**Src/controller/Article.js**

/\*\*

 \* Controller extracts information (query string parameters, path parameters etc., makes a call a call to the service to get data, and sends the response)

 \*/

// import { fetchArticles } from '../services/Articles.cjs';

import {

  fetchArticles,

  fetchArticleById,

  addArticle,

  updateArticle,

  removeArticle,

  addcomment,

  fetchcomments

} from "../services/Articles.js";

import HttpError from '../utils/HttpError.js';

// /Articles?sort=name&order=asc&minPrice=10&minRating=4&page=2&q=garden

const getArticles = (req, res, next) => {

  // Extract minPrice from query string and log it as well

  let { sort, order, page, q } = req.query;

  // convert page to integer, and set a default of 1 if it is not passed / incorrect

  let pageInt = parseInt(page); // parseInt( 'hello' ) -> NaN, parseInt( undefined ) -> NaN

  if (isNaN(pageInt)) {

    pageInt = 1;

  }

  // if sort field is not passed, we assume it is sorted by name

  if (!sort) {

    sort = "name";

  }

  // if sorting order is not passed, we assume it is sorted in ascending order

  if (!order) {

    order = "asc";

  }

  fetchArticles(sort, order, pageInt, q)

    .then((Articles) => {

        res.json(Articles);

    })

    .catch((err) => {

        // eg. DB connection error

        // 500 -> Internal Server Error

        const httpError = new HttpError( err.message, 500 );

        next( httpError );

    });

};

// GET /products/:\_id

// Example: /Articles/61dd7711cbf3176114766adb

const getArticleById = (req, res, next) => {

  const { \_id } = req.params;

  fetchArticleById(\_id)

    .then((Article) => {

      if( !Article ) {

          // 404 -> Not Found

          const httpError = new HttpError( 'Article with given id does not exist', 404 );

          next( httpError );

          return;

      }

      res.json(Article);

    })

    .catch((err) => {

      if( err.kind === 'ObjectId' ) {

          // 400 -> BAD REQUEST

          const httpError = new HttpError( 'Invalid Article id', 400 );

          next( httpError );

      } else {

          // eg. DB connection error

          // 500 -> Internal Server Error

          const httpError = new HttpError( err.message, 500 );

          next( httpError );

      }

    });

};

// POST /Articles

const postArticle = (req, res, next) => {

  const { body } = req;

  console.log( Object.keys( body ) );

  // check if the body is an empty object

  if( Object.keys( body ).length === 0 ) {

    const httpError = new HttpError( 'Request body is empty. Article details are missing.', 400 );

    next( httpError );

    return;

  }

  addArticle( body )

    .then(Article => {

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

    })

    .catch(err => {

      const httpError = new HttpError( err.message, 400 );

      next( httpError );

    });

};

// PUT /Articles/:\_id

const putArticle = ( req, res, next ) => {

  const { body } = req;

  const { \_id } = req.params;

  // check if the body is an empty object

  if( Object.keys( body ).length === 0 ) {

    const httpError = new HttpError( 'Request body is empty.Article details are missing.', 400 );

    next( httpError );

    return;

  }

  updateArticle( \_id, body )

    .then(Article => {

      res.json( Article);

    })

    .catch(err => {

      if( err.kind === 'ObjectId' ) {

        // 400 -> BAD REQUEST

        const httpError = new HttpError( 'Invalid Article id', 400 );

        next( httpError );

      } else {

          // eg. DB connection error

          // 500 -> Internal Server Error

          const httpError = new HttpError( err.message, 500 );

          next( httpError );

      }

    });

};

const deleteArticle = ( req, res, next ) => {

  const { \_id } = req.params;

  removeArticle( \_id )

    .then(() => {

      res.status(204).json();

    })

    .catch(err => {

      if( err.kind === 'ObjectId' ) {

        // 400 -> BAD REQUEST

        const httpError = new HttpError( 'Invalid Article id', 400 );

        next( httpError );

      } else {

          // eg. DB connection error

          // 500 -> Internal Server Error

          const httpError = new HttpError( err.message, 500 );

          next( httpError );

      }

    });

};

// POST /:\_id/reviews

const postcomment = (req, res, next) => {

  const { \_id } = req.params;

  const { body } = req;

  // check if the body is an empty object

  if( Object.keys( body ).length === 0 ) {

    const httpError = new HttpError( 'Request body is empty. comment details are missing.', 400 );

    next( httpError );

    return;

  }

  addcomment( \_id, body )

    .then(Article => {

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

    })

    .catch(err => {

      const httpError = new HttpError( err.message, 400 );

      next( httpError );

    });

};

const getcomments = (req, res, next) => {

  const { \_id } = req.params;

  fetchcomments( \_id )

    .then((comments) => {

        res.json(comments);

    })

    .catch((err) => {

        const httpError = new HttpError( err.message, 500 );

        next( httpError );

    });

}

export { getArticles, getArticleById, postArticle, putArticle, deleteArticle, postcomment, getcomments };

**Src/Controller/Pages/Article.js**

import { fetchArticles } from '../../services/Article.js';

const getArticlesPage = ( req, res, next ) => {

    fetchArticles()

        .then(Articles => {

            res.render( 'Articles', {

                Articles: Articles

            });

        });

};

export {

    getArticlesPage

};

**Src/Controller/Pages/Index.js**

const getHomePage = ( req, res, next ) => {

    res.render( 'index' );

};

export {

    getHomePage

};

**Src/Data/Article.json**

[

    {

      "author": "Steve",

      "title": "How to Portion Control (How to Lose Weight with Portion Control)",

      "abstract": "There are all sorts of “tips” and “tricks” out there, but today we’re going to show you the best method.",

      "body": "Leaf rake with 48-inch wooden handleFor many of our Online Coaching clients, we create portion control strategies for weight loss that don’t suck. Today, we’ll share them with you too!.",

      "imageUrl": "hhttps://www.nerdfitness.com/wp-content/uploads/2020/06/seafood-chart-768x276.jpg",

      "createdAt":"January 18, 2022",

      "comments": [

        {

          "commenter": "naruto",

          "title": "awesome",

          "comment": "2022-01-10",

          "created at": "It is a great product"

        }

      ]

    },

    {

      "author": "Steve",

      "title": "How to Portion Control (How to Lose Weight with Portion Control)",

      "abstract": "There are all sorts of “tips” and “tricks” out there, but today we’re going to show you the best method.",

      "body": "Leaf rake with 48-inch wooden handleFor many of our Online Coaching clients, we create portion control strategies for weight loss that don’t suck. Today, we’ll share them with you too!.",

      "imageUrl": "hhttps://www.nerdfitness.com/wp-content/uploads/2020/06/seafood-chart-768x276.jpg",

      "createdAt":"January 18, 2022",

      "comments": [

        {

          "commenter": "naruto",

          "title": "awesome",

          "comment": "2022-01-10",

          "created at": "It is a great product"

        }

      ]

    }

  ]

**Src/Data/Index.js**

import mongoose from 'mongoose';

// create the models

import '../models/Article.js';

import '../models/User.js';

const connectionStr = `mongodb://localhost:27017/online\_store`;

// connect() returns a Promise object

mongoose

    .connect( connectionStr )

    .then(() => {

        console.log( `connected to the DB` );

    })

    .catch(err => {

        console.log( err.message );

    });

**Src/Models/Article.js**

import mongoose from 'mongoose';

import commentSchema from './comments.js';

// schema for Article

const ArticleSchema = new mongoose.Schema({

    comments: {

        type: [ commentSchema ],

    },

    author: {

        type: String,

        required: true,

    },

     title: {

        type: String,

        required: true,

    },

    Abstract: {

        type: String,

        required: false

    },

    body: {

        type: String,

        required: false

    },

    imageUrl: {

        type: String,

        required: false

    },

    createdAt: {

        type: Date,

        required: true,

        default: Date.now

    },

});

// register the model

// collection - Articles

mongoose.model( 'Article', ArticleSchema );

**Src/Model/Comments.js**

import mongoose from 'mongoose';

const commentSchema = new mongoose.Schema({

    commenter: {

        type: string,

        required: true,

    },

    title: {

        type: String

    },

    comment: {

        type: string,

    },

    createdAt: {

        type: Date,

        required: true,

        default: Date.now

    },

});

export default commentSchema;

**Src/Routes/Article.js**

import express from 'express';

import {

    getArticles,

    getArticleById,

    postArticle,

    putArticle,

    deleteArticle,

    postcomments,

    getcomments

} from '../controllers/Articles.js';

// create a router

const router = express.Router();

// any user

router.get(    '/'    , getArticles );

router.get(    '/:\_id', getArticleById );

// authorized for role = 'admin'

router.post(   '/'    ,  postArticle );

router.put(    '/:\_id',  putArticle );

router.delete( '/:\_id',  deleteArticle );

// any user

router.get(    '/:\_id/comments', getcomments );

// authenticated users (logged-in users)

router.post(   '/:\_id/comments',  postcomments);

// module.exports = router

// exporting single item in ES2015 export syntax

export default router;

**Src/Routes/Pages/Article.js**

import express from 'express';

import {

    getArticles,

    getArticleById,

    postArticle,

    putArticle,

    deleteArticle,

    postcomments,

    getcomments

} from '../controllers/Articles.js';

// create a router

const router = express.Router();

// any user

router.get(    '/'    , getArticles );

router.get(    '/:\_id', getArticleById );

// authorized for role = 'admin'

router.post(   '/'    ,  postArticle );

router.put(    '/:\_id',  putArticle );

router.delete( '/:\_id',  deleteArticle );

// any user

router.get(    '/:\_id/comments', getcomments );

// authenticated users (logged-in users)

router.post(   '/:\_id/comments',  postcomments);

// module.exports = router

// exporting single item in ES2015 export syntax

export default router;

**Src/Route/Pages/Index.js**

import express from 'express';

import { getHomePage } from '../../controllers/pages/index.js';

const router = express.Router();

router.get( '/', getHomePage );

export default router;

**Src/Services/Article.cjs**

/\*\*

 \* Service talks to the database (data layer)

 \*/

const Articles = require( '../data/Articles' );

const fetchArticles = () => Articles;

module.exports = {

    fetchArticles

};

**Src/Service/Article.js**

/\*\*

 \* Service talks to the database (data layer)

 \*/

import mongoose from "mongoose";

import config from "../utils/config.js";

const Article = mongoose.model("Article");

const fetchArticles = (sort, order, page, q) => {

  const skipClause = (page - 1) \* config.PAGE\_SIZE;

  const filterClause = {};

  if (q) {

    filterClause.description = {

      $regex: q,

      $options: "i", // ignore case

    };

  }

  console.log(filterClause);

  const sortClause = {

    [sort]: order === "desc" ? -1 : 1,

  };

  return Article

          .find(filterClause)

          .select( 'Author  title abstract body imageUrl createdAt' )

          .sort(sortClause)

          .skip(skipClause)

          .limit(config.PAGE\_SIZE);

};

const fetchArticleById = (\_id) => {

  return Article.findById(\_id);

};

const addArticle = ( Article ) => {

  return Article.create(Article );

};

const updateArticle = ( \_id, newArticleDetails ) => {

  // By default everything is $set

  // {

  //   $set: newProductDetails

  // }

  // NOTE: If you want to remove some property you need to create an updateClause this way and pass it instead to Product.findByIdAndUpdate

  // const updateClause = {

  //   $unset: {

  //     description: true

  //   }

  // };

  return Article.findByIdAndUpdate( \_id, newArticleDetails, { new: true, runValidators: true } );

};

const removeArticle = ( \_id ) => {

  return Article.findByIdAndRemove( \_id );

};

const addcomment = ( \_id, comment ) => {

  return Article.findByIdAndUpdate(

    \_id,

    {

      $push: {

        comments: comment

      }

    },

    {

      new: true,

      runValidators: true

    }

  );

};

const fetchcomment = ( \_id ) => {

  return Article

          .findById( \_id )

          .select( 'comments' )

          .then(Articlecomments => Articlecomments.comments)

}

export {

  fetchArticles,

  fetchArticleById,

  addArticle,

  updateArticle,

  removeArticle,

  addcomment,

  fetchcomment

};

**Src/until/Config.js**

const config = {

    PAGE\_SIZE: 5

};

export default config;

**Src/Until/HttpError.js**

class HttpError extends Error {

    constructor( message, statusCode ) {

        super( message ); // this.message = message;

        this.statusCode = statusCode;

    }

};

export default HttpError;

**Src/Views/Article.ejs**

<%- include( "header" ) -%>

<h1 class="my-5">

    List of Article

    <br />

    <small class="h4">The Awesome Online Store</small>

</h1>

<div class="row">

    <% for( let i = 0; i < Article.length; i++ ) { %>

        <div class="col-12 col-lg-3 my-3">

            <div class="card p-4">

                <img src="<%=Articles[i].imageUrl%>" class="card-img-top" style="width: 40%; margin: 0 auto;">

                <div class="card-body">

                  <h5 class="card-title">

                    <%=Articles[i].name%>

                  </h5>

                  <div class="card-text">

                    <%=Articles[i].description%>

                  </div>

                  <a href="#" class="btn btn-primary">Go somewhere</a>

                </div>

              </div>

        </div>

    <% } %>

</div>

<%- include( "footer" ) -%>

**Src/Views/footer.ejs**

        <script src="bootstrap-5.1.3-dist/js/bootstrap.min.js"></script>

    </body>

</html>

**Src/Views/Header.ejs**

<!-- shortcut - ! -->

<!DOCTYPE html>

<html lang="en">

    <head>

        <meta charset="UTF-8">

        <meta http-equiv="X-UA-Compatible" content="IE=edge">

        <meta name="viewport" content="width=device-width, initial-scale=1.0">

        <title>Amazin</title>

        <link rel="stylesheet" href="/bootstrap-5.1.3-dist/css/bootstrap.min.css">

    </head>

    <body class="container">

        <%- include( "navigation-menu" ) -%>

**Src/Views/Index.ejs**

<%- include( "header" ) -%>

<h1 class="my-5">

    Amazin

    <br />

    <small class="h4">The Awesome Online Store</small>

</h1>

<p>

    Welcome to Amazin. You can search through more than 2 crore products. If you cannot find something here, you cannot get it anywhere else!

</p>

<p>

    Author: narutouzumaki

</p>

<p>

    Tech stack:

    <ul>

        <li>Node JS</li>

        <li>Express</li>

        <li>MongoDB using Mongoose</li>

        <li>EJS</li>

    </ul>

</p>

<%- include( "footer" ) -%>

**Src/Views/Nagigation-menu.ejs**

<nav class="navbar navbar-expand-lg navbar-dark bg-primary">

  <div class="container-fluid">

    <a class="navbar-brand" href="/">Amazin</a>

    <button

      class="navbar-toggler"

      type="button"

      data-bs-toggle="collapse"

      data-bs-target="#navbarNav"

      aria-controls="navbarNav"

      aria-expanded="false"

      aria-label="Toggle navigation"

    >

      <span class="navbar-toggler-icon"></span>

    </button>

    <div class="collapse navbar-collapse" id="navbarNav">

      <ul class="navbar-nav">

        <li class="nav-item">

          <a class="nav-link" aria-current="page" href="/page/Articls">Articles</a>

        </li>

      </ul>

    </div>

  </div>

</nav>

**Src/index.js**

// set up temporary environment variables

import dotenev from 'dotenv';

dotenev.config();

// connect to the DB

import './data/index.js';

import express from 'express';

import ArticlesRouter from './routes/Articles.js';

import indexRouter from './routes/pages/index.js';

import ArticlesPageRouter from './routes/pages/Articles.js';

import HttpError from './utils/HttpError.js';

import fs from 'fs';

import path from 'path';

const app = express();

app.set( 'views', path.join( process.cwd(), 'src', 'views' ) );

app.set( 'view engine', 'ejs' );

// custom middleware

app.use(( req, res, next ) => {

    const logMessage = `${req.method} ${req.url} at time ${new Date().toTimeString()}\n`;

    fs.writeFile( path.join( process.cwd(), 'src', 'server.log' ), logMessage, { encoding: 'utf-8', flag: 'a+' }, ( err ) => {

        if( err ) {

            console.log( err.message );

        }

        next(); // pass control to the next middleware

    });

});

app.use( express.static( path.join( process.cwd(), 'src', 'public' ) ) );

// extracts JSON data from request body and sets it up on req.body

app.use( express.json() );

// extracts JSON data from a form and sets it up on req.body

app.use( express.urlencoded( { extended: false } ) );

app.use( '/Articles', ArticlesRouter );

app.use( '/', indexRouter );

app.use( '/page/Articles', ArticlesPageRouter );

app.use(( req, res, next ) => {

    const httpError = new HttpError( 'Resource not found', 404 );

    next( httpError );

});

// error-handling middleware

app.use(( httpError, req, res, next ) => {

    res.status( httpError.statusCode || 500 ).json({

        message: httpError.message

    });

});

const PORT = process.env.PORT || 4000;

app.listen( PORT, err => {

    if( err ) {

        console.log( error.message );

        return;

    }

    console.log( `Started http://localhost:${PORT}` );

});

**Src/Index.old.js**

import express from 'express';

const app = express();

app.get( '/products', ( req, res ) => {

    res.send( 'We will send the list of Articles' );

});

// Send a message when client requests for /users

app.get( '/users', ( req, res ) => {

    res.send( 'We will send the list of users' );

});

const PORT = process.env.PORT || 4000;

app.listen( PORT, err => {

    if( err ) {

        console.log( error.message );

        return;

    }

    console.log( `Started http://localhost:${PORT}` );

});