Modul 05

Web Server dan Express.js

TUJUAN PEMBELAJARAN

1. Mampu memahami dan menjelaskan tentang Web Server
2. Mampu memahami dan menjelaskan tentang Express.JS sebagai Web Framework

Hardware & Software

1. PC (*Personal Computer)* dengan akses Internet
2. JavaScript
3. Visual Studio Code atau IDE lainnya yang mendukung JavaScript
4. NPM
5. Express.JS

URAIAN MATERI

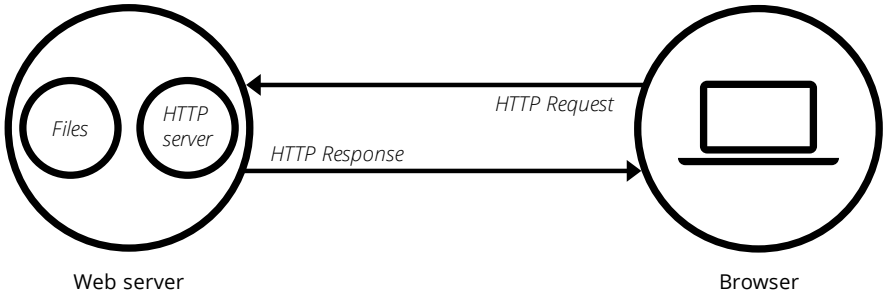
A. Web Server

Server web (web server) adalah entitas berupa komputer fisik atau perangkat lunak yang berfungsi sebagai pusat penyimpanan dan pengiriman data dalam bentuk halaman web kepada pengguna yang meminta akses. Ketika seorang pengguna mengakses suatu situs web, permintaannya diteruskan ke server web yang bersangkutan. Selanjutnya, server web ini mengolah permintaan tersebut dan mengirimkan halaman web yang sesuai ke perangkat pengguna melalui jaringan internet. Istilah *web server* dapat merujuk pada perangkat keras atau perangkat lunak, atau keduanya yang saling bekerja bersama.

- Dari sisi perangkat keras, *web server* adalah komputer yang menyimpan perangkat lunak server web dan file komponen situs web (misalnya, dokumen HTML, gambar, stylesheet CSS, dan file JavaScript). Sebuah server web terhubung ke Internet dan mendukung pertukaran data fisik dengan perangkat lain yang terhubung ke web.
- Dari sisi perangkat lunak, server web mencakup beberapa bagian yang mengendalikan bagaimana pengguna web mengakses file-file yang dihosting. Paling minimum adalah server HTTP. Sebuah server HTTP adalah perangkat lunak yang memahami URL (alamat web) dan

HTTP (protokol yang digunakan oleh browser untuk melihat halaman web). Sebuah server HTTP dapat diakses melalui nama domain situs web yang disimpan di dalamnya, dan mengirimkan konten dari situs web yang dihosting tersebut ke perangkat pengguna akhir.

Pada level paling dasar, setiap kali browser membutuhkan sebuah file yang di-host di server web, browser tersebut meminta file melalui HTTP. Ketika permintaan mencapai server web yang tepat (perangkat keras), server HTTP (perangkat lunak) menerima permintaan tersebut, menemukan dokumen yang diminta, dan mengirimkannya kembali ke browser, juga melalui HTTP. (Jika server tidak menemukan dokumen yang diminta, maka akan mengembalikan respons 404).



Gambar 1. Proses komunikasi browser dan web server menggunakan HTTP protocol

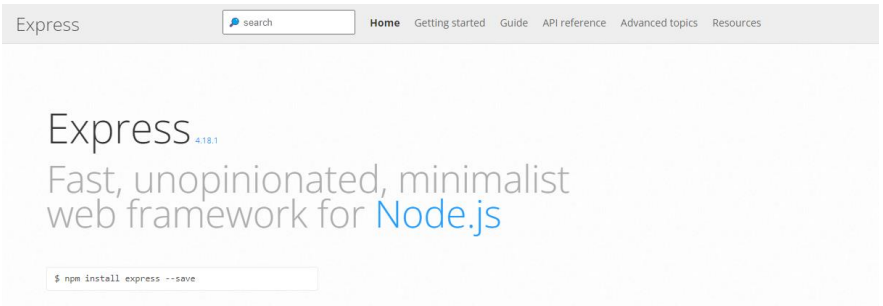
Untuk mempublikasikan sebuah situs web, diperlukan server web yang bersifat statis maupun dinamis.

- Sebuah server web statis, atau *stack*, terdiri dari komputer (perangkat keras) dengan server HTTP (perangkat lunak). Hal ini disebut "statis" karena server ini mengirimkan file yang di-host-nya apa adanya ke browser Anda.
- Sebuah server web dinamis terdiri dari server web statis ditambah perangkat lunak tambahan, yang paling umum adalah server aplikasi dan database. Hal ini disebut "dinamis" karena server aplikasi memperbarui file yang di-host sebelum mengirimkan konten ke browser Anda melalui server HTTP.

Sebagai contoh, untuk menghasilkan halaman web akhir yang bisa dilihat di browser, server aplikasi mungkin mengisi sebuah *template* HTML dengan konten dari database. Situs seperti Wikipedia memiliki ribuan halaman web. Biasanya, jenis situs seperti ini terdiri dari hanya beberapa *template* HTML dan sebuah database besar, bukan ribuan dokumen HTML statis. Pengaturan ini membuatnya lebih mudah untuk merawat dan mengirimkan konten.

B. ExpressJS

Express.js, atau sering disebut hanya "Express," adalah framework web yang berjalan di atas bahasa pemrograman JavaScript. Ini merupakan salah satu framework yang paling populer dan umum digunakan untuk pengembangan aplikasi web di sisi server (server-side) dengan menggunakan bahasa JavaScript.



Gambar 2. Halaman web Expressjs yang dapat diakses melalui https://expressjs.com/

Express.js dan Node.js memiliki hubungan erat, di mana Express.js adalah sebuah framework web yang dibangun di atas platform Node.js. Ini berarti Express.js adalah salah satu dari banyak framework yang dapat digunakan untuk mengembangkan aplikasi web dengan menggunakan Node.js sebagai lingkungan runtime-nya.

Node.js adalah platform runtime JavaScript yang memungkinkan Anda menjalankan kode JavaScript di sisi server (server-side). Ini sangat efisien untuk menangani banyak permintaan I/O yang bersifat non-blocking, yang merupakan fitur kunci dalam pengembangan aplikasi web yang skalabel dan responsif. Express.js, di sisi lain, adalah framework yang dirancang khusus untuk memudahkan pengembangan aplikasi web dengan menggunakan Node.js. Ini menyediakan alat dan fitur tingkat tinggi untuk mengatur permintaan HTTP, menangani rute, mengelola sesi, dan banyak lagi. Express.js memungkinkan pengembang untuk membangun aplikasi web dengan cepat dan efisien dengan menggunakan Node.js sebagai basisnya. Pada tabel berikut dapat dilihat perbedaan express.js dan node.js

Tabel 1. Perbedaan Express.js Vs Node.js

Aspek Pembeda	**Node.js**	**Express.js**
Kegunaan	Digunakan untuk mengembangkan frontend dan backend dari sebuah aplikasi web.	Digunakan untuk mengembangkan backend dari sebuah aplikasi web.
Definisi	Ini adalah platform pengembangan *(development platform)*	Ini adalah kerangka kerja web *(web framework)*
Sisi Pemrograman	Digunakan untuk pemrograman sisi klien dan sisi server.	Digunakan untuk pemrograman sisi server.
Fitur	Fitur yang lebih sedikit dibandingkan.	Lebih banyak fitur daripada Node.js.
Core technology[1]	Dibangun di atas mesin V8 Google.	Dibangun di atas Node.js
Ditulis dalam Bahasa?	C, C++, JavaScript	Express.js ditulis dalam JavaScript

Mendukung Bahasa?	TypeScript, CoffeeScript, Ruby	JavaScript
Performa	Kinerja lebih baik daripada Express.js.	Secara relatif, tidak sebaik kinerja Node.js
Ketersediaan *controller provision*[2]	Tidak	Ya
Mendukung *routing*[3]?	Tidak	Ya
Coding Time	Membutuhkan lebih banyak waktu	Membutuhkan lebih sedikit waktu

Catatan:
1. teknologi inti yang membangun aplikasi
2. *'controller provision'* merujuk pada komponen yang mengelola aliran data dan interaksi dalam sebuah aplikasi dan bertanggung jawab untuk menangani input pengguna dan mengkoordinasikan Tindakan antara model (data) dan tampilan (antarmuka pengguna).
3. *'routing'* menentukan bagaimana berbagai URL atau *endpoint* berhubungan dengan fungsi *(function)* atau penanganan *(handlers)* tertentu, memungkinkan pengembang untuk mendefinisikan bagaimana aplikasi merespons permintaan klien yang berbeda.

Meski tabel diatas menunjukkan perbedaan node.js dan express.js, pada dasarnya keduanya tidak berdiri sendiri, melainkan saling bekerjasama dalam pengembangan sebuah aplikasi. Umumnya Express.js digunakan dalam pengembangan aplikasi-aplikasi berikut seperti single-page apps, reusable apps, aplikasi *middleware*, RESTful APIs, Melayani dan Mengakses Berkas Statis Menggunakan Browser, Apliksi web Enterprise dan aplikasi web e-commerce. Adapun beberapa keunggulan Express.js adalah sebagai berikut.

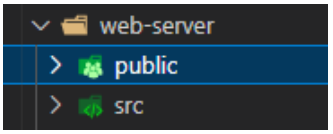
1. **Durasi Pengembangan yang Lebih Singkat**
   Istilah *'Express'* sendiri mengindikasikan bahwa Express.js dikenal karena pengembangannya yang cepat. Express.js menggunakan bahasa JavaScript, yang juga memungkinkan pemrograman yang cepat.
2. **Penanganan Kesalahan *(Error Handling)* Express.js**
   Proses penanganan kesalahan Express.js dirancang sedemikian rupa sehingga mampu mendeteksi dan memperbaiki *bug* dalam kode yang bersifat sinkron dan asinkron.
3. **Penanganan Permintaan I/O *(I/O Request Handling)* yang Sesuai**
   Tergantung pada musimnya, bisnis aplikasi berbasis *on-demand* dapat menerima ratusan atau bahkan ribuan notifikasi per hari. Jika dibangun dalam kerangka kerja backend Express, aplikasi-aplikasi ini memiliki keunggulan, karena kerangka kerja ini cukup kuat untuk menerima berbagai permintaan Masukan *(Input)* dan Keluaran *(Output)* secara bersamaan.
4. **Kerangka Kerja yang bersifat non-preskriptif *(Unopinionated Framework)***
   Pengembang menyukai kerangka kerja Express, karena tidak memiliki aturan ketat tentang penempatan atau urutan komponen kode.

5. **Instalasi yang Cepat & Mudah**
   Menginstal kerangka kerja adalah tugas yang sangat menjengkelkan dan membutuhkan waktu. Namun, menginstal Express.js cukup mudah. Selain itu, ia juga lebih sederhana untuk diatur, dikonfigurasi, dan disesuaikan.
6. **Kemampuan untuk Membuat Server API REST Express.js *(REST Expressjs API Server)***
   Peran utama dari server API REST adalah untuk mengakses dan menggunakan data. Dan tidak ada yang lebih memahami pentingnya membuat server API REST daripada seorang pengembang atau seorang ahli IT. Express memungkinkan pengguna pengembang untuk membuatnya dengan tingkat keamanan yang maksimal.
7. **Integrasi Basis Data *(Database Integration)* yang Mudah**
   Express adalah kerangka kerja backend yang mudah dikelola. Programmer dapat mengintegrasikan basis data apa pun dengan mudah seperti MongoDB, Redis, MySQL.

LATIHAN

a. Instalasi Express.js dan membuat halaman menggunakan fungsi app.get

1. Buatlah folder baru dengan nama **web-server** dan bukalah folder tersebut pada aplikasi visual studio code anda.
2. Buka direktori ke folder dengan cara **klik kanan**, lalu pilih **Open in Integrated Terminal.** Lalu ketikan perintah **`npm init -y`** untuk meng-*generate* file **package.json**
3. Selanjutnya install **express.js** versi **terbaru** yaitu versi 4.18.2 menggunakan perintah npm yang telah anda pelajari sebelumnya. Silakan cek package express.js melalui halaman ini https://www.npmjs.com/package/express. Buka **modul sebelumnya** jika anda lupa perintah atau cara menginstall package dari npm
4. Anda akan melihat bahwa folder baru **node_modules** dan file baru **package-lock.json** telah berhasil di *generate*
5. Buatlah **dua** buah folder baru dalam folder **web-server** melalui visual studio code anda. Beri nama folder pertama dengan **public** dan folder kedua dengan **src**



Gambar 3 Direktori aplikasi web-servers

6. Didalam folder **src**, buatlah file baru dengan nama **app.js**
7. Ketikan kode berikut ini

```
const express = require('express')

const app = express()

//ini halaman/page utama
```

```
app.get('', (req, res) => {
    res.send('Selamat datang di halaman utama')
})
//ini halaman bantuan/FAQ (Frequently Asked Questions)
app.get('/bantuan', (req, res) => {
    res.send('Ini halaman bantuan')
})
```

Catatan:

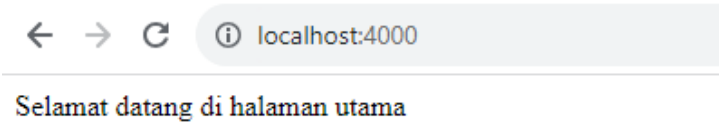
a. **`app = express()`** menunjukan pembuatan sebuah variable constant baru dengan nama **app** dan diambil dari fungsi express yang diimport dari expressjs
b. Method **`app.get`** mendefinisikan sebuah rute HTTP GET menggunakan objek app dari Express.js
c. **`(req, res) => { ... }`** Bagian dari kode ini mendefinisikan sebuah fungsi callback yang akan dieksekusi ketika rute diakses. Dalam Express.js, fungsi ini memiliki dua argumen: req (objek *request*) dan res (objek *response*). Di dalam fungsi ini, Anda mendefinisikan apa yang harus terjadi ketika pengguna mengakses rute ini.
d. **`res.send(...)`** digunakan untuk mengirimkan respons kembali ke klien (browser pengguna)
e. Expressjs memiliki banyak jenis method yang dapat diakses lebih jauh melalui dokumentasi berikut https://expressjs.com/en/4x/api.html

8. Lanjutkan koding diatas untuk membuat dua halaman lagi yaiut halaman **infoCuaca** dan halaman **tentang**. Lakukan sebagaimana mana baris kode halaman **bantuan** diatas
9. Lalu tambahkan baris kode berikut **diakhir** semua kode diatas

```
app.listen(4000, () => {
    console.log('Server berjalan pada port 4000.')
})
```

Anda dapat mengganti server port 4000 dengan 3000, 5000 atau yang lainnya

10. Buka direktori ke file app.js tersebut melalui terminal dengan cara **klik kanan**, lalu pilih **Open in Integrated Terminal**
11. Pastikan anda berada pada direktori **src,** lalu ketikan perintah **`node app.js`** untuk menjalankan program tersebut
12. Bukalah web browser anda lalu ketikan **http://localhost:4000/.** Angka 4000 dapat diganti sesuai dengan port yang telah anda *set* pada bari kode `app.listen`. Akan tampil seperti gambar dibawah ini



Gambar 4. Tampilan localhost:4000

13. Cobalah mengakses halaman lainnya dengan menambahkan url halaman dibelakang url utama. Contoh: http://localhost:4000/infoCuaca. Coba lakukan juga untuk halaman bantuan dan halaman tentang

b. Integrasi HTML dan JSON pada Express.js

1. Cobalah ganti teks **`selamat datang di halaman utama`** dengan kode html <h1> sehingga menjadi

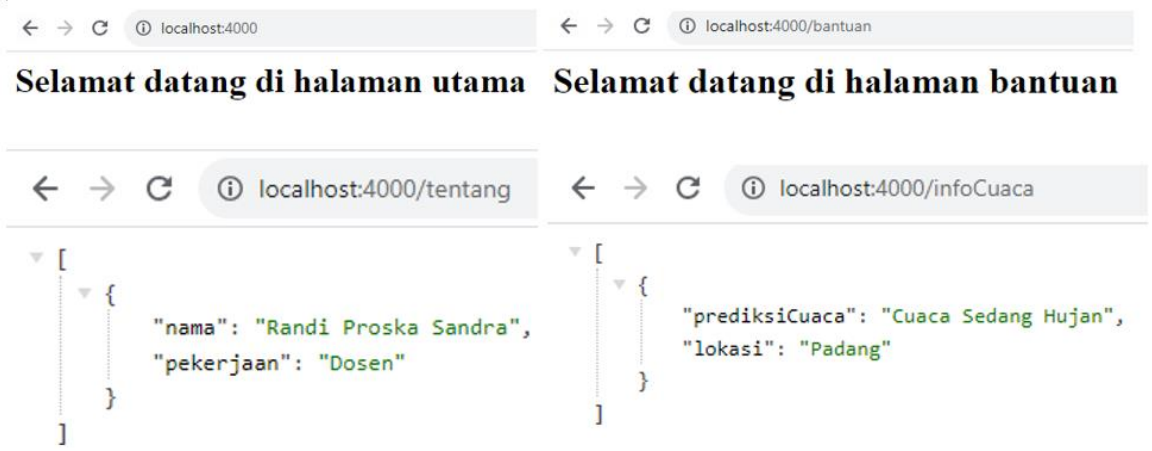
```
res.send('<h1>Selamat datang di halaman utama</h1>')
```

2. Jalankan file app.js menggunakan perintah **`nodemon app.js`** dan lihat hasilnya pada browser. **PENTING!** Untuk selanjutnya, anda akan menggunakan perintah **nodemon** untuk menjalankan program bukan lagi **node**. Hal ini bertujuan agar perubahan dapat dilihat secara realtime setiap anda mengubah baris kode.
3. Selanjutnya cobalah ganti kode pada halaman **tentang** dengan kode berikut

```
app.get('/tentang, (req, res) => {
    res.send([{
        nama: 'Randi Proska Sandra',
        pekerjaan: 'Dosen'
    }])
})
```

Catatan: Teks pada nama dan pekerjaan dapat anda sesuaikan dengan keinginan anda.

4. Bukalah aplikasi anda di browser dan pastikan anda beradal url halaman **tentang.** http://localhost:4000/tentang
5. Perhatikan bahwa tampilan pada halaman utama berbeda dengan halaman tentang. Hal ini karena halaman utama menggunakan format HTML sedangkan halaman tentang menggunakan format JSON
6. Ubah juga teks untuk halaman **bantuan** dan halaman **infoCuaca.** Tambahkan teks pada halaman **bantuan** dengan kode HTML <h1> dan teks infoCuaca dengan kode JSON yang berisi dua objek yaitu **`prediksiCuaca: 'cuaca berpotensi hujan'`** dan **`lokasi: 'Padang'`**, sama seperti yang anda lakukan pada halaman **tentang**.
7. Bukalah setiap halaman di browser anda dan pastikan bahwa semuanya menampilkan tampilan yang diinginkan.



Gambar 5. Halaman utama (kiri atas), halaman bantuan (kanan atas), halaman tentang (kiri bawah) dan halaman infoCuaca (kanan bawah)

8. **PENTING!** Untuk selanjutnya, setiap kali anda melakukan perubahan pada program anda, pastikan untuk mengecek halaman yang anda ubah. Hal ini bertujuan untuk memastikan konten yang tampil sudah sesuai dengan yang diinginkan dan tidak ada *error* yang muncul

c. Akses Static Assets pada Expressjs menggunakan Path module

1. Buatlah file baru pada folder **public** dan beri nama **index.html** dan tambahkan kode html berikut

```
<!DOCTYPE html>

<html>
<head></head>
   <body>
       <h1>Ini adalah halaman utama statis</h1>
   </body>
</html>
```

2. Selanjutnya, pada file **app.js** di folder **src**, tambakan kode berikut ini

```
1  const path = require('path')
2  const express = require('express')
3
4  const app = express()
5  const direktoriPublic = path.join(__dirname, '../public')
6
7  app.use(express.static(direktoriPublic))
```

Gambar 6. Potongan baris kode pada file app.js

Pastikan! anda hanya menambahkan kode yang belum ada pada baris program anda. Jangan menghapus kode yang sudah ada

Catatan:

a. Module **path** menyediakan utilitas untuk bekerja dengan path file dan direktori yang dapat dipelajari lebih jauh melalui dokumentasi berikut ini https://nodejs.org/api/path.html#path
b. **`path.join()`** adalah method untuk menggabungkan semua segmen path yang diberikan menggunakan pemisah yang spesifik, kemudian melakukan normalisasi terhadap jalur yang dihasilkan.
c. **`__dirname`** menunjukkan bahwa yang diakses adalah direktori dalam hal ini adalah direktori public. Jika anda mengakses file maka akan menggunakan **`__filename`**
d. **`express.static`** dapat dipahami melalui dokumentasi berikut http://expressjs.com/en/api.html#express.static

3. Lanjutkan program dengan membuat dua file html lagi pada folder public dengan nama **tentang.html** dan **bantuan.html.** Silakan isi file html dengan baris kode html seperti pada halaman **index.html**. Gantilah teks pada baris kode **<h1>** agar sesuai dengan nama halaman.
4. Silakan akses aplikasi pada browser anda untuk melihat tampilannya. Cobalah akses masing-masing halaman dengan menambahkan .html dibelakang. Contoh: http://localhost:4000/tentang.html

5. Selanjutnya buatlah **folder baru** pada folder **public** dengan nama **css.** Lalu, dalam folder tersebut, buatlah file baru dengan nama **styles.css**. Lalu tambahkanlah kode berikut ini

```
h1 {
    color: grey;
}

img {
    width: 250px;
}
```

6. Selanjutnya tambahkan baris kode berikut pada file **index.html** pada didalam heading <head>. Lakukan yang sama untuk file **bantuan.html** dan **tentang.html**

```
<link rel="stylesheet" href="/css/styles.css">
```

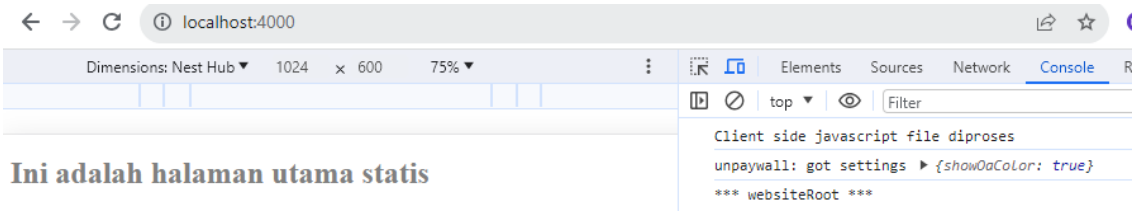
7. Buatlah satu **folder baru** lagi di folder public dengan nama **js**. Lalu buatlah **file baru** dalam folder tersebut dengan nama **app.js** dan tambahkan kode berikut ini

```
console.log('Client side javascript file diproses')
```

8. Lalu tambahkan lagi baris kode berikut pada file index.html, tepat dibawah baris kode pada nomor 6

```
<script src="/js/app.js"></script>
```

9. Jalankan program anda di browser. Sembari membuka halaman utama, cobalah inspect elemen pada browser chrome/firefox anda dan perhatikan bahwa fille **app.js** dalam folder **js** telah di proses

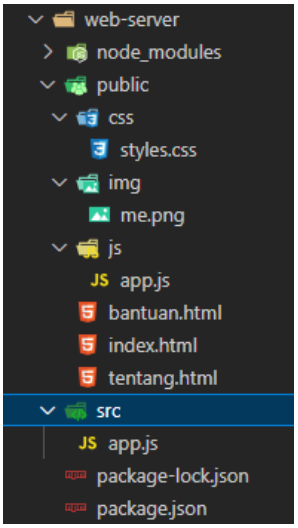


Gambar 7. Tampilan pemrosesan file javascript sisi client pada browser

10. Selanjutnya buatlah folder baru lagi dengan nama **img.** Lalu masukanlah sebuah gambar berformat **.png** ke folder tersebut. Direkomendasikan memasukan foto anda.
11. Lalu tambahkan baris kode berikut pada file **tentang.html** tepat dibawah kode <h1> pada bagian <body>

```
<img src="/img/ISI_DENGAN_NAMA_FILE_GAMBAR_ANDA.png">
```

12. Silakan buka localhost pada browser anda untuk mengakses halaman tentang, bantuan dan halaman utama. Pastikan akhir halaman anda berekstensi **.html.** Contoh: http://localhost:4000/tentang.html
13. Keseluruhan direktori yang ada pada aplikasi anda adalah seperti pada gambar dibawah ini



Gambar 8. Direktori dan file pada aplikasi web server

d. Templating pada Expressjs menggunakan Handlebars.js

Pada bagian ini, anda akan menggunakan **handlebars**. Handlebars.js adalah perluasan dari bahasa templating Mustache yang diciptakan oleh Chris Wanstrath. Handlebars.js dan Mustache keduanya adalah bahasa templating yang tidak memiliki logika yang menjaga pemisahan antara tampilan dan kode. Handlebars dapat dikatakan sebagai sebuah mesin template (template engine) yang dapat digunakan bersama dengan Express.js untuk menghasilkan tampilan dinamis dalam aplikasi web. Handlebars memungkinkan untuk memisahkan logika aplikasi dari tampilan pada aplikasi berbasis web. Handlebars menggunakan sintaksis yang mudah dibaca dan dapat memasukkan variabel, loop, kondisi, dan lainnya ke dalam template HTML.

1. Silakan stop nodemon anda pada terminal di visual studio code dengan menekan **`ctrl + c`**
2. Lakukan instalasi **hbs** menggunakan perintah npm. hbs adalah sebuah mesin tampilan *(view engine)* express.js untuk handlebars.js. Silakan akses informasi lebih jauh terkait hbs melalui url berikut https://www.npmjs.com/package/hbs
3. Setelah instalasi selesai jalankan kembali aplikasi web-server anda menggunakan perintah **nodemon app.js**
4. Bukalah file **app.js** yang ada di folder **src** dan tambahkan baris kode berikut tepat diatas baris kode **`app.use(express.static(direktoriPublic))`**

```
app.set('view engine', 'hbs')
```

PENTING! lalu, jadikan baris kode **app.use** diatas menjadi **komentar**

5. Ubah kembali kode pada baris **app.get** halaman **utama** dan app.get halaman **tentang** pada file **app.js** di folder **src** dengan kode berikut

```
//ini halaman utama
app.get('', (req, res) => {
    res.render('index', {
        judul: 'Aplikasi Cek Cuaca',
        nama: 'Randi Proska Sandra'
    })
```

```
})
//ini halaman tentang
app.get('/tentang', (req, res) => {
    res.render('tentang', {
        judul: 'Tentang Saya',
        nama: 'Randi Proska Sandra'
    })
})
```

6. Lanjutkan program, lakukan hal yang sama untuk halaman **bantuan**. Pada baris kode halaman bantuan, tambahkan satu objek lagi yaitu **`teksBantuan: 'ini adalah teks bantuan'.`** Khusus halaman infoCuaca biarkan saja seperti kode awal.
7. Lalu, buatlah folder baru pada aplikasi web-server anda dan beri nama **views.** Didalam folder ini, buatlah file baru dengan nama **index.hbs**
8. Salinlah semua baris kode yang ada pada file **index.html** ke file **index.hbs.** Lalu gantilah kode yang ada dalam <body> dengan baris kode berikut

```
<h1>{{judul}}</h1>
<p>Dikembangkan oleh {{nama}}</p>
```

Catatan:
`{{title}}` menunjukkan bahwa file index.hbs akan mengakses title yang pada baris kode di file app.js. Proses pengaksesan ini telah diatur oleh handlebars. Hal ini memungkinkan penggunaan kode secara berulang.

9. Buat dua buah file lagi dalam folder **views** dengan nama **bantuan.hbs** dan **tentang.hbs.**
10. Lalu salinlah baris kode **bantuan.html** ke **bantuan.hbs** dan ubahlah baris kode yang ada didalam <body> dengan kode berikut

```
<h1>Bantuan Apa yang anda butuhkan?</h1>
<p>{{teksBantuan}}</p>
<p>Dikembangkan oleh {{nama}}</p>
```

11. Lalu salin baris kode **tentang.html** ke **tentang.hbs** dan ubahlah baris kode yang ada didalam <body> dengan kode berikut

```
<h1>{{judul}}</h1>
<img src="/img/ISI_DENGAN_NAMA_FILE_GAMBAR_ANDA.png">
<p>Dikembangkan oleh {{nama}}</p>
```

12. Jika langkah-langkah diatas sudah selesai, buatlah folder baru didalam folder **public** dengan nama **file_html.** Lalu pindahkan semua file **.html** kedalam folder ini. Hal ini bertujuan agar aplikasi tidak membaca file .html lagi melainkan file **.hbs**. Dalam praktik sebenarnya, anda dapat menghapus semua file .html karena tidak akan digunakan lagi. Selain itu file .html ini juga bersifat statis. Namun, dalam praktikum ini file .html tetap dibiarkan ada sebagai bukti apa yang sudah anda kerjakan.
13. **Ubahlah nama** folder views dengan nama **templates.** Lalu pahami *error* yang tampil pada browser anda. *Error* ini terjadi karena expressjs menganggap folder views sebagai lokasi default untuk file-file yang terkait dengan tampilan website kita. Hal ini tidak efektif jika memiliki banyak file yang terkait dengan tampilan (views)

14. Untuk mengatasinya maka kita harus meng-kustomisasi path ke direktori folder views. Tambahkanlah kode berikut kedalam file **app.js** yang ada folder **src.** **Penting!** Cukup tambahkan baris kode yang belum ada pada baris kode anda

```
1  const path = require('path')
2  const express = require('express')
3  const hbs = require('hbs')
4
5  const app = express()
6
7  // Mendefinisikan jalur/path untuk konfigurasi Express
8  const direktoriPublic = path.join(__dirname, '../public')
9  const direktoriViews = path.join(__dirname, '../templates')
10
11 // Setup handlebars engine dan lokasi folder views
12 app.set('view engine', 'hbs')
13 app.set('views', direktoriViews)
14
15 // Setup direktori statis
16 app.use(express.static(direktoriPublic))
```

Gambar 9. Potongan baris kode file app.js langkah d14

Baris kode baru yang seharusnya anda tambahkan berada pada baris 3, 9 dan 13.

15. Cek kembali program anda melalui browser apakah *error* sudah teratasi.

e. Mengatur Tampilan Aplikasi menggunakan sistem templating

Bagian ini hanya melanjutkan **bagian d Langkah 15** yang bertujuan untuk mengatur tampilan aplikasi anda sehingga lebih menarik.

1. Silakan hentikan nodemon pada terminal visual studi code dengan menekan **`ctrl + c`**
2. Lalu jalankan kembali dengan perintah berikut **`nodemon app.js -e js,hbs.`** **PENTING!** Untuk selanjutnya, anda akan menggunakan perintah ini untuk menjalankan program anda.
3. Selanjutnya, buatlah **dua** folder baru didalam folder **templates.** Beri nama folder tersebut dengan **views** dan **partials.** Folder **views** akan berisi file tampilan utama, sementara folder **partials** nantinya akan berisi file tampilan partial seperti header dan footer. Hal ini diperlukan untuk membuat beberapa baris kode program menjadi dapat digunakan kembali *(reusable)*.
4. Masukan file **index.hbs**, **bantuan.hbs** dan **tentang.hbs** ke folder **views**
5. Buatlah dua buah file didalam folder **partials.** File pertama beri nama **header.hbs** dan yang kedua **footer.hbs**
6. Sesuaikanlah baris kode berikut pada file app.js anda. Pastikan anda hanya mengubah yang perlu diubah.

```
// Mendefinisikan jalur/path untuk konfigurasi Express
const direktoriPublic = path.join(__dirname, '../public')
const direktoriViews = path.join(__dirname, '../templates/views')
const direktoriPartials = path.join(__dirname, '../templates/partials')

// Setup handlebars engine dan lokasi folder views
app.set('view engine', 'hbs')
app.set('views', direktoriViews)
hbs.registerPartials(direktoriPartials)
```

Gambar 10. Potongan baris kode file app.js langkah e6

Pada tampilan diatas seharusnya anda hanya perlu mengubah baris 9, 10 dan 15

7. Masukan baris kode berikut pada file **header.hbs**

```
<header>
    <h1>{{judul}}</h1>

    <a href="/">Home</a>
    <a href="/tentang">Tentang</a>
    <a href="/bantuan">Bantuan</a>
</header>
```

Gambar 11. Potongan baris kode header.hbs langkah e7

8. Lalu, masukan baris kode berikut pada file **footer.hbs**

```
<footer>
    <p>Dikembangkan oleh by {{nama}}</p>
</footer>
```

9. Unduh icon cuaca berikut ini https://cdn-icons-png.flaticon.com/512/1779/1779940.png dan letakan pada folder **img** di aplikasi web-server anda. Jika anda ingin menggunakan gambar lainnya, silakan cari di google dan pastikan formatnya **.png** dan tidak memiliki *background*
10. Tambahkanlah baris kode berikut pada didalam bagian **<head>** pada file **index.hbs**

```
<title>Aplikasi Cek Cuaca</title>
<link rel="icon" href="/img/cuaca.png">
<link rel="stylesheet" href="/css/styles.css">
<script src="/js/app.js"></script>
```

Silakan sesuaikan teks yang di blok dengan warna merah dengan nama file icon anda

11. Lakukan kembali langkah 8 untuk file **bantuan.hbs** dan **tentang.hbs**. Ubahlah teks yang di blok hijau sesuai dengan judul halaman masing-masing.
12. Ubalah kode pada file style.css dengan kode berikut ini

```
body {
    color: #333333;
    font-family: arial;
    max-width: 650px;
    margin: 0 auto;
    padding: 0 16px;

    display: flex;
    flex-direction: column;
    min-height: 100vh
```

```
}

.main-content {
    flex-grow: 1;
}

footer {
    color: #888888;
    border-top: 1px solid #eeeeee;
    margin-top: 16px;
    padding: 16px 0;
}

header {
    margin-top: 16px;
    margin-bottom: 48px;
}

h1 {
    font-size: 64px;
    margin-bottom: 16px;
}

header a {
    color: #888888;
    margin-right: 16px;
    text-decoration: none;
}

.portrait {
    width: 250px;
}
```

13. Ubah lagi kode pada file index.hbs, tentang.hbs dan bantuan.hbs yang ada dalam bagian **<body>** dengan kode berikut
    a) Baris kode didalam **<body>** pada file **index.hbs**

```
    <div class="main-content">
        {{>header}}
        <p>Aplikasi ini digunakan untuk mengecek Cuaca!</p>
    </div>
    {{>footer}}
```

Gambar 12. Potongan baris kode index.hbs langkah e13a

    b) Baris kode didalam **<body>** pada file **tentang.hbs**

```
    <div class="main-content">
        {{>header}}
        <img class="portrait" src="/img/GANTI_DENGAN_NAMA_FILE.png">
    </div>
    {{>footer}}
```

Gambar 13. Potongan baris kode tentang.hbs langkah e13b

c) Baris kode didalam **<body>** pada file **bantuan.hbs**

```
<div class="main-content">
    {{>header}}
    <p>{{teksBantuan}}</p>
</div>
{{>footer}}
```

Gambar 14. Potongan baris kode tentang.hbs langkah e13c

Catatan:

{{>header}} dan **{{>footer}}** digunakan mengakses file header dan footer yang ada pada folder partials

14. Berikut adalah seharusnya tampilan akhir aplikasi anda

Gambar 15. Tampilan akhir aplikasi web-server – halaman home (kiri), halaman tentang (tengah) dan halaman bantuan (kanan)

f. Wildcard route pada Expressjs

Expressjs memungkinkan pembuatan routing menggunakan wildcard seperti tanda *. Hal ini biasa digunakan programmer untuk menunjukan bahwa sebuah halaman (page) tidak ada seperti error 404.

1. Silakan tambahkan baris kode berikut pada file **app.js** yang ada didalam folder **src.** Letakan tepat **diatas app.listen** dan **dibawah app.get** untuk halaman infoCuaca

```
app.get('/bantuan/*', (req, res) => {
    res.render('404', {
        judul: '404',
        nama: 'Randi Proska Sandra',
        pesanKesalahan: 'Artikel yang dicari tidak ditemukan.'
    })
})

app.get('*', (req, res) => {
    res.render('404', {
        judul: '404',
        nama: 'Randi Proska Sandra',
        pesanKesalahan: 'Halaman tidak ditemukan.'
    })
})
```

Gambar 16. Potongan baris kode file app.js langkah f1 tentang wildcard

Catatan:

a. **app.get('/bantuan/*'……..)** digunakan untuk menampilkan pesan kesalahan ketika sebuah halaman artikel pada url …./bantuan tidak ditemukan
b. **app.get('*'……..)** digunakan untuk menampilkan pesan kesalahan ketika sebuah halaman di url manapun tidak ditemukan

2. Selanjutnya, buatlah file baru pada folder templates/**views** dengan nama **404.hbs** dan tambahkan baris kode berikut ini

```
<!DOCTYPE html>
<html>
<head>
    <title>404</title>
    <link rel="icon" href="/img/cuaca.png">
    <link rel="stylesheet" href="/css/styles.css">
    <script src="/js/app.js"></script>
</head>
<body>
    <div class="main-content">
        {{>header}}
        <p>{{pesanKesalahan}}</p>
    </div>
    {{>footer}}
</body>
</html>
```

3. Silakan ketikan url berikut pada browser anda http://localhost:4000/tes dan http://localhost:4000/bantuan/tes. Lalu pahamilah apa yang ditampilkan.

TUGAS

1. Perhatikan bahwa didalam kode anda pada file **index.hbs, tentang.hbs, bantuan.hbs** dan **404.hbs** terjadi penggunaan beberapa baris kode berulang pada bagian **<head>**. Buatlah kode tersebut menjadi lebih efisien dengan dengan menampungnya pada file **.hbs** terpisah didalam folder **partials**. Baris kode pada bagian **<title>teks</title>** harus tetap bersifat dinamis sehingga ketika halaman diganti maka title halaman juga berganti.

REFERENCES

1. Casciaro, M., & Mammino, L. (2020, July). *Node. Js Design Patterns* (3rd ed.) [Packtpub]. Packt Publishing. https://www.packtpub.com/product/nodejs-design-patterns-third-edition/9781839214110
2. Pasquali, S. (2013, November). *Mastering Node.js* [Packtpub]. Packt Publishing. https://www.packtpub.com/product/mastering-nodejs/9781782166320
3. Leka, M. (2020, August 12). *Exploring the JavaScript Ecosystem: Popular Tools, Frameworks, and Libraries.* Medium. Retrieved October 1, 2023, from

https://mirzaleka.medium.com/exploring-javascript-ecosystem-popular-tools-frameworks-libraries-7901703ec88f

4. Panchal, M. (2022, May 4). *What Is ExpressJS In Node JS? – Backend Framework For Web Apps.* Excellent Webworld. Retrieved October 1, 2023, from https://www.excellentwebworld.com/what-is-expressjs-in-node-js/
5. MDN (Mozilla Developer Network). (2023, July 3). *What is a web server?*. Mozilla Developer. Retrieved October 1, 2023, from https://developer.mozilla.org/en-US/docs/Learn/Common_questions/Web_mechanics/What_is_a_web_server
6. OpenJS Foundation. (n.d.). *Express routing* (StrongLoop/IBM, Trans.). Expressjs. Retrieved October 1, 2023, from https://expressjs.com/en/guide/routing.html