Module 2 Lab 2: Basic Full Stack Labs

Ryan M.K. Burke-Jimenez

Arizona State University

IFT 458

Professor Dinesh Sthapit

September 10th, 2023

Module 2 Lab 2: Basic Full Stack Labs

This lab is designed to implement a basic full stack environment onto a local host for testing and learning purposes.

# Environment Setup

Visual Studio Code was the environment used to set up the files for inspection. Upon creating a new directory for the project to be held, NPM was used to initialize the project and create a package.json file. NPM was also utilized to install express, body-parser, and ejs, all modules that are necessary for this assignment. After this, three main files were created for the demonstration: app.js (main file), index.ejs, and result.ejs. Index.ejs and result.ejs are both held in a separate folder named “views” that is held in the same directory as app.js. App.js utilizes Node.js while index.ejs and result.ejs utilize HTML (Figure 1). The project was subsequently committed and pushed onto GitHub for assessment. When the local server is operational on port 4000, inspecting the elements will show the corresponding HTML source code in the web browser, in this case, Google Chrome. The environment is now successfully established for further analysis.

A screenshot of a computer program

Description automatically generated

Figure - index.ejs, result.ejs, app.js

## Exercise 1: Analyzing .ejs Code vs. HTML Rendered in the Browser

The purpose of this exercise is to observe and analyze the difference between an .ejs document with a rendered .html document that is present in a browser’s element inspection.

## .ejs Code Analysis

There are two .ejs files that exist in this project, however, only one is relevant for inspection: result.ejs. Result.ejs contains JavaScript within the HTML, which is due to the files being .ejs rather than .html files. EJS is short for Embedded JavaScript and is a powerful way to generate HTML with JavaScript components. To embed JavaScript within the HTML file, it must be present between <%= variable %> and is between HTML element paragraphs with the result being <p>Variable: <%= variable%></p>. The HTML element <p></p> may be just for aesthetics where any other HTML element will suffice. The <%=%> structure is necessary for the .ejs file to understand that JavaScript is present in the code and will set variables to the given placeholders, in this case, various mathematical operations are variables, and the calculations are completed within the app.js file. The variables are set in result.ejs when index.ejs communicates with app.js through the post function (Figure 2). App.js requests information that is entered in the UI of index.ejs and with the inputs app.js can respond with the

A screenshot of a computer program

Description automatically generated

2. Calculations complete, response acknowledged. Render and results sent back to result.ejs. Variables set with results sent from app.js

1. “Calculate” in UI pressed, POST function requested. Index.ejs sends information to app.js.

Figure - Communication route for result.ejs

calculations when the “calculate” button in the UI is pressed. Once pressed, result.ejs is rendered and receives the variables from the POST function where the results are displayed in the UI.

### HTML Code Analysis

After careful analysis of the .ejs file, analysis of the rendered HTML file is next for observation. In the element inspection of the UI, the source code of result.ejs is present, however, no JavaScript is shown in the HTML file (Figure 3). In the HTML file, the results of the POST call from app.js are displayed but the JavaScript in result.ejs is not. The results being present rather than their variable names were to be expected, however, the <%=%> structure is not shown within the HTML file which is rather intriguing. This must mean that once the result.ejs file receives the result of the POST function, EJS knows to fully replace <%= variable%> with the result without any complication. Other than this, the .ejs and the HTML file are similar in structure and nothing else is modified.

A screenshot of a computer

Description automatically generated

Figure - HTML file inspection

Compare and Contrast. In structure, HTML and EJS are similar in that HTML can be generated in both file types without any issue. All syntax and elements used in HTML may be used in EJS, however, it is not reciprocal. EJS has the ability to hard code JavaScript into the source code while using HTML as the base code for the file. This can be a very powerful tool in the event that the user needs to output JavaScript while developing a website with HTML that has dynamic content as a function of the website (Ikechukwu, 2018). In the event that a developer needs to provide users with a frontend UI with the proper workings of the backend embedded into the file, EJS is the most beneficial way to accomplish this.

**Exercise 2: Arrays and Objects**

References

Ikechukwu, L. (2020, August 13). Using EJS as a Template Engine in your Express App - Linda Ikechukwu - Medium. *Medium*. <https://medium.com/@Linda_Ikechukwu/https-medium-com-linda-ikechukwu-using-ejs-as-a-template-engine-in-your-express-app-cb3d82c15e17#:~:text=What%20is%20EJS%3F,interpolate%20(concatenate)%20strings%20effectively>.