**How To Make a Books App Using Mongoose and Node.js**

1. Fork and clone existing repo in Ubuntu:

<https://git.generalassemb.ly/sf-wdi-43/mongoose-books-app.git>

1. Make a branch using git checkout -b my-work-sprint-1
2. Open up server.js to see what’s in it. server.js is the connection between routes and database.
3. Before starting node server, npm install to install necessary files.
4. Create book.js file in models folder.
5. Create BookSchema:

var mongoose = require(‘mongoose’);

var Schema = mongoose.Schema;

var BookSchema = newSchema({

title: String,

author: {

type: Schema.Types.ObjectId

ref: ‘Author’

},

image: String,

releaseDate: String

});

var Book = mongoose.model(‘Book’, BookSchema);

module.exports = Book;

1. Look in index.js file in models folder.
2. index.js will import each model and export an object called exports with keys representing each of our models. That way we can require the entire directory and get all of our models! Thus, we import and export with the following code: module.exports.Book = require(“./book.js”);
3. 
4. Look at seed.js
5. Run seed.js file. If it doesn’t work, look in book.js. It should look like



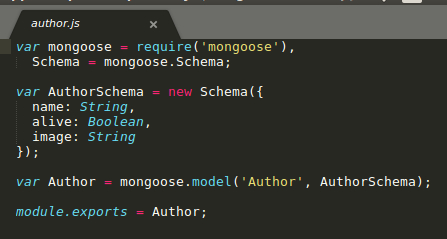
1. Open server.js and add var db = require(‘./models’) at the top near “SETUP and CONFIGURATION”.
2. Delete hard-coded books array.
3. Find books index route and replace with



1. Restart node js and mongod. You should see all books from database from seed.js upon refreshing.
2. Create new file models/author.js. This is where you:

* Require mongoose
* Have your AuthorSchema object
* Assign variable Author to mongoose model called Author with the AuthorSchema
* Export module Author

As shown below:

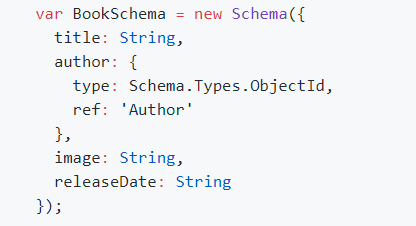


1. Export author.js and book.js in index.js, as shown belown:

module.exports.Author = require(“./author.js”);

module.exports.Book = require(“./book.js”);

1. Update Book Schema:



1. Add authors\_list array data

var authors\_list = [

{

name: "Harper Lee",

alive: false

},

{

name: "F Scott Fitzgerald",

alive: false

},

{

name: "Victor Hugo",

alive: false

},

{

name: "Jules Verne",

alive: false

},

{

name: "Sheryl Sandberg",

alive: true

},

{

name: "Tim Ferriss",

alive: true

},

{

name: "John Steinbeck",

alive: false

},

{

name: "William Shakespeare",

alive: false

}

];

1. Update seed.js file to create books and authors functions and connect the two together. Remove all other functions and replace with:

db.Author.remove({}, function(err, authors) {

console.log('removed all authors');

db.Author.create(authors\_list, function(err, authors){

if (err) {

console.log(err);

return;

}

console.log('recreated all authors');

console.log("created", authors.length, "authors");

db.Book.remove({}, function(err, books){

console.log('removed all books');

books\_list.forEach(function (bookData) {

var book = new db.Book({

title: bookData.title,

image: bookData.image,

releaseDate: bookData.releaseDate

});

db.Author.findOne({name: bookData.author}, function (err, foundAuthor) {

console.log('found author ' + foundAuthor.name + ' for book ' + book.title);

if (err) {

console.log(err);

return;

}

book.author = foundAuthor;

book.save(function(err, savedBook){

if (err) {

console.log(err);

}

console.log('saved ' + savedBook.title + ' by ' + foundAuthor.name);

});

});

});

});

});

});

1. Edit server.js. Change a few routes as follows:

// get all books

app.get('/api/books', function (req, res) {

// send all books as JSON response

db.Book.find()

// populate fills in the author id with all the author data

.populate('author')

.exec(function(err, books){

if (err) { console.log("index error: " + err); }

res.json(books);

});

});

// create new book

app.post('/api/books', function (req, res) {

// create new book with form data (`req.body`)

var newBook = new db.Book({

title: req.body.title,

image: req.body.image,

releaseDate: req.body.releaseDate,

});

// this code will only add an author to a book if the author already exists

db.Author.findOne({name: req.body.author}, function(err, author){

newBook.author = author;

// add newBook to database

newBook.save(function(err, book){

if (err) {

console.log("create error: " + err);

}

console.log("created ", book.title);

res.json(book);

});

});

});

1. Open your web browser developer tools. Try to add a character to one of the books. You should see an error message like: jquery.min.js:4 POST http://localhost:3000/api/books/56fc1e8a8d4bcdb3e5e0092e/characters 404 (Not Found)
2. Add CharacterSchema to models/book.js with one name attribute:

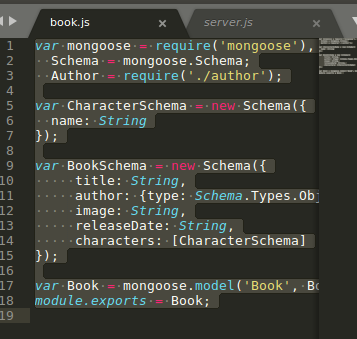
// models/book.js

var CharacterSchema = new Schema({

name: String

});

1. Access embedded Schema in BookSchema:



1. Add a new route to server.js

app.post('/api/books/:book\_id/characters', function (req, res) {

// Get book id from url params (`req.params`)

var bookId = req.params.book\_id;

db.Book.findById(bookId)

.populate('author') // Reference to author

// now we can worry about saving that character

.exec(function(err, foundBook) {

console.log(foundBook);

if (err) {

res.status(500).json({error: err.message});

} else if (foundBook === null) {

// Is this the same as checking if the foundBook is undefined?

res.status(404).json({error: "No Book found by this ID"});

} else {

// push character into characters array

foundBook.characters.push(req.body);

// save the book with the new character

foundBook.save();

res.status(201).json(foundBook);

}

}

);

});