COMP 9783 Front-end Development

Lab-2-5 - EJS Template Language. (4% of the course mark)

Name:

Student Number:

EJS.js is a lightweight, easy-to-use templating engine for JavaScript that allows developers to embed JavaScript code directly into HTML templates. It enables the dynamic generation of HTML content based on data provided by the server or application logic. With EJS, developers can create reusable template files containing HTML markup mixed with JavaScript logic, which can be rendered dynamically to produce dynamic web pages.

Lab objectives:

1. To understand how to dynamically generate HTML content based on data from the server or application.
2. Promote code reusability by separating HTML markup from JavaScript logic.
3. EJS.js is commonly used with Node.js for server-side rendering of dynamic web pages.

Create an Express.js app:

1. On **VSCode**, **create** a **folder** named **Simple-Express-EJS-App**.
2. Open the **terminal** and change the directory to **Simple-Express-EJS-App**.
1. Initialize the app by typing **npm init -y** and **press enter**.

Note: npm init -y is a quick shortcut to initialize the package.json file. It will initialize the app with default values.

3. Type **npm install express** and **press enter**.
4. Type **npm install ejs** and **press enter**.

5. Type **npm install -D nodemon** and **press enter**.

Note: The -D option means the package would be installed as a developer dependency.

6. Open **package.json** and look for **devDependencies**. Take a screenshot and name it **devDependencies.png**.
7. Look for the **scripts** attribute and add this code:

"start": "nodemon index.js"

8. Save the changes.
9. Create a file named **index.js** and enter the following code:

```
const express = require("express");
const app = express();
const port = 3000;
const APP_NAME = "Simple-Express-EJS-App";

app.set("view engine", "ejs");

const data = {
  header: { title: "header data..." },
  content: {
    title: "content data...",
    northAmericanCountries: [
      { name: "Canada", code: "CA" },
      { name: "Mexico", code: "MX" },
      { name: "United States", code: "US" },
    ],
  },
  footer: { title: "footer data..." },
};

app.get("/", (req, res) => {
```

```
  res.render("index", { data: data });
});

app.listen(port, () => {
  console.log(`${APP_NAME} listening on port ${port}`);
});
```

10. Save the changes.
11. Create a folder named: **views**.
12. Inside the views folder, create the following files:
    a. content.ejs
    b. header.ejs
    c. footer.ejs
    d. Index.ejs
13. For **content.ejs** add the following code and save the changes:

```
<section class="section">
  <div class="container">
    <div class="box">
      <p class="is-size-2 has-text-centered"><%= content.title %></p>
      <table class="table mx-auto">
        <thead>
          <tr>
            <th>Name</th>
            <th>Code</th>
          </tr>
        </thead>
        <tbody>
          <!-- prettier-ignore -->
          <% content.northAmericanCountries.forEach((country) => { %>
          <tr>
            <td><%= country.name %></td>
            <td><%= country.code %></td>
          </tr>
          <% }) %>
```

```
        </tbody>
      </table>
    </div>
  </div>
</section>
```

14. For **header.ejs** add the following code and save the changes:

```
<section class="section">
  <div class="container">
    <div class="box"><p class="is-size-2"><%= header.title %></p></div>
  </div>
</section>
```

15. For **footer.ejs** add the following code and save the changes:

```
<section class="section">
  <div class="container">
    <div class="box"><p class="is-size-2"><%= footer.title %></p></div>
  </div>
</section>
```

16. For **index.ejs** add the following code and save the changes:

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1" />
    <title></title>
    <link
      rel="stylesheet"
      href="https://cdn.jsdelivr.net/npm/bulma@1.0.0/css/bulma.min.css"
    />
  </head>
  <body>
```

```
    <!-- prettier-ignore -->
    <%- include ('header', { header: data.header }); %>
    <!-- prettier-ignore -->
    <%- include ('content', { content: data.content }); %>
    <!-- prettier-ignore -->
    <%- include ('footer', {  footer: data.footer }); %>
  </body>
</html>
```

17. On the terminal type **npm run start** and **press enter**. This should **start the app** via **nodemon**.

Note: nodemon is a utility for Node.js that helps in automatically restarting the Node application when changes are detected in the source code. It is particularly useful during development to streamline the process of code modification and testing without manually restarting the server every time a change is made.

18. Take a screenshot of the command line and name it **nodemon.png**.
19. Open your browser and enter http://localhost:3000 and take a screenshot and name it **ejs.png**.
20. Modify any value of variable **data** on **index.js** and save the changes. SInce js files are monitored for changes, it should automatically restart the app.
21. Open your browser and enter http://localhost:3000, ensure that the changes made are visible, take a screenshot and name it **ejs-modified.png**.
22. Modify the value of variable **data** on **index.js** and **add any new attribute** and **value for each element** of the **northAmericanCountries** object save the changes. SInce js files are monitored for changes, it should automatically restart the app.
23. Update **content.ejs** to display the new attribute and value added in the previous step.
24. By default **.ejs** files are not monitored by nodemon, type **rs** and **press enter** to restart the app.

25. Open your browser and enter http://localhost:3000, ensure that the changes made are visible, take a screenshot and name it **ejs-new-attribute-value.png**.

Submission:

1. Use the html template: **index.html** and **write HTML codes for each screenshot**:
    a. Write a title and short description.
    b. Display the screenshot.
2. Create a new folder named **html** and copy all the **HTML**, **CSS** and **PNG** files used in the previous step.
3. Create a new folder named **app** and copy everything from **Simple-Express-EJS-App**, except for the **node_modules** folder.

Note: Do not include the node_modules folder, this is not needed as I can just run npm install to reinstall the packages.

4. Create a new folder named **submit** and copy both the **html** and **app folders**.
5. Create a **zip file** of the **submit folder**.
6. Submit the **zip file** to **GBC - D2L**.