2022

PBW

Modul 5 – NodeJS

Raymond Chandra Putra, S.T.,M.T.
IF UNPAR
4/26/2022

PETUNJUK PENGERJAAN

1. JANGAN MENGUBAH NAMA FILE!
2. Jawablah pertanyaan-pertanyaan (berwarna oranye) yang ada pada modul. Simpan jawaban Anda dalam file M05xxyyy.txt.
3. Upload folder (di zip) dan file txt Anda ke IDE sesuai jam pengumpulan yang tertera.

NodeJS

Pada modul ini kita akan berpindah dari penggunaan javascript di sisi klien ke sisi server. Javascript mulai dapat digunakan sebagai Bahasa pemrograman pada sisi server sejak 2009 sejak kemunculan teknologi NodeJS. Pastikan NodeJS yang dibutuhkan sudah terinstal. Pada kesempatan ini kita akan mencoba membuat aplikasi web server sederhana dengan menggunakan NodeJS. Untuk pengerjaan ini silahkan baca ikuti instruksi berikut dengan baik.

Bagian I Penyiapan Workspace

Pada bagian ini dilakukan proses penyiapan workspace dengan instruksi sebagai berikut.

1. Siapkan folder pengerjaan kosong dengan nama M05xxyyy.
2. Di dalam folder ini bukalah command line / terminal dan masukan perintah “npm init”. Selanjutnya ada beberapa hal yang perlu diisikan yaitu:

Package name	M05
Description	Modul NodeJS
Author	Nama dan NPM

   Jika sudah berhasil, maka akan muncul file bernama package.json yang berisi data yang sebelumnya dimasukkan.
3. Selanjutnya kita akan mencoba menambahkan suatu library bernama nodemon yang berfungsi mendeteksi perubahan kode secara otomatis sehingga kita tidak perlu mematikan dan menyalakan server nodeJS setiap kali ada perubahan.
   Untuk itu ketikan perintah “npm install nodemon --save-dev”. Jika berhasil, maka akan muncul folder node_modules yang berisi nodemon dan semua dependencies yang dibutuhkan juga akan muncul file package-lock.json.
   a. Mengapa untuk library seperti nodemon dimasukkan sebagai devDependencies bukan dependencies?
4. Buka file package.json. Pada file ini cari bagian scripts yang terdapat script test. Tambahkan script kedua dengan nama “start” yang berisi “nodemon index.js”.
5. Pada workspace tambahkan file dengan nama index.js. Lalu pada terminal jalankan perintah “npm run start”.
   a. Perintah apa yang dimasukkan untuk memanggil scripts test di package.json?

Bagian II Web Server

Pada bagian ini kita akan membuat web server sederhana pada file **index.js**. Ikutilah petunjuk berikut.

1. Sebagai permulaan, ketikkan kode berikut yang memanfaatkan HTTP API:

```
const http = require('http');

const server = http.createServer((request, response) => {
    response.statusCode = 200;
    response.setHeader('Content-Type', 'text/plain');
    response.end("Hello World");
});

server.listen(8080, () => {
  console.log('Ready!');
});
```

Gambar 1 – File index.js

2. Buka browser anda dan lakukan request ke web server yang anda buat.
    a. URL apakah yang anda gunakan untuk memunculkan teks Hello World?
    b. Port berapa yang digunakan untuk web server ini?
    c. Javascript module apa yang digunakan pada kode ini?
3. Ubahlah kode ini agar menggunakan ES6Module.
    a. Perubahan apa saja yang perlu dilakukan?
4. Kita akan memisahkan kode createServer menjadi 2 bagian sebagai berikut yang selanjutnya kita beri nama router.

```
const router = (request, response) => {
    response.statusCode = 200;
    response.setHeader('Content-Type', 'text/plain');
    response.end("Hello World");
}
const server = http.createServer(router);
```

Gambar 2 – Fungsi router

5. Selanjutnya agar lebih rapih lagi, maka pindahkan fungsi router tersebut ke file lain yaitu **router.js**. Lalu fungsi router ini diubah menjadi module. Dengan dipindahkan fungsi router tersebut, maka pada file **index.js** tersisa kode sebagai berikut.

```
import http from 'http';
import { router } from './router.js'

const server = http.createServer(router);

server.listen(8080, () => {
  console.log('Ready!');
});
```

Gambar 3 – File index.js

Bagian III File System API

Pada bagian ini kita akan menggunakan API lain yaitu File System. Pada zip soal, terdapat folder **public** yang berisi halaman web sederhana. Masukkan folder ini ke dalam workspace untuk menghemat waktu.

1. Tujuan dari bagian ini adalah, jika kita halaman ini dibuka maka akan diarahkan ke halaman index.html dan jika diklik link About maka akan server akan menyediakan halaman about.html. Jika diakses halaman lain maka akan keluar tulisan “Not Found”.
2. Untuk tujuan itu NodeJS harus mampu mengakses file html yang sudah dibuat dengan menggunakan File System API.
3. Bukalah file **router.js** lalu tambahkan kode berikut di dalam fungsi router.

```
console.log("URL: " + request.url);
console.log("Method: "+ request.method);
```

Gambar 4 – Variable request

Lalu jalankan halaman tersebut. Lakukan eksplorasi dengan mengubah2 url pada browser.

   a. Berdasarkan pengamatan anda, apa fungsi variable request ini?

4. Dengan mengetahui kegunaan variable request ini, hapus implementasi fungsi router dan tambahkan 3 percabangan:
   a. Jika url “/” dan method GET: tampilkan teks “index”.
   b. Jika url “about.html” dan method GET: tampilkan teks “about”.
   c. Lainnya: tampilkan teks “Not Found” dan ubah statusCode menjadi 404.
5. Tambahkan import untuk File System API yang berada di module “fs” di file **router.js**. Terdapat beberapa implementasi untuk API ini. Pertama kita coba untuk yang synchronous.

```
import fs from 'fs';
```

6. Lalu ubahlah kode pada percabangan pertama di instruksi nomor 4 dengan kode sebagai berikut.

```
const content = fs.readFileSync('./public/index.html', 'utf8');
response.statusCode = 200;
response.setHeader('Content-Type', 'text/html');
response.end(content);
```

Gambar 5 – Fungsi fs synchronous.

Lakukan juga penyesuaian untuk halaman about. Dan bukalah halaman ini pada browser.

   a. Pada halaman about terdapat kode javascript. Jelaskan perbedaan eksekusi kode ini dibandingkan dengan index.js dan router.js.
   b. (Challenge opsional) Apa yang harus dilakukan jika kode javascript di about.html diakses secara external dengan membuat file script.js

7. Selanjutnya, kita akan mencoba mengubah implementasi API tersebut dengan menggunakan asynchronous callback sebagai berikut.

```
fs.readFile('./public/index.html', 'utf8', (err, content) => {
    response.statusCode = 200;
    response.setHeader('Content-Type', 'text/html');
    response.end(content);
});
```

Gambar 6 – Fungsi fs menggunakan callback

Lakukan juga perubahan untuk halaman about. Lalu coba kembali pada browser.

   a. Jelaskan permasalah penggunaan fungsi synchronous terutama ketika mengakses file dibandingkan proses asynchronous.

8. Terakhir, API ini juga menyediakan versi promise yang **lebih disarankan** untuk di masa yang akan datang. Akan tetapi, module ini tersedia di tempat lain sehingga pada bagian import harus diubah menjadi 'fs/promises'.

```
fs.readFile('./public/index.html', 'utf8')
    .then( (content) => {
        response.statusCode = 200;
        response.setHeader('Content-Type', 'text/html');
        response.end(content);
    });
```

Gambar 7 – Fungsi fs menggunakan promise.

Lakukan juga perubahan untuk halaman about. Lalu coba kembali pada browser.

9. **Challenge wajib**: Ubahlah implementasi promise pada nomor 8 dengan menggunakan format async/await.

Folder workspace ini dikumpulkan menjadi format zip dan diberi nama M05xxyyy.zip.

= **SELAMAT MENGERJAKAN** =