Pengenalan Web Service Berbasis REST

Arsitektur REST

- Gaya arsitektur software REST dideskripsikan dalam 6 buah batasan, yaitu :
  1) Uniform interface
  2) Stateless
  3) Cache-able
  4) Client-server
  5) Layered system
  6) Code on demand (optional)

(1) Uniform interface

- Batasan uniform interface mendeskripsikan interface antara client dan server.
- Batasan uniform interface ini menggunakan 4 buah prinsip, di antaranya :
  - Resource based
  - Manipulation of resource through representation
  - Self descriptive message
  - Hypermedia as the engine of application (HATEOAS)

Resource based

- Setiap resource dalam server diidentifikasi menggunakan URI sebagai resource identifier.
- Server nantinya tidak mengirimkan database secara langsung ketika diminta oleh client, tetapi akan mengirimkan resource tersebut dalam format lainnya, semisal menggunakan format JSON atau XML.

Manipulation of resource through representation

- Ketika client mengetahui representasi dari resource tertentu dalam suatu server, client tersebut dapat melakukan modifikasi ataupun penghapusan resource asalkan client tersebut memiliki permission untuk melakukan operasi-operasi tersebut.

Self descriptive message

- Setiap pesan yang dikirimkan memiliki informasi yang cukup tentang bagaimana cara memproses informasi tersebut.
- Misalkan ketika sebuah server mengirimkan data dalam format JSON, maka informasi terkait data yang dikirimkan akan disertakan dalam header yang dikirimkan.

Hypermedia as The Engine of Application State (HATEOAS)

- HATEOAS dapat berarti :

  Jika dibutuhkan, link terhadap suatu resource dapat disertakan pada body / header response.

(2) Stateless

- Pada layanan web berbasis REST, statelessness merupakan kunci utama. Artinya, bahwa state untuk menangani request ada di dalam request itu sendiri / disertakan beserta dengan request yang dikirimkan.
- State bisa berada pada URI, parameter, body, ataupun header.
- Ketika server telah berhasil memproses request, sebagian state akan dikirimkan kembali ke client melalui header, status, ataupun response body.

(3) Cache-able

- Sebagaimana dalam World Wide Web, client dapat melakukan proses cache terhadap response.

(4) Client-server

- Sebagaimana telah disebutkan sebelumnya, uniform interface akan memisahkan bagian client dengan bagian server.
- Bagian client tidak harus berfokus pada data storage, sehingga aplikasi client dapat lebih portabel.
- Sementara itu, bagian server tidak harus berfokus pada user interface ataupun user state, sehingga aplikasi dari sisi server dapat menjadi lebih sederhana dan lebih skalabel.

(5) Layered System

- Client tidak harus langsung terhubung dengan server, tetapi dapat melalui server perantara.
- Server perantara dapat berperan sebagai load balancer ataupun sebagai shared cache.
- Layer dapat juga berperan untuk keamanan sistem.

(6) Code on Demand (Optional)

- Server dapat melakukan extend ataupun kustomisasi dari suatu fungsi yang diminta oleh pihak client dengan mengirimkan logic yang dapat dieksekusi, misalnya Java Applet ataupun javascript.

Beberapa tips jika akan mengembangkan layanan web berbasis REST →

Gunakan verb-verb yang dimiliki oleh HTTP (HTTP Methods) sesuai dengan maknanya

- Setiap client dapat mengirimkan method HTTP kepada server seperti GET, POST, PUT, dan DELETE.
- Dalam hal ini, setiap operasi harus disesuaikan dengan method tersebut, misalkan ketika client mengirimkan permintaan method GET untuk suatu resource tententu, maka ketika permintaan tersebut direspon oleh server, tidak boleh terjadi adanya perubahan terhadap resource data.

Gunakan nama resource yang masuk akal (sensible)

- Menggunakan resource yang masuk akal (sensible), seperti /post/23 dan bukan /api?type=post&id=23, akan meningkatkan kejelasan terhadap apa sebenarnya resource yang dapat diakses oleh client.
- Yang harus diperhatikan disini, nama suatu resource harus berupa kata benda (noun), gunakan HTTP methods sebagai aksi terhadap resource tersebut.

XML dan JSON

- Format yang umum dipakai sebagai respon dari suatu layanan web berbasis REST adalah JSON.
- Meskipun demikian, ada baiknya pada layanan web yang kita kembangkan juga menyediakan format yang lain, semisal XML.
- Nantinya, user dapat memilih format apa yang akan dikirimkan oleh pihak server.

Perhatikan aspek connectedness

- Salah satu prinsip dari REST adalah connectedness (melalui hypermedia links).
- Meskipun layanan web tetap dapat diimplementasikan tanpa memperhatikan aspek connectedness, namun dengan adanya link terhadap suatu resource tertentu, maka hal tersebut akan membuat suatu layanan web menjadi lebih self-destrictive.
- Contohnya adalah ketika server menerima request POST kemudian membuat resource baru, maka server dapat mengirimkan header location yang merupakan URL dari resource yang baru saja terbentuk.

Beberapa Definisi Istilah dalam REST

Idempotence

- Pada layanan web berbasis REST, jika suatu operasi dikatakan idempotence, maka client dapat melakukan operasi tersebut berkali-kali dan client akan mendapatkan hasil yang sama (idem).
- Contoh HTTP methods yang bersifat idempotent adalah PUT, DELETE, GET, HEAD, OPTIONS, dan TRACE.

Safety

- Jika suatu operasi pada layanan web berbasis REST dikatakan safety, maka operasi tersebut tidak akan menghasilkan efek tertentu.
- Artinya, client dapat memanggil operasi-operasi yang bersifat safety tanpa harus cemas akan terjadi perubahan terhadap suatu resource pada server.
- Contoh HTTP methods yang bersifat Safe diantaranya GET, HEAD, OPTIONS, dan TRACE.
- Perhatikan bahwa setiap operasi yang bersifat safe juga bersifat idempotent, tetapi tidak berlaku sebaliknya.

HTTP Verbs (methods)

- Pada bagian uniform interface, telah sedikit disinggung tentang HTTP Verbs yang merupakan operasi terhadap suatu resource.
- Terdapat beberapa HTTP methods/verbs yang sangat umum digunakan, diantaranya POST, GET, PUT, dan DELETE.
- Nantinya POST akan dipetakan menjadi operasi create, GET → read, PUT → update, DELETE → delete (CRUD).

GET

- Metode HTTP GET digunakan untuk mendapatkan / membaca representasi dari suatu resource.
- Jika tidak ada permasalahan ketika request GET dikirimkan kepada server oleh pihak client, maka pihak server akan mengirimkan representasi berupa format JSON ataupun XML serta akan mengirimkan HTTP Status Code dengan nilai 200 (OK).

GET

- Jika terjadi error, maka server akan mengirimkan kode 404 (not found) atau 400 (bad request).
- GET bersifat idempotent dan safe.
- Contoh request get misalkan GET

  http://www.example.com/customer/12345

PUT

- Biasanya HTTP method PUT digunakan untuk melakukan update (edit) dari suatu resource.
- Meskipun demikian, PUT juga dapat digunakan untuk membuat resource baru.
- Jika PUT akan digunakan untuk membuat suatu resource baru pada server, maka ID dari suatu resource harus ditentukan sendiri oleh pihak client.

PUT

- Jika operasi PUT berhasil dilakukan, maka server juga akan mengirimkan HTTP status code 200 (OK) atau 204 (No Content) jika tidak ada data yang dikirimkan dari server ke client.
- Jika operasi PUT digunakan untuk membuat suatu resource, maka kode yang dikirimkan adalah 201 (created), serta body response adalah bersifat opsional untuk dikirimkan.

PUT

- Header Location yang berisi URL juga tidak perlu dikirimkan karena client sudah tahu ID dari resource yang baru saja dibuat.
- PUT tidak bersifat safe karena operasi ini akan mengubah resource, tetapi bersifat idempotent. Artinya jika operasi PUT dipanggil berkali-kali untuk mengupdate suatu resource maka hasil yang didapatkan adalah sama.

POST

- HTTP method POST digunakan untuk membuat suatu resource baru pada server.
- Ketika operasi berhasil dilakukan, maka server akan mengirimkan HTTP Status Code 201 (Created) dengan mengembalikan Location pada bagian header yang isinya merupakan link dari resource yang baru saja dibuat.
- POST bersifat tidak safe dan tidak idempotent.

DELETE

- HTTP method delete berfungsi untuk menghapus suatu resource.
- Jika operasi ini berhasil dilakukan, maka server akan mengirimkan HTTP Status Code 200 (OK) ataupun 204 (No Content) tanpa response body.