



Pengenalan dan Penggunaan Express.js untuk membangun REST API Sederhana





Zulka Ali Ajuab, S.Kom

2013 - 2017 : Instruktur di BBLKI Serang

2017 - Sekarang : Instruktur di BBPVP Bekasi



Tujuan Pembelajaran

Peserta pelatihan dapat membuat RESTful API sederhana tanpa menggunakan Database dalam lingkungan Node.js

Manfaat Pembelajaran

Peserta dapat membuat RESTful API dengan menggunakan Database dalam lingkungan Node.js

Express.JS

- Express.js adalah framework web untuk **Node.js** yang digunakan untuk:
 - Membuat aplikasi web atau API RESTful
 - Menangani HTTP request dan response
 - Menyusun middleware (fungsi penghubung antara permintaan dan respons)



Node.JS dan Express.JS

Node.js

- Node.js adalah lingkungan runtime JavaScript di sisi server. Artinya kita bisa menjalankan JavaScript di luar browser, seperti untuk membuat aplikasi backend (server), command-line tools, automation, dan lainnya
- Node.js memungkinkan kita menggunakan JavaScript untuk :
 1. Menangani permintaan HTTP
 2. Membaca dan menulis File
 3. Berinteraksi dengan database
 4. Menjalankan logika server, API, dan sebagainya

Ciri Khas Node.js

1. Non-blocking / Asynchronous I/O
Node.js sangat cocok untuk aplikasi yang membutuhkan banyak koneksi secara bersamaan karena tidak menunggu satu proses selesai sebelum memproses lainnya
2. Single-threaded dengan event Loop
Meskipun menggunakan satu thread, Node.js sangat efisien karena menggunakan event loop untuk menangani banyak request secara bersamaan.
3. Berbasis Chrome V8 Engine
Mesin Javascript yang sangat cepat (sama dengan yang digunakan Google Chrome)
4. NPM (Node Package Manager)
Node.js memiliki ekosistem pustaka terbesar di dunia (via npm) yang memudahkan pengembangan

Instalasi Node.js

- Di Windows Installer Node.js bisa didapatkan dari situs resminya yaitu : <https://nodejs.org>
- Unduh versi LTS (Long Term Support)
- Lalu jalankan file installer yang telah diunduh dan ikuti langkah instalasi hingga selesai
- Buka Command Prompt, lalu ketikkan script di bawah untuk memastikan Node.js dan npm berhasil terinstall

```
node -v  
npm -v
```


Kenapa Menggunakan Express.js??

Kelebihan	Penjelasan
 Sederhana & Minimalis	Struktur kode sangat ringkas dan mudah dipahami.
 Cepat	Ringan, tanpa banyak abstraksi yang rumit.
 Middleware Support	Bisa menyisipkan fungsi tambahan ke alur request/response.
 Routing Fleksibel	Routing sangat mudah dan powerful (REST API sangat cocok).
 Ekosistem Luas	Banyak plugin atau library yang mendukung Express.

Contoh Penggunaan Express.js

Untuk melakukan proses instalasi Express.js dalam lingkungan Node.js langkah-langkahnya adalah sebagai berikut :

1. Buat folder proyek dan berikan nama sesuai dengan proyek yang akan dibuat
2. Inisialisasi proyek dengan **npm init** atau **npm init -y** dengan menggunakan **Command Prompt**. Opsi **-y** akan langsung membuat package.json dengan nilai default, tanpa perlu menjawab pertanyaan satu persatu
3. Ketikkan perintah **npm install express**. Perintah ini akan menambahkan Express ke folder node_modules dan secara otomatis mencatatnya di package.json

Contoh Penggunaan Express.js (lanjutan)

Setelah dua proses di atas, maka akan terbentuk file dan folder di dalam folder proyek, yaitu :

1. package.json
2. package-lock.json
3. node_modules/

package.json

File ini adalah inti dari proyek Node.js. Fungsinya :

- Menyimpan informasi proyek (nama, versi, deskripsi, dll)
- Menyimpan dependensi (seperti Express) dalam bagian "dependencies"
- Menyimpan skrip, misalnya untuk menjalankan server (npm start)
- Memudahkan developer lain mengetahui dan menginstall dependensi cukup dengan np install
- Contoh isinya :

```
{
  "name": "myapp",
  "version": "1.0.0",
  "main": "index.js",
  "scripts": {
    "start": "node index.js"
  },
  "dependencies": {
    "express": "^4.18.2"
  }
}
```

package-lock.json

File ini otomatis dibuat saat kita menginstall package dengan npm.

Fungsinya :


- Mengunci versi pasti dari semua package (termasuk sub-dependensi)
- Memastikan bahwa setiap orang yang menginstall proyek ini akan mendapatkan versi yang sama persis dari semua library
- Berguna untuk keamanan dan stabilitas proyek, apalagi saat deploy ke production

node_modules/ (folder)

Folder ini adalah tempat semua library yang kita install disimpan.

Fungsinya :

- Tempat semua file dari Express dan dependensinya berada.
- Saat kita `require('express')`, Node.js akan mencari di folder ini.
- Ukurannya bisa sangat besar, karena bisa memuat banyak library dan sub-library.
- Catatan penting : `node_modules/` tidak perlu diupload ke Github atau dikirim ke orang lain. Cukup kirim `package.json` dan `package-lock.json`, lalu orang lain tinggal jalankan **npm install**, maka semua library akan otomatis diunduh ulang,



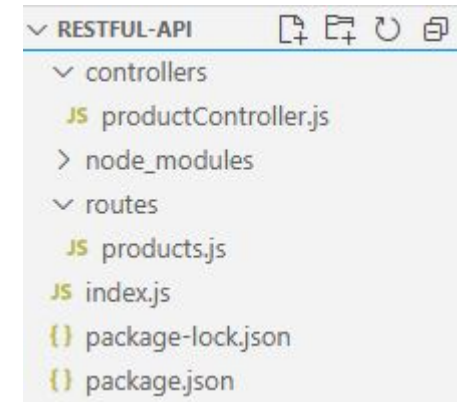
Membuat RESTFul API dengan Express.js tanpa menggunakan Database

Pengantar Projek

- Projek yang akan dibuat adalah backend Application yang dapat dijalankan lewat aplikasi yang dapat men-consume API seperti Postman dan semacamnya.
- Untuk penjelasan RESTful API dapat dilihat pada unit kompetensi yang lain
- Untuk struktur folder dan file, tidak ada keharusan mutlak dari Express.js sendiri. Express sangat fleksible, kita dapat Meletakkan semua kode dalam 1 file (index.js) atau memecah jadi folder seperti routes/, controllers/, models/, dll. Tetapi,
- Struktur folder yang rapi sangat penting jika proyek semakin besar, bekerja dalam tim lain, ingin menerapkan prinsip *separation of concern*

Langkah-langkah membuat aplikasi

1. Buat projek node baru dengan nama restful-api
2. Inisial projek dan install express.js
3. Diluar file dan folder yang terbentuk, buatlah beberapa file dan folder seperti :
 - **index.js** -> Entry point
 - **routes/**
 - **products.js** -> Routing RESTfull
 - **controllers/**
 - **productController.js** -> Logic Pemrosesan



Langkah-langkah membuat aplikasi

4. Pada index.js tambahkan script di bawah ini :

```
const express = require('express');
const app = express();
const productRoutes = require('./routes/products');

app.use(express.json()); // Middleware parsing JSON
app.use('/products', productRoutes); // Prefix endpoint

const PORT = 3000;
app.listen(PORT, () => {
  console.log(`Server RESTful berjalan di http://localhost:${PORT}`);
});
```

Langkah-langkah membuat aplikasi

5. Pada products.js tambahkan script di bawah ini :

```
const express = require('express');
const router = express.Router();
const controller = require('../controllers/productController');

// Routing RESTful
router.get('/', controller.getAllProducts);
router.get('/:id', controller.getProductById);
router.post('/', controller.createProduct);
router.put('/:id', controller.updateProduct);
router.delete('/:id', controller.deleteProduct);

module.exports = router;
```

Langkah-langkah membuat aplikasi

6. Pada **controllers/productController** tambahkan script di bawah ini, untuk membuatkan inisialisasi awal variabel products dalam bentuk array :

```
let products = [  
  { id: 1, name: 'Produk A', price: 10000 },  
  { id: 2, name: 'Produk B', price: 15000 }  
];
```


Langkah-langkah membuat aplikasi

7. Masih **controllers/productController** tambahkan script di bawah ini, untuk membuat logic mendapatkan seluruh product (getAllProducts):

```
exports.getAllProducts = (req, res) => {  
  res.status(200).json(products);  
};
```

Langkah-langkah membuat aplikasi

8. Masih **controllers/productController** tambahkan script di bawah ini, untuk membuat logic mendapatkan product by Id(getProductById):

```
exports.getProductById = (req, res) => {  
  const id = parseInt(req.params.id);  
  const product = products.find(p => p.id === id);  
  if (!product) return res.status(404).json({ message: 'Produk tidak ditemukan' });  
  res.status(200).json(product);  
};
```

Langkah-langkah membuat aplikasi

9. Masih **controllers/productController** tambahkan script di bawah ini, untuk membuat logic Create produk baru(createProduct):

```
exports.createProduct = (req, res) => {  
  const { name, price } = req.body;  
  if (!name || typeof price !== 'number') {  
    return res.status(400).json({ message: 'Input tidak valid' });  
  }  
  
  const newProduct = {  
    id: products.length ? Math.max(...products.map(p => p.id)) + 1 : 1,  
    name,  
    price  
  };  
  
  products.push(newProduct);  
  res.status(201).json(newProduct);  
};
```


Langkah-langkah membuat aplikasi

10. Masih **controllers/productController** tambahkan script di bawah ini, untuk membuat logic update product (updateProduct):

```
exports.updateProduct = (req, res) => {  
  const id = parseInt(req.params.id);  
  const product = products.find(p => p.id === id);  
  if (!product) return res.status(404).json({ message: 'Produk tidak ditemukan' });  
  
  const { name, price } = req.body;  
  if (!name || typeof price !== 'number') {  
    return res.status(400).json({ message: 'Input tidak valid' });  
  }  
  
  product.name = name;  
  product.price = price;  
  res.status(200).json(product);  
};
```

Langkah-langkah membuat aplikasi

11. Masih **controllers/productController** tambahkan script di bawah ini, untuk membuat logic menghapus produk (deleteProduct):

```
exports.deleteProduct = (req, res) => {  
  const id = parseInt(req.params.id);  
  const index = products.findIndex(p => p.id === id);  
  if (index === -1) return res.status(404).json({ message: 'Produk tidak ditemukan' });  
  
  products.splice(index, 1);  
  res.status(204).send();  
};
```

Langkah-langkah membuat aplikasi

12. Jalankan Server dengan mengetikkan **node index.js**:

```
PS D:\> cd "C:\Users\user\Documents\restful-api" & node index.js
Server RESTful berjalan di http://localhost:3000
```

13. Test aplikasi dengan aplikasi **Postman** atau extension **Thunder Client** di **Vs.Code**
14. Untuk menjalankannya url yang digunakan adalah `http://localhost:3000` + Endpoint RESTfull seperti pada list di bawah:

Method	Endpoint	Keterangan
GET	/products	Ambil semua produk
GET	/products/:id	Ambil satu produk
POST	/products	Tambah produk baru
PUT	/products/:id	Update produk tertentu
DELETE	/products/:id	Hapus produk tertentu

Langkah-langkah membuat aplikasi

15. Sebagai contoh untuk mengakses semua produk, maka urlnya adalah :

The screenshot displays a REST client interface with the following components:

- Method:** A dropdown menu showing "GET".
- Endpoint:** A text input field containing "http://localhost:3000/products".
- Buttons:** A "Send" button and a "Klik" label.
- Tabs:** "Query", "Headers", "Auth", "Body", "Tests", and "Pre Run".
- Query Parameters:** A section with a checkbox labeled "parameter" and a "value" field.
- Status:** "Status: 200 OK", "Size: 83 Bytes", and "Time: 11 ms".
- Response:** A JSON array of two product objects.

Red annotations highlight specific parts of the interface:

- A red arrow points to the "GET" method, labeled "Method".
- A red arrow points to the endpoint URL, labeled "Endpoint".
- A red box surrounds the "Send" button, labeled "Klik".
- A red box surrounds the JSON response, labeled "Hasil / Response".

```
1  [  
2    {  
3      "id": 1,  
4      "name": "Produk A",  
5      "price": 10000  
6    },  
7    {  
8      "id": 2,  
9      "name": "Produk B",  
10     "price": 15000  
11   }  
12  ]
```

Langkah-langkah membuat aplikasi

16. Untuk mengakses produk by id, maka urlnya adalah :

GET ⌵ http://localhost:3000/products/1 Send		Status: 200 OK Size: 40 Bytes Time: 4 ms	
<u>Query</u> Headers ² Auth Body Tests Pre Run		<u>Response</u> Headers ⁶ Cookies Results Docs	
Query Parameters		<pre>1 { 2 "id": 1, 3 "name": "Produk A", 4 "price": 10000 5 }</pre>	
<input type="checkbox"/>	parameter		value

Langkah-langkah membuat aplikasi

17. Untuk mengakses createProduk, maka urlnya adalah :

The screenshot displays a REST client interface with the following components:

- Method:** POST (labeled 1)
- URL:** http://localhost:3000/products (labeled 2)
- Body:** JSON (labeled 3 and 4). The JSON content is:

```
{  "name": "Produk Baru",  "price": 15000}
```

 (labeled 5).
- Send Button:** A blue button to execute the request (labeled 6).
- Status:** 201 Created (labeled 6).
- Size:** 43 Bytes.
- Time:** 4 ms.
- Response:** A JSON object:

```
{  "id": 3,  "name": "Produk Baru",  "price": 15000}
```

 (labeled 6).

Kesimpulan

- Pembuatan aplikasi backend (RESTful API) baik yang terhubung dengan database atau tidak, dapat menggunakan library Express.js dalam lingkungan Node.js
- Struktur Folder dalam aplikasi perlu dipahami dengan baik, agar dapat dilakukan scale up ketika aplikasi bertambah besar
- Untuk pengujian aplikasi backend dapat menggunakan aplikasi PostMan dan extension Thunder Client pada Vs.Code



**TERIMA
KASIH**

