**Clone book-starter @ 12 June 2019**

Npm install

Nodemon

Mongod (make sure you’re in the right directory. This runs the mongo server on port 27010. This should run automatically when you start up the mac )

Display single entry

Add a route in In Routes.js, Add

router.get("/authors/:id", AuthorController.show)

Add in a method in author\_controller.js, add

async function show(req, res) {

// req.params.id is to get the id from the request object

let author = await AuthorModel.findById(req.params.id)

res.render('layout', {

view: 'author/show',

title: 'Author',

author

})

}

Update the exports at the bottom of author\_controller.js

module.exports = {

create,

index,

make,

show

}

In layout.ejs, add the general pattern (best practice), is the title o the page could consist the site title and the page value.

<title>Books R Us - <%= title %></title>

Add a view for showing a single author. Create show.ejs in view > authors

* Rendering a single author here

Add into show.ejs

* <h1><%= author.name %></h1>
* <h3><%= author.gender %></h3>
* <p><%= author.bio %></p>
* <hr>
* <a href="/authors">Back to Authors</a>

In index.ejs add

<a href="/authors/<%= author.\_id %>">Show Details</a>

The author ID here corresponds to the route created in routes.js

router.get("/authors/:id", AuthorController.show)

*Problem with create new author not working*

In routes currently, the /authors/new thinks it is matching with /authors/:id.

router.get("/authors", AuthorController.index)

router.get("/authors/:id", AuthorController.show) //newly added

router.post("/authors", AuthorController.create)

router.get("/authors/new", AuthorController.make)

To fix this, move the make new author route above the show individual author route. ***The order of the routes matter***!

router.get("/authors", AuthorController.index)

router.get("/authors/new", AuthorController.make)

router.get("/authors/:id", AuthorController.show) //newly added

router.post("/authors", AuthorController.create)

**Improving the routes.js**

Rename routes.js to author\_routes.js

It will change the app.use statement in app.js

app.use(require("./author\_routes"))

Add a path to the use the routes. Only use the routes only for path that starts with authors

app.use('/authors', require("./author\_routes"))

Likewise, we can then update the routes.sj to remove all /authors in the path designation

router.get("/", AuthorController.index)

router.get("/new", AuthorController.make)

router.get("/:id", AuthorController.show)

router.post("/", AuthorController.create)

**Add a delete author function**

On routes.js,a dd

router.delete("/:id", AuthorController.destroy)

on author\_controller, add a delete authors method

async function destroy(req, res) {

await AuthorModel.findByIdAndRemove(req.params.id)

res.redirect("/authors")

}

On author\_controller.js, update the exports function

module.exports = {

create,

index,

make,

show,

destroy

}

Need to pass the delete request. Go into layout.ejs, and add script. (this can be in its own file)

<body>

<%- include(view) -%>

<script>

// event handler

function deleteAuthor(id) {

console.log(id)

}

</script>

</body>

Update index.ejs by adding a link to delete that calls the deleteAuthor method defined in layout.ejs

<h1>Authors</h1>

<a href="/authors/new">Create New Author</a>

<% for (author of authors) { %>

<div>

<p>Name: <%= author.name %></p>

<p>Bio: <%= author.bio %></p>

<p>Gender: <%= author.gender %></p>

<a href="/authors/<%= author.\_id %>">Show Details</a> |

<a href="" onclick="deleteAuthor('<%= author.\_id %>')">Delete Author</a>

</div>

<hr />

<% } %>

Note, it needs to be single quotes around the author\_id inside the method call to deleteAuthor. The empty quotes in href allows you to delete.

Now update the javascript in layout.ejs to retrieve the authors by its id

<body>

<%- include(view) -%>

<script>

// event handler

function deleteAuthor(id) {

fetch(`/authors/${id}`)

}

</script>

</body>

AT the end of the delete function, it still redirects, because the page is reloaded.

Update layout ejs

<body>

<%- include(view) -%>

<script>

// event handler

function deleteAuthor(id) {

// this is the HTTP request,

// when you delete it, and it resolves, redirect it to the authors

// The redirection has to be done on the clientside and not serverside because this is an ajax call.

fetch(`/authors/${id}`, {

method: 'delete'

}).then(res => location.href = '/authors')

}

</script>

</body>

**To do all of the above in the easier way 🡪 npm i method-override**

Call it in your app.js by adding to the top

const methodOverride = require("method-override")

On App.js, use the methodOverride

app.use(methodOverride('\_method'))

In index.ejs, update the delete link as per follows

<a href="/authors/<%= author.\_id %>?\_method=DELETE">Delete Author</a>

You can use change this to \_method=PUT for editing

Comment out the <script> javascript </script> in layout.ejs

Note: By default methodOverride will only work on post requests. We need to get it to work on delete requests as well. So you need to pass in the methods for POST and GET to get it to work specifically. Otherwise you will not be able to delete the entry.

So go back to app.js, update the use statement for methodOverride

app.use(methodOverride('\_method', {

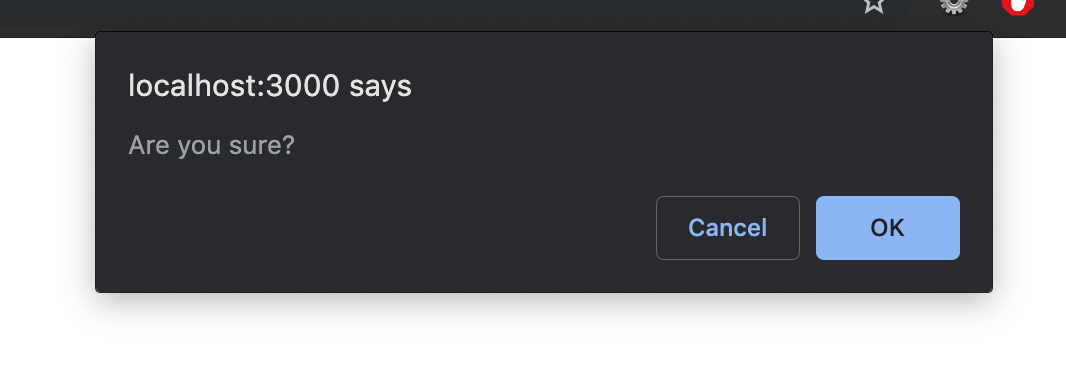
methods: ['POST', 'GET']

}))

**Resolving: no user notification that an element has been deleted or will be deleted**

On **index.ejs**, update the delete link as per below with confirmation alert box to pop up

<a href="/authors/<%= author.\_id %>?\_method=DELETE" onclick="return confirm('Are you sure?')">Delete Author</a>



**Add books models to authors – hwo to do relationshisp on mongo**

Create a schema and model

On models folder > create book\_model.js

Copy all of author\_model into book\_model. We will modify the code for book\_model to the below

const mongoose = require("mongoose")

// create a schema

const BookSchema = new mongoose.Schema({

// set properties as title

title: {

type: String,

required: true

},

published: {

type: Date,

default: true

},

author: {

// this field will be the ID of an object. It is a UID (long hex).

type: mongoose.Schema.ObjectId,

ref: 'author'

}

})

// export the schema because you might us it elsewhere

const BookSchema = mongoose.model("book", BookSchema)

module.exports = {

BookModel,

BookSchema

}

**Do routes**

Copy author\_routes.js to book\_routes.js

The routes remain the same for book (CRUD operations)

Replace all mentions of “authors” to “book”, case-sensitive

const express = require("express")

const router = express.Router()

const BookController = require("./controllers/book\_controller")

router.get("/", BookController.index)

router.get("/new", BookController.make)

router.get("/:id", BookController.show)

router.post("/", BookController.create)

router.delete("/:id", BookController.destroy)

module.exports = router

on App.js, add below under the app.use(‘/authors) statement

app.use('/books', require("./book\_routes"))

**Do Controller**

Duplicate author\_controller.js to book\_controller.js

Replace all mentions of “authors” to “book”, case-sensitive

Modify the **create** function with ‘title, published, author’ properties as defined in the model as per below

async function create(req, res) {

//logic for creating a resource

let {

title,

published,

author

} = req.body

let book = await BookModel.create({

title,

published,

author

})

.catch(err => res.status(500).send(err))

res.redirect("/books")

}

In the make function add in an array of author to the system. So you must create an author before you create a book on the system. When you create a new book, you should be able to choose from a drop down lsit of existing authors to chose from . modify the make function in book\_controller.js

function make(req, res) {

//shows the form to create the resource

const authors = await AuthorModel.find()

res.render('layout', {

view: 'book/new',

title: 'New Book',

authors // create a list of all authors so that you can create a new book by choosing from a list of existing authors

})

// res.render("book/new")

}

**View**

Book views are essentially a copy of the author views

Go through each file in the book vies folder and replace all mention of author to book

Update index.js with

<p>Title: <%= book.title %></p>

<p>Published: <%= book.published %></p>

<p>Author: <%= book.author.name %></p>

It is book.author.name not book.author because we don’t want to output the entire author object ( youd on’t need the bio, gender for instance)

Update show.ejs

<h1><%= book.title %></h1>

<h3><%= book.published %></h3>

<p><%= book.author.name %></p>

<hr>

<a href="/books">Back to Books</a>

In Index.js, change the link to author to a link that when you click on it will go to the author page

<p>Author: <a href="/authors/{%= book.author.\_id %>"><%= book.author.name %></a></p>

Likewise, modify show.ejs to the below

<h1><%= book.title %></h1>

<h3><%= book.published %></h3>

<p>Author: <a href="/authors/{%= book.author.\_id %>"><%= book.author.name %></a></p>

<hr>

<a href="/books">Back to Books</a>

On book\_controller, add to the top. We do this because in make function we are referring to the AuthorModel

const {

AuthorModel

} = require("../models/author\_model")

On views >book > new.ejs modify the labels to title, published and author

For name, this remains the same as authors.

For published, this should be a date field

<div>

<label>Published</label>

</div>

<div>

<input type="date" name="published" />

</div>

For authors, modify the below to call from the authors model . The value of the dropdown list is the author id that we want to pass onto the form so that the system knows which author to choose. But visually, the option should be the author’s name .

<div>

<label>Author</label>

<input>

</div>

<div>

<select name="author">

<% for (author of authors) { %>

<option value="<%= author.\_id %>"><%= author.name %></option>

<% } %>

</select>

</div>

Express doesn’t do relationship by default. You must provide all the details. Use a method called populate

ON book\_controller.js, modify the index method by adding to the end of the books initialisation. Unlike rails which does it automatically or you, this tells express to go to the database, search for the object “author” (identified as the foreign key in the book\_model.js) to insert into the book object.

let books = await BookModel.find().populate('author')

on book\_controller.js, modify the show function by adding to the end of the book initialisation

let book = await BookModel.findById(req.params.id).populate('author')