Nama : Muchamad Maulana Irfan Arifandi

Nim : 22.01.53.0020

Matkul : Web Service

1. Get

Sebuah permintaan GET mengambil data dari web server dengan menentukan parameter di bagian URL dari permintaan. Jika Anda memeriksa contoh permintaan HTTP bawah ini, kami minta index.html, dan melewati report\_id parameter.

Contoh saat menggunakan GET:

* Anda mengakses URL murni demi melihat data. Anda bisa menganggapnya sebagai menggunakan pernyataan SELECT SQL. Anda meminta data dari server web tanpa maksud memperbarui data apapun.
* Anda perlu URL untuk menjadi 'bookmarkable'. Pada dasarnya HTTP GET dianggap diulangi, yang memungkinkan permintaan untuk dicoba aman dan tanggapan-cache.
* Anda tidak keberatan permintaan diulang. Misalnya pengguna mengunjungi URL yang sama lebih dari sekali.

Contoh ketika tidak menggunakan GET:

* Anda lewat data sensitif seperti username, password, nomor jaminan sosial, dll
* Anda mengirimkan data dalam jumlah besar. Meskipun tidak ada batas karakter didefinisikan dalam spesifikasi HTTP untuk panjang URL, IE 4 misalnya hanya mendukung URL panjang maksimum ~ 2000 karakter menggunakan permintaan GET.
* Anda perlu memperbarui sesuatu pada server, misalnya mengirimkan formulir yang akan memperbarui alamat pengguna atau keranjang belanja.

1. Post

Sebuah permintaan HTTP POST memanfaatkan badan pesan untuk mengirim data ke server web. Jika Anda memeriksa contoh permintaan HTTP POST di bawah ini, Anda akan melihat bahwa kita mengirimkan permintaan HTTP POST dengan tubuh pesan 'userid = mo & password = mypassw' untuk login.jsp (login.jsp akan menjadi sebuah aplikasi yang ke depan server web permintaan untuk).

Contoh saat menggunakan POST:

* Anda memiliki sejumlah besar data untuk mengirim ke server web (ukuran data akan melebihi batas URL dari metode GET).
* Anda mengirim data sensitif seperti uesrnames, password, nomor jaminan sosial dan lain-lain
* Anda mengubah keadaan data dalam aplikasi web. Misalnya, keranjang belanja melacak item yang Anda beli.

Contoh ketika tidak menggunakan POST:

* URL yang Anda melewati memiliki persyaratan menjadi 'bookmarkable'. Jika keadaan perubahan URL, maka pengguna tidak akan dapat mengambil, atau melihat data itu itu adalah mantan negara.
* Permintaan Anda perlu idempotent. Perhatikan bahwa permintaan POST bisa idempotent, namun itu praktik yang lebih baik untuk menggunakan PUT (jika metode permintaan HTTP ini didukung oleh web server dan client).

1. Put

PUT mirip dengan POST memanfaatkan badan pesan untuk mentransfer data. Namun, ada beberapa perbedaan mendasar antara keduanya. Pertama PUT dianggap idempotent, kedua tindakan seorang PUT ini selalu ditetapkan untuk URI tertentu, akhirnya PUT adalah untuk memuat data untuk sumber daya itu. Dengan kata lain Anda harus tahu lokasi yang tepat dari mana data yang Anda kirimkan akan diambil nanti.

Contoh kapan harus menggunakan PUT:

* Masukan adalah idempotent, jadi pada dasarnya jika Anda perlu untuk mengakomodasi untuk skenario di mana ada permintaan yang disampaikan beberapa kali tapi hasilnya harus sama untuk setiap pengiriman, penggunaan PUT.Ini bisa berguna untuk membuat user baru misalnya. Jika Anda mengirim permintaan PUT untuk membuat user Joe Smith beberapa kali, permintaan terakhir harus memiliki hasil yang sama seolah-olah dikirim pertama.
* Anda memiliki URI khusus yang Anda mengirim data ke. Sebagai contoh:

POST URI:

http://hostname.com/users/new

PUT URI:

<http://hostname.com/users/privacyyou>

Contoh ketika tidak menggunakan PUT:

* PUT tidak boleh digunakan untuk permintaan non idempotent (jika keadaan sumber daya kemungkinan akan berubah setiap kali permintaan dikirim).
* Ada baiknya untuk diingat bahwa dalam kasus bentuk html, kebanyakan browser tidak mendukung PUT / DELETE metode. Diharapkan POST / GET digunakan.Beberapa kerangka Tenang seperti Ruby on Rails misalnya memerlukan penggunaan PUT / DELETE, namun ini Metode HTTP hanya terowongan melalui Metode POST HTTP.

PUT /somedatabase/some\_doc\_id HTTP/1.1

Content-Length: 240

Content-Type: application/json

{

"Subject":"Resume",

"Author":"Mo",

"Body":"Find my resume attached"

}

1. Delet

Metode HTTP DELETE dapat digunakan untuk menghapus sumber daya dari server.Umumnya digunakan dalam dua skenario. Skenario Fist adalah jika Anda mengikuti standar tenang dalam mengembangkan aplikasi web Anda. Kedua hal itu dapat digunakan saat DELETE diaktifkan pada web-server dan Anda ingin mengikuti standar HTTP untuk menghapus sumber daya. Sangat penting untuk dicatat namun yang dapat Anda gunakan HTTP POST untuk menangani tindakan HTTP DELETE juga, keputusan dipersempit ke opsi yang diuraikan di atas.

1. Option

Pilihan ini berguna untuk mencari tahu mana metode HTTP dapat diakses oleh klien.Tergantung pada bagaimana web server Anda mencoba untuk menyambung ke dikonfigurasi, administrator mungkin hanya memiliki POST dan GET metode HTTP diakses. Sementara metode HTTP lain seperti DELETE, TRACE, dll dinonaktifkan.

1. Patch

Dalam konteks HTTP request methods, PATCH adalah salah satu metode yang digunakan untuk melakukan modifikasi sebagian pada sumber daya yang sudah ada di server. Ini berbeda dengan metode PUT, di mana PUT digunakan untuk mengganti seluruh sumber daya, sedangkan PATCH hanya memodifikasi bagian-bagian tertentu dari sumber daya tersebut.

Contoh penggunaan metode PATCH:

Jika Anda memiliki data pengguna seperti berikut:

{

"id": 1,

"nama": "Ali",

"email": "ali@example.com"

}

Misalkan Anda hanya ingin memperbarui alamat email, maka dengan PATCH, Anda hanya perlu mengirimkan data yang berubah, misalnya:

{

"email": "ali.baru@example.com"

}

Dengan demikian, hanya properti email yang diperbarui, sementara data lainnya tetap sama.

Karakteristik utama PATCH:

* Modifikasi sebagian: Hanya bagian yang perlu diubah yang dikirimkan ke server, tidak perlu mengirimkan seluruh data.
* Idempotency: Walaupun PATCH seringkali tidak idempotent, beberapa implementasi bisa menjadikannya idempotent (artinya, melakukan request yang sama berulang kali akan menghasilkan hasil yang sama).
* Efisiensi: Karena hanya bagian yang diubah yang dikirim, metode ini lebih efisien dalam hal penggunaan bandwidth.

1. Head

HTTP Metode HEADgunakan untuk mengambil informasi tentang URL dari web server. Jadi misalnya jika Anda mengirim permintaan HEAD, Anda akan menerima respon dari server web yang berisi informasi yang sama seperti yang Anda lakukan dengan HTTP POST tidak termasuk data tubuh. Berikut adalah contoh :

HEAD /de HTTP/1.1[CRLF]

Host: www.google.com[CRLF]

Connection: close[CRLF]

User-Agent: Web-sniffer/1.0.37 [CRLF]

Accept-Encoding: gzip[CRLF]

Accept-Charset: ISO-8859-1,UTF-8;q=0.7,\*;q=0.7[CRLF]

Cache-Control: no-cache[CRLF]

Accept-Language: de,en;q=0.7,en-us;q=0.3[CRLF]

Referer: http://web-sniffer.net/[CRLF]