Week 5 Lab Class: MongoDB

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

© Fabio Ciravegna, University of Sheffield

Mongodb

- Open start and select Mongo DB
  - this will open a shell
- In the shell create the directory

```
U:\>mkdir data\db
```

- Then call mongod.exe

```
U:\>mongod.exe
2018-03-23T15:48:18.369+0000 I CONTROL  [initand
t host=TENC8CBB8085E16
2018-03-23T15:48:18.369+0000 I CONTROL  [initand
```

© Fabio Ciravegna, University of Sheffield

The University Of Sheffield.

Mongodb is on port

C:\Windows\system32\cmd.exe - mongod

```
C:\Program Files\MongoDB\Server\3.2\bin>mongod
2016-05-31T19:32:06.016+0530 I CONTROL  [main] Hotfix KB2731284 or later update
is not installed, will zero-out data files
2016-05-31T19:32:06.018+0530 I CONTROL  [initandlisten] MongoDB starting : pid=6
804 port=27017 dbpath=C:\data\db\ 64-bit host=INDLAPTOP0312
2016-05-31T19:32:06.018+0530 I CONTROL  [initandlisten] targetMinOS: Windows 7/W
indows Server 2008 R2
2016-05-31T19:32:06.018+0530 I CONTROL  [initandlisten] db version v3.2.6
2016-05-31T19:32:06.018+0530 I CONTROL  [initandlisten] git version: 05552b562c7
a0b3143a729aaa0838e558dc49b25
2016-05-31T19:32:06.018+0530 I CONTROL  [initandlisten] OpenSSL version: OpenSSL
 1.0.1p-fips 9 Jul 2015
2016-05-31T19:32:06.018+0530 I CONTROL  [initandlisten] allocator: tcmalloc
2016-05-31T19:32:06.018+0530 I CONTROL  [initandlisten] modules: none
2016-05-31T19:32:06.018+0530 I CONTROL  [initandlisten] build environment:
2016-05-31T19:32:06.018+0530 I CONTROL  [initandlisten]     distmod: 2008plus-ss
l
2016-05-31T19:32:06.019+0530 I CONTROL  [initandlisten]     distarch: x86_64
2016-05-31T19:32:06.019+0530 I CONTROL  [initandlisten]     target_arch: x86_64
2016-05-31T19:32:06.019+0530 I CONTROL  [initandlisten] options: {}
2016-05-31T19:32:06.021+0530 I -        [initandlisten] Detected data files in C
:\data\db\ created by the 'wiredTiger' storage engine, so setting the active sto
rage engine to 'wiredTiger'.
2016-05-31T19:32:06.022+0530 I STORAGE  [initandlisten] wiredtiger_open config:
create,cache_size=4G,session_max=20000,eviction=(threads_max=4),config_base=fals
e,statistics=(fast),log=(enabled=true,archive=true,path=journal,compressor=snapp
y),file_manager=(close_idle_time=100000),checkpoint=(wait=60,log_size=2GB),stati
stics_log=(wait=0),
2016-05-31T19:32:06.848+0530 I NETWORK  [HostnameCanonicalizationWorker] Startin
g hostname canonicalization worker
2016-05-31T19:32:06.848+0530 I FTDC     [initandlisten] Initializing full-time d
iagnostic data capture with directory 'C:/data/db/diagnostic.data'
2016-05-31T19:32:06.854+0530 I NETWORK  [initandlisten] waiting for connections
on port 27017
```

© Fabio Ciravegna, University of Sheffield

On a mac

- Download mongo to your computer
- launch mongo
  - 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

© Fabio Ciravegna, University of Sheffield

Today's Lab Class

- We will modify the exercise 1 from week 2
  - 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

© Fabio Ciravegna, University of Sheffield

The Week 2's exercise

My Form

First name: Mickey

Last name: Mouse

Year of Birth:

Submit

{"name":"Mickey","surname":"Mouse","dob":1909,"age":109}

```
function sendAjaxQuery(url, data) {
    $.ajax({
        url: url ,
        data: data,
        dataType: 'json',
        type: 'POST',
        success: function (dataR) {
            // no need to JSON parse the result, as we are using
            // dataType:json, so JQuery knows it and unpacks the
            // object for us before returning it
            var ret = dataR;
            // in order to have the object printed by alert
            // we need to JSON stringify the object
            document.getElementById('results').innerHTML= JSON.stringify(ret);
        },
        error: function (xhr, status, error) {
            alert('Error: ' + error.message);
```

© Fabio Ciravegna, University of Sheffield

routes/index.js

body-parser to access the data from req

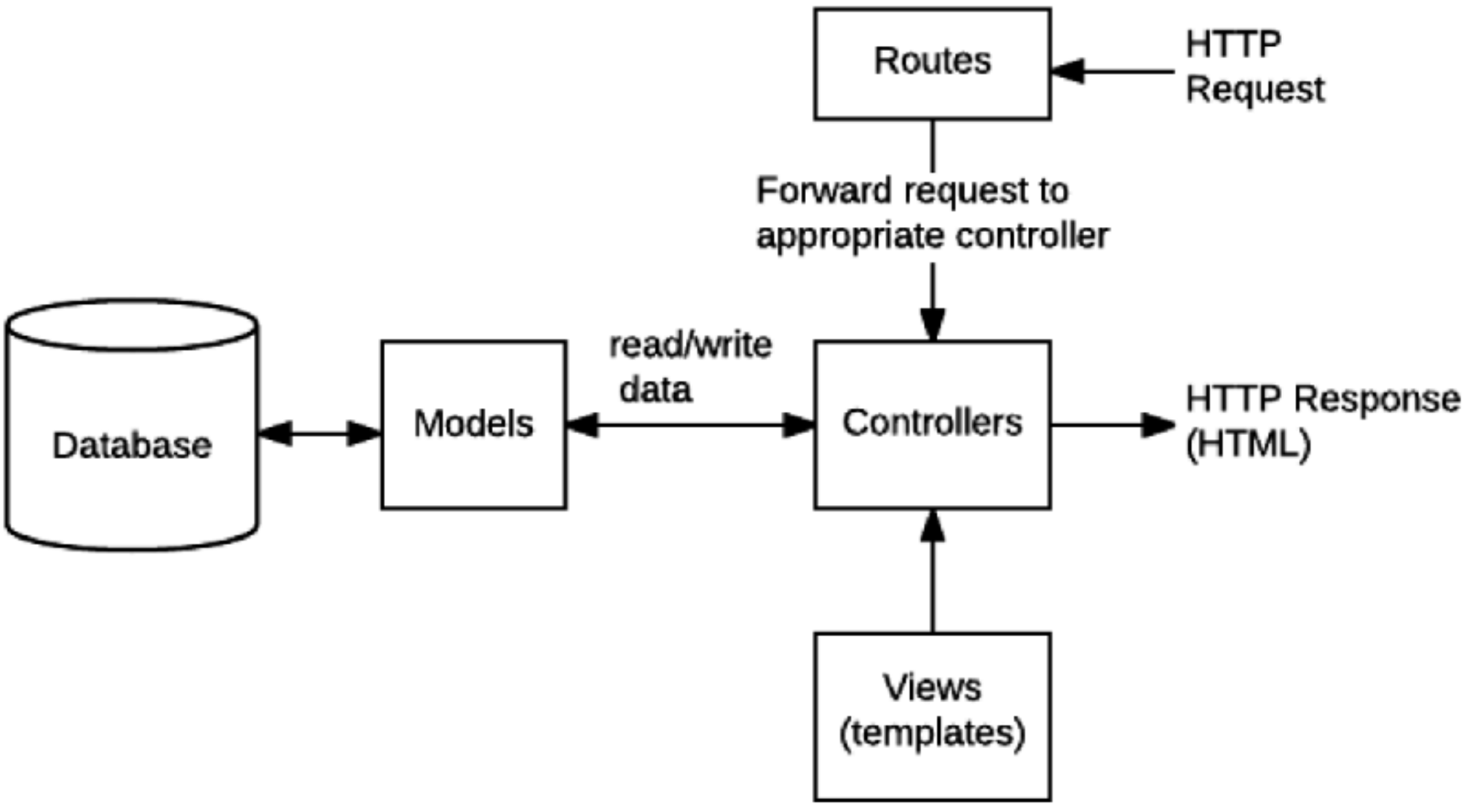
```
router.post('/index', function(req, res, next) {
    var userData = req.body;
    if (userData == null) {
        res.status(403).send('No data sent!')
    } else if (!isNumeric(userData.year)) {
        res.status(403).send('Year is invalid!')
    }
    const year = (new Date()).getFullYear()
    userData.age = year - parseInt(userData.year);
    res.setHeader('Content-Type', 'application/json');
    res.send(JSON.stringify(userData));
});
```

Calculation of age from the data provided

© Fabio Ciravegna, University of Sheffield

Using the db

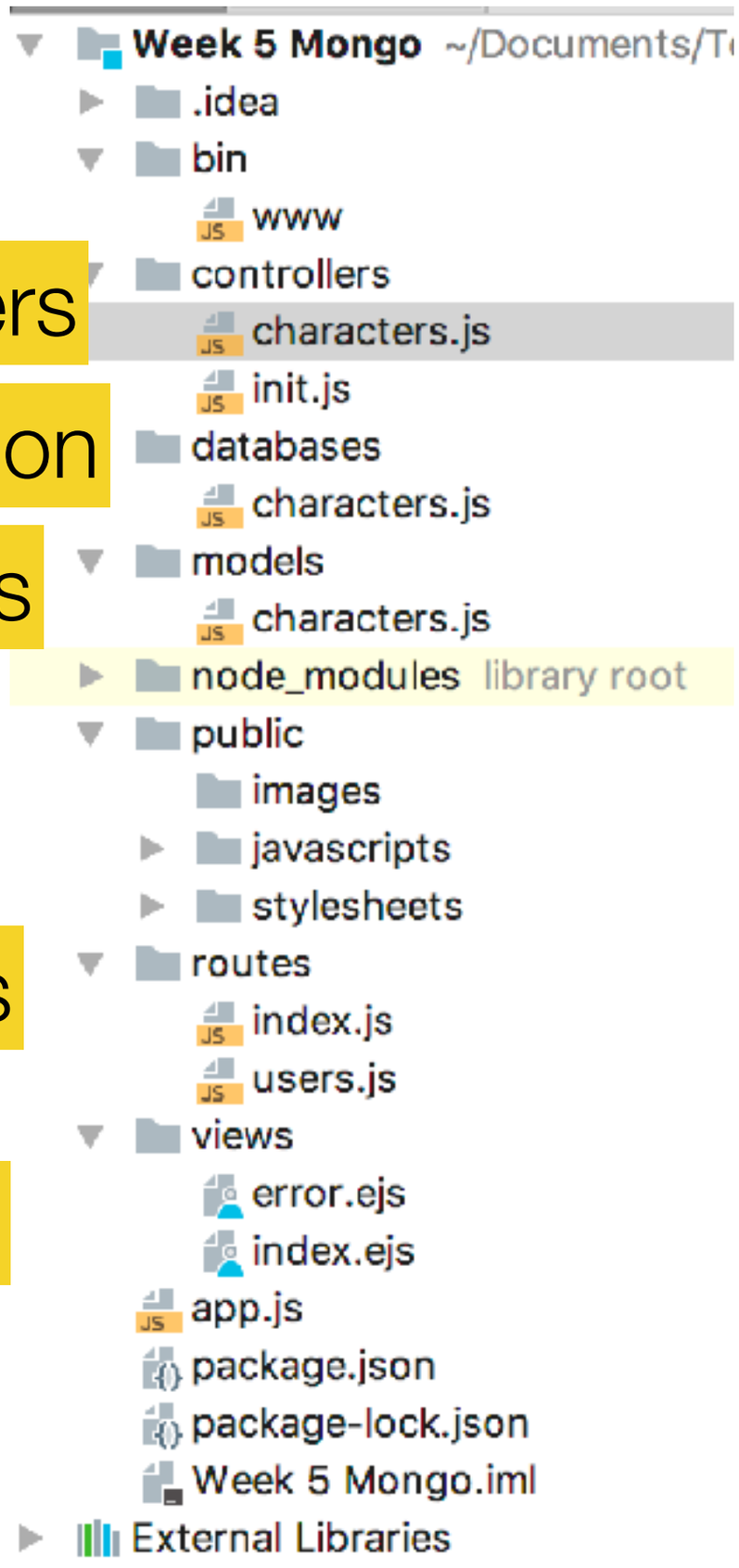
- Remember the nodes organisation for MongoDB



© Fabio Ciravegna, University of Sheffield

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



© Fabio Ciravegna, University of Sheffield

databases/characters.js

- Containing the definition of the db

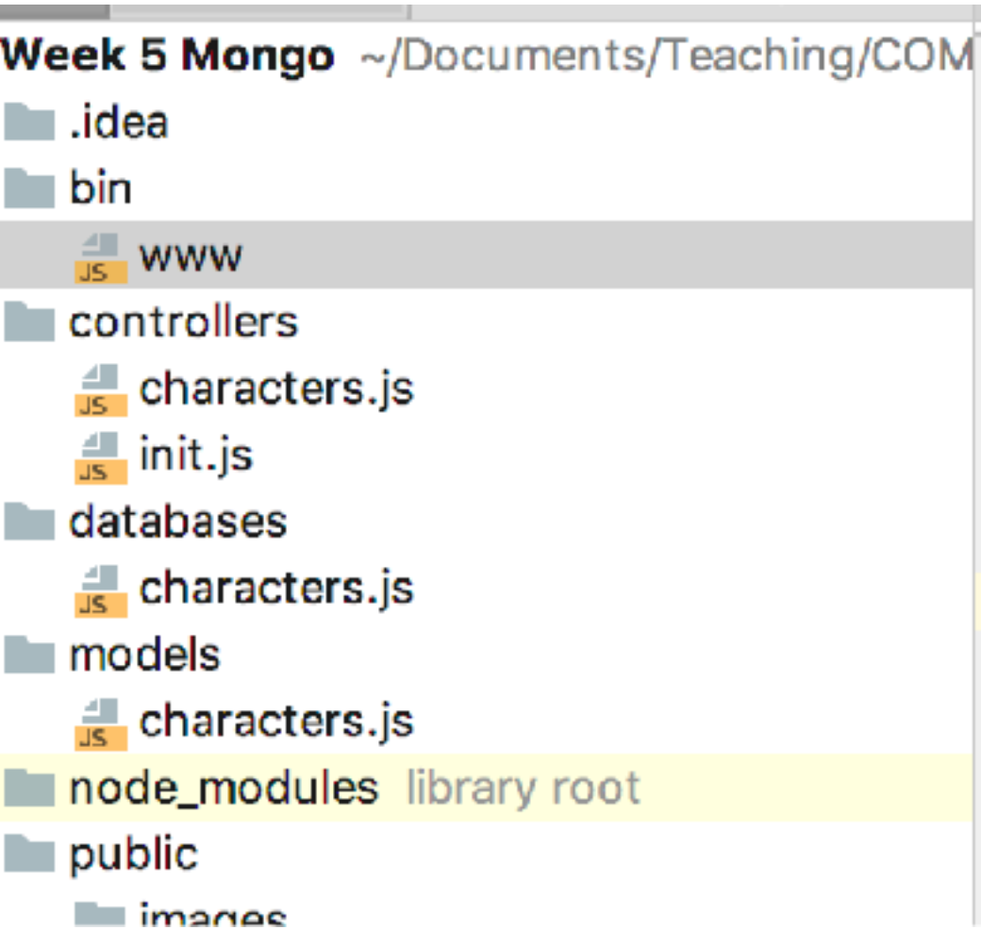
```
var mongoose = require('mongoose');
var ObjectId = require('mongodb').ObjectID;
var bcrypt = require('bcryptjs');

//The URL which will be queried. Run "mongod.exe" for this to connect
//var url = 'mongodb://localhost:27017/test';
var mongoDB = 'mongodb://localhost:27019/characters';

mongoose.Promise = global.Promise;
mongoose.connect(mongoDB);
var db = mongoose.connection;
//Bind connection to error event (to get notification of connection errors)
db.on('error', console.error.bind(console, 'MongoDB connection error:'));

// db.dropDatabase();

//  MORE GENERAL WAY WOULD BE TO CALL:
// try {
//     var connection = mongoose.createConnection(mongoDB);
//     console.log("connection to mongodb worked!");
// }catch (e) {
// console.log('error in db connection: ' +e.message)
// }
//
// WHICH WOULD ALLOW MULTIPLE CONNECTIONS
```

it is either 27017 or 27019

it creates the db if not existing

© Fabio Ciravegna, University of Sheffield

load it in www/bin




```
1   #!/usr/bin/env node
2
3   /**
4    * Module dependencies.
5    */
6
7   var app = require('../app');
8   var debug = require('debug')('week-5-mongo:server
9   var http = require('http');
10  var database= require('../databases/characters')
11  /**
12   * Get port from environment and store in Express
13   */
14
15  var port = normalizePort(process.env.PORT || '300
```

© Fabio Ciravegna, University of

Schema and Model

- under models/character.js

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

var Schema = mongoose.Schema;

var 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}
    }
);

// 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

© Fabio Ciravegna, University of Sheffield

routes/index.js

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

```
var express = require('express');
var router = express.Router();
var bodyParser= require("body-parser");
```

load the controllers (see next slide)

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


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

router.post('/index', character.getAge);

module.exports = router;
```

when a post arrives call the function getAge in the controller

© Fabio Miravegna, University of Sheffield

Temporary init

- We initially load the data for Mickey Mouse
- we will add new elements dynamically later on
- RUN this just once and then comment it

```
var mongoose = require('mongoose');
var 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();
    var character = new Character({
        first_name: 'Mickey',
        family_name: 'Mouse',
        dob: dob
    });
    console.log('dob: '+character.dob);

    character.save(function (err, results) {
        console.log(results._id);
    });
}
```

load the model

create a character

save it into the DB

controllers/characters.js

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

exports.getAge = function (req, res) {
    var userData = req.body;
    if (userData == null) {
        res.status(403).send('No data sent!')
    }
    try {
        Character.find({first_name: userData.firstname, family_name: userData.lastname},
            'first_name family_name dob age',
            function (err, characters) {
                if (err)
                    res.status(500).send('Invalid data!');
                var character =null;
                if (characters.length>0) {
                    var firstElem = characters[0];
                    character = {name: firstElem.first_name, surname: firstElem.family_name,
                        dob: firstElem.dob, age: firstElem.age};
                }
                res.setHeader('Content-Type', 'application/json');
                res.send(JSON.stringify(character));
            });
    } catch (e) {
        res.status(500).send('error '+ e);
    }

}
```

called by the post in routes/index.js

querying Mongo using name and surname

returning name, surname, dob and age

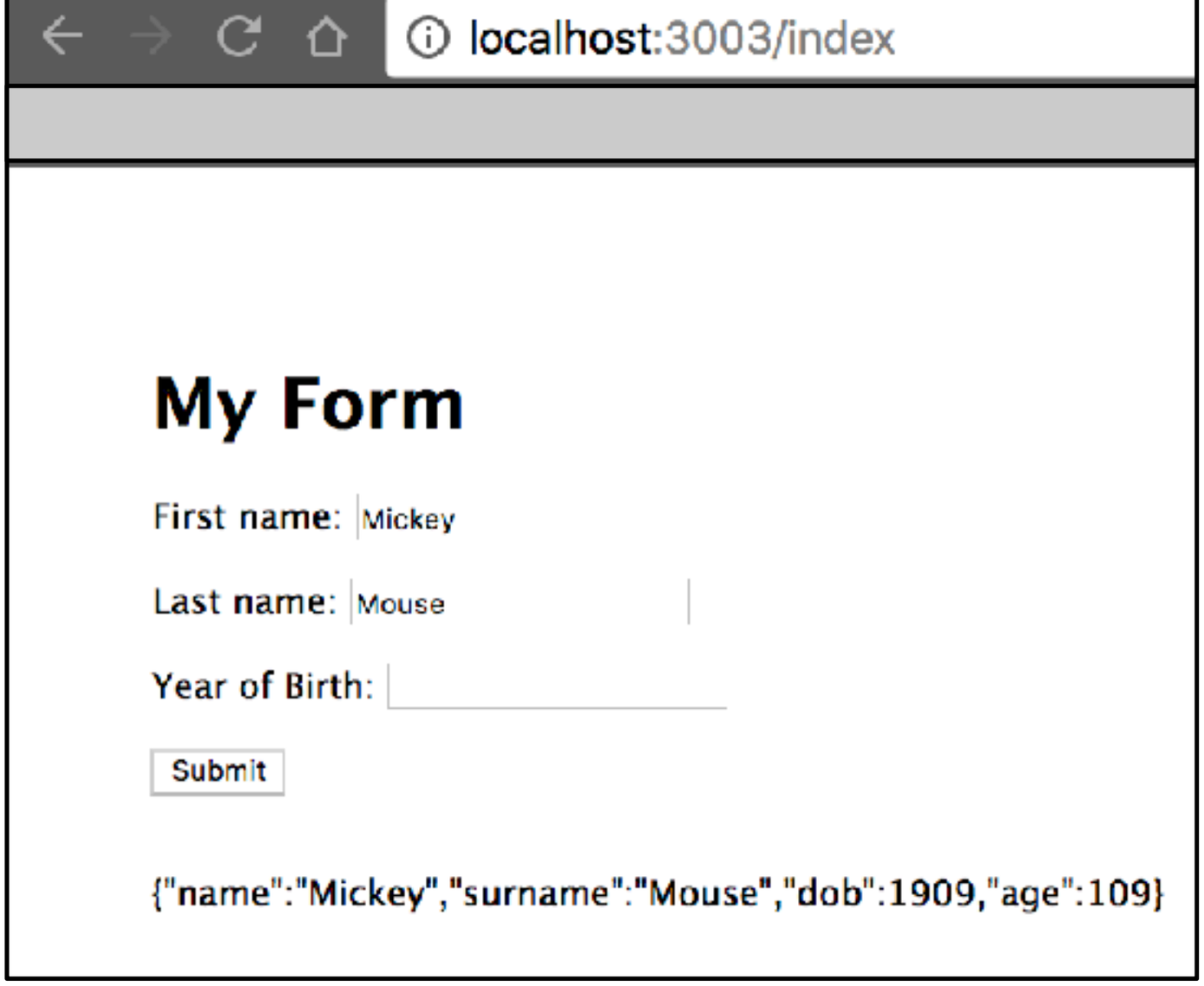
mongo returns a list - get the first element

map the received structure into the output structure

© Fabio Ciravegna, University of Sheffield

The University Of Sheffield.

That's it!



If you run it on http://localhost:3000
(check the port under www/bin, as I may have changed it in the solution - I see in the image above it is 3003!!)
with Mickey Mouse as input

you will get his age

© Fabio Ciravegna, University of Sheffield

The University Of Sheffield.

Part 2

Insert elements dynamically

© Fabio Ciravegna, University of Sheffield

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 ajax 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

© Fabio Ciravegna, University of Sheffield

- 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

© Fabio Ciravegna, University of Sheffield

Now insert Minnie Mouse

My Form

First name: Minnie

Last name: Mouse

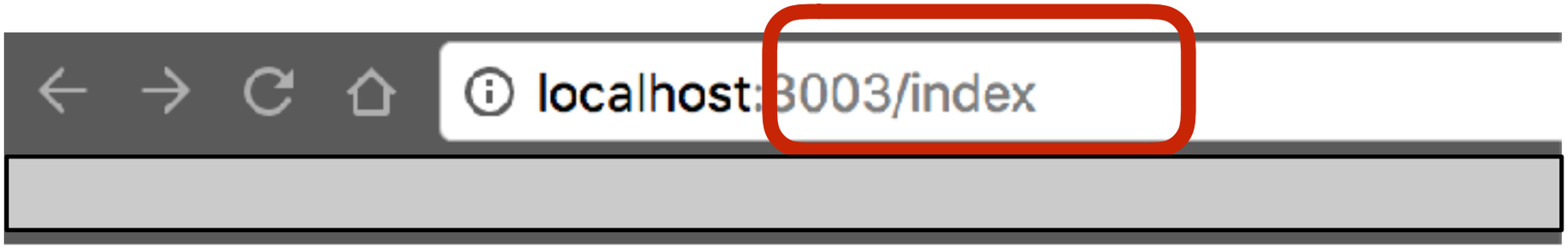
Year of Birth: 1900

Submit

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

© Fabio Ciravegna, University of Sheffield

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}

© Fabio Ciravegna, University of Sheffield

Good Luck

The solution will be on Mole in the next hours

© Fabio Ciravegna, University of Sheffield