**Afternoon Challenge in Express-Bookstore @ 14 June**

Create two more routes in author\_routes.js

* View the form itself
* Post that data to

Add to bottom of author\_routes.js before the export

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

router.put("/:id/edit", AuthorController.update)

on author\_controller.js, add edit and update to the export statement at the bottom

module.exports = {

create,

index,

make,

show,

destroy,

edit,

update,

}

The existing make function is closest to what we want to achieve in edit function, because the make function sees a copy of the form.

Copy and paste make function and rename as edit.

Change the view being rendered to “author/edit” not “author/new”. And title is “edit author (remember to make the function async always)

async function edit(req, res) {

//shows the form to create the resource

res.render('layout', {

view: 'author/edit',

title: 'Edit Author'

})

}

The edit function, we will have the form fields. We will also need id (in the background) and current existing information to be prepopulated in the form. To do that, we have to retrieve the actual author object based on the passed through id (from the route we set up “/:id/edit”.

To do all that, update the edit function as per below

async function edit(req, res) {

let author = await AuthorModel.findById(req.params.id) //route to edit includes an id already

//shows the form to create the resource

res.render('layout', {

view: 'author/edit',

title: 'Edit Author',

author //the id is passed through to the view here

})

}

In Views > Authors, copy and paste the new.ejs and rename as edit.ejs.

In edit.ejs, change the form method to post (not put, because it won’t be allowed). You will need to pass through the author\_id when the form is submitted as well. We use methodOverride to tell Express that we actually want the PUT method, not POST.

<form method="post" action="/authors/<%= author.\_id %>?\_method=PUT">

To put in the existing values into the edit form, we update the value attributes in each of the <input> fields.

<input type="text" name="<%= author.name %>" />

<textarea name="bio"><%= author.bio%></textarea>

This will enable the chosen choice to appear selected in the drop down list

<select name="gender">

<option value="<%= author.gender %>" hidden selected><%= author.gender %></option>

<option value="male">male</option>

<option value="female">female</option>

<option value="non binary">non binary</option>

</select>

**The Update function**

Create function is what receives the post of the form and is what is closest to the update method (to post what has been inputted in edit function).

Copy and paste create function and rename as update.

Update the newly added update function with the below.

This is telling the Authormodel to find the author by the id and update the fields with what is passed in through req.body

let author = await AuthorModel.findByIdAndUpdate(req.params.id, {

name,

bio,

gender

})

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

Update Views > Author > **index.ejs**

<a href="/authors/<%= author.\_id %>/edit">Edit</a> |

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

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

List all authors books on author-show page and make link to the show-book page

Go to author\_controller.js first, show needs a list of books written by author to pass it to the view. Update show method

async function show(req, res) {

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

let books = await BookModel.findOne({

author: req.params.id // criteria

})

res.render('layout', {

view: 'author/show',

title: 'Author',

author,

books

})

}

On Author\_controller.js, require in the bookModel

const {

BookModel

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

Go to views folder > authors > show.ejs

<h2>Books</h2>

<% for (book of books) { %>

<a href="/books/<%= book.\_id %>"><%= book.title %> (<%= book.published.getYear() %>)</a>

<% } %>

<%= book.published.year() %> draws only the year of the date. The date is stored as a javascript object. .getYear() is a method to draw the year off that date object.

**Add photo**

npm I express-fileupload

Go into app.js

const fileUpload = require('express-fileupload')

Down the page on app.js, insert it as middleware

app.use(fileUpload())

Puts the files in memory and then gives us a files property to the request object which allows us to access them and make use of them.

We want to save the file to the file system.

Create a new folder on root called “public”. Inside public, create a folder “images”. All static assets (not routes). We also put front-end javascript, stylesheets in public.

We need to tell Express where to serve static assets from by adding the below to app.js.

app.use(express.static('public'))

Any images dropped into public>images can be accessed by localhost:3000/images/picture.jpg

On author\_controller.js, to add images to controller, we need to modify modify, update, edit functions.

In the create function, we add console.log to test the file upload first

console.log(req.file.image)

req.file.image refers to the request object > file inside that request object> uploaded from the input named “image”

In Views > authors > view.ejs, add to the form

<div>

<label>Image </label>

</div>

<div>

<input type="file" name="image" />

</div>

**Multer – image file upload**

multer is not an app middleware. But a route middleware. Need ot require it in our routes file instead of apps. Js. It deals with multi-part form.

Add to author\_routes.js

const multer = require('multer')

also tell multer where to add the files to (define destination) in author\_routes.js

const upload = multer({

dest: 'public/images'

})

In author\_routes.js, modify the create route as per below

Tells the upload object to upload a single image from the <input> called image

router.post("/", upload.single('image'), AuthorController.create)

in views > author> new.ejs, modify the form (do the same for edit.ejs)

<form method="post" action="/authors" enctype="multipart/form-data">

Add to author\_model

image: {

type: String,

}

On Author\_controller, add to create function

const image = req.file.path.replace('public', '')

let author = await AuthorModel.create({

name,

bio,

gender,

image

})

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

ON view > Author > show.ejs, add under h1

<img src="<%= author.image %>" />

On author\_routes.js

const path = require('path')

on author\_routes.ks

const storage = multer.diskStorage({

destination: function (req, file, callback) {

callback(null, 'public/images')

},

filename: function (req, file, callback) {

callback(null, req.params.id + path.extname(file.originalname))

}

})

Path package draws out the original name extension

Req.parms.id is coming in as a url id (not mongodb id)

In author)routes.js, amend the multer require in

const upload = multer({

storage

})