Static Files, Forms, JSON, Postman

Software Development Bootcamp

Topic

Serving And Handling A Simple HTML Form With Express

Express Middleware

Express middleware are functions that have access to the request object (req), the response object (res)

Express Middleware: `urlencoded`

```
app.use(express.urlencoded({extended: true})
```

- Parses incoming request bodies with URL-encoded payloads
- Populates the **`req.body`** property with the parsed data
- **`extended: true`** allows for parsing of nested objects
- Necessary for handling form submissions

Express Middleware: `static`

```
app.use(express.static('public'))
```

- Serves static files such as HTML, CSS, images, and JavaScript files
- Automatically serves files from the specified directory
- Can specify multiple static directories
- Order matters: Express looks for files in the order directories are set

Setting Up The Express Server

- In this example we are using **require** syntax to access express
- We use express middleware `urlencoded`, and `static` before defining our routes

```
const express = require('express');
const app = express();
const port = 3000;

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

// Serve static files from the 'public'
directory
app.use(express.static('public'));

app.listen(port, () => {
 console.log(`Server running at
http://localhost:${port}`);
});
```

Creating The HTML Form

- Create a folder named **public** and inside that folder an **index.html** file with this form
- The form's `action` attribute matches the route in our Express app

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport"
content="width=device-width,
initial-scale=1.0">
    <title>Simple Form Example</title>
</head>
<body>
    <h1>User Registration</h1>
    <form action="/submit-form" method="POST">
        <label for="username">Username:</label>
        <input type="text" id="username"
name="username" required><br><br>

        <label for="email">Email:</label>
        <input type="email" id="email"
name="email" required><br><br>

        <button type="submit">Register</button>
    </form>
</body>
</html>
```

Handling Form Submission

- We destructure the `req.body` object to access the form data we provided
- We use the provided form data in our response `res.send(...${username}...)`

```
app.post('/submit-form', (req, res) => {
   const { username, email } = req.body;
   console.log('Form submitted:', { username, email });

   // Here you would typically save the data to a database
   // For this example, we'll just send a simple response using the username provided in the form
   res.send(`Thank you, ${username}! Your registration was successful.`);
 });
```

Exercise

Reading Form Data

Topic

Sending JSON

Express Middleware: `json`

Express provides built-in middleware for parsing JSON data in incoming requests. This middleware is crucial for handling JSON payloads in modern web applications

`app.use(express.json())`

- Parses incoming request bodies with JSON payloads
- Populates the **`req.body`** property with the parsed data
- Sets **Content-Type** to **application/json** for responses
- Throws an error if invalid JSON is sent

Sending a JSON Response

res.json() is a method in Express.js used to send JSON responses. It automatically sets the Content-Type header to application/json.

Key Points About res.json()

- Converts the parameter to a JSON string using JSON.stringify()
- Sends the response with the correct content-type header
- Can handle objects, arrays, strings, booleans, numbers, null, and undefined
- Automatically ends the response process

res.json() Example

Express will automatically:

- Convert the `users` array into a JSON string
- Set the **Content-Type** header to '**application/json**'
- Send the response to the client

```
// Define route handler for GET requests to '/api/users'
app.get('/api/users', (req, res) => {
  // Create users array containing two user objects
  const users = [
    { id: 1, name: 'Alice' },
    { id: 2, name: 'Bob' }
  ];
  // send the users array as a JSON response
  res.json(users);
});
```

What Is Postman?

Postman is a popular API development and testing tool that simplifies the process of sending HTTP requests and analyzing responses. Postman helps ensure that your endpoints are sending correctly formatted JSON data before integrating with a frontend application.

Postman

Key Features Of Postman

- User-friendly interface for crafting and sending HTTP requests
- Supports various HTTP methods (GET, POST, PUT, DELETE, etc.)
- Allows setting headers, request bodies, and query parameters
- Provides a way to save and organize requests for reuse
- Offers environment variables for managing different configurations
- Includes tools for automated testing and API documentation

Using Postman

1. Create a new request by selecting the HTTP method and entering the URL
2. Add any necessary headers, query parameters, or request body
3. Send the request and view the response, including status code, headers, and body
4. Save the request for future use or share it with team members

Why Use Postman?

- Testing APIs during development
- Debugging issues with API requests and responses
- Creating and running automated API tests
- Generating API documentation
- Collaborating with team members on API projects

Exercise

Postman Install