REST Web Service

Dalam mengembangkan web service, Anda perlu menetapkan arsitektur apa yang hendak diadaptasi. Dengan menetapkan arsitektur, client dan server lebih mudah dalam berkomunikasi karena memiliki pola atau gaya yang konsisten. Salah satu arsitektur web service yang banyak digunakan saat ini adalah *REST*.

REST atau **RE**presentational **S**tate **T**ransfer adalah salah satu gaya arsitektur yang dapat diadaptasi ketika membangun web service. Arsitektur ini sangat populer digunakan karena pengembangannya yang relatif mudah. REST menggunakan pola request-response dalam berinteraksi, artinya ia memanfaatkan protokol HTTP seperti yang sudah kita pelajari di materi sebelumnya.

Dalam implementasinya arsitektur REST benar-benar memisahkan peran client dan server, bahkan keduanya tidak harus saling mengetahui. Artinya ketika terjadi perubahan besar di sisi client, tidak akan berdampak pada sisi server, begitu juga sebaliknya.

REST API

Sebagian dari kalian mungkin mengenal REST dengan sebutan RESTful API. Yups, memang benar! RESTful merupakan sebutan untuk web services yang menerapkan arsitektur REST. REST juga merupakan API (*application program interface*) karena ia digunakan untuk menjembatani antara sistem yang berbeda (client dan server).

> API atau Application Program Interface merupakan antarmuka yang menjadi perantara antara sistem aplikasi yang berbeda. API tak hanya dalam bentuk Web Service, bisa saja berupa SDK (Software Development Kit) ataupun lainnya.

Berikut beberapa sifat yang menjadi kunci pada REST API.

- **Client-Server** : Ini merupakan hal yang paling mendasar dalam membangun REST API. Server harus bisa merespons permintaan yang dilakukan client, baik itu respons berhasil ataupun gagal. Komunikasi client dan server dilakukan melalui protokol HTTP.
- **Stateless** : REST API tidak boleh menyimpan keadaan (state) apa pun terkait client. Seluruh state harus tetap disimpan di client. Artinya, tidak ada session di REST API. Permintaan yang dilakukan client harus mengandung informasi yang jelas. Jangan berharap RESTful API akan menyimpan informasi dari permintaan sebelumnya untuk digunakan di permintaan selanjutnya.
- **Cacheable** : Agar dapat merespons permintaan dengan cepat, sebaiknya REST API menerapkan prinsip cache. Sehingga setiap permintaan tidak melulu mengambil dari database.
- **Layered** : Ketika REST API server memiliki arsitektur yang kompleks, client seharusnya tidak perlu tahu bagaimana server melayaninya.

Selain itu, sebelum membangun REST API, kita perlu mengenal dahulu bagaimana konsep-konsep penting yang harus diterapkan dalam membangun arsitektur ini. Apa saja?

Singkatnya, ketika membangun REST API kita harus memperhatikan empat poin berikut:

- Format request dan response.
- HTTP Verbs/Methods.
- HTTP Response code.

- URL Design.

Yuk kita bahas lebih detail poin-poin tersebut pada materi selanjutnya.