    required: true
  },
  postDate: {
    type: Date,
    "default": Date.Now
  },
});
const gameSchema = mongoose.Schema({
...
  reviews: [reviewSchema]
});
...
```

Mongoose

GetAll

GetOne

Use Mongoose to get all Games, simpler way of doing things.
Modify file /api/controllers/games.controller.js

```
const mongoose= require("mongoose");
const Game= mongoose.model(process.env.GAME_MODEL);

const getAll= function(req, res) {
    let offset= 0;
    let count= 5;
    if (req.query && req.query.offset) {
        offset= parseInt(req.query.offset, 10);
    }
    if (req.query && req.query.count) {
        offset= parseInt(req.query.count, 10);
    }
    Game.find().exec(function(err, games) {
        console.log("Found games", games.length);
        res.json(games);
    });
}
```

Mongoose

GetAll

GetOne

Use Mongoose to get all Games, simpler way of doing things.
Modify file /api/controllers/games.controller.js

```
const mongoose= require("mongoose");
const Game= mongoose.model(process.env.GAME_MODEL);

const getAll= function(req, res) {
    let offset= 0;
    let count= 5;
    if (req.query && req.query.offset) {
        offset= parseInt(req.query.offset, 10);
    }
    if (req.query && req.query.count) {
        offset= parseInt(req.query.count, 10);
    }
    Game.find().skip(offset).limit(count).exec(function(err, games) {
        console.log("Found games", games.length);
        res.json(games);
    });
}
```

Mongoose
GetAll
GetOne

Use Mongoose to get one Game, simpler way of doing things.

Modify file /api/data/games-controller.js

```
const getOne= function(req, res) {
    const gameId= req.params.gameId;
    Game.findById(gameId).exec(function(err, game) {
        res.status(200).json(game);
    });
}
```

Mongoose GET

Sub-document

Sub-documents

GetAll

Sub-documents

GetOne

Add a route to the sub-document (based on REST rules).
Separate Controllers into logical collections.
Add a Controller for the sub-document.
Modify file /api/routes/index.js

```
…
const publisherController= require("../controllers/publisher.controllers");
…
router.route("/games/:gameId/publisher")
  .get(publisherController.getOne);
…
```

Add file /api/controllers/publisher.controllers.js

```
const mongoose= require("mongoose");
const Game= mongoose.model(process.env.GAME_MODEL);
const getOne= function(req, res) {
  console.log("GET One Publisher Controller");
  const gameId= req.params.gameId;
  Game.findById(gameId).select("publisher").exec(function(err, game) {
    console.log("Found publisher ", game.publisher, " for Game ", game);
    res.status(200).json(game.publisher);
  });
}
module.exports= {
  getOne : getOne
}
```

Mongoose GET

Sub-document

Sub-documents

GetAll

Sub-documents

GetOne

Add a route to the sub-document (based on REST rules).
Separate Controllers into logical collections.
Add a Controller for the sub-document.
Modify file /api/routes/index.js

```
...
const reviewsController= require("../controllers/reviews.controllers");
...
router.route("/games/:gameId/reviews")
  .get(publisherController.getAll);
...
```

Add file /api/controllers/reviews.controllers.js

```
const mongoose= require("mongoose");
const Game= mongoose.model(process.env.GAME_MODEL);
const getAll= function(req, res) {
  console.log("GET Reviews Controller");
  const gameId= req.params.gameId;
  Game.findById(gameId).select("reviews").exec(function(err, game) {
    console.log("Found reviews ", game.reviews, " for Game ", game);
    res.status(200).json(game.reviews);
  });
}
module.exports= {
  getOne : getOne
}
```

Mongoose GET Sub-document Sub-documents GetAll Sub-documents GetOne

Add a route to the sub-document (based on REST rules).
Separate Controllers into logical collections.
Add a Controller for the sub-document.
Modify file /api/routes/index.js

```
...
router.route("/games/:gameId/reviews/:reviewId")
   .get(publisherController.getAll);
...
```

Add file /api/controllers/reviews.controllers.js

```
...
const getOne= function(req, res) {
   console.log("GET One Publisher Controller");
   const gameId= req.params.gameId;
   const reviewId= req.params.reviewId;
   Game.findById(gameId).select("reviews").exec(function(err, game) {
      console.log("Found review ", game.review.id(reviewId), " for Game ", game);
      res.status(200).json(game.review.id(reviewId));
   });
}
module.exports= {
   getAll: getAll,
   getOne : getOne
}
```