# Body Request | Belajar Membuat Aplikasi Back-End untu..



## Body Request

Ketika client melakukan permintaan dengan method POST atau PUT, biasanya permintaan tersebut memiliki sebuah data yang disimpan pada body request. Data pada body bisa berupa format teks, JSON, berkas gambar, atau format lainnya. Data tersebut nantinya digunakan oleh server untuk diproses di database atau disimpan dalam bentuk objek utuh.

Ketahuilah bahwa http.clientRequest merupakan turunan dari readable stream, yang di mana untuk mendapatkan data body akan sedikit sulit dibandingkan dengan mendapatkan data header. Data di body didapatkan dengan teknik stream, seperti yang sudah kita ketahui, teknik ini memanfaatkan EventEmitter untuk mengirimkan bagian-bagian datanya. Dalam kasus http.clientRequest event data dan end-lah yang digunakan untuk mendapatkan data body.

Berikut adalah contoh bagaimana mendapatkan data body:

```
1. const requestListener = (request, response) => {
2.     let body = [];
3.     request.on('data', (chunk) => {
4.         body.push(chunk);
5.     });
6.     request.on('end', () => {
7.         body = Buffer.concat(body).toString();
8.     });
9. };
```

Mari kita bedah kodenya.

- Pertama, kita deklarasikan variabel body dan inisialisasikan nilainya dengan array kosong. Ini berfungsi untuk menampung buffer pada stream.
- Lalu, ketika event data terjadi pada request, kita isi array body dengan chunk (potongan data) yang dibawa callback function pada event tersebut.
- Terakhir, ketika proses stream berakhir, maka event end akan terbangkitkan. Di sinilah kita mengubah variabel body yang sebelumnya menampung buffer menjadi data sebenarnya dalam bentuk string melalui perintah Buffer.concat(body).toString().

*Huft!* Cukup melelahkan yah untuk mendapatkan data melalui teknik stream. Guna memantapkan pemahaman, mari kita praktikan pada proyek web server sebelumnya.

## Latihan Mendapatkan Body Request

Di latihan sebelumnya, web server yang kita buat sudah berhasil merespons sesuai request method yang dilakukan client. *That's nice!*

Nah, di latihan kali ini kita akan coba mendapatkan data pada body request ketika client mengirimkan request menggunakan method POST.

Buatlah web server merespons permintaan method POST dengan menampilkan sapaan dan nama berdasarkan data body yang dikirim oleh client. Bila client mengirimkan nama "Dicoding", maka respons akan menampilkan "Hai, Dicoding!".

Client akan mengirimkan data nama tersebut menggunakan format JSON seperti ini:

```
1. { "name": "Dicoding" }
```

Namun sebelum itu, agar latihan lebih fokus terhadap bagaimana mendapatkan data pada body, kita hapus dulu logika method yang sebenarnya belum kita butuhkan, seperti PUT dan DELETE.

Jadi, silakan buka berkas server.js pada proyek **nodejs-web-server** dan hapuslah logika method PUT dan DELETE. Sehingga, berkas server.js tampak lebih ringkas seperti ini:

```
 1. const http = require('http');
 2. const requestListener = (request, response) => {
 3.     response.setHeader('Content-Type', 'text/html');
 4.     response.statusCode = 200;
 5.     const { method } = request;
 6.     if(method === 'GET') {
 7.         response.end('<h1>Hello!</h1>');
 8.     }
 9.     if(method === 'POST') {
10.         response.end('<h1>Hai!</h1>');
11.     }
12. };
13. const server = http.createServer(requestListener);
14. const port = 5000;
15. const host = 'localhost';
16. server.listen(port, host, () => {
17.     console.log(`Server berjalan pada http://${host}:${port}`);
18. });
```

Selanjutnya, kita bisa mulai menuliskan logika stream di dalam blok POST.

```
if(method === 'POST') {
  let body = [];
  request.on('data', (chunk) => {
      body.push(chunk);
  });
  request.on('end', () => {
      body = Buffer.concat(body).toString();
      response.end(`<h1>Hai, ${body}!</h1>`);
  });
}
```

Perhatikan kode di atas! Kita memindahkan proses respons di dalam callback event end. Hal ini diperlukan karena data body siap dikonsumsi hanya ketika event end telah dibangkitkan. Dalam arti lain, server tidak akan mengirimkan respons bila proses *stream* belum selesai.

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

```
curl -X POST -H "Content-Type: application/json" http://localhost:5000 -d "{\"name\": \"Dicoding\"}"
```

Server akan menanggapi dengan hasil berikut:

```
<h1>Hai, {"name": "Dicoding"}!</h1>
```

Tunggu, ini bukan hasil yang kita harapkan. Body masih bernilai data string JSON. Data ini masih perlu kita olah lagi agar bisa mendapatkan nilai name yang sebenarnya. Gunakanlah JSON.parse() untuk mengubah JSON string menjadi JavaScript objek. Sesuaikan kembali kode pada blok POST menjadi seperti ini (lihat kode yang ditebalkan):

```
if(method === 'POST') {
  let body = [];
  request.on('data', (chunk) => {
    body.push(chunk);
  });
  request.on('end', () => {
    body = Buffer.concat(body).toString();
    const { name } = JSON.parse(body);
    response.end(`<h1>Hai, ${name}!</h1>`);
  });
}
```

Simpan perubahan pada berkas **server.js**; jalankan ulang server dengan perintah npm run start; dan coba lagi lakukan permintaan ke server dengan menggunakan method POST.

```
curl -X POST -H "Content-Type: application/json" http://localhost:5000 -d "
{\"name\": \"Dicoding\"}"
```

maka output-nya akan:

```
<h1>Hai, Dicoding!</h1>
```

*Voila!* Inilah hasil yang kita harapkan! Anda bisa kirimkan permintaan POST lain dengan data nama Anda sendiri. Cobalah, apakah hasilnya sesuai atau tidak?

```
curl -X POST -H "Content-Type: application/json" http://localhost:5000 -d "
{\"name\": \"Dimas\"}"
// output: <h1>Hai, Dimas!</h1>
```

ini konten buat note nanti