Path Parameter

Mari kita berbicara mengenai teknik routing lebih lanjut. Path dalam routing bisa dikatakan sebagai alamat yang digunakan client untuk melakukan permintaan ke server. Alamat atau path yang dibuat biasanya merupakan teks verbal yang dapat dimengerti oleh client. Tak jarang hanya dengan membaca path dari sebuah tautan kita langsung mengerti apa yang client minta kepada server.

Sebagai contoh, ketika membaca tautan GitHub ini https://github.com/dicodingacademy, Anda pasti mengerti bahwa client ingin meminta server untuk menampilkan profil github dari username “dicodingacademy”.

Contoh lain, dari alamat https://twitter.com/maudyayunda, coba Anda tebak kira-kira apa yang diminta client ke server? Jika Anda berpikir client meminta profil twitter kak Maudy Ayunda, yups, Anda tepat sekali!

Twitter dan Github menggunakan pendekatan yang sama dalam menampilkan halaman profil. Mereka memanfaatkan *username* sebagai bagian dari path untuk melakukan permintaan ke server. Terbayang tidak *sih* bagaimana mereka melakukannya? Di saat mereka memiliki pengguna yang banyak, apakah mereka menetapkan route secara satu per satu berdasarkan username untuk setiap penggunanya? Tentu tidak!

Untuk melakukan hal tersebut, Twitter dan Github menggunakan teknik path parameter. Di Hapi Framework teknik tersebut sangat mudah untuk diterapkan. Cukup dengan membungkus path dengan tanda { }. Sebagai contoh:

```
1.  server.route({
2.      method: 'GET',
3.      path: '/users/{username}',
4.      handler: (request, h) => {
5.          const { username } = request.params;
6.          return `Hello, ${username}!`;
7.      },
8.  });
```

Seperti yang Anda lihat di atas, pada properti `path` terdapat bagian path yang ditulis `{username}`. Itu berarti, server memberikan bagian teks tersebut untuk client manfaatkan sebagai parameter.

Nantinya parameter ini akan disimpan sebagai properti pada `request.params` yang dimiliki handler dengan nama sesuai yang Anda tetapkan (username). Sebagai contoh, bila Anda melakukan permintaan ke server dengan alamat ‘`/users/harry`’, maka server akan menanggapi dengan ‘Hello, harry!’.

Pada contoh kode di atas, nilai path parameter wajib diisi oleh client. Bila client mengabaikannya dengan melakukan permintaan pada alamat ‘`/users`’, maka server akan mengalami eror.

DIBANTU
Privacy - Terms

parameter:

```
1.  server.route({
2.      method: 'GET',
3.      path: '/users/{username?}',
4.      handler: (request, h) => {
5.          const { username = 'stranger' } = request.params;
6.          return `Hello, ${username}!`;
7.      },
8.   });
```

Sekarang bila client meminta pada alamat ‘ /users/dicoding ’, server menanggapi dengan ‘Hello, dicoding!’; dan bila client meminta hanya pada path ‘ /users ’, server akan menanggapinya dengan ‘Hello, stranger!’.

Anda bisa menetapkan lebih dari satu path parameter. Namun, penting untuk Anda ketahui bahwa optional path parameter hanya dapat digunakan di akhir bagian path saja. Artinya, jika Anda menetapkan optional path di tengah-tengah path parameter lain contohnya /{one?}/{two} , maka path ini dianggap tidak valid oleh Hapi.

Latihan Path Parameter

Sekarang Anda sudah tahu apa itu path parameter, saatnya kita coba praktikkan pada web server yang sudah dibuat.

Pada latihan kali ini, kita akan membuat route baru dengan nilai path /hello/{name?} . Bila client melampirkan nilai path parameter, server harus mengembalikan dengan pesan “Hello, name!”. Namun bila tidak, server harus mengembalikan dengan nilai
“Hello, stranger!”. Sudah paham? Yuk kita mulai!

Buka berkas **routes.js** dan tambahkan route baru seperti ini (lihat kode yang ditebalkan).

routes.js

DIBANTU
Privacy - Terms

```
    {
        method: 'GET',
        path: '/',
        handler: (request, h) => {
            return 'Homepage';
        },
    },
    {
        method: '*',
        path: '/',
        handler: (request, h) => {
            return 'Halaman tidak dapat diakses dengan method tersebut';
        },
    },
    {
```

Di dalam handler , dapatkan nilai path parameter melalui properti request.params . Kita manfaatkan object destructuring untuk mendapatkan nilainya. Jangan lupa berikan nilai default “stranger”.

Lalu, kembalikan fungsi handler dengan pesan sesuai dengan ketentuan yah. Sehingga, fungsi handler tampak seperti ini:

```
{
    method: 'GET',
    path: '/hello/{name?}',
    handler: (request, h) => {
        const { name = "stranger" } = request.params;
        return `Hello, ${name}!`;
    },
},
```

Simpan perubahan pada berkas **routes.js**; coba jalankan kembali server dengan perintah npm run start ; dan lakukanlah permintaan melalui curl atau browser pada path **/hello/dicoding** dan **/hello**.

```
curl -X GET http://localhost:5000/hello/dicoding
// output: Hello, dicoding!
curl -X GET http://localhost:5000/hello
// output: Hello, stranger!
```

Good Job! Anda berhasil menerapkan path parameter!

DIBANTU

Sebelumnya Selanjutnya

Privacy - Terms

dicoding

dicoding

Dicoding Space
Jl. Batik Kumeli No.50, Sukaluyu,
Kec. Cibeunying Kaler, Kota Bandung
Jawa Barat 40123

Decode Ideas
Discover Potential

> Tentang Kami

Blog
Reward
Showcase

Hubungi Kami
FAQ

Penghargaan

image
click bila
belum muncul

© 2021 Dicoding | Dicoding adalah merek milik PT Presentologics, perusahaan induk dari PT Dicoding Akademi Indonesia.

Terms • Privacy

DIBANTU

Privacy - Terms