REST API (Representational State Transfer)

What is REST API?

- REST API adalah salah satau pendekatan arsitektur yang digunakan dalam pengembangan layanan web (web service).
- REST API berfokus pada sumber daya (resources) dan bagaimana sumber daya diakses melalui permintaan HTTP.

Prinsip Utama REST

1. **Resources**: Semua konsep adalah sumber daya, misalnya data, gambar, teks, atau layanan.
2. **Operasi pada Sumber Daya**: Sumber daya dapat diakses, dibaca, diperbarui, atau dihapus menggunakan operasi standar HTTP.
3. **Representasi**: Sumber daya direpresentasikan dalam format tertentu, seperti JSON atau XML.
4. **Stateless**: Setiap permintaan dari klien ke server harus mengandung semua informasi yang diperlukan untuk memahami permintaan tersebut. Server tidak menyimpan status klien antara permintaan.

Contoh Sumber Daya

Dalam REST API, sumber daya dapat berupa apa saja yang dapat diidentifikasi dan didefinisikan, seperti:

- Produk dalam toko online
- Posting blog
- Profil pengguna
- Layanan peramalan cuaca
- Dan banyak lagi...

Operasi pada Sumber Daya

REST API menggunakan operasi standar HTTP seperti:

- `GET` untuk membaca sumber daya.
- `POST` untuk membuat sumber daya baru.
- `PUT` atau `PATCH` untuk memperbarui sumber daya yang ada.
- `DELETE` untuk menghapus sumber daya.

Representasi Sumber Daya, URL, dan Metode

Sumber Daya	Metode HTTP	URL
Daftar Produk	`GET`	`/api/products`
Detail Produk	`GET`	`/api/products/123`
Tambah Produk	`POST`	`/api/products`
Perbarui Produk	`PUT` atau `PATCH`	`/api/products/123`
Hapus Produk	`DELETE`	`/api/products/123`

Representasi Sumber Daya

- Representasi sumber daya dalam REST API umumnya menggunakan format data seperti JSON atau XML.
- Ini memungkinkan klien dan server untuk berkomunikasi dengan format yang sama.

Representasi Sumber Daya

Format XML

```xml
<user>
  <id>123</id>
  <name>John Doe</name>
  <email>john.doe@example.com</email>
</user>
```

Format JSON

```json
{
  "id": 123,
  "name": "John Doe",
  "email": "john.doe@example.com"
}
```

Stateless

- REST API dianggap "stateless" karena setiap permintaan dari klien ke server harus mengandung semua informasi yang diperlukan.
- Server tidak menyimpan status klien antara permintaan.
- Hal ini memungkinkan server untuk memproses permintaan secara independen tanpa mempertimbangkan status sebelumnya.

Keuntungan REST API

- Kesederhanaan: Memahami konsep REST API relatif mudah.
- Keterbukaan: Format data umum memudahkan integrasi dengan berbagai platform.
- Fleksibilitas: REST API dapat digunakan dengan berbagai bahasa pemrograman.
- Dukungan: REST API didukung oleh banyak library dan framework.

Contoh Kode REST API

```
const http = require("http")

const products = [
  { id: 1, name: "Produk A" },
  { id: 2, name: "Produk B" },
  { id: 3, name: "Produk C" },
]

const server = http.createServer()

server.on("request", (req, res) => {
  if (req.method === "GET" && req.url === "/api/products") {
    res.writeHead(200, { "Content-Type": "application/json" })
    res.end(JSON.stringify(products))
  }
})

server.listen(3000, () => {
  console.log("Server berjalan di http://localhost:3000/")
})
```

Contoh Kode REST API

```
server.on("request", (req, res) => {
  if (req.method === "POST" && req.url === "/api/products") {
    let data = ""

    req.on("data", (chunk) => {
      data += chunk
    })

    req.on("end", () => {
      const newProduct = JSON.parse(data)
      res.writeHead(201, { "Content-Type": "application/json" })
      res.end(JSON.stringify(newProduct))
    })
  }
})
```

Contoh Kode REST API

```
server.on("request", (req, res) => {
  if (req.method === "PUT" && req.url.startsWith("/api/products/")) {
    const productId = parseInt(req.url.split("/").pop())

    let data = ""

    req.on("data", (chunk) => {
      data += chunk
    })

    req.on("end", () => {
      const updatedProduct = JSON.parse(data)

      // Perbarui produk dengan ID sesuai (tidak ditampilkan di sini).

      res.writeHead(200, { "Content-Type": "application/json" })
      res.end(JSON.stringify(updatedProduct))
    })
  }
})
```

Contoh Kode REST API

```
server.on("request", (req, res) => {
  if (req.method === "DELETE" && req.url.startsWith("/api/products/")) {
    const productId = parseInt(req.url.split("/").pop())

    // Hapus produk dengan ID sesuai (tidak ditampilkan di sini).

    res.writeHead(200)
    res.end()
  }
})
```

Latihan

Lengkapi contoh kode REST API pada contoh sebelumnya untuk menangani operasi `GET`, `POST`, `PUT`, dan `DELETE` pada sumber daya `/api/products`.

Ketentuan:

- Data produk disimpan dalam sebuah array.
- Setiap produk memiliki properti `id` dan `name`.
- Operasi `GET` menampilkan seluruh data produk.
- Operasi `POST` menambahkan data produk baru.
- Operasi `PUT` memperbarui data produk berdasarkan `id`.
- Operasi `DELETE` menghapus data produk berdasarkan `id`.
- Tampilan pesan error jika sumber daya tidak ditemukan atau data tidak valid.

Latihan

Buat sebuah aplikasi REST API sederhana untuk mengelola data buku dengan ketentuan:

- Data buku disimpan dalam sebuah array.
- Setiap buku memiliki properti `id`, `title`, `author`, dan `year`.
- Aplikasi dapat menampilkan seluruh data buku.
- Aplikasi dapat menampilkan data buku berdasarkan `id`.
- Aplikasi dapat menambahkan data buku baru.
- Aplikasi dapat memperbarui data buku berdasarkan `id`.
- Aplikasi dapat menghapus data buku berdasarkan `id`.