



JS

# AsrulDev - Node JS

---

## Pertemuan 1: Mengenal Express JS

[Dasar-dasar Jaringan](#)

[Server dan Client](#)

[Membuat server HTTP](#)

[Install Express JS](#)

[Membangun Web Server](#)

[Membuat Rute](#)

[Mengenal Request](#)

[Response Express](#)

[Menggunakan Router](#)

[Menggunakan Controller](#)

## Pertemuan 2: CRUD - MySQL

[CREATE](#)

[READ](#)

[UPDATE](#)

[DELETE](#)

---

## Pertemuan 1: Mengenal Express JS

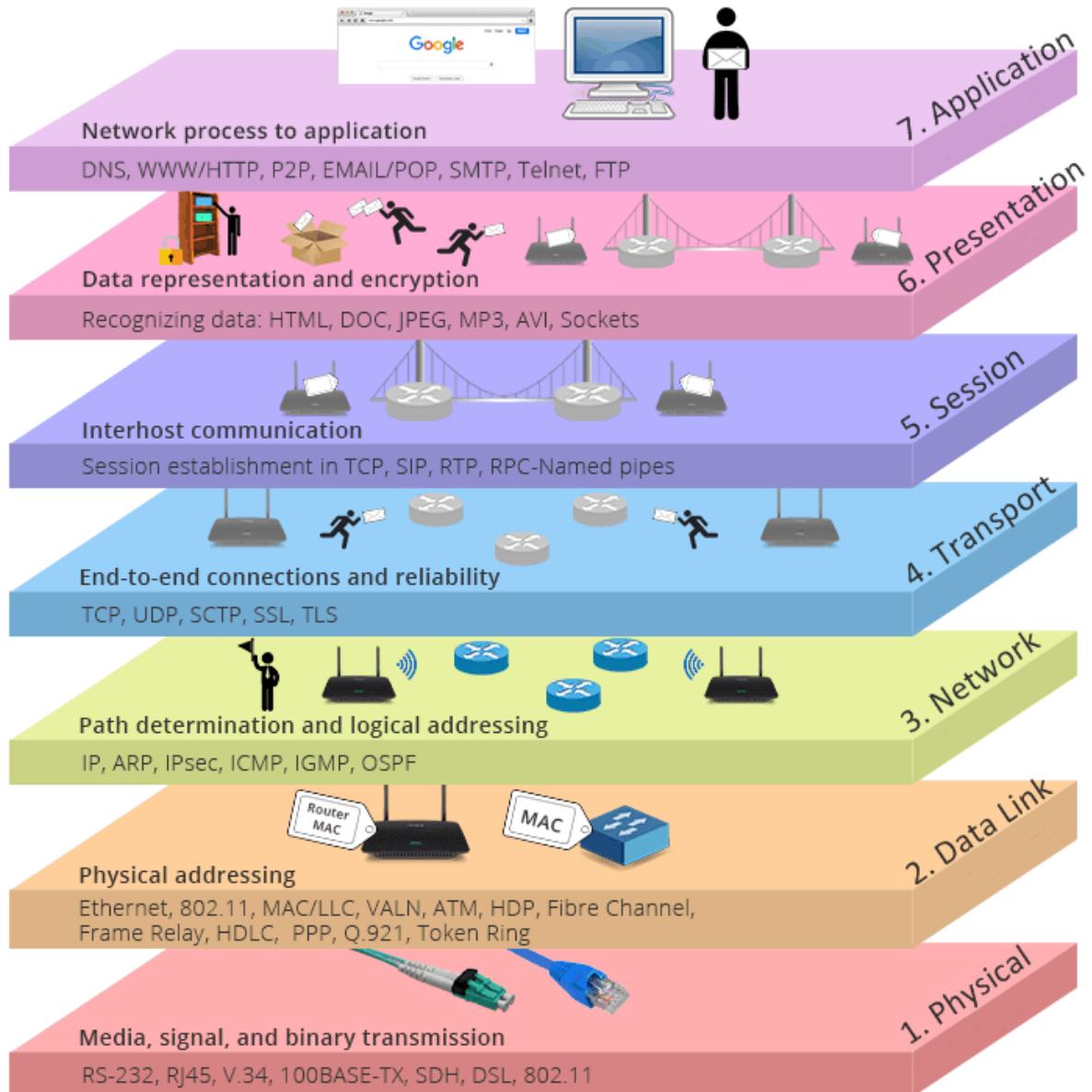
Hampir semua karir saya, saya telah bekerja secara eksklusif sebagai frontend (sisi client). Merancang layout, responsif, membuat komponen, dan lainnya. Karena itu saya hampir tidak pernah berurusan dengan request routing atau

HTTP secara langsung. Maka dengan ini saya sempatkan menulis agar tidak lupa dikemuan hari. Yok.

## **Dasar-dasar Jaringan**

HTTP (Hypertext Transfer Protocol) adalah protocol komunikasi yang digunakan dalam jaringan komputer. Internet sendiri memiliki banyak protokol seperti SMTP (Simple Mail Transfer Protocol), FTP (File Transfer Protocol), POP3 (Post Office Protocol 3) dan lainnya.

Protokol-protokol ini memungkinkan perangkat dengan hardware atau software yang berbeda dapat berkomunikasi satu sama lain karena memberikan format pesan yang jelas, aturan, sintaks, dll. Ini berarti selama perangkat mendukung protokol tersebut, maka ia dapat berkomunikasi dengan perangkat lainnya dalam jaringan.



Sumber: Medium

Sistem operasi biasanya menyediakan dukungan HTTP sehingga tidak perlu menginstall aplikasi tambahan untuk mengakses web.

HTTP, yang digunakan untuk menjalankan web, berbeda. Ini dikenal sebagai protokol tanpa koneksi, karena didasarkan pada mode operasi permintaan / response. Browser web membuat permintaan ke server untuk gambar, font, konten dll. Tetapi begitu permintaan terpenuhi, koneksi antara browser dan server terputus.

## Server dan Client

Istilah server mungkin sedikit membingungkan bagi orang yang baru mengenal industri karena dapat merujuk pada perangkat keras (komputer fisik yang menampung semua file dan perangkat lunak yang diperlukan oleh situs web) atau perangkat lunak (program yang memungkinkan pengguna untuk mengakses file-file itu di web ).

Hari ini, kita akan berbicara tentang sisi perangkat lunak. Tetapi pertama-tama, beberapa definisi. URL adalah singkatan dari **Universal Resource Locator**, dan terdiri dari 3 bagian: protokol , server dan file yang diminta .

Protokol HTTP mendefinisikan beberapa metode yang dapat digunakan browser untuk meminta server melakukan banyak tindakan berbeda, yang paling umum adalah GET dan POST. Saat pengguna mengklik tautan atau memasukkan URL di field alamat, browser mengirim GET request ke server untuk mengambil resource yang ditentukan dalam URL.

Server perlu tahu cara memproses HTTP request ini untuk mengambil file yang benar kemudian mengirimkannya kembali ke browser yang memintanya sebagai response.

## Membuat server HTTP

Berikut cara membuat server dengan express

```
const express = require("express");
const app = express();

app.use(express.static("public"));

app.get("/", (request, response) => {
  response.sendFile(__dirname + "/public/index.html");
});

app.listen(5000);
```

## Install Express JS

Untuk menggunakan express js, sebelumnya kamu harus pastikan bahwa di laptop telah terinstall Node Js dan NPM.

Buka terminal dan buat sebuah folder tempat kamu akan meletakkan project tersebut anggap saja foldernya bernama **belajar-api**.

Setelah itu buat initial project dengan perintah berikut.

```
npm init
```

atau dengan tambahan flag `-y` dan menyatakan bahwa flag `-y` kamu setuju generate otomatis yang dilakukan npm untuk membuat file **package.json**

```
npm init -y
```

File package.json adalah file yang berisi semua informasi package atau library yang dipakai untuk membuat atau membangun aplikasi atau web tersebut.

Sejauh ini kita sudah berhasil untuk membuat project yang akan dibangun, selanjutnya kita akan menginstall `express.js`, cara installnya berikut ini.

```
npm install express --save
```

Kalau dilihat kembali file `package.json` maka akan ada express yang baru saja kita install seperti berikut.

```
{
  "name": "belajar-api",
  "version": "1.0.0",
  "description": "",
  "main": "server.js",
  "scripts": {
    "test": "echo \\"\\Error: no test specified\\\" && exit 1"
  },
  "keywords": [],
  "author": "AsrulDev",
  "license": "ISC",
  "dependencies": {
    "express": "^4.17.1"
  }
}
```

Kamu selesai melakukan install express js, silahkan cobakan dan tanyakan intruktur jika ada yang belum dimengerti...

## Membangun Web Server

Web server adalah server yang bertugas untuk menjalankan web agar dapat melakukan request dari user dan memberikan response sebagai keluaran.

Setelah menginstall express js, sekarang buatlah file baru dengan nama `server.js` yang isinya sebagai berikut.

```
const express = require("express");
const app = express();

app.listen(5000, (req, res) => {
  console.log("Server is running with <http://localhost:5000>");
});
```

Setelah itu jalankan server dengan perintah

```
node server.js
```

Jika terminal menampilkan `Server is running with <http://localhost:5000>` maka kamu bisa cek menggunakan web browser atau postman untuk membuktikan server telah berjalan dengan baik.

## Membuat Rute

Rute adalah jalan atau jalur yang mengarahkan user atau pengguna melakukan request ke web server sehingga mendapat response dari web server tersebut.

Untuk menjalankan rute ini terlebih dahulu kita harus memahami `HTTP Methode` secara sederhana.

1. GET
2. POST
3. PUT / PATCH
4. DELETE

Penggunaan HTTP Methode ini pada express telah menyediakan `function` yang bisa dipakai yaitu: `get()` , `post()` , `put()` , `delete()` , `patch()`.

```
app.get("/", (req, res) => {
  /* */
});
app.post("/", (req, res) => {
  /* */
});
app.put("/", (req, res) => {
  /* */
});
app.delete("/", (req, res) => {
  /* */
});
app.patch("/", (req, res) => {
  /* */
});
```

Berikut cara penggunaannya:

```
const express = require("express");
const app = express();

app.get("/", (req, res) => {
  res.send("Halo Aa Academy, ini web pertama saya");
});

app.listen(5000, (req, res) => {
  console.log("Server is running with <http://localhost:5000>");
});
```

Ubah kembali rute dan tambahkan rute `/about` sehingga kode menjadi seperti berikut

```
// ...
app.get('/', (req, res) => {
  res.send('ini rute utama atau root')
})

app.get('/about', (req, res) => {
  res.send('ini rute /about')
})
// ...
```

Ketika server dijalankan, kita telah memiliki 2 rute yaitu rute `/` dan `/about`

Catatan:

- `req` adalah parameter yang berfungsi sebagai request dari user ke server
- `res` adalah parameter yang berfungsi sebagai response dari server untuk user

## Mengenal Request

Express telah menyediakan property-property yang bertindak sebagai request, berikut beberapa daftar request yang disediakan oleh express:

### Property Request Express

Aa Properti	☰ Deskripsi
<code>.app</code>	Referensi object pada express
<code>.cookies</code>	Berisi informasi cookie yang dikirim, ini dapat digunakan jika menggunakan middleware <code>cookie-parser</code>
<code>.hostname</code>	Informasi hostname web server
<code>.ip</code>	Informasi IP server
<code>.method</code>	HTTP Methode yang digunakan
<code>.params</code>	Menampilkan informasi sesuai dengan nama parameter
<code>.path</code>	Nemampilkan informasi jalur URL
<code>.protocol</code>	Menampilkan protocol request
<code>.query</code>	Objek yang berisi informasi query dari request yang berlangsung
<code>.secure</code>	true jika request merupakan request yang secure (digunakan pada HTTPS)
<code>.signedCookies</code>	Berisi signed cookies oleh request, dapat digunakan jika menggunakan middleware <code>cookie-parser</code>
<code>.xhr</code>	Bernilai true jika request adalah XMLHttpRequest

Aa Properti	☰ Deskripsi
.body	Berisi data yang dituliskan pada body request
.header	Berisi data yang dituliskan pada header request
.baseUrl	Informasi pengalaman utama web server
.originalUrl	Informasi pengalaman request dilakukan

Contoh penggunaan `.query`

```
?name=Asrul
```

Tulis berikut pada kode

```
app.get("/about", (req, res) => {
  res.send("ini rute /about " + req.query.name);
});
```

Jika dijalankan akan menghasilkan seperti berikut ini

```
?name=Asrul
```

Tulis berikut pada code dan ketika ad query pada url dengan key **nama** maka akan diterima oleh server sebagai request.

```
app.get("/about", (req, res) => {
  res.send(
    "ini rute /about " + req.query.name + " dan umurnya " + req.query.age
  );
});
```

## Response Express

Sama halnya dengan request, `function` response sangat banyak yang disediakan oleh express.

## Method Response

Aa Properti	☰ Deskripsi
<u>.send()</u>	Mengirim dan menampilkan data string ke user
<u>.end()</u>	Mengakhiri response server
<u>.status()</u>	Informasi HTTP status, seperti 200, 404, dll
<u>.json()</u>	Menjadikan response sebagai data JSON
<u>.render()</u>	Merender file sebagai HTML
<u>.download()</u>	Memberikan response sebagai file yang dapat diunduh
<u>.type()</u>	Menyetting tipe response server
<u>.redirect()</u>	Mengalihkan halaman ke halaman yang lain

Contoh penggunaan `.redirect()`

```
app.get("/profile", (req, res) => {
  res.redirect("/login");
});

app.get("/login", (req, res) => {
  res.send("silahkan Login terlebih dahulu");
});
```

Silahkan cobakan dan baca cara penggunaannya dari beberapa sumber, seperti [expressjs.com](http://expressjs.com) atau serching dari google.

## Menggunakan Router

Router telah kita bahas pada awal pembuatan server, apa gunanya dan bagaimana pembuatannya. Tetapi kita menuliskannya pada file yang sama dengan server, sehingga jika aplikasi web tersebut semakin besar maka pengelolaannya akan semakin sulit. Untuk itu kita akan pisahkan sfile yang menangani `server` dan file yang menangani `router`.

Berikut adalah kode pada file `server.js` yang telah dibuat.

```
const express = require("express");
const app = express();

app.get("/", (req, res) => {
    res.send("ini rute utama atau root");
});

app.get("/profile", (req, res) => {
    res.redirect("/login");
});

app.get("/login", (req, res) => {
    res.send("silahkan Login terlebih dahulu");
});

app.get("/about", (req, res) => {
    res.send(
        "ini rute /about " + req.query.name + " dan umurnya " + req.query.age
    );
});

app.listen(5000, (req, res) => {
    console.log("Server is running with <http://localhost:5000>");
});
```

Kita akan pisahkan menjadi 2 file dengan nama file `server.js` dan `router.js`

file `server.js`

```
const express = require("express");
const app = express();

app.use("/", require("./router"));

app.listen(5000, (req, res) => {
    console.log("Server is running with <http://localhost:5000>");
});
```

File `router.js`

```
const router = require("express").Router();

router.get("/", (req, res) => {
```

```

    res.send("ini rute utama atau root");
});

router.get("/profile", (req, res) => {
    res.redirect("/login");
});

router.get("/login", (req, res) => {
    res.send("silahkan Login terlebih dahulu");
});

router.get("/about", (req, res) => {
    res.send(
        "ini rute /about " + req.query.name + " dan umurnya " + req.query.age
    );
});

module.exports = router;

```

Jika dijalankan maka akan menghasilkan hal yang sama dengan sebelumnya, tetapi penulisan lebih rapi.

## Menggunakan Controller

Setelah kita pisahkan file server dan router, ternyata router kita masih melakukan proses logika yang seharusnya hanya dilakukan oleh controller.

File `router.js`

```

const router = require("express").Router();

router.get("/", (req, res) => {
    res.send("ini rute utama atau root");
});

router.get("/profile", (req, res) => {
    res.redirect("/login");
});

router.get("/login", (req, res) => {
    res.send("silahkan Login terlebih dahulu");
});

router.get("/about", (req, res) => {
    res.send(
        "ini rute /about " + req.query.name + " dan umurnya " + req.query.age
    );
});

```

```
});  
  
module.exports = router;
```

Ubah `router.js` dan tambahkan file `controller.js` sehingga menjadi.

File `Router.js`

```
const router = require("express").Router();  
const { index, profile, login, about } = require("./controller");  
  
router.get("/", index);  
router.get("/profile", profile);  
router.get("/login", login);  
router.get("/about", about);  
  
module.exports = router;
```

File `Controller.js`

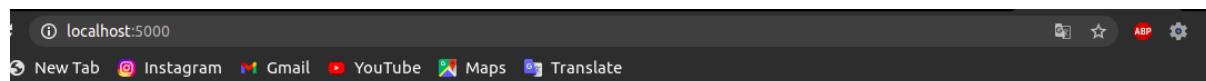
```
const index = (req, res) => {  
    res.send("ini rute utama atau root");  
};  
  
const profile = (req, res) => {  
    res.redirect("/login");  
};  
  
const login = (req, res) => {  
    res.send("silahkan Login terlebih dahulu");  
};  
  
const about = (req, res) => {  
    res.send(  
        "ini rute /about " + req.query.name + " dan umurnya " + req.query.age  
    );  
};  
  
module.exports = {  
    index,  
    profile,  
    login,  
    about,  
};
```

Jika tidak ada kesalahan dalam penulisan kode, maka hasilnya akan sama dengan yang sebelumnya.

## Pertemuan 2: CRUD - MySQL

Sebelum mulai siapkan terlebih dahulu:

1. Nodejs yang telah terinstall
2. Mysql server, boleh pakai xampp, lampp ataumamp
3. VS Code, sublime, atom atau lainnya sebagai editor
4. Browser
5. Semangat yang tinggi



Product List			
Product ID	Product Name	Price	Action
2	HP Nokia	700000	<button>Edit</button> <button>Delete</button>

Pertama sekali anda harus membuat projek baru pada sebuah folder, misal saya membuat folder `crud-express` dan masuk kedalam folder tersebut.

```
mkdir crud-express
cd crud-express
```

Buatlah project node pada folder tersebut sehingga menghasilkan `package.json` dengan kode `npm init`, isi nama project dan beberapa pertanyaan seperti versi, main file, author, descripsi, dll sehingga hasilnya seperti berikut.

```
{  
  "name": "crud-express",  
  "version": "1.0.0",  
  "description": "crud express js dan mysql",  
  "main": "server.js",  
  "scripts": {  
    "start": "nodemon",  
    "test": "echo \\\"Error: no test specified\\\" && exit 1"  
  },  
  "author": "Asrul Harahap",  
  "license": "ISC"  
}
```

Install package express, mysql, dan hbs dengan cara :

```
npm install express  
npm install mysql  
npm install hbs
```

atau dengan cara berikut

```
npm install express mysql hbs
```

Sehingga hasilnya menjadi:

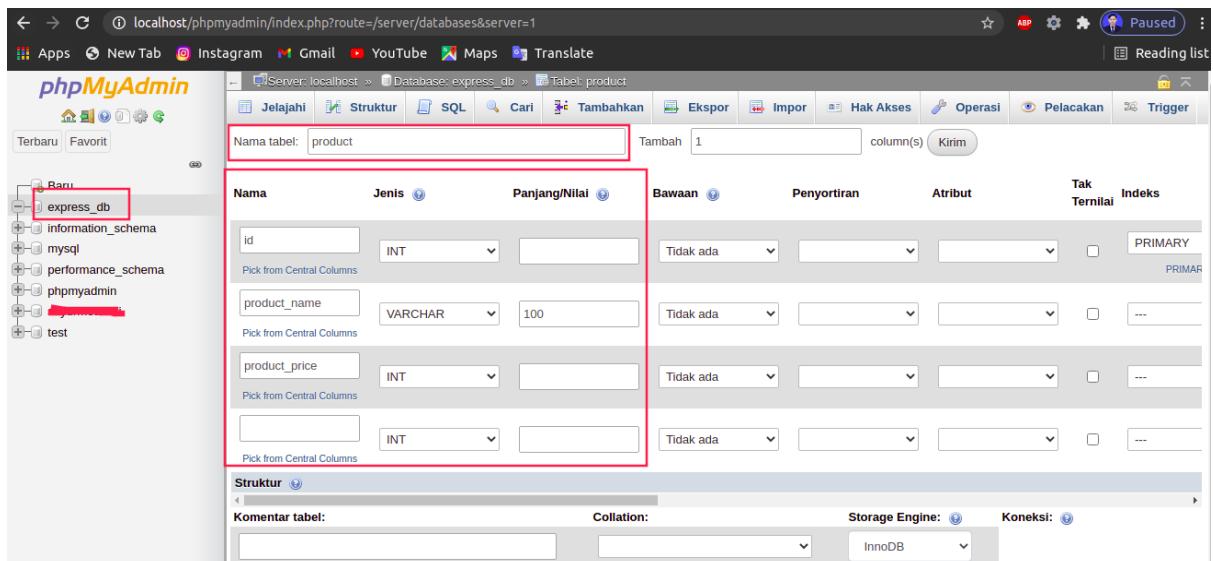
```
{  
  "name": "crud-express",  
  "version": "1.0.0",  
  "description": "crud express js dan mysql",  
  "main": "index.js",  
  "scripts": {  
    "start": "nodemon",  
    "test": "echo \\\"Error: no test specified\\\" && exit 1"  
  },  
  "author": "Asrul Harahap",  
  "license": "ISC",  
  "dependencies": {  
    "express": "^4.17.1",  
    "hbs": "^4.0.6",  
    "mysql": "^2.17.1"  
  }  
}
```

Buatlah file server `server.js` untuk menjalankan server nodejs. Tulis kode berikut

```
const express = require("express");
const app = express();

//server listening
app.listen(5000, () => {
    console.log("Server is running at port 5000");
});
```

Kemudian buatlah database pada mysql, boleh menggunakan phpmyadmin biar lebih cepat dengan nama **express\_db** (hanya contoh saja).



The screenshot shows the phpMyAdmin interface for a MySQL database named 'express\_db'. A new table named 'product' is being created. The table structure is as follows:

Nama	Jenis	Panjang/Nilai	Bawaan	Penyortiran	Atribut	Tak Ter nilai	Indeks
id	INT		Tidak ada			<input checked="" type="checkbox"/>	PRIMARY
product_name	VARCHAR	100	Tidak ada			<input type="checkbox"/>	---
product_price	INT		Tidak ada			<input type="checkbox"/>	---
	INT		Tidak ada			<input type="checkbox"/>	---

Below the table structure, there are fields for 'Struktur', 'Komentar tabel:', 'Collation:', 'Storage Engine:', and 'Koneksi:'.

Edit kembali `server.js` dan tambahkan kode untuk koneksi express dengan mysql seperti berikut ini.

```
const mysql = require("mysql");

//konfigurasi koneksi
const conn = mysql.createConnection({
    host: "localhost",
```

```
    user: "root",
    port: "3306",
    password: "root",
    database: "express_db",
  });

//connect ke database
conn.connect((err) => {
  if (err) throw err;
  console.log("Mysql Connected bro...");
});
```

Jika kita jalankan dengan cara `node index.js` maka tampilannya seperti berikut:

```
Server is running at port 5000
Mysql Connected...
```

Kita tahu bahwa server sudah jalan pada <http://localhost:5000> dan juga sudah terkoneksi dengan database. Selanjutnya kita set untuk view sehingga dapat ditampilkan dengan html pada browser. Tambah beberapa kode berikut pada file `index.js`

```
const path = require("path");

app.set("views", path.join(__dirname, "views"));
app.set("view engine", "hbs");
```

Tambahkan kode berikut pada `server.js` sebagai controller

## CREATE

```
app.use(express.json());
app.use(express.urlencoded({ extended: false }));

app.post("/save", (req, res) => {
  let data = {
    product_name: req.body.product_name,
    product_price: req.body.product_price,
  };
});
```

```
let sql = "INSERT INTO product SET ?";
conn.query(sql, data, (err, results) => {
  if (err) throw err;
  res.redirect("/");
});
});
```

## READ

```
app.get("/", (req, res) => {
  let sql = "SELECT * FROM product";
  conn.query(sql, (err, results) => {
    if (err) throw err;
    res.render("list", {
      results: results,
    });
  });
});
```

## UPDATE

```
app.post("/update", (req, res) => {
  let sql =
    "UPDATE product SET product_name=' " +
    req.body.product_name +
    "' , product_price=' " +
    req.body.product_price +
    "' WHERE id=" +
    req.body.id;
  conn.query(sql, (err, results) => {
    if (err) throw err;
    res.redirect("/");
  });
});
```

## DELETE

```
app.post("/delete", (req, res) => {
  let sql = "DELETE FROM product WHERE id=" + req.body.product_id + "";
  conn.query(sql, (err, results) => {
    if (err) throw err;
    res.redirect("/");
  });
});
```

Kemudian buat file untuk ditampilkan ke client pada folder views dengan nama file `list.hbs`

```
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>CRUD Express JS</title>
  <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.1.3/css/bootstrap.min.css" integrity="sha384-MCw98/SFnGE8fJT3GXwEOngsV7Zt27NXFoaoApmYm81iuXoPkFOJwJ8ERdknLPMO" crossorigin="anonymous">
</head>
<body>
  <div class="container">
    <h2 class="text-center">Product List</h2>
    <button class="btn btn-success" data-toggle="modal" data-target="#myModalAdd">
      Add New
    </button>
    <table class="table table-striped" id="mytable">
      <thead>
        <tr>
          <th>Product ID</th>
          <th>Product Name</th>
          <th>Price</th>
          <th>Action</th>
        </tr>
      </thead>
      <tbody>
        {{#each results}}
          <tr>
            <td>{{ id }}</td>
            <td>{{ product_name }}</td>
            <td>{{ product_price }}</td>
            <td>
              <a href="javascript:void(0);" class="btn btn-sm btn-info edit" data-id="{{ id }}" data-product_name="{{ product_name }}" data-product_price="{{ product_price }}>Edit</a>
              <a href="javascript:void(0);" class="btn btn-sm btn-danger delete" data-id="{{ id }}>Delete</a>
            </td>
          </tr>
        {{/each}}
      </tbody>
    </table>
  </div>

  <!-- Modal Add Produk-->
  <form action="/save" method="post">
    <div class="modal fade" id="myModalAdd" tabindex="-1" role="dialog" aria-labelledby="exampleModalLabel" aria-hidden="true">
      <div class="modal-dialog" role="document">
        <div class="modal-content">
          <div class="modal-header">
            <h5 class="modal-title" id="exampleModalLabel">Add New Product</h5>
            <button type="button" class="close" data-dismiss="modal" aria-label="Close">
              <span aria-hidden="true">&times;</span>
            </button>
          </div>
          <div class="modal-body">
            <input type="text" class="form-control" placeholder="Product Name" name="product_name">
            <input type="number" class="form-control" placeholder="Product Price" name="product_price">
          </div>
          <div class="modal-footer">
            <button type="button" class="btn btn-secondary" data-dismiss="modal">Cancel
            <button type="submit" class="btn btn-primary">Save
          </div>
        </div>
      </div>
    </form>
  </div>
</body>
</html>
```

```

                </button>
            </div>
            <div class="modal-body">
                <div class="form-group">
                    <input type="text" name="product_name" class="form-control" placeholder="Product Name" required>
                </div>

                <div class="form-group">
                    <input type="text" name="product_price" class="form-control" placeholder="Price" required>
                </div>
                <div class="modal-footer">
                    <button type="button" class="btn btn-secondary" data-dismiss="modal">Close</button>
                    <button type="submit" class="btn btn-primary">Save</button>
                </div>
            </div>
        </div>
    </form>

    <!-- Modal Update Produk-->
    <form action="/update" method="post">
        <div class="modal fade" id="EditModal" tabindex="-1" role="dialog" aria-labelledby="exampleModalLabel" aria-hidden="true">
            <div class="modal-dialog" role="document">
                <div class="modal-content">
                    <div class="modal-header">
                        <h5 class="modal-title" id="exampleModalLabel">Edit Product</h5>
                        <button type="button" class="close" data-dismiss="modal" aria-label="Close">
                            <span aria-hidden="true">&times;</span>
                        </button>
                    </div>
                    <div class="modal-body">
                        <div class="form-group">
                            <input type="text" name="product_name" class="form-control" product_name placeholder="Product Name" required>
                        </div>

                        <div class="form-group">
                            <input type="text" name="product_price" class="form-control" placeholder="Price" required>
                        </div>
                        <div class="modal-footer">
                            <input type="hidden" name="id" class="product_id">
                            <button type="button" class="btn btn-secondary" data-dismiss="modal">Close</button>
                            <button type="submit" class="btn btn-primary">Update</button>
                        </div>
                    </div>
                </div>
            </div>
        </form>

```

```

<!-- Modal Delete Produk-->
<form id="add-row-form" action="/delete" method="post">
    <div class="modal fade" id="DeleteModal" tabindex="-1" role="dialog" aria-labelledby="myModalLabel" aria-hidden="true">
        <div class="modal-dialog">
            <div class="modal-content">
                <div class="modal-header">
                    <h5 class="modal-title" id="myModalLabel">Delete Product</h5>
                    <button type="button" class="close" data-dismiss="modal" aria-label="Close"><span aria-hidden="true">&times;</span></button>
                </div>
                <div class="modal-body">
                    <strong>Anda yakin mau menghapus data ini?</strong>
                </div>
                <div class="modal-footer">
                    <input type="hidden" name="product_id" class="form-control product_id2" required>
                    <button type="button" class="btn btn-default" data-dismiss="modal">Close</button>
                    <button type="submit" class="btn btn-success">Delete</button>
                </div>
            </div>
        </div>
    </form>

    <script src="<https://code.jquery.com/jquery-3.3.1.slim.min.js>" integrity="sha384-q8i/X+965Dz00rT7abK41JStQIAqVgRVzpbzo5smXKp4YfRvH+8abTE1Pi6jizo" crossorigin="anonymous"></script>
    <script src="<https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.14.3/umd/popper.min.js>" integrity="sha384-ZMP7rVo3mIykV+2+9J3UJ46jBk0WLauUAdn689aCwoqbBJiSnjAK/l8WvCWPIPM49" crossorigin="anonymous"></script>
    <script src="<https://stackpath.bootstrapcdn.com/bootstrap/4.1.3/js/bootstrap.min.js>" integrity="sha384-ChfqqxuZUCnJSK3+MXmPNIyE6ZbWh2IMqE241rYiqJxyMiZ6OW/JmZQ5stwEULTy" crossorigin="anonymous"></script>
    <script>
        $(document).ready(function(){
            //tampilkan data ke modal untuk edit
            $('#mytable').on('click', '.edit', function(){
                var product_id = $(this).data('id');
                var product_name = $(this).data('product_name');
                var product_price = $(this).data('product_price');
                $('#EditModal').modal('show');
                $('.product_name').val(product_name);
                $('.price').val(product_price);
                $('.product_id').val(product_id);
            });
            //tampilkan modal hapus record
            $('#mytable').on('click', '.delete', function(){
                var product_id = $(this).data('id');
                $('#DeleteModal').modal('show');
                $('.product_id2').val(product_id);
            });
        });
    </script>

```

```
</body>  
</html>
```

Selengkapnya lihat source code <https://github.com/saturdaycode/mmc-crud-express-mysql>