Method/Verb Request

dicoding.com/academies/261/tutorials/14677

Web server yang sudah kita buat pada latihan sebelumnya sudah berhasil merespons dan menampilkan data dalam dokumen HTML. Namun, tahukah Anda bahwa web server yang kita buat belum mengenali sepenuhnya permintaan yang diberikan oleh client?

Maksudnya, meskipun client meminta dengan path atau method yang berbeda, server akan merespons dengan data yang sama. Server kita saat ini tidak peduli permintaan datang seperti apa, dia akan mengembalikan data yang sama. Anda bisa mencobanya sendiri melalui cURL dengan menggunakan HTTP method yang berbeda.

```
curl -X POST http://localhost:5000
// output: <h1> Halo HTTP Server!</h1>
curl -X PUT http://localhost:5000
// output: <h1> Halo HTTP Server!</h1>
curl -X DELETE http://localhost:5000
// output: <h1> Halo HTTP Server!</h1>
curl -X GET http://localhost:5000
// output: <h1> Halo HTTP Server!</h1>
```

> Ketika mencobanya pastikan HTTP Server Anda sedang berjalan. Bila dalam keadaan terhenti, jalankan kembali server dengan perintah npm run start pada Terminal proyek nodejs-web-server.

Hal tersebut wajar karena kita memang belum menuliskan logika dalam menangani permintaan dari method yang berbeda. Lalu, bagaimana caranya agar bisa melakukan hal tersebut?

Fungsi request listener menyediakan dua parameter yakni request dan response. Fokus ke parameter request, parameter ini merupakan instance dari http.ClientRequest yang memiliki banyak properti di dalamnya.

Melalui properti-propertinya ini kita dapat mengetahui seperti apa karakteristik dari permintaan HTTP yang dilakukan oleh client. Seperti method yang digunakan, path atau alamat yang dituju, data yang dikirimkan (bila ada), dan informasi lain mengenai permintaan tersebut.

Untuk mendapatkan nilai method dari request, gunakanlah properti request.method seperti ini:

```
const requestListener = (request, response) => {
    const method = request.method;
};
```

Atau, Anda bisa menggunakan cara yang lebih *clean* dengan menggunakan *object destructuring* seperti ini:

```
1. const requestListener = (request, response) => {
2.     const { method } = request;
3. };
```

Properti method bernilai tipe method dalam bentuk string. Nilainya dapat berupa “GET”, “POST”, “PUT”, atau method lainnya sesuai dengan yang client gunakan ketika melakukan permintaan. Dengan memiliki nilai method, kita bisa memberikan respons berbeda berdasarkan tipe method-nya.

```
1. const requestListener = (request, response) => {
2.     const { method } = request;
3.     if(method === 'GET') {
4.         // response ketika GET
5.     }
6.     if(method === 'POST') {
7.         // response ketika POST
8.     }
9.     // Anda bisa mengevaluasi tipe method lainnya
10. };
```

Sekali lagi, tidak hanya properti method, objek request kaya akan properti dan fungsi lain di dalamnya. Anda dapat mengeksplorasi properti atau fungsi lainnya pada halaman dokumentasi Node.js tentang HTTP Client Request.

Latihan Handling Request

Sekarang Anda sudah tahu kan bagaimana cara mendapatkan nilai method dari permintaan client? Yuk terapkan pada web server yang sudah kita buat.

Pada latihan ini, kita akan melakukan konfigurasi agar server mengembalikan respons dengan kata “hello” dalam berbagai bahasa sesuai method yang digunakan client.

Silakan buka kembali berkas server.js pada proyek **nodejs-web-server**. Lalu, sesuaikan respons berdasarkan request method yang dilakukan oleh client.

```
const http = require('http');
const requestListener = (request, response) => {
    response.setHeader('Content-Type', 'text/html');
    response.statusCode = 200;
    const { method } = request;
    if(method === 'GET') {
        response.end('<h1>Hello!</h1>');
    }
    if(method === 'POST') {
        response.end('<h1>Hai!</h1>');
    }
    if(method === 'PUT') {
        response.end('<h1>Bonjour!</h1>');
    }
    if(method === 'DELETE') {
        response.end('<h1>Salam!</h1>');
    }
};
const server = http.createServer(requestListener);
const port = 5000;
const host = 'localhost';
server.listen(port, host, () => {
    console.log(`Server berjalan pada http://${host}:${port}`);
});
```

Simpan perubahan pada berkas **server.js**; jalankan ulang server dengan perintah npm run start; dan coba lakukan permintaan ke server dengan menggunakan method yang berbeda melalui cURL.

Hasilnya akan tampak seperti ini:

```
curl -X GET http://localhost:5000
// output: <h1>Hello!</h1>
curl -X POST http://localhost:5000
// output: <h1>Hai!</h1>
curl -X PUT http://localhost:5000
// output: <h1>Bonjour!</h1>
curl -X DELETE http://localhost:5000
// output: <h1>Salam!</h1>
```

ini konten buat note nanti