**Experiment Number: 6 Date: 03-03-2025**

# **Integrate a Template Engine (EJS/Pug) with Express and Render Dynamic HTML Views**

## **PRE LAB EXERCISE**

### **Objective:**

* Understand the role of **template engines** in web applications.
* Learn how to integrate **EJS (Embedded JavaScript) or Pug** with Express.js to render **dynamic web pages**.
* Implement a simple **Express.js server** that dynamically generates HTML pages using data.

### **QUESTIONS:**

1. **What is a template engine, and why is it used in web applications?**

A **template engine** is a software component used in web development to generate dynamic HTML pages. It allows developers to embed programming logic into HTML templates, enabling the separation of **presentation (UI)** from **business logic**.

1. **Compare EJS and Pug in terms of syntax and usability.**

| **Feature** | **EJS (Embedded JavaScript)** | **Pug (formerly Jade)** |
| --- | --- | --- |
| **Syntax** | HTML with <%= %> & <% %> | Indentation-based, no closing tags |
| **Readability** | Similar to HTML, beginner-friendly | Compact but harder for complex HTML |
| **Ease of Use** | Easy for HTML & JS users | Requires learning new syntax |
| **Logic Handling** | Uses JS expressions (if, for) | More readable syntax for logic |
| **Whitespace** | No strict rules | Indentation-sensitive |
| **Performance** | Slightly slower (full HTML parsing) | Faster (minimal syntax) |
| **Extensibility** | Supports partials & layouts | Supports mixins & includes |
| **Best for** | Keeping HTML structure intact | Cleaner, less verbose templates |

1. **How does a template engine improve code reusability in web applications?**

* Enable reusable components like **partials** (e.g., headers, footers).
* Supports **layouts** for consistent structure across pages.
* Reduces duplicate code by allowing **dynamic content injection**.

1. **What is the difference between server-side rendering (SSR) and client-side rendering (CSR)?**

|  |  |  |
| --- | --- | --- |
| **Feature** | **Server-Side Rendering (SSR)** | **Client-Side Rendering (CSR)** |
| **Rendering Location** | HTML is generated on the server and sent to the client. | HTML is built dynamically in the browser using JavaScript. |
| **Performance** | Faster initial page load but may have slower interactions. | Slower initial load but smoother interactions after loading. |
| **SEO (Search Engine Optimization)** | Better for SEO since search engines can crawl fully rendered HTML. | Requires extra effort for SEO (e.g., prerendering or SSR in frameworks like Next.js). |
| **User Experience** | Page loads fully on request but refreshes are required for updates. | Feels more responsive; only parts of the page update without a full reload. |
| **Best Used For** | Static content, blogs, e-commerce, SEO-heavy websites. | Single-page applications (SPAs), dashboards, interactive web apps. |

1. **How do you pass dynamic data from an Express route to a template engine?**

Use the res.render() method in Express:

app.get('/users', (req, res) => {

    const users = [{ name: "Alice", age: 25 }, { name: "Bob", age: 30 }];

    res.render('users', { users });  // Passing 'users' data to the template

});

This makes users accessible inside the template for dynamic rendering

## **IN LAB EXERCISE**

### **Objective:**

* Set up an **Express.js server**.
* Use **EJS or Pug** to generate **dynamic web pages**.
* Pass **dynamic data** from the server to the frontend.

### **Resources Required:**

* **Node.js**, **Express.js**, **EJS/Pug**, **Postman**, **Browser (Chrome/Firefox)**.

### **Step 1: Install Required Packages**

npm init -y

npm install express ejs pug

A screen shot of a computer

AI-generated content may be incorrect.

### **Step 2: Set Up Express Server**

**Create server.js**

A screen shot of a computer program

AI-generated content may be incorrect.

### **Step 3: Create the Views Folder and Template Files**

**Create views/users.ejs**

A computer screen shot of a program code

AI-generated content may be incorrect.

### **Step 4: Run the Server and Test the Output**

1. Start the server:

node server.js



1. Open a browser and visit:

<http://localhost:3000/users>

1. The **list of users** will be dynamically displayed on the webpage.

A screenshot of a computer

AI-generated content may be incorrect.

### **Using Pug as an Alternative Template Engine**

1. Modify **server.js** to use **Pug** instead of **EJS**:

app.set('view engine', 'pug');

app.set('views', path.join(\_\_dirname, 'views'));

**server.js**

const express = require('express');

const app = express();

const path = require('path');

// Set Pug as the template engine

app.set('view engine', 'pug');

app.set('views', path.join(\_\_dirname, 'views'));

// Sample Data

const users = [

    { name: "Alice", age: 25 },

    { name: "Bob", age: 30 },

    { name: "Charlie", age: 22 }

];

// Route to Render Pug Template

app.get('/users', (req, res) => {

    res.render('users', { users });

});

app.listen(3000, () => console.log('Server running on http://localhost:3000'));

1. Create **views/users.pug**:

A black screen with white text

AI-generated content may be incorrect.

1. Restart the server and visit http://localhost:3000/users to see the same **dynamic content** but rendered with **Pug** instead of **EJS**.

A screenshot of a computer

AI-generated content may be incorrect.

## **POST LAB EXERCISE**

### **QUESTIONS:**

1. **What are the advantages of using a template engine over plain HTML?**

* **Dynamic Content**: Allows inserting variables, loops, and conditionals to generate dynamic pages.
* **Code Reusability**: Supports partials and layouts, reducing redundant code.
* **Separation of Concerns**: Keeps logic separate from the view layer, improving maintainability.
* **Improved Readability**: Reduces clutter compared to embedding JavaScript directly in HTML.

1. **How does EJS handle loops and conditionals?**

* **Loops** (forEach example):

<% users.forEach(user => { %>

    <li><%= user.name %> - <%= user.age %> years old</li>

<% }) %>

* **Conditionals** (if-else example):

<% if(user.age > 25) { %>

    <p>Older than 25</p>

<% } else { %>

    <p>25 or younger</p>

<% } %>

1. **What are the key differences between EJS and Pug?**

|  |  |  |
| --- | --- | --- |
| **Feature** | **EJS** | **Pug** |
| **Syntax** | Uses standard HTML with embedded JS (<%= %>) | Uses indentation-based, minimal syntax |
| **Readability** | Familiar for HTML users | More concise but different from HTML |
| **Usability** | Easy to integrate with existing HTML | Shorter, but has a learning curve |
| **Extensibility** | Supports partials and layouts | Supports partials, layouts, and mixins |
| **Performance** | Slightly slower due to HTML parsing | Faster due to compact syntax |

1. **How can you pass form data from an HTML page to an Express.js route?**

* **Add a form in an EJS or Pug template**

<form action="/submit" method="POST">

    <input type="text" name="username" placeholder="Enter your name">

    <button type="submit">Submit</button>

</form>

* **Enable body parsing in Express**

const express = require('express');

const app = express();

app.use(express.urlencoded({ extended: true })); // Parses form data

* **Handle the form submission in Express**

app.post('/submit', (req, res) => {

    const username = req.body.username;

    res.send(`Hello, ${username}!`);

});

1. **How can template engines be used to create layouts and partials for reusability?**

* **Layouts**: Define a base structure and extend it in child templates.
* **Partials**: Create reusable components like headers and footers.

**Example in EJS**:

1. **Create views/header.ejs**

<header><h1>My Website</h1></header>

1. **Include it in a page (views/home.ejs)**

<%- include('header') %>

<p>Welcome to my website</p>

**ASSESSMENT PATTERN.**

|  |  |  |
| --- | --- | --- |
| **Description** | **Max Marks** | **Marks Awarded** |
| Pre Lab Exercise | **5** |  |
| In Lab Exercise | **10** |  |
| Post Lab Exercise | **5** |  |
| Viva | **10** |  |
| **Total** | **30** |  |
| **Faculty Signature** | |  |