**Exercise02\_03\_01 – Step 1**

This exercise will consist of coding the front-end parts of our application.



1. Create a new folder for the exercise called ***Exercise02\_03\_01***. Copy all of the files into it from the previous Exercise02\_02\_01. Open your IDE and build 2 new folders inside the project. One will be called ***public***, and it will be used to store our ***static*** content. The second will be called views, and we will build our dynamic content in it. Download the ***exercise resources*** from Google Classroom. Unzip them and place them into the ***public*** folder. There will be two ***CSS*** files and an ***image***.
2. We need to install an external third-party module that we will need with NPM. We will be building our dynamic content pages with EJS, so we need to install it:  
   ***npm install ejs –save***Check **package.json** to make sure the dependencies are correct.
3. Let’s get ***index.js*** ready to serve ***static*** resources from our ***public*** folder. For that, we employ an Express ***app.use()*** method to indicate the location of the static files. Place the following code just above the ***app.listen()*** code:  
   ***app.use(express.static(\_\_dirname + '/public'));***Run your server to make sure that everything is working and the server is listening on the correct port. Test it in the browser, then terminate it and the server.

**Exercise02\_03\_01 – Step 2**



1. Let’s get ***index.js*** ready to serve ***dynamic*** resources. We will keep these resources in the ***views*** folder. Because we are using ***EJS*** as our dynamic templating engine, we will have to let the server know that. We can do that in ***index.js*** with an express ***app.set()*** method. This method is used to set ***name/value*** pairs, and express knows what ***‘view engine’*** is. Place the following code just below our variable declarations:  
   var app = express();  
    ***app.set('view engine', 'ejs');***
2. Let’s start building our ***views***. The first one will be a simple ***login*** page, ***login.ejs***, in the ***views*** folder, just straight HTML. We will build it with an ***ejs*** extension, because that is the ***view*** ***engine*** we specified in our server. Scaffold some HTML and build it out. Notice that it knows where to find our CSS when we ***link*** it. Notice also that we are providing a hyperlink to our ***/auth/twitter*** route:  
   ***<!doctype html>  
   <html>  
   <head>  
    <title>Login</title>  
    <meta charset="UTF-8">  
    <meta name="viewport" content="initial-scale=1.0">  
    <link rel="stylesheet" href="login.css";  
   </head>  
   <body>  
    <div>  
    <h1>Twitter Notes</h1>  
    <h2>Your Personal Friends Diary</h2>  
    <a href="/auth/twitter">  
    <img src="sign-in-with-twitter-gray.png"></a>  
    </div>  
   </body>  
   </html>***
3. Let’s build a function that will clear out our credentials for security. We can place it in ***authenticator.js***, just below our ***getCredentials()*** function:  
    ***clearCredentials: function () {  
    twitterCredentials.oauth\_token = "";  
    twitterCredentials.oauth\_token\_secret = "";  
    twitterCredentials.access\_token = "";  
    twitterCredentials.access\_token\_secret = "";  
    twitterCredentials.twitter\_id = "";  
    },***
4. Now let’s build a route in ***index.js*** for this to endpoint ***/login***. For security, it will clear out our credentials. Place it just above the ***app.use()*** call for our static content as follows:  
   ***app.get('/login', function(req, res) {  
    authenticator.clearCredentials();  
    res.render('login');   
   });***  
   app.use(express.static(\_\_dirname + '/public'));  
   Run your server and give the browser the ***login*** url: ***localhost:8080/login***. We should get our new page. Click the button and we should be off and running to our app.
5. Just below that. let’s build a route to endpoint ***/logout***. For security, it will clear out our credentials:  
   ***app.get('/logout', function(req, res) {  
    authenticator.clearCredentials();  
    res.redirect('/login');   
   });***  
   Run your server and give the browser the ***login*** url: ***localhost:8080/login***. Go through the ***authorization*** process, then route the browser to ***/allfriends***. Lastly route the browser to ***/logout***. Now we can see our eventual application flow.
6. Next let’s do some surgery in ***index.js*** to prepare the way for the final routing of our application. Just above the ***/logout*** route, let’s define a function called ***renderMainPageFromTwitter()***.We will move all of our ***waterfall*** code to it. Scaffold it as follows:  
   ***function renderMainPageFromTwitter(req, res) {  
      
   }***  
   ***Cut*** all of the code out of the ***/allfriends*** route and paste it into this new function.
7. Now let’s modify what is left of the ***/allfriends*** route as follows:  
   ***app.get('/allfriends', function(req, res) {  
    renderMainPageFromTwitter(req, res);  
   });***  
   Once again run your server and give the browser the ***login*** url: ***localhost:8080/login***. Go through the ***authorization*** process, then route the browser to ***/allfriends***. Lastly route the browser to ***/logout***. Now we can see our eventual application flow. Everything should still be working.

**Exercise02\_03\_01 – Step 3**



1. Now let’s start to build the main page that will end up displaying our data. It will be in the ***views*** folder, and be called ***index.ejs***. It will be capable of loading dynamic data at runtime:  
   <head>  
    ***<title>Friends</title>***  
    <meta charset="UTF-8">  
    <meta name="viewport" content="initial-scale=1.0">  
    ***<link rel="stylesheet" href="style.css">***</head>  
   <body>  
    ***<div id="wrapper">  
    <a href="/logout"><button type="button"   
    class="logout">Logout</button></a>  
    <div id="friends-column" class="column">  
    <h1>Your Followers</h1>  
    <ul id="friends">  
      
    </ul>  
    </div>  
    <div id="notes-column" class="column">  
    <h1>Notes</h1>  
    <ul id="notes”></ul>  
    </div>  
    </div>***</body>  
   </html>  
   We will come back later and embed some JavaScript into this file using EJS.
2. We can now modify our ***/allfriends*** route to become ***/*** route. We will first need to check our credentials for security, and ***redirect*** the user to the login page if we do not have them. If we have them, we will go get our Twitter data:  
   ***app.get('/', function(req, res) {  
    var credentials = authenticator.getCredentials();  
    if (!credentials.access\_token || !credentials.access\_token\_secret)   
    {  
    return res.redirect('/login');  
    }  
    console.log('Loading friends from Twitter');  
    renderMainPageFromTwitter(req, res);});***  
   Once again run your server and give the browser the ***/*** url: ***localhost:8080/***. Because we have no credentials, we should see the ***login*** page. Sign in and go through the ***authorization*** process, then route the browser back to ***/***. again. We should get a ***console.log()*** message and we should see our friends data on the Web page. Lastly route the browser to ***/logout***.
3. Let’s make some minor changes to support our new routing. Return to the ***/auth/twitter*** route and make the following changes to the results of the ***authenticate()*** callback:  
   app.get(url.parse(config.oauth\_callback).path, function (req, res) {  
    authenticator.authenticate(req, res, function (err) {  
    ***if (err) {  
    res.redirect('/login');  
    } else {  
    res.redirect('/');  
    }*** });  
   });  
   Once again run your server and give the browser the ***/*** url: ***localhost:8080/***. Sign in and go through the ***authorization*** process. Now we should automatically see our friends data. Lastly, route the browser to ***/logout***.
4. Now let’s do some EJS programming. We will use it to embed JavaScript in our ***index.ejs*** page to handle the display of our dynamic content. We are going to get the ***template*** ready to receive content from our server, then make use of it. First we will implement a JavaScript style ***forEach*** loop to populate our ***<ul>*** element. We use EJS ***code*** tags ***<% %>***. It will expect to receive an array named ***friends***. Let’s program it as follows:  
    <ul id="friends">  
    ***<% friends.forEach(function(friend) { %>  
      
    <% }); %>*** </ul>
5. Now we can dynamically build our ***<li>*** elements, adding attributes, images, and innerHTML data using EJS ***variable*** tags ***<%= %>***, as follows:   
    <% friends.forEach(function(friend) { %>  
    ***<li uid="<%= friend.twitter\_id %>" class="friend">  
    <img class="profile-image"   
    src="<%= friend.profile\_image\_url %>">  
    <p>  
    <span class="name">  
    <%= friend.name %></span>  
    <span class="screen-name">  
    <%= friend.screen\_name %></span>  
    </p>  
    <p class="location"><%= friend.location %></p>  
    </li>*** <% }); %>
6. Return to ***index.js*** and let’s modify the renderMainPageFromTwitter() function. First let’s remove the two ***console.log()*** debug functions that show the ***friends*** array lengths. Now we need to convert our ***friends*** array to get it into a format that we can pass to our EJS template. To do this we will use the JavaScript ***array.map()*** method. Add the following code directly below the ***friends.sort()*** method:  
    ***friends = friends.map(function(friend) {  
    return {  
    twitter\_id: friend.id\_str  
    for\_user: credentials.twitter\_id,  
    name: friend.name,  
    screen\_name: friend.screen\_name,  
    location: friend.location,  
    profile\_image\_url: friend.profile\_image\_url  
    };  
    });*** res.send(friends);  
   Once again we will test to make sure we didn’t break anything. Run your server and give the browser the ***/*** url: ***localhost:8080/***. Sign in and go through the ***authorization*** process. Now we should see our friends data. You would have to look carefully, but you can see that it is now formatted differently. Lastly, route the browser to ***/logout***.
7. Now let’s try to render the content onto our EJS template. Directly below our previous function call, we will replace the ***res.send()*** call. Notice that we will render the page, passing it our newly formatted friends array as an object as the second argument:  
    ***res.render('index', { friends: friends });***Magic! Scroll down through your friends. Try the ***Logout*** button. We have completed our consuming of Twitter’s RESTful APIs! We now have a cleanly flowing Web app, with the ability to display dynamic content through an EJS template as a front-end!