# MODUL PERKULIAHAN

**Mata Kuliah** : Pengembangan Aplikasi Web  
**Materi** : Membuat Server HTTP dengan Node.js dan Express.js  
**Pertemuan** : 02  
**Fakultas** : Teknik  
**Program Studi** : Teknologi Informasi  
**Kode MK** : AntaraTI501P  
**Disusun Oleh** : Ir.Asroni, S.T., M.Eng

## DAFTAR ISI

1. [PEMBAHASAN](#pembahasan)
2. [PREREQUISITES](#prerequisites)
3. [PRAKTIKUM](#praktikum)
   * [A. Tujuan Praktikum](#a-tujuan-praktikum)
   * [B. Materi](#b-materi)
   * [C. Metode Praktikum](#c-metode-praktikum)
   * [D. Langkah-langkah Praktikum](#d-langkah-langkah-praktikum)
4. [TUGAS](#tugas)

## PEMBAHASAN

1. Pengenalan HTTP dan Server Web:
   * Konsep dasar protokol HTTP
   * Struktur request dan response HTTP
   * Peran server web dalam aplikasi web
2. Membuat Server HTTP dengan Node.js:
   * Menggunakan modul http bawaan Node.js
   * Menangani request dan mengirim response
   * Routing sederhana
3. Pengenalan Express.js:
   * Apa itu Express.js dan kelebihannya
   * Instalasi Express.js
   * Struktur dasar aplikasi Express.js
4. Routing di Express.js:
   * Penanganan metode HTTP (GET, POST, PUT, DELETE)
   * Route parameters dan query strings
   * Middleware routing
5. Middleware di Express.js:
   * Konsep dan fungsi middleware
   * Built-in middleware
   * Custom middleware
6. Templating Engine:
   * Pengenalan EJS (Embedded JavaScript)
   * Rendering views dengan data dinamis
7. Penanganan Form dan JSON:
   * Parsing body request
   * Validasi input
8. Pengenalan RESTful API:
   * Prinsip-prinsip REST
   * Implementasi API sederhana dengan Express.js

## PREREQUISITES

Beberapa perangkat lunak yang digunakan untuk menunjang proses pembelajaran:

1. Node.js (versi LTS terbaru)
2. NPM (Node Package Manager)
3. Text editor (Visual Studio Code, Sublime Text, atau Atom)
4. Postman atau tools API testing lainnya

## PRAKTIKUM

### A. Tujuan Praktikum

Praktikum ini bertujuan untuk memperkenalkan mahasiswa pada pembuatan server HTTP menggunakan Node.js dan Express.js. Mahasiswa akan mempelajari cara membuat server sederhana, menangani routing, menggunakan middleware, dan membuat API sederhana.

### B. Materi

1. Membuat server HTTP dengan Node.js
2. Pengenalan dan penggunaan Express.js
3. Routing dan middleware di Express.js
4. Templating dengan EJS
5. Penanganan form dan JSON
6. Pembuatan RESTful API sederhana

### C. Metode Praktikum

1. Penjelasan materi oleh instruktur
2. Demonstrasi langsung pembuatan server dan API
3. Praktik mandiri oleh mahasiswa
4. Diskusi dan tanya jawab

### 

### D. Langkah-langkah Praktikum

1. Membuat Server HTTP dengan Node.js
   * Buat file simple-server.js:
   * const http = require('http');  
       
     const server = http.createServer((req, res) => {  
      if (req.url === '/') {  
      res.writeHead(200, { 'Content-Type': 'text/plain' });  
      res.end('Hello, World!');  
      } else if (req.url === '/api/data') {  
      res.writeHead(200, { 'Content-Type': 'application/json' });  
      res.end(JSON.stringify({ message: 'This is JSON data' }));  
      } else {  
      res.writeHead(404, { 'Content-Type': 'text/plain' });  
      res.end('404 Not Found');  
      }  
     });  
       
     const PORT = 3000;  
     server.listen(PORT, () => {  
      console.log(`Server running at http://localhost:${PORT}/`);  
     });
   * Jalankan server dengan perintah: node simple-server.js
   * Akses http://localhost:3000 dan http://localhost:3000/api/data di browser
2. Membuat Server dengan Express.js
   * Inisialisasi proyek baru dan instal Express:
   * mkdir express-demo  
     cd express-demo  
     npm init -y  
     npm install express
   * Buat file app.js:
   * const express = require('express');  
     const app = express();  
     const PORT = 3000;  
       
     app.get('/', (req, res) => {  
      res.send('Hello from Express!');  
     });  
       
     app.get('/api/data', (req, res) => {  
      res.json({ message: 'This is JSON data from Express' });  
     });  
       
     app.listen(PORT, () => {  
      console.log(`Express server running at http://localhost:${PORT}/`);  
     });
   * Jalankan server dengan perintah: node app.js

\*\*\*Karena kita sudah membuat project node server di pertemuan 1, maka silahkan lanjut ke step berikut:

1. Middleware untuk Body Parser dan CORS
   * Modifikasi app.js atau server.js pada project node server:
   * const express = require('express');
   * const cors = require('cors');
   * const app = express();
   * const PORT = 3001;
   * // Middleware
   * app.use(cors());
   * app.use(express.json());
   * app.use((req, res, next) => {
   * console.log(`${new Date().toISOString()} - ${req.method} ${req.url}`);
   * next();
   * });
   * app.get('/', (req, res) => {
   * res.send('Home Page for API');
   * });
   * app.listen(PORT, () => {
   * console.log(`Express server running at http://localhost:${PORT}/`);
   * });
2. RESTful API Sederhana
   * Buat folder dengan nama **routes,** kemudian buat file **books.js** di dalamnya untuk menampung semua rute terkait buku, isi dengan code berikut:
   * const express = require('express');
   * const router = express.Router();
   * let books = [
   * {id: 1, title: 'Book 1', author: 'Author 1'},
   * {id: 2, title: 'Book 2', author: 'Author 2'}
   * ];
   * router.get('/', (req, res) => {
   * res.json(books);
   * });
   * router.get('/:id', (req, res) => {
   * const book = books.find(b => b.id === parseInt(req.params.id));
   * if (!book) return res.status(404).send('Book not found');
   * res.json(book);
   * });
   * router.post('/', (req, res) => {
   * const { title, author } = req.body;
   * if (!title || !author) {
   * return res.status(400).json({ message: 'Title and author are required' });
   * }
   * const book = {
   * id: books.length + 1,
   * title,
   * author
   * };
   * books.push(book);
   * res.status(201).json(book);
   * });
   * module.exports = router;
   * Modifikasi **server.js** untuk menggunakan API router, tambahkan baris code berikut:

Panggil logika *routing* yang khusus menangani "buku" dari file lain:

* + const bookRoutes = require('./routes/books');
  + app.use('/api/books', bookRoutes);
  + Baris ini menggunakan atau "memasang" *router* yang sudah diimpor tadi ke dalam aplikasi Express utama.

1. Uji endpoint menggunakan PostMan atau ThunderClient (extention VS Code)
   * Buka Postman dan buat *request* baru. Pastikan node server sudah jalan.
   * Di kolom URL, masukkan alamat lengkap ke *endpoint* Anda:

<http://localhost:3001/api/books>

A screenshot of a computer

AI-generated content may be incorrect.

* + Di sebelah kiri kolom URL, ubah metode HTTP dari **GET** (default) menjadi **POST,** untuk menjalankan create data books:
  + Karena Anda akan mengirim data, klik pada tab **Body** di bawah kolom URL. Pilih opsi **raw.**
  + Di menu *dropdown* di sebelah kanan, ubah Text menjadi **JSON**. Ini sangat penting agar Postman mengirim data dalam format yang benar.

A screenshot of a computer

AI-generated content may be incorrect.

## TUGAS

1. Buat sebuah aplikasi web sederhana menggunakan Express.js yang memiliki fitur CRUD untuk manajemen buku perpustakaan.
   * Implementasikan routing untuk operasi CRUD (Create, Read, Update, Delete) buku
   * Gunakan array atau file JSON sebagai penyimpanan data sementara
   * Implementasikan validasi input untuk setiap operasi CRUD
2. Implementasikan middleware untuk logging setiap request yang masuk ke server.
   * Log harus mencakup timestamp, method HTTP, dan URL yang diakses
3. Implementasikan error handling yang baik.
   * Gunakan middleware untuk menangani 404 Not Found
   * Implementasikan global error handler
4. Tambahkan Folder TUGAS diluar folder project node dan react:

* Buat file Tugas2.md
* Isi dengan screenshot API endpoint CRUD seperti dibawah ini:

A screenshot of a computer

AI-generated content may be incorrect.

1. Push project ke GitHub dengan Repository yang sudah ada.
2. Kirim link file **Tugas2.md** dari repository GitHub ke dalam MyKlass praktikum PAW.