



Modul 06

JSON HTTP Endpoints

TUJUAN PEMBELAJARAN

1. Mampu memahami dan menjelaskan tentang JSON HTTP endpoints menggunakan Expressjs

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. JSON HTTP Endpoints

JSON HTTP Endpoints adalah titik akhir atau URL pada sebuah aplikasi web yang digunakan untuk mengirim atau menerima data dalam format JSON melalui protokol HTTP (Hypertext Transfer Protocol). JSON, singkatan dari JavaScript Object Notation, adalah format data ringan yang banyak digunakan dalam pertukaran data antara aplikasi web dan layanan web. Format JSON sangat populer dalam pengembangan web karena kemudahannya dibaca oleh manusia dan diurai oleh komputer. Format ini bersifat serbaguna dan dapat digunakan untuk mengirim data struktural seperti objek, array, string, angka, dan lainnya. JSON digunakan secara luas dalam pengembangan aplikasi web modern untuk mengirim dan menerima data secara efisien.

JSON HTTP Endpoints digunakan dalam pengembangan web untuk berbagai tujuan, termasuk:

1. Mengirim Data: JSON HTTP Endpoints digunakan untuk mengirim data dari klien (biasanya browser) ke server. Misalnya, ketika seorang pengguna mengisi formulir di

halaman web, data tersebut dapat dikirim ke server dalam format JSON melalui HTTP untuk diproses.

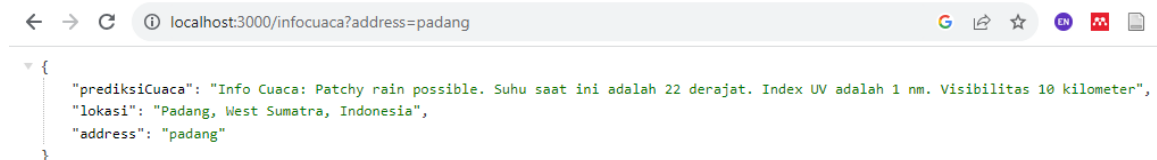
2. Menerima Data: Server juga dapat menyediakan JSON HTTP Endpoints yang memungkinkan klien untuk mengambil data dalam format JSON. Misalnya, sebuah aplikasi dapat memiliki API (*Application Programming Interface*) yang mengembalikan data dalam format JSON ketika diminta oleh klien.
3. Interaksi dengan Layanan Web: JSON HTTP Endpoints dapat digunakan untuk berkomunikasi antara berbagai layanan web. Misalnya, ketika sebuah layanan web diintegrasikan dengan layanan pihak ketiga seperti Twitter atau Facebook, pengembang mungkin akan berinteraksi dengan JSON HTTP Endpoints yang disediakan oleh layanan tersebut untuk mengambil atau mengirim data.

Sebagai contoh pada url mapbox API berikut menggabungkan berbagai informasi berikut:

`https://api.mapbox.com/geocoding/v5/mapbox.places/padang.json?access_token=KODE_TOKEN_DISINI`

- Protokol yang digunakan: **HTTPS**
- Lokasi jaringan dari server API: **api.mapbox.com/geocoding/v5/mapbox.places/**
- Titik akhir URL Target: **padang.json?access_token=KODE_TOKEN_DISINI**

Pada praktikum ini anda akan membuat HTTP Endpoints dalam format JSON menggunakan aplikasi web-server pada modul sebelumnya. Dimana ketika anda mengakses halaman localhost melalui browser misalnya <http://localhost:3000/infocuaca?address=padang> maka akan menampilkan file dalam format JSON seperti pada gambar berikut.



Gambar 1. Tampilan localhost JSON HTTP Endpoints

Pada URL diatas terlihat bahwa URL tersebut merupakan kombinasi antara protokol **HTTP** yang digunakan, lokasi jaringan dari server web (termasuk port TCP) yaitu **localhost:3000** dan URL target yaitu **infocuaca?address=padang**

B. Perbedaan HTTP Endpoints dan API Endpoints

JSON HTTP Endpoints seringkali merupakan bagian dari API (Application Programming Interface) Endpoints, tetapi keduanya tidak selalu sama. Berikut adalah penjelasan untuk memahami perbedaannya

1. JSON HTTP Endpoints: adalah titik akhir atau URL pada sebuah aplikasi web yang digunakan untuk mengirim atau menerima data dalam format JSON melalui protokol HTTP. JSON HTTP Endpoints biasanya merujuk pada cara data dikirimkan atau diterima melalui HTTP dalam format JSON. Hal ini adalah bagian penting dari API, tetapi tidak mengatur semua aspek API.

2. API Endpoints: adalah serangkaian titik akhir atau URL yang digunakan untuk berkomunikasi dengan aplikasi atau layanan. Hal ini dapat menggunakan berbagai format data, termasuk JSON, XML, atau bahkan teks biasa. API Endpoints mengatur cara aplikasi atau layanan berinteraksi dengan satu sama lain atau dengan pengguna. API Endpoint mungkin dapat mengirim atau menerima data dalam format JSON, tetapi mereka juga dapat bekerja dengan format data lainnya.

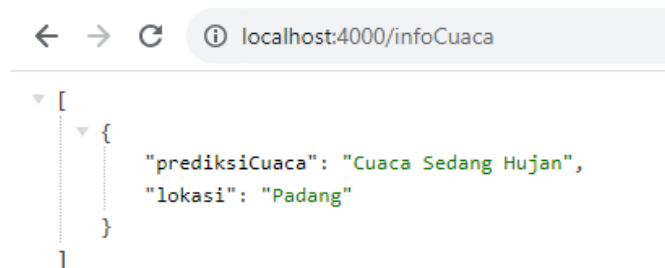
Secara ringkas JSON HTTP Endpoints adalah salah satu cara untuk mengimplementasikan komunikasi antara aplikasi atau layanan melalui API, dengan fokus pada penggunaan format data JSON dan protokol HTTP. Namun, dalam konteks API yang lebih luas, Anda dapat memiliki API Endpoints yang menggunakan format data yang berbeda, seperti XML, dan bahkan dapat menggunakan protokol selain HTTP, seperti WebSocket.

LATIHAN

A. Membuat JSON HTTP Endpoints

a. Req Query

1. Bukalah aplikasi web-server anda pada visual studio code. Pastikan anda berada pada direktori **src**, lalu jalankan aplikasi menggunakan perintah **nodemon app.js -e js,hbs**
2. Bukalah URL <http://localhost:4000/infoCuaca> pada web browser anda perhatikan bahwa ini akan menampilkan file JSON dengan format statis sebagaimana yang ada pada baris kode **app.get('/infoCuaca'.....)**



Gambar 2. Tampilan localhost pada URL infoCuaca

3. Pada tahap selanjutnya kita akan menggunakan **req.query** sebagai salah satu objek permintaan (*request*) dalam expressjs. **req.query** diisi dengan string kueri *request* yang ditemukan dalam URL. String kueri ini berbentuk pasangan kunci-nilai (*key-value*) dan dimulai setelah tanda tanya dalam URL. Jika terdapat lebih dari satu string kueri, mereka dipisahkan dengan tanda "ampersand." Contoh dapat dilihat di bawah ini.

`https://proska.com/user?nama=Randi&isAuthor=Proska`

Dari kode di atas, string kueri yang digunakan adalah "nama" dan "isAuthor". Ketika permintaan ini dilakukan, objek `req.query` akan diisi dengan string kueri tersebut.

```
req.query = {name: "Theodore", isAuthor: true}
```

4. Cobalah ganti baris kode pada `app.get('/infocuaca'.....)` dengan kode berikut ini

```
app.get('/infocuaca', (req, res) => {  
  if (!req.query.address) {  
    return res.send({  
      error: 'Kamu harus memasukan lokasi yang ingin dicari'  
    })  
  }  
  
  res.send({  
    prediksiCuaca: 'Cuaca Sedang Hujan',  
    lokasi: 'Padang',  
    address: req.query.address  
  })  
})
```

5. Cobalah akses di browser dengan mengetikan url berikut <http://localhost:4000/infoCuaca?address=padang>.
6. Pahami apa yang ditampilkan pada langkah 5. Lalu cobalah mengakses <http://localhost:4000/infoCuaca>. Anda akan melihat bahwa aplikasi menampilkan pesan untuk memasukan lokasi yang ingin dicari. Hal ini terjadi karena anda tidak memasukan kueri **address**

b. Integrasi API Weatherstack dan Mapbox kedalam JSON HTTP Endpoints

1. **Hentikan** nodemon anda, lalu lakukan instalasi **postman-request** dengan perintah **npm** sebagaimana yang pernah anda lakukan pada **modul 3**. Berikut adalah halamannya <https://www.npmjs.com/package/postman-request>
2. Setelah instalasi selesai, jalankan kembali aplikasi dengan perintah **nodemon**
3. Selanjutnya, buatlah folder baru didalam folder **src** dengan nama **utils**.
4. Lalu buatlah **dua** file baru dalam folder tersebut dengan nama **geocode.js** dan **prediksiCuaca.js**
5. Masukan kode berikut pada file **geocode.js**

```
const request = require('postman-request')  
  
const geocode = (address, callback) => {  
  const url = 'https://api.mapbox.com/geocoding/v5/mapbox.places/' +  
    encodeURIComponent(address) + '.json?access_token=API_KEY_DISINI'  
  
  request({ url: url, json: true }, (error, response) => {  
    if (error){  
      callback('Tidak dapat terkoneksi ke layanan', undefined)  
    } else if (response.body.features.length === 0){
```

```
        callback('Tidak dapat menemukan lokasi. Lakukan
pencarian lokasi yang lain', undefined)
    } else {
        callback(undefined, {
            latitude : response.body.features[0].center[1],
            longitude : response.body.features[0].center[0],
            location : response.body.features[0].place_name
        })
    }
})
}

module.exports = geocode
```

PENTING! Jika anda menggunakan API **positionstack**, maka silakan menyesuaikan **url API** nya dan juga seluruh bagian kode yang mengandung **response.body**. Silakan akses kembali file API positionstack melalui browser anda untuk melihat data yang ingin anda akses.

6. Lalu masukan kode berikut pada file **prediksiCuaca.js**

```
const request = require('postman-request')

const forecast = (latitude, longitude, callback) => {
    const url =
    'http://api.weatherstack.com/current?access_key=API_KEY_DISINI&quer
y='+encodeURIComponent(latitude)+'&'+encodeURIComponent(longitude)+'
&units=m';
    request({ url: url, json: true }, (error, response) => {
        if(error) {
            callback('Tidak dapat terkoneksi ke layanan',
undefined)
        } else if(response.body.error) {
            callback('Tidak dapat menemukan lokasi',undefined)
        } else {
            callback(undefined,
                'Info Cuaca: ' +
response.body.current.weather_descriptions[0] + '. ' +
                'Suhu saat ini adalah ' +
response.body.current.temperature + ' derajat. ' +
                'Index UV adalah ' +
response.body.current.uv_index + ' nm. ' +
                'Visibilitas ' + response.body.current.visibility +
                ' kilometer'
            )
        }
    })
}

module.exports = forecast
```

7. Selanjutnya, ubahlah baris kode `app.get('/infocuaca'.....)` pada file `app.js` yang ada di folder `src` dengan kode berikut ini

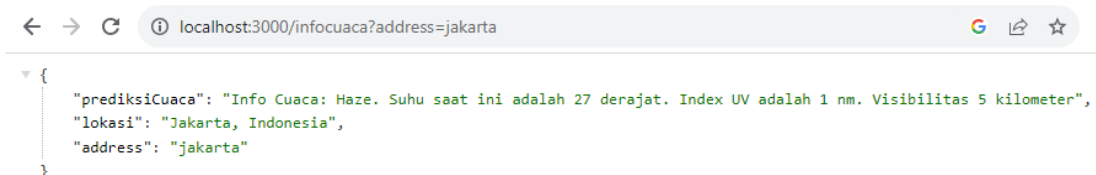
```
app.get('/infocuaca', (req, res) => {  
  if(!req.query.address){  
    return res.send({  
      error:'Kamu harus memasukan lokasi yang ingin dicari'  
    })  
  }  
  geocode(req.query.address, (error, { latitude, longitude,  
location } = {})) => {  
    if (error){  
      return res.send({error})  
    }  
    forecast(latitude, longitude, (error, dataPrediksi) => {  
      if (error){  
        return res.send({error})  
      }  
      res.send({  
        prediksiCuaca: dataPrediksi,  
        lokasi: location,  
        address: req.query.address  
      })  
    })  
  })  
})
```

8. Perlu diperhatikan juga bahwa anda harus meng-*import* file `geocode` dan `prediksiCuaca` dengan menambahkan baris kode berikut pada awal kode `app.js`

```
4 const geocode = require('./utils/geocode')  
5 const forecast = require('./utils/prediksiCuaca')
```

Gambar 3. Variabel `geocode` dan `forecast` untuk import file `geocode` dan `prediksiCuaca`

9. Cobalah mengakses url <http://localhost:3000/infocuaca?address=jakarta> berikut dibrowser anda. Cobalah mengganti kata 'jakarta' dengan keyword lainnya. Perhatikanlah bahwa objek `prediksiCuaca`, `lokasi`, dan `address` sekarang bersifat dinamis, bukanlah lagi statis



```
{  
  "prediksiCuaca": "Info Cuaca: Haze. Suhu saat ini adalah 27 derajat. Index UV adalah 1 nm. Visibilitas 5 kilometer",  
  "lokasi": "Jakarta, Indonesia",  
  "address": "jakarta"  
}
```

Gambar 4. Akses JSON HTTP Endpoints yang bersifat dinamis

B. Menampilkan data API menggunakan Method Fetch

Pada bagian ini, anda akan membuat form pencarian lokasi agar anda dapat mengecek cuaca berdasarkan lokasi yang dicari. Anda akan menggunakan method **fetch**. Di Node.js, `fetch` bukanlah sebuah method bawaan (*built-in method*), tetapi biasanya digunakan dalam

lingkungan browser, bukan pada *runtime* Node.js. **fetch** adalah API JavaScript yang digunakan untuk mengambil (*fetch*) data dari sumber eksternal seperti server web atau API REST. API ini sangat umum digunakan di lingkungan browser untuk membuat permintaan HTTP.

1. Bukalah file **public/js/app.js**, kemudian masukanlah kode berikut ini

```
const weatherform = document.querySelector('form')
const search = document.querySelector('input')
const pesanSatu = document.querySelector('#pesan-1')
const pesanDua = document.querySelector('#pesan-2')

// pesanSatu.textContent = 'From javascript'

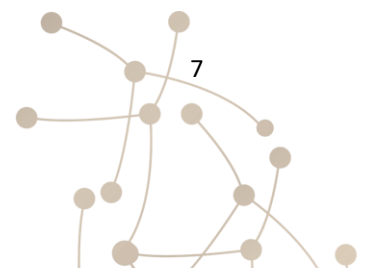
weatherform.addEventListener('submit', (e) => {
  e.preventDefault()
  const location = search.value

  pesanSatu.textContent = 'Sedang mencari lokasi ..'
  pesanDua.textContent = ''

  fetch('http://localhost:3000/infocuaca?address='+
location).then((response)=>{
  response.json().then((data)=>{
    if(data.error){
      pesanSatu.textContent = data.error
    } else {
      pesanSatu.textContent = data.lokasi
      pesanDua.textContent = data.prediksiCuaca
    }
  })
})
})
```

Catatan:

- a. **querySelector** diatas adalah sebuah metode yang digunakan di lingkungan browser pada JavaScript untuk mengakses elemen-elemen HTML di halaman web dengan menggunakan selector CSS.
 - b. Perintah **fetch(...)** diatas digunakan untuk mengambil data dari JSON HTTP Endpoints yang telah kita buat sebelumnya. Jadi, pada dasarnya kita akan menggunakan data API yang telah kita buat sendiri untuk kemudian kita tampilkan pengguna
2. Lalu ubahlah baris kode pada bagian **<body>** pada file **index.hbs** dengan kode berikut ini



```
<body>
  <div class="main-content">
    {{>header}}
    <p>Gunakan website ini untuk menemukan informasi cuaca!</p>
    <form>
      <input placeholder="Masukan lokasi">
      <button>Cari Lokasi</button>
    </form>
    <p id="pesan-1"></p>
    <p id="pesan-2"></p>
  </div>
  {{>footer}}
  <script src="/js/app.js"></script>
</body>
```

Gambar 5. Kode form pencarian pada file index.hbs

PENTING!

- Perhatikan bahwa javascript **/js/app.js** diletakan pada bagian akhir, bukan lagi berada pada bagian **<head>** seperti sebelumnya. Hal ini bertujuan agar file tersebut di eksekusi setelah proses pencarian pada form dilakukan
 - Hapuslah baris script ini dari bagian **<head>** di semua file .hbs diantaranya **bantuan.hbs**, **tentang.hbs** dan **404.hbs**. Hal ini dilakukan karena file ini hanya dibutuhkan pada **index.hbs** saja.
3. Tambahkan lah kode berikut ini pada file **styles.css**

```
55 input{
56   border: 2px solid #cccccc;
57   padding: 8px;
58 }
59
60 button {
61   cursor: pointer;
62   border: 0px solid #cccccc;
63   background: #888888;
64   color: white;
65   padding: 9px;
66 }
```

Gambar 6. Kode style untuk form input dan button

4. Silakan akses aplikasi cek cuaca anda pada browser. Lakukan pencarian dengan mengetikan lokasi atau tempat yang ingin dicari, misalnya: **'Jakarta'** atau **'Universitas Negeri Padang'**. Pastikan bahwa tampilannya adalah seperti berikut ini

Aplikasi Cek Cuaca

[Home](#) [Tentang Saya](#) [Bantuan](#)

Gunakan website ini untuk menemukan informasi cuaca!

Universitas Negeri Padang

Cari Lokasi

State University Of Padang, Jalan hamka, Padang, West Sumatra 25132, Indonesia

Info Cuaca: Patchy rain possible. Suhu saat ini adalah 21 derajat. Index UV adalah 1 nm.
Visibilitas 10 kilometer

Dibuat oleh Randi Proska

Gambar 7. Tampilan akhir aplikasi cek cuaca apabila pencarian ditemukan

5. Coba juga untuk mengetik tanda ? pada kotak pencarian dan pastikan bahwa yang tampil adalah seperti gambar berikut ini

Gunakan website ini untuk menemukan informasi cuaca!

Cari Lokasi

Tidak dapat menemukan lokasi. Lakukan pencarian lokasi yang lain

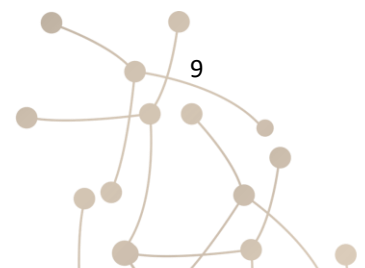
Gambar 8. Tampilan akhir aplikasi cek cuaca apabila pencarian tidak ditemukan

TUGAS

1. Carilah template HTML yang menarik di google ataupun platform lainnya, kemudian gantilah tampilan pada halaman **tentang saya**. Anda dapat mengubah tampilan ini dengan memodifikasi file **styles.css** (jika diperlukan) dan yang terutama modifikasi file **tentang.hbs**. Sesuaikan tampilan ini dengan keinginan anda. Tujuan halaman ini adalah menampilkan informasi tentang aplikasi dan pengembang aplikasi
2. Ubah juga tampilan pada halaman **bantuan** dengan memodifikasi file **bantuan.hbs**. Tujuan halaman ini adalah sebagai **FAQ (Frequently Asked Questions)** yang berisi list pertanyaan tentang aplikasi yang kita kembangkan

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>