Day04 Deployment

part 1 – Folder Setup

1. Copy the folder called **Part12** from **Day03** paste it where all the other days are and rename it to **Day04**.
2. Copy the folder from **Day02** paste it inside of the folder you just renamed to **Day04**. Rename **Day02** inside of **Day04** to just **HTML**. It should contain all the HTML/JS/CSS from the second day of sessions.
3. Your **Day04** folder should look like the image below, before running **npm install**.

|  |
| --- |
|  |

1. Run **npm install** to create the **node**\_**modules** folder and setup the environment.
2. First task is to serve the **HTML** pages we built on **Day02**. Now that we have a **controller** file, everything that goes to the client will pass through there and the root function will handle serving our HTML page when users land on our root which at the moment is <http://localhost/8000>. We would need to change line 3 of controller.js to serv the index.html file instead.

|  |
| --- |
| **const Weight = require('../models/employees);**  **exports.getdefault=function(req, res){**  **res.send(‘../HTML/index.html’);**  **};** |

1. Execute **node http\_server** and go to the address from #5. Of course this will not work because whatever is between the single quotes will simply print out on the browser screen. Node has a different function to handle html files, it is called **sendFile()** and it is attached to the response object.

|  |
| --- |
| **const Weight = require('../models/employees);**  **exports.getdefault=function(req, res){**  **res.sendFile(‘../HTML/index.html’);**  **};** |

Remember to stop and start the service

1. This did not work but it provided some clues, something about a path. In order to serve static pages, we need the **path** package, so in the controller.js file, declare a variable and point it to the **path** package.

|  |
| --- |
| **const Weight = require('../models/employees);**  **const path = require("path");**  **exports.getdefault=function(req, res){** |

1. The path object has a method called **join()** which we can use to obtain the current path of the application. If we then concatenate the root path with the path where our **HTML** files live, we can finally obtain a true absolute path to our files

|  |
| --- |
| **const path = require("path");**  **exports.getdefault=function(req, res){**  **res.sendFile(path.join(\_\_dirname + '/../HTML/index.html'));**  **};** |

1. Although the html file is served, it appears to not know that CSS and JS exists, we need to let Express know that these files exist and that it should use them. Open routes.js and include the following lines:

|  |
| --- |
| **module.exports = function(app){**  **const express= require('express');**  **app.use(express.static(\_\_dirname + '/../HTML'));**  **let controller = require('../controllers/controller');** |

Although we had to require **Express** in the http\_server.js file, we still have to do it again in this file. Also we just need the static method to know where the directory is that contains our html/css/js files.

Note: if the above code does not work in U18, use this instead:  
**app.use(express.static(\_\_dirname + './../HTML'));**

1. At this point, if you navigate to team weights in the navigation menu, you should see some records there already. If you do not see anything then you would need to complete the teamweights.html file.
2. Open the scripts.js file inside of the scripts folder. Make sure that the AngularJS controller is obtaining its data from the URL and not the JSON text file.

|  |
| --- |
| **let file = "json.txt";**  **let url = "http://localhost:8000/getallrecords";**  **//**  **let app = angular.module('SkillsApp', [] );**  **app.controller('Weights', function($scope, $http) {**  **$http.get(url).then(function(response){**  **$scope.allWeights = response.data;** |

part 2 – Configuring Team Weights

1. We did not have to do much work with teamweights.html, but we can improve the display a bit. Open that html file and go to where the **h2** element is, should be around line 25. Take note of the name of the div tag that will be used to build the new HTML that will then display the documents from our database.

|  |
| --- |
| **<h2>Showing records for team</h2>**  **<div id="records"></div>**  **<button id="getData">Get Records</button>** |

Give it an id as well.

1. Now we can target that id in the css, so create a new style for our display of records:

|  |
| --- |
| **#records {**  **margin-left:34px;**  **width:80%;**  **}** |

Refresh the teamweights.html file and adjust the CSS to your liking.

1. We may also want to change the background color of alternating rows just for easier reading, again you can play with the background colors of this code:

|  |
| --- |
| **#records div:nth-child(odd) {**  **background: lightgray;**  **display: block;**  **margin:5px 0px;**  **}** |

1. In the scripts.js file, change the displayData() function to display as a div tag instead of a p tag.

|  |
| --- |
| **function displayData(arr) {**  **let outHTML = "";**  **for(let i=0; i < arr.length; i++){**  **outHTML+="<div>"+arr[i].empName + " weighed " + arr[i].empWeight + " Kgs</div>";**  **}**  **document.getElementById("records").innerHTML = outHTML;**  **}** |

part 3 – adding a new record (jQuery)

The HTML file enterweight.html is being used to add a new record to the database. We have to configure that file to work with the **putnewdoc** endpoint. Before continuing make sure that the jQuery CDN files are linked in the head tags of the html doc

|  |
| --- |
| **<head>**  **<title> Skillsoft Weight Tracker </title>**  **<link rel="stylesheet" type="text/css" href="styles/styles.css" />**  **<meta content="text/html;charset=utf-8" http-equiv="Content-Type">**  **<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.3.1/jquery.min.js"></script>**  **<script src="https://cdn.jsdelivr.net/npm/jquery-validation@1.19.0/dist/jquery.validate.js"></script>**  **</head>** |

1. Open the scripts.js file and enter the following function.

|  |
| --- |
| **function putNewDoc() {**  **let endpoint = 'http://localhost:8000/putnewdoc';**  } |

Notice the endpoint

1. We will be using **ajax** to implement the post to our endpoint. The ajax function will take an object with all the parameters it needs

|  |
| --- |
| **function putNewDoc() {**  **let endpoint = 'http://localhost:8000/putnewdoc';**  **$.ajax({**  **})**  **}** |

1. Here are some of the values the ajax function needs.

|  |
| --- |
| **function putNewDoc() {**  **let endpoint = 'http://localhost:8000/putnewdoc';**  **$.ajax({**  **type:**  **dataType:**  **url:**  **data:**  **success:**  **})**  **}** |

1. This will be a POST request, so that value goes next to type. datatype is of course JSON and the endpoint is already defined. After that the data itself will be wrapped up in a json object and finally we will handle success.

|  |
| --- |
| **function putNewDoc() {**  **let endpoint = 'http://localhost:8000/putnewdoc';**  **$.ajax({**  **type: "POST",**  **dataType: 'json',**  **url: endpoint,**  **data: {**    **},**  **success:**  **})**  **}** |

1. The data part is now filled out by extracting the values in the form fields. For success, we pass a function that will be loaded with data returned from our API call

|  |
| --- |
| **function putNewDoc() {**  **let endpoint = 'http://localhost:8000/putnewdoc';**  **$.ajax({**  **type: "POST",**  **dataType: 'json',**  **url: endpoint,**  **data: {**  **empName: jQuery("#empName").val(),**  **empWeight: jQuery("#empWeight").val()**  **},**  **success: function (response) {**  **console.log(JSON.parse(response.data));**  **}**  **})**  **}** |

1. Here is the entire function. The final done function can be chained to the ajax function.

|  |
| --- |
| **function putNewDoc() {**  **let endpoint = 'http://localhost:8000/putnewdoc';**  **$.ajax({**  **type: "POST",**  **dataType: 'json',**  **url: endpoint,**  **data: {**  **empName: jQuery("#empName").val(),**  **empWeight: jQuery("#empWeight").val()**  **},**  **success: function (response) {**  **console.log(JSON.parse(response.data));**  **}**  **})**  **}** |

part 4 – Returning One Document

We do not yet have an API to return just one document for example we search for just Axle. We would need to create an endpoint route and a controller function to handle this.

1. Create a route first in routes.js

|  |
| --- |
| **app.route('/getbyformname').post(controller.getbyformname);** |

Notice that this is a POST request

1. Start writing a function to correspond to the route above, in the controller.js file

|  |
| --- |
| **exports.getbyformname=function(req, res){**  **};** |

1. We will extract the name to be found using req.body.empName

|  |
| --- |
| **exports.getbyformname=function(req, res){**  **let empToFind = req.body.empName;**  **};** |

1. Use the Weight object and the find() method to obtain the document

|  |
| --- |
| **exports.getbyformname=function(req, res){**  **let empToFind = req.body.empName;**  **Weight.find({ });**  **};** |

1. Complete the function by passing in the name to be found and a function to handle the results

|  |
| --- |
| **exports.getbyformname=function(req, res){**  **let empToFind = req.body.empName;**  **Weight.find({empName:empToFind}, function(err, results) {**  **if (err)**  **res.send(err);**  **res.json(results);**  **});**  **};** |

1. We can test the new api using Postman
2. Once the api works, we can turn our attention to the myweights.html file and configure that for searching. We can use some of the same HTML code from teamweights, so the two pair of div tags that displayed records will work for myweights also

|  |
| --- |
| **<input id="empName" type="text" />**  **</div>**  **</form>**  **<div id="records"></div>**  **<div>**  **<button onclick="getDoc()">Find my records</button>**  **</div>** |

1. We now have to write the getDoc() function in scripts.js. below is the basic setup, we have our endpoint and our one form field defined. We also have the fetch structure in place

|  |
| --- |
| **function getDoc(){**  **let endpoint = 'http://localhost:8000/getbyformname';**  **let empName = document.getElementById('empName').value;**  **fetch(endpoint, {**  **})**  **};** |

1. We know it’s a post request and we can also send along headers to the server to let the server know what kind of data to expect

|  |
| --- |
| **function getDoc(){**  **let endpoint = 'http://localhost:8000/getbyformname';**  **let empName = document.getElementById('empName').value;**  **fetch(endpoint, {**  **method: "POST",**  **headers: {**  **},**  **})**  **};** |

1. Now we can define the header and also grab the value the user types into the one field we have on this form

|  |
| --- |
| **function getDoc(){**  **let endpoint = 'http://localhost:8000/getbyformname';**  **let empName = document.getElementById('empName').value;**  **fetch(endpoint, {**  **method: "POST",**  **headers: {**  **"Content-Type": "application/x-www-form-urlencoded"**  **},**  **body: "empName=" + empName**  **})**  **};** |

1. There will now be a then() method to deal with the fetch promise and a catch method as well

|  |
| --- |
| **fetch(endpoint, {**  **method: "POST",**  **headers: {**  **"Content-Type": "application/x-www-form-urlencoded"**  **},**  **body: "empName=" + empName**  **}).then(function(response) {}**  **).catch(function (error) {}**  **)** |

1. The first then() method will take a function parameter which will be loaded with the response from our api call. We can simply call the displayData() method we wrote earlier to display the data. The catch() method will be invoked if there is an error

Here is the complete function

|  |
| --- |
| **function getDoc(){**  **let endpoint = 'http://localhost:8000/getbyformname';**  **let empName = document.getElementById('empName').value;**  **fetch(endpoint, {**  **method: "POST",**  **headers: {**  **"Content-Type": "application/x-www-form-urlencoded"**  **},**  **body: "empName=" + empName**  **}).then(function(response) {**  **response.json()**  **.then(function(data) {**  **displayData(data);**  **});**  **}**  **).catch(function (error) {**  **console.log('Request failed', error);**  **})**  **};** |