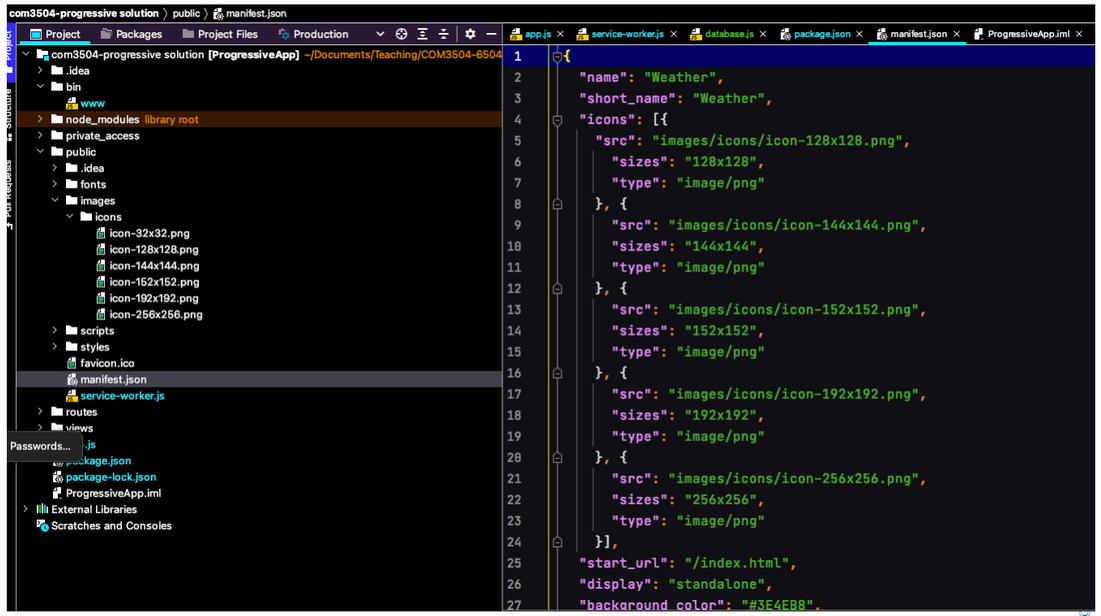
Week 5 Lab Class: PWA

PWA

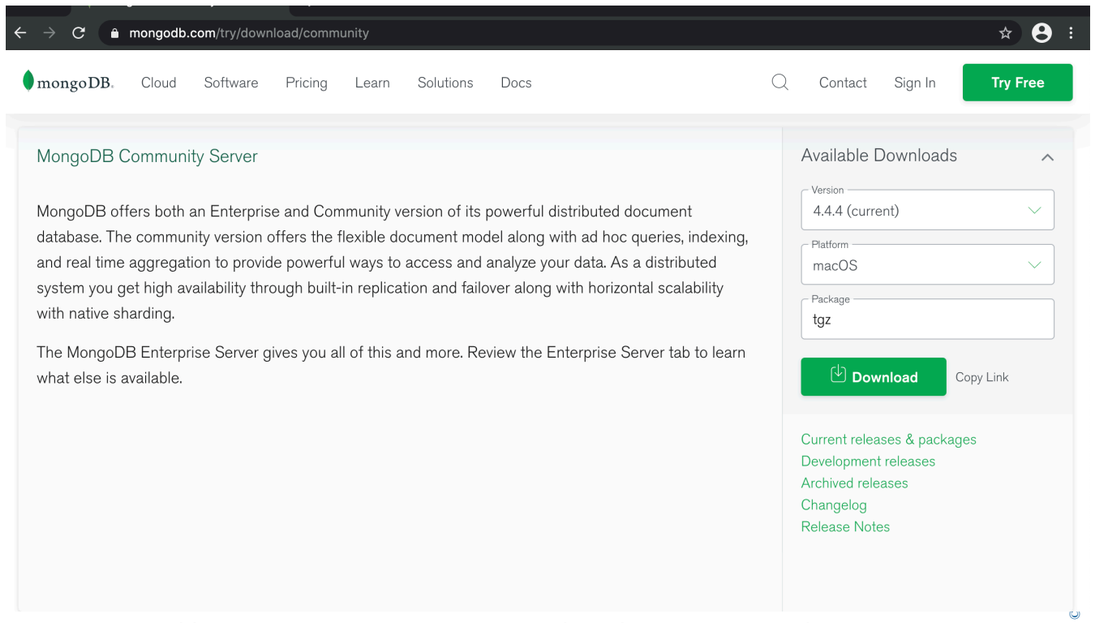
- You are given a PWA
  - it is the usual one we used last week for the service workers and the Indexed DB
  - Inspect:
    - the manifest file
    - the icons
- There is not a specific exercise - just make sure to understand how a PWA works



Week 3 Lab Class: MongoDB

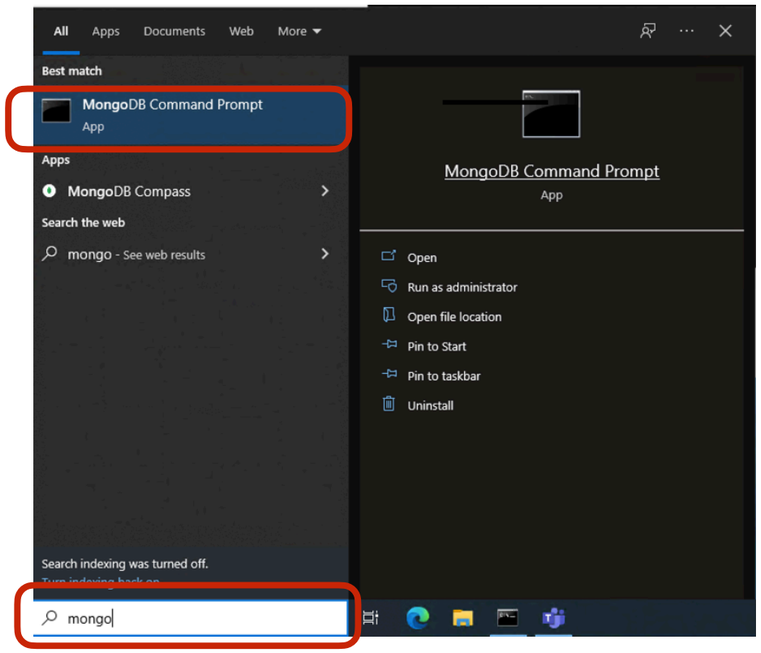
Prof Fabio Ciravegna
Department of Computer Science,
The University of Sheffield
f.ciravegna@shef.ac.uk

Download MongoDb

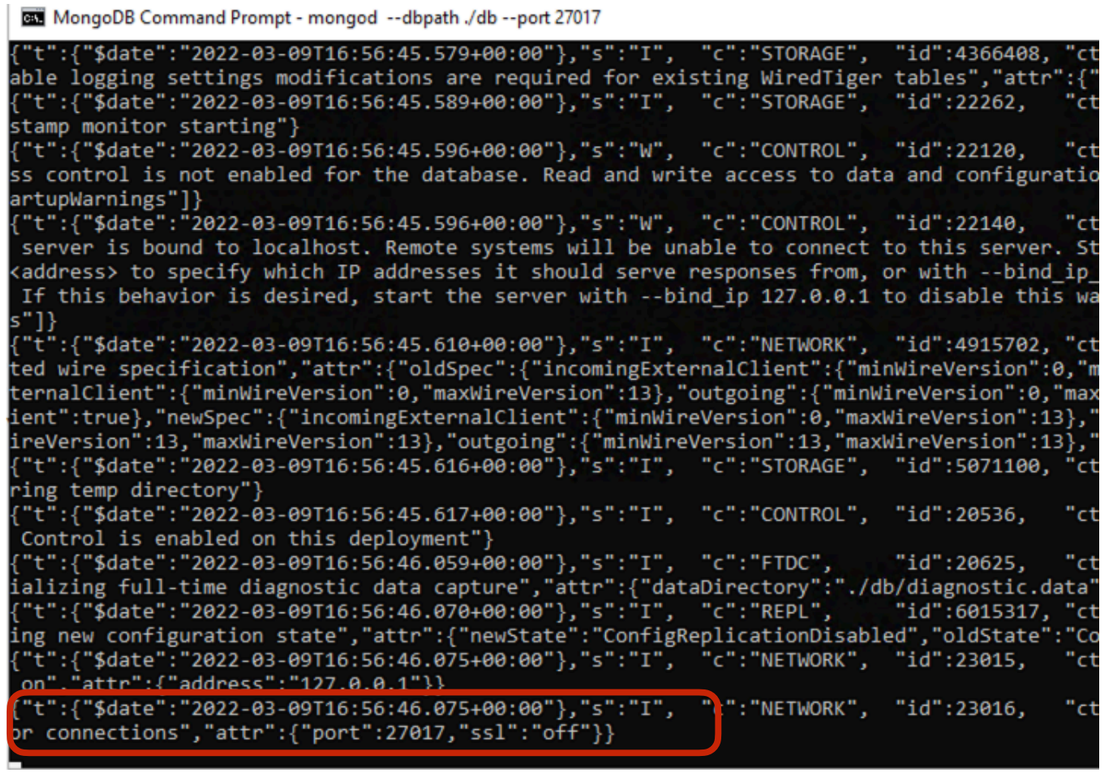


https://www.mongodb.com/try/download/community

Windows



- to start mongo db
  - open the mongodb prompt
  - C:\> mkdir data/db
  - C:\> cd data
  - C:\> mkdir db
  - C:> >mongod --dbpath db --port 27017
    - this will make sure that mongo is on port 27017 and that the db directory is data/db



on a Mac

```
administrator@Saturn Downloads % tar -xvzf mongodb-macos-x86_64-4.4.4.tgz
x mongodb-macos-x86_64-4.4.4/LICENSE-Community.txt
x mongodb-macos-x86_64-4.4.4/MPL-2
x mongodb-macos-x86_64-4.4.4/README
x mongodb-macos-x86_64-4.4.4/THIRD-PARTY-NOTICES
x mongodb-macos-x86_64-4.4.4/bin/install_compass
x mongodb-macos-x86_64-4.4.4/bin/mongo
x mongodb-macos-x86_64-4.4.4/bin/mongod
x mongodb-macos-x86_64-4.4.4/bin/mongos
administrator@Saturn Downloads %
```

Move the created folder into a suitable place, e.g. your home directory or somewhere under /lib

on a Mac (2)

- mkdir ~/data/db
- sudo <**path to your mongo instance**>/bin/mongod --dbpath ~/data/db --port 27017
- if you want to keep it running:
- sudo <*path to your mongo instance*>/bin/mongod --dbpath /data/db --port 27017 — fork --logpath mongolog.log

Today's Lab Class

- We will modify one of the exercises from week 1
  - the one about getting the age of Micky Mouse
  - however, this time we will use MongoDb to get the age of Mickey Mouse
- We will first analyse a potential solution
  - Then you will be required to modify the solution to enhance it with an additional functionality

A Past Exercise (modified)

Get Character Age

First name:

Last name:

Year of Birth:

Submit

routes/index.js

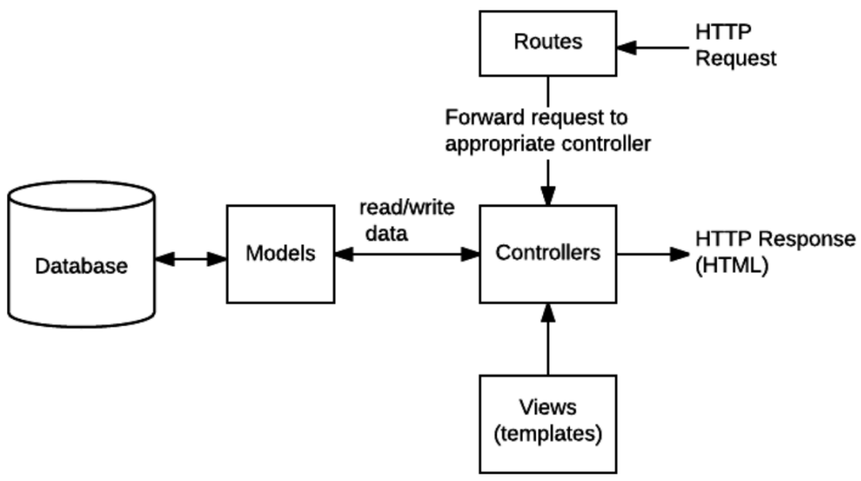
```
router
    .get('/index', function (req, res, next) {
        res.render('index', {title: 'Get Character Age'});
    })
    .post('/index', character.getAge);

module.exports = router;
```


Axios call

```
function sendAxiosQuery(url, data) {
    axios.post(url, data)
        .then((dataR) => {// no need to JSON parse the result, as we are using
            // we need to JSON stringify the object
            document.getElementById( elementId: 'results').innerHTML = JSON.stringify(dataR.data)
        })
        .catch(function (response) {
            alert(response.toJSON());
        })
}
```


Using the db

- Remember the nodes organisation for



The program

- The program is organised in that way
- There is a database called 'characters'
  - which has a model called 'Character' representing name, surname and year of birth of each character

controllers

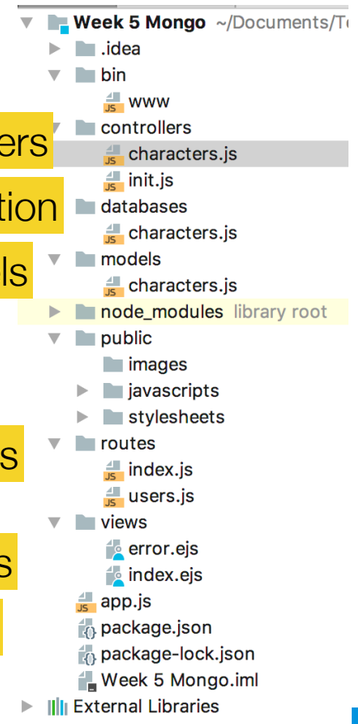
database definition

models

routes

views

app



databases/characters.js

- Containing the creation/connection to the db

```
const mongoose = require('mongoose');

//The URL which will be queried. Run "mongod.exe" for this to connect
//var url = 'mongodb://localhost:27017/test';
const mongoDB = 'mongodb://localhost:27017/characters';

mongoose.Promise = global.Promise;

connection = mongoose.connect(mongoDB, {
    useNewUrlParser: true,
    useUnifiedTopology: true,
    checkServerIdentity: false,
})
    .then(() => {
        console.log('connection to mongodb worked!');
    })
    .catch((error) => {
        console.log('connection to mongodb did not work! ' + JSON.stringify(error));
    });
```

it creates the db if not existing

it runs either on port 27017 or 27019 modify to fit your computer's

load the DB in www/bin

```
/**
 * Module dependencies.
 */

var app = require('../app');
var debug = require('debug')('week-5-mongo:server');
var http = require('http');
var database= require('../databases/characters')
```


Schema and Model

- under models/character.js

```
const mongoose = require('mongoose');

const Schema = mongoose.Schema;

const Character = new Schema(
    {
        first_name: {type: String, required: true, max: 100},
        family_name: {type: String, required: true, max: 100},
        dob: {type: Number},
        whatever: {type: String} //any other field
    }
);

// Virtual for a character's age
Character.virtual('age')
    .get(function () {
        const currentDate = new Date().getFullYear();
        const result= currentDate - this.dob;
        return result;
    });

Character.set('toObject', {getters: true, virtuals: true});

module.exports = mongoose.model('Character', Character
```

schema definition

dob will contain the year of birth (e.g. 1908)

age is a dynamic value (it changes every year) so we define it as a dynamic field

remember to export

routes/index.js

- The code is moved from the routes file to the controllers

```
const express = require('express');
const router = express.Router();

var character = require('../controllers/characters');
var initDB = require('../controllers/init');
initDB.init();


/* GET home page. */
router
    .get( path: '/index', handlers: function (req : Request<P, ResBody, ReqBody, R
        res.render( view: 'index', options: {title: 'Get Character Age'});
    })
    .post( path: '/index', character.getAge);

module.exports = router;
```

load the controllers (see next slide)

when a post arrives call the function getAge in the controller

Temporary init (controllers/init.js)

- We initially load the initial data for Mickey Mouse
- we will add new elements dynamically later on
- RUN these few lines just once and then comment them out

```
const mongoose = require('mongoose');
const Character = require('../models/characters');

exports.init= function() {
    // uncomment if you need to drop the database

    // Character.remove({}, function(err) {
    //     console.log('collection removed')
    // });

    const dob=new Date(1908, 12, 1).getFullYear();
    let character = new Character({
        first_name: 'Mickey',
        family_name: 'Mouse',
        dob: dob
    });
    // console.log('dob: '+character.dob);

    character.save()
        .then ((results) => {
            console.log("object created in init: "+ JS
        })
        .catch ((error) => {
            console.log(JSON.stringify(error));
        });
}
```

load the model

create a character

save it into the DB

controllers/characters.js

```
let Character = require('../models/characters');

exports.getAge = function (req, res) {
    let userData = req.body;
    if (userData == null)
        res.status(403).json('No data sent!')

    Character.find({first_name: userData.firstname, family_name: userData.lastname},
        'first_name family_name dob age')
        .then(characters => {
            let character = null;
            if (characters.length > 0) {
                let firstElem = characters[0];
                character = {
                    name: firstElem.first_name, surname: firstElem.family_name,
                    dob: firstElem.dob, age: firstElem.age
                };
                res.json(character.age);
            } else {
                res.json("not found");
            }
        })
        .catch((err) => {
            res.status(500).send('Invalid data or not found!' + JSON.stringify(err));
        });
}
```

called by the post in routes/index.js

querying Mongo using name and surnar

returning name, surname, dob and age

mongo returns a list - get the first element

map the received structure into the output structure

return from the server!!!

That's it!

If you run it on http://localhost:3004
with Mickey Mouse as input

you will get his age

Part 2

Insert elements dynamically

Create a new form

- That enables inserting characters
  - this will be a post to /insert
- The form will be in a new ejs view
  - call it views/insert.ejs
  - the form will be identical to the one used for querying
    - so just copy index.ejs into insert.ejs
- The axis call will be identical
  - you just need to point the URL to a new route
    - let's call this route /insert
    - I suggest you modify the onSubmit method in javascript/index.js to receive a parameter which is the url.
    - For example

```
in index.ejs <form id="xForm" onsubmit="onSubmit('/index')">
in insert.ejs <form id="xForm" onsubmit="onSubmit('/insert')">
```

    - then modify the method onSubmit in javascripts/index.js

- Steps:
  - define a route under routes/index.js
    - for both get and post

```
/* GET home page. */
router.get('/insert', function(req, res, next) {
  res.render('insert', { title: 'My Form' });
});

router.post('/insert', character.insert);
```

  - define a new exported function under controllers/characters.js

```
exports.insert = function (req, res) {…}
```

    - here you will call the save MongoDB method
      - hint - see controller/init.js for suggestions

Now insert Minnie Mouse

My Form

First name: Minnie

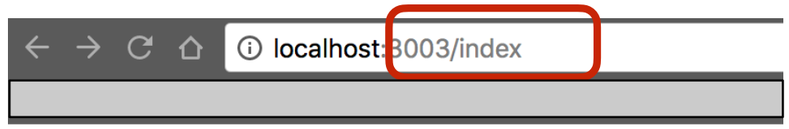
Last name: Mouse

Year of Birth: 1900

Submit

{"_id":"5ab2efe56dbc38bb61d4ca1e","first_name":"Minnie","family_name":"Mouse","dob":1900,"

Now search for Minnie in the DB



My Form

First name: Minnie

Last name: Mouse

Year of Birth: 1900

Submit

{"name":"Minnie","surname":"Mouse","dob":1900,"age":118}

The University Of Sheffield.

Good Luck

The solution is on Blackboard