NODE.JS WORKSHOP
HARI 3: WEB FRAMEWORK
NODE.JS

Adityo Pratomo (Framework)
Mozilla, 1 Oktober 2016

REVIEW

- API Node JS
  - Module
  - HTTP
  - fs

- Membuat server dengan Node.js
- Bekerja low level dengan paket HTTP

MATERI HARI INI

- Express: Web Framework untuk Node.js
- Membuat Backend sederhana dengan Express
- Memahami Arsitektur MVC

WEB FRAMEWORK

- Kerangka yang membantu menyusun sistem back-end sebuah website
- Dalam Node.js, ia membantu mengabstraksi sistem back-end dengan bekerja di atas koneksi dan packet handling yang sudah dijalankan oleh API Node.js
- Kita tidak lagi perlu membuat sistem routing untuk setiap jenis request yang masuk

WEB FRAMEWORK

EXPRESS: WEB FRAMEWORK UNTUK NODE.JS

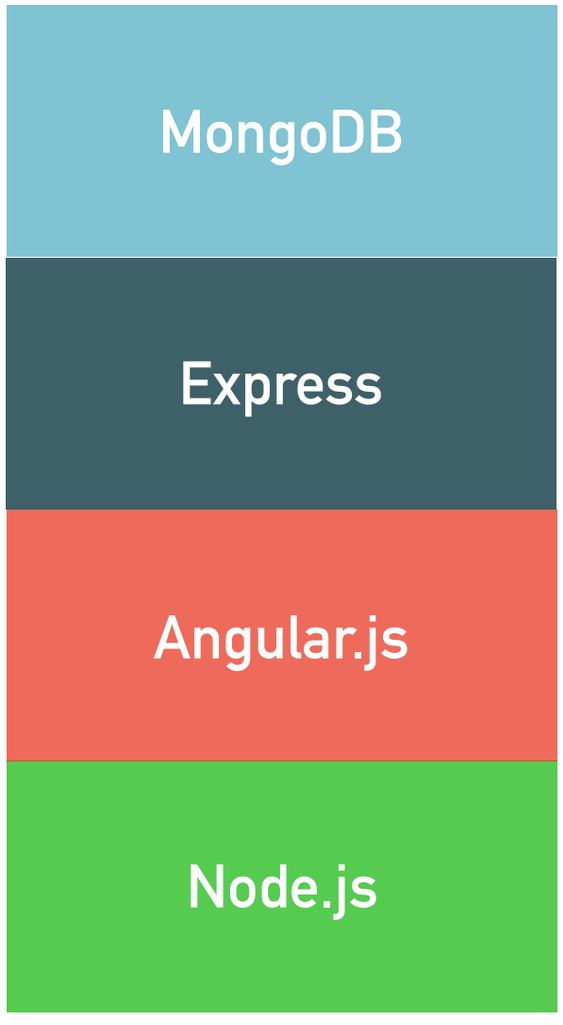
- Express adalah salah satu web framework yang bisa digunakan untuk Node.js
- Penggunaannya cukup sederhana
- Memiliki beragam tool yang membantu development
- Menggunakan berbagai middleware yang bisa digunakan sesuai kebutuhan, ataupun dibuat sendiri
- expressjs.com

EXPRESS: WEB FRAMEWORK UNTUK NODE.JS

- Express bukan satu-satunya web framework untuk Node.js
- Contoh lain:
  - Hapi
  - Koa.js
  - Sails
- Kita menggunakan Express karena developmentnya paling matang dan paling banyak digunakan untuk proyek Node saat ini
- Proyek milik Node.js Foundation

MEAN STACK

- Bersama Node, Express menjadi salah satu dari 4 komponen penyusun MEAN Stack
- MEAN adalah sebuah teknologi web development full stack (back-end dan front-end) yang semuanya disusun dengan teknologi berbasis JavaScript
- MongoDB, Express, Angular, Node
- Alternatif dari LAMP (Linux, Apache, MySQL, PHP)



MENGGUNAKAN EXPRESS

- Install Express dan Express Generator

```
npm install express --save
npm install express-generator -g
```

- Inisiasi project Express

```
express --ejs mySite
```

- Pergi ke direktori project Express

```
cd mySite
```

- Install dependensi

```
npm install
```

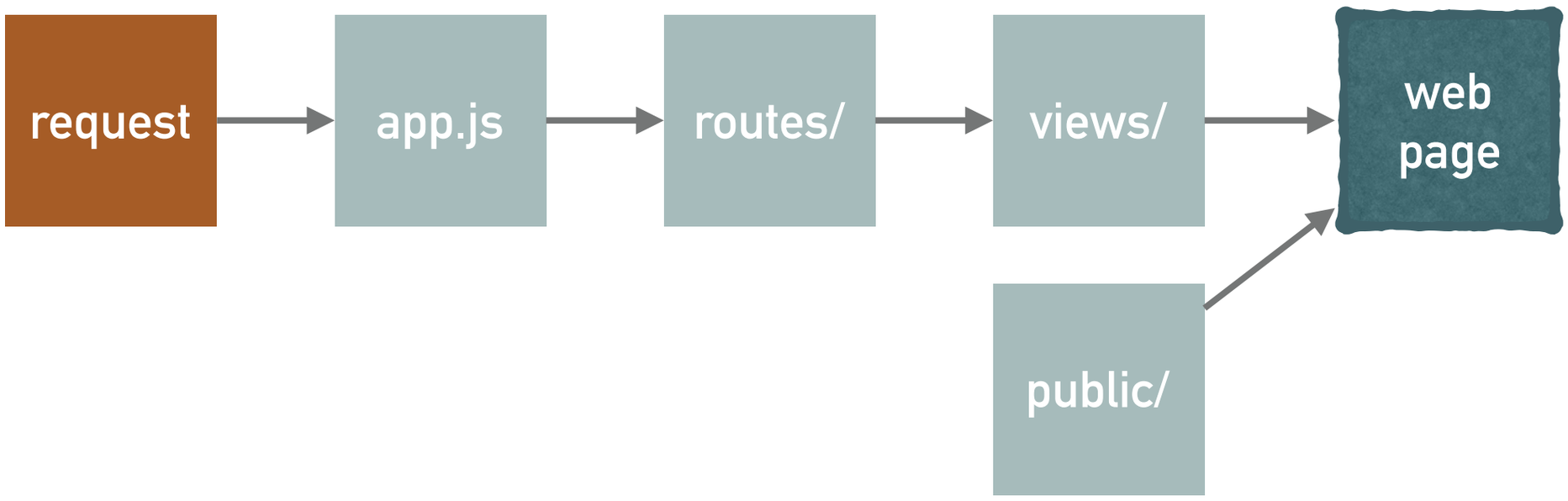
- Jalankan project Express

```
npm start
```

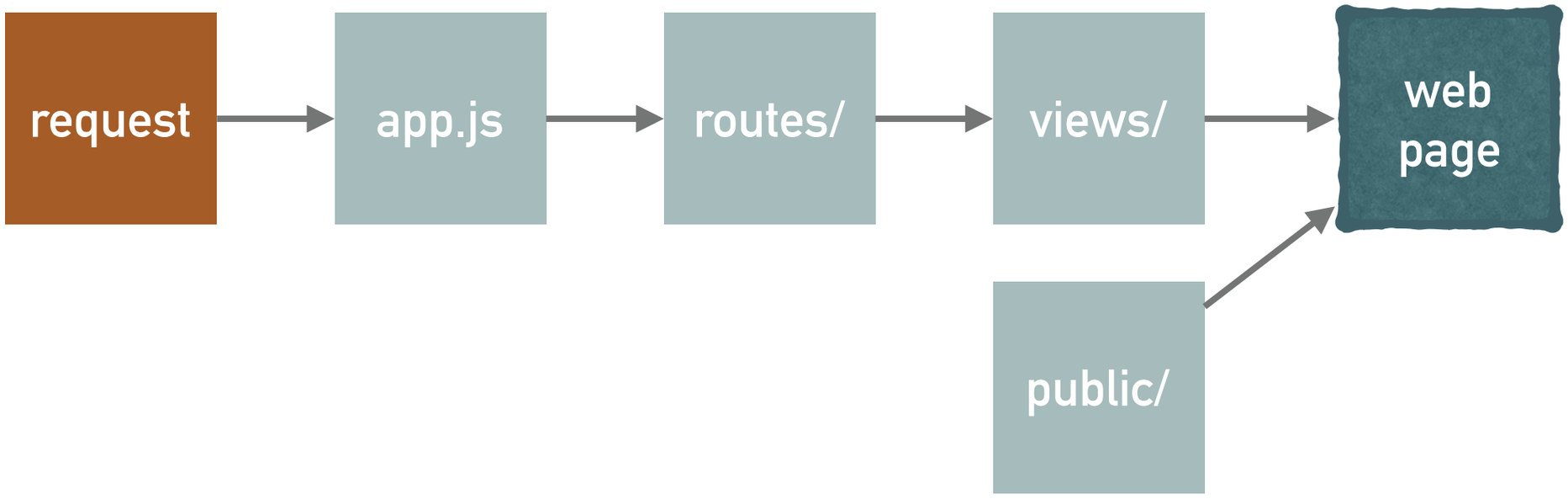
KOMPONEN PENTING EXPRESS

- app.js
  - File induk yang mengatur keseluruhan aplikasi
- routes/
  - folder berisi semua routing dalam aplikasi
- views/
  - folder berisi template halaman/komponen halaman
- public/
  - berisi file statis yang digunakan oleh aplikasi

ALUR DEFAULT EXPRESS



ALUR DEFAULT EXPRESS



- app.js mengatur agar setiap request, ditangani oleh request handler tertentu, termasuk yang didefinisikan dalam routes.
- routes akan mengatur views apa yang harus ditampilkan oleh request handler tertentu.
- views akan mengumpulkan data + file static yang akan direnderdi halaman web

TEMPLATE VIEWS

- Views berisi template yang akan direnderer menjadi file HTML saat keseluruhan app dibuild
- Kita menggunakan template ejs
- Dalam ejs, semua variabel ditulis <%= variabel %>
- Selain itu, di dalam tag template, kita bisa menulis pernyataan JS <% if(number) { var x = 10; } %>
- Di luar itu, file ditulis sebagaimana menulis HTML biasa, dengan tag HTML biasa pula

TEMPLATE VIEWS

- Selain Ejs, ada template lain yang bisa digunakan Express:
  - Jade (default, namun deprecated)
  - Pug
  - Mustache
  - Handlebars
  - etc.
- Template mana yang digunakan, bisa diset saat membuat project
- Kita cukup memilih 1 template favorit

LATIHAN 1

- Ubah agar halaman default selalu menampilkan 1 angka acak. Pesannya adalah “Hello, (angka_acak)”
- Ubah agar halaman /users berisi 5 nama pahlawan nasional

MEMBUAT APLIKASI DENGAN EXPRESS

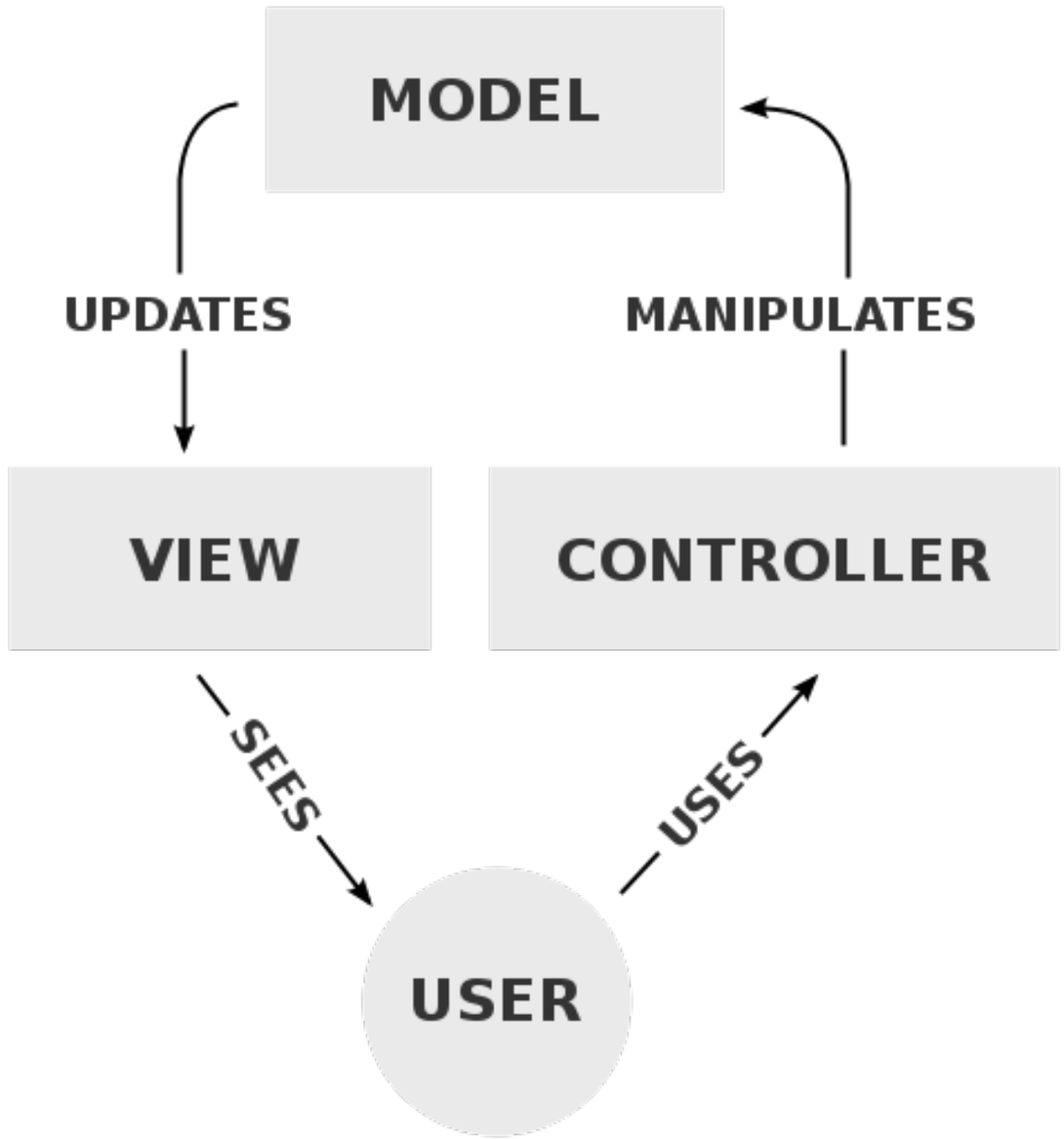
- Mari menggunakan Express untuk membuat sebuah aplikasi web sederhana
- Aplikasi ini adalah sebuah pencatat sederhana
- Tujuannya adalah:
  - Mempelajari arsitektur MVC di Express
  - Menunjukkan bagaimana melakukan operasi CRUD sederhana di Express
  - Lebih memahami cara kerja Express

ARSITEKTUR MVC

- Sebuah arsitektur yang banyak digunakan untuk aplikasi yang menggunakan GUI
- Memisahkan komponen-komponen yang memiliki tanggung jawab masing-masing, se-independen mungkin
- Tujuan penggunaannya adalah membuat aplikasi yang lebih modular dan rapi

ARSITEKTUR MVC

- Model: menyimpan data serta logic utama aplikasi
- View: menampilkan informasi yang berasal dari model(data)
- Controller: menerima input dari user dan mengubahnya menjadi perintah untuk model



ARSITEKTUR MVC DI EXPRESS

- Sejauh ini, kita telah melihat implementasi View dan Controller di aplikasi Express default:
  - *View*: diwakili oleh isi dari folder views/
  - *Controller*: diwakili oleh folder routes/ di mana ia menangani setiap request yang masuk dan langsung mengarahkan view yang berkaitan
- Kita perlu mengimplementasikan sendiri representasi ***Model*** di aplikasi kita

ARSITEKTUR MVC DI EXPRESS

- *Model:*
  - Menyimpan data aplikasi
  - Mengubah data sesuai input dari controller
  - Memberikan data untuk view
  - Berada di modul terpisah dari controller

ARSITEKTUR MVC DI EXPRESS

- *Model:*
  - Disimpan dalam folder yang kita buat sendiri, bernama “models” dan setingkat dengan routes dan views
  - Memiliki API yang memberikan fungsi untuk melakukan operasi CRUD (Create, Read, Update, Destroy) pada data
  - Kite implementasikan dengan membuat fungsi create, read, update dan destroy dalam model

MEMBUAT APLIKASI

- Buat aplikasi baru dengan express, beri nama “notes”
- Ikuti petunjuk sebelumnya

```
express --ejs notes
cd notes
npm install
```

MEMBUAT MODEL

- Buat direktori bernama models
- Buat file bernama notes.js dalam direktori tersebut

```
var notes = [];
exports.update = exports.create = function(key, title, body) {
    notes[key] = {
        title: title,
        body: body
    };
}
exports.read = function(key) {
    return notes[key];
}
exports.destroy = function(key) {
    delete notes[key];
}
exports.keys = function() {
    return Object.keys(notes);
}
```

MEMBUAT MODEL

- Ini adalah model temporer
- Setiap note memiliki key untuk identifikasi

```
var notes = [];
exports.update = exports.create = function(key, title, body) {
    notes[key] = {
        title: title,
        body: body
    };
}
exports.read = function(key) {
    return notes[key];
}
exports.destroy = function(key) {
    delete notes[key];
}
exports.keys = function() {
    return Object.keys(notes);
}
```

MENAMPILKAN HALAMAN DEPAN APLIKASI

- Di dalam app.js, impor model notes sebagai modul

```
var notes = require (‘./routes/notes’);
```

- Ubah routes/index.js

```
var notes = require('../models/notes');

/* GET home page. */
router.get('/', function(req, res, next) {
    res.render('index', {
        title: 'Express',
        notes: notes
    });
});
```

MENAMPILKAN HALAMAN DEPAN APLIKASI

- Ubah views/index.ejs menjadi:

```
<% include top %>
<%
var keys = notes.keys();
if (keys) {
     keys.forEach(function(key) {
       var note = notes.read(key);
       %><p><%= key %>:
       <a href="/noteview?key=<%= key %>"><%= note.title %></a>
       </p><%
   });
} %>
<% include bottom %>
```

- Tambahkan file routes/notes.js

```
var notes = require('../models/notes');
```

MENAMPILKAN HALAMAN DEPAN APLIKASI

- Buat file views/top.ejs

```
<html>
    <head>
        <title><%= title %></title>
        <link rel='stylesheet' href='/stylesheets/style.css' />
    </head>
    <body>
        <h1><%= title %></h1>
        <div class='navbar'>
        <p><a href='/'>Home</a> | <a href='/noteadd'>ADD Note</a></p>
        </div>
```

- Buat file views/bottom.ejs

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

MENAMPILKAN HALAMAN DEPAN APLIKASI

- Kini halaman depan bisa ditampilkan

```
npm start
```

MENAMBAHKAN NOTE BARU (CREATE)

- Tambahkan di app.js

```
app.get(‘/noteadd’, notes.add);
```

- Ubah routes/notes.js

```
var notes = require(‘../models/notes');

exports.add = function(req, res, next) {
  res.render('noteedit', {
     title: "Add a Note",
     docreate: true,
     notekey: "",
        note: undefined
   });
}
```

MENAMBAHKAN NOTE BARU

- Tambahkan file views/noteedit.ejs

```
<% include top %>
<form method='POST' action=‘/notesave'>

<input type='hidden' name='docreate'
   value='<%=docreate ? "create" : “update"%>'>

<p>Key: <input type='text' name='notekey'
   value='<%= note ? notekey : "" %>’></p>

<p>Title: <input type='text' name='title'
   value='<%= note ? note.title : "" %>' /></p>

<br/><textarea rows=5 cols=40 name='body' ><%=
   note ? note.body : "" %></textarea>

<br/><input type='submit' value='Submit' />
</form>
<% include bottom %>
```

MENAMBAHKAN NOTE BARU

- Tambahkan di app.js

```
app.get('/notesave', notes.save);
```

- Tambahkan di routes/notes.js

```
exports.save = function(req, res, next) {
    if (req.body.docreate === "create") {
        notes.create(req.body.notekey,
            req.body.title, req.body.body);
    } else {
        notes.update(req.body.notekey,
            req.body.title, req.body.body);
    }
    res.redirect('/noteview?key='+req.body.notekey);
}}
```

- Kini aplikasi bisa dijalankan

MEMBACA NOTE (READ)

- Tambahkan di app.js

```
app.get(‘/noteview’, notes.view);
```

- Tambahkan di routes/notes.js

```
exports.view = function(req, res, next) {
  var note = {};
  if (req.query.key) {
      note = notes.read(req.query.key);
  }
  res.render('noteview', {
      title: note ? note.title : "",
      notekey: req.query.key,
      note: note
  });
}
```

MEMBACA NOTE (READ)

- Tambahkan views/noteview.ejs

```
<% include top %>
<p><%= note ? note.body : "" %></p>
<p>Key: <%= notekey %></p>
<% if (notekey) { %>
    <hr/>
    <p><a href="/notedestroy?key=<%= notekey %>">Delete</a>
    | <a href="/noteedit?key=<%= notekey %>">Edit</a></p>
<% } %>
<% include bottom %>
```

- Kini aplikasi bisa dijalankan

MENGUPDATE NOTE (UPDATE)

- Tambahkan di app.js

```
app.get('/noteedit', notes.edit);
```

- Tambahkan di routes/notes.js

```
exports.edit = function(req, res, next) {
    var note = undefined;
    if (req.query.key) {
        note = notes.read(req.query.key);
    }
    res.render('noteedit', {
        title: note ? ("Edit " + note.title) : "Add a Note",
        docreate: note ? false : true,
        notekey: req.query.key,
        note: note
    });
}
```

MENGHAPUS NOTE (DELETE)

- Note tidak akan langsung terhapus, tapi diberikan dulu halaman konfirmasi sebelum penghapusan
- Tambahkan di app.js

```
app.get('./notedestroy', notes.destroy);
```

- Tambahkan di routes/notes.js

```
exports.destroy = function(req, res, next) {
    var note = undefined;
    if (req.query.key) {
        note = notes.read(req.query.key);
    }
    res.render('notedestroy', {
        title: note ? note.title : "",
        notekey: req.query.key,
        note: note
    });
}
```

MENGHAPUS NOTE (DELETE)

- Tambahkan views/notedestroy.ejs

```
<% include top %>
<form method='POST' action='/notedodestroy'>
<input type='hidden' name='notekey'value='<%= note ? notekey : ""
%>'>
<p>Delete <%= note.title %> ?</p>
<br/><input type='submit' value='DELETE' />
    <a href="/noteview?key=<%= notekey %>">Cancel</a>
</form>
<% include bottom %>
```

MENGHAPUS NOTE (DELETE)

- Tambahkan di app.js

```
app.get('/notedodestroy', notes.dodestroy);
```

- Tambahkan di routes/notes.js

```
exports.dodestroy = function(req, res, next) {
    notes.destroy(req.body.notekey);
    res.redirect('/');
}
```

- App sudah selesai dan bisa digunakan :)

LATIHAN 2

- Tambahkan 1 field lagi di note yang berisi tanggal
- Kalau kesulitan, field ini bisa diisi sendiri oleh user, tanpa perlu datepicker sendiri

TAMBAHAN

- File CSS bisa digunakan untuk mengubah tampilan aplikasi
- File ini berada di public/stylesheets/style.css
- Dideklarasikan di bagian <head> di file top.ejs