Daftar Isi

NodeJS

- **Documentation**
  https://www.npmjs.com/

- **Core Modules**
  https://nodejs.org/dist/latest-v13.x/docs/api/

- **NPM**
  - $ npm init : inisialisasi
  - $ npm i <package> : instal package
  - $ npm instal <package>@<versi> : instal package dengan versi pilihan
  - $ npm unistall <package> : unistal package

- **Nodemon**
  - instal lokal :
    $ npm install --save-dev nodemon@2.0.7

  - konfigurasi di package.json : menambahkan scripts “start” : “nodemon app”

```
"scripts": {
  "start" : "nodemon app.js",
  "test": "echo \"Error: no test specified\" && exit 1"
}
```

  - running server :
    $ npm start

ExpressJS

- **Documentation**

  NPM : https://www.npmjs.com/
  ExpressJS : https://expressjs.com/
  Express-validator : https://express-validator.github.io/docs/

- **Installing**

  1. $ mkdir myapp : membuat folder project
  2. $ cd myapp : masuk kedalam folder project
  3. $ npm init : inisialisasi folder
  4. $ npm i express : memasang framwork express
  5. $ npm i ejs : memasang view engine (ejs)
  6. $ npm i –g nodemon : memasang package nodemon

- **Starter**

```
const express = require('express')

const app = express()
const port = 3000

// Konfigurasi / setup
app.set('view engine', 'ejs'); // EJS
app.use(express.static('public')); // Built-in middleware
app.use(express.urlencoded({ extended: true })); // Built-in middleware, untuk proses tambah data contact

app.get('/', (req, res) => {
 res.render('index', { title: 'Halaman Home', nama: 'Moch Ihsan Saepulloh' });
});

app.listen(port, () => {
 console.log(`Example app listening on port ${port}`)
})
```

- **Templating Engine**
  - Cara 1 :
    1. buat folder layouts, berisikan file (tidak perlu seperti yield):
       - header.ejs
       - navbar.ejs
       - footer.ejs

    2. penggunaan di views :

```
<%- include('layouts/header') %>
<%- include('layouts/navbar') %>

<h1>Halaman Contact</h1>

<%- include('layouts/footer') %>
```

  - Cara 2 : (withPackage)
    1. $ npm install express-ejs-layouts : memasang package layouts
    2. konfigurasi dan setup di app.js :

```
const express = require('express');
const expressLayouts = require('express-ejs-layouts'); // Templating Layouts
const app = express();
const port = 3000;

// EJS
app.set('view engine', 'ejs');
// Templating Layouts
app.use(expressLayouts);

// Routing
app.get('/', (req, res) => {
  res.render('index', { nama: 'Moch Ihsan Saepulloh', title: 'Halaman Home' });
})

// Menggunakan templating layouts (package)
app.get('/about', (req, res) => {
  res.render('about', {
    title: 'Halaman About',
    layout: 'layouts/main-layout'
  });
})
```

    3. penggunakan di views (tidak perlu include) :

```
<h1>Halaman About</h1>
```

- **Validation**
  - installing package :
    1. $ npm i express-validator

  - require di app.js :

```
const { body, validationResult, check } = require('express-validator');
```

  - penggunaan route app.js :

```
// proses tambah data contact (insert)
app.post('/contact', [
 // rules validation
 body('nama').custom((value) => {
   const duplikat = cekDuplikat(value);
   if(duplikat) {
     throw new Error('Nama contact sudah digunakan.');
   }

   return true;
 }),
 check('email', 'Email tidak valid.').isEmail(),
 check('nohp', 'No HP tidak valid.').isMobilePhone('id-ID'),
], (req, res) => {
 const errors = validationResult(req);
 if(!errors.isEmpty()) {
   res.render('add-contact', {
     title: 'Form Tambah Contact',
     errors: errors.array()
   });
 } else {
   addContact(req.body);
   req.flash('msg', 'Data contact berhasil ditambahkan.');
   res.redirect('/contact');
 }
})
```

- khusus cekDuplikat di contacts.js :
  1. membuat function :

```
// cek duplikat
const cekDuplikat = (nama) => {
  const contacts = loadContact();
  return contacts.find((contact) => contact.nama === nama);
}

module.exports = { loadContact, findContact, addContact, cekDuplikat };
```

  2. load modules :

```
module.exports = { loadContact, findContact, addContact, cekDuplikat };
```

  3. require modules di app.js :

```
const { loadContact, findContact, addContact, cekDuplikat } = require('./utils/contacts');
```

- menampilkan pesan error di view :

```
<% if (typeof errors != 'undefined') { %>
<div class="alert alert-danger" role="alert">
  <ul>
    <% errors.forEach(error => { %>
      <li><%= error.msg %></li>
    <% }) %>
  </ul>
</div>
<% } %>
```

- **Flash Message**
  - installing package :
    1. $ npm i express-session
    2. $ npm i cookie-parse
    3. $ npm i connect-flash

  - require di app.js :

```
// Flash Message
const session = require('express-session');
const cookieParser = require('cookie-parser');
const flash = require('connect-flash');
```

  - konfigurasi di app.js :

```
// Konfigurasi Flash Message
app.use(cookieParser('secret'));
app.use(session({
  cookie: { maxAge: 6000 },
  secret: 'secret',
  resave: true,
  saveUninitialized: true
}));
app.use(flash());
```

  - jadikan di app.js untuk dikirim ke view :

```
// halaman contact (getAll)
app.get('/contact', (req, res) => {
  const contacts = loadContact();
  res.render('contact', {
    title: 'Halaman Contact',
    contacts,
    msg: req.flash('msg') // Flash Messange
  });
})
```

  - penggunaan di view :

```
<% if (msg.length !== 0) { %>
<div class="alert alert-success" role="alert"><%= msg %> </div>
<% } %>
```

MongoDB

- **Documentation**
  https://mongoosejs.com/

- **Installing**
  1. $ npm i mongoose : Memasang package database mongodb
  2. $ npm i method-override : Memasang method override (method bajakan)

- **Setup**
  1. Membuat folder dan file : utils/db.js

```
const methodOverride = require('method-override'); // Method Override

const mongoose = require('mongoose');
mongoose.connect('mongodb://127.0.0.1:27017/<namaDatabase>', {
  useNewUrlParser: true,
  useUnifiedTopology: true,
  useCreateIndex: true
});
```

  2. Require & setup model Contact di app.js

```
// mongoose
require('./utils/db');
const Contact = require('./model/contact');

// setup method override
app.use(methodOverride('_method'));
```

  3. Membuat folder dan file : public/contact.js

```
const mongoose = require('mongoose');

// Membuat schema / collection
const Contact = mongoose.model('Contact', {
  nama: {
    type: String,
    required: true,
  },
  nohp: {
    type: String,
    required: true,
  },
  email: {
    type: String,
  },
});

module.exports = Contact;
```

- **Create**

1. Routing di app

```
// halaman form tambah contact (insert)
app.get('/contact/add', (req, res) => {
  const contacts = loadContact();
  res.render('add-contact', {
    title: 'Form Tambah Contact',
  });
});

// proses tambah data contact (insert)
app.post('/contact', [
  // rules validation
  body('nama').custom(async (value) => {
    const duplikat = await Contact.findOne({ nama: value });
    if(duplikat) {
      throw new Error('Nama contact sudah digunakan.');
    }

    return true;
  }),
  check('email', 'Email tidak valid.').isEmail(),
  check('nohp', 'No HP tidak valid.').isMobilePhone('id-ID'),
], (req, res) => {
  const errors = validationResult(req);
  if(!errors.isEmpty()) {
    res.render('add-contact', {
      title: 'Form Tambah Contact',
      errors: errors.array()
    });
  } else {
    Contact.insertMany(req.body, (error, result) => {
      req.flash('msg', 'Data contact berhasil ditambahkan.');
      res.redirect('/contact');
    });
  }
});
```

2. Penggunaan di view

```
<h2>Form Tambah Data Contact</h2>

<% if (typeof errors != 'undefined') { %>
<div class="alert alert-danger" role="alert">
  <ul>
    <% errors.forEach(error => { %>
      <li><%= error.msg %></li>
    <% }) %>
  </ul>
</div>
<% } %>

<form method="post" action="/contact">
  <div class="mb-3">
    <label for="nama" class="form-label">Nama</label>
    <input type="text" class="form-control" id="nama" name="nama" required>
  </div>
  <div class="mb-3">
    <label for="email" class="form-label">Email</label>
    <input type="email" class="form-control" id="email" name="email" required>
  </div>
  <div class="mb-3">
    <label for="nohp" class="form-label">No HP</label>
    <input type="text" class="form-control" id="nohp" name="nohp" required>
  </div>
  <button type="submit" class="btn btn-primary">Tambah Data</button>
</form>
```

- **Read (getAll)**

1. Routing di app.js

```
// halaman contact (getAll)
app.get('/contact', async (req, res) => {
 const contacts = await Contact.find();

 res.render('contact', {
  title: 'Halaman Contact',
  contacts,
  msg: req.flash('msg') // Flash Messange
 });
});
```

2. Penggunaan di view

```
<% if (contacts.length === 0) { %>
  <tr>
    <td colspan="4">
      <div class="alert alert-danger fw-bold" role="alert">
        Data contact masih kosong.
      </div>
    </td>
  </tr>
<% } %>

<% no=1 %>
<% contacts.forEach(contact => { %>
<tr>
  <th scope="row"><%= no++ %> </th>
  <td><%= contact.nama %></td>
  <td><%= contact.nohp %></td>
  <td>
    <a href="/contact/<%= contact.nama %>">
      <i class="bi bi-info-circle"></i>
    </a>
    <a href="/contact/edit/<%= contact.nama %>">
      <i class="bi bi-pencil"></i>
    </a>
    <form action="/contact?_method=DELETE" method="post">
      <input type="hidden" name="nama" value="<%= contact.nama %>">
      <button type="submit" onclick="return confirm('yakin?')">
        <i class="bi bi-trash"></i>
      </button>
    </form>
  </td>
</tr>
<% }) %>
```

- **Read (getById)**

1. Routing di app.js

```
// halaman detail contact (getById)
app.get('/contact/:nama', async (req, res) => {
  const contact = await Contact.findOne({ nama: req.params.nama });

  res.render('detail', {
    title: 'Halaman Detail Contact',
    contact,
  });
});
```

2. Penggunaan di view

```
<% if (!contact) { %>
<div class="alert alert-danger fw-bold" role="alert">
  Data contact tidak tersedia.
  <br>
  <a href="/contact" class="alert-link fw-normal">kembali</a>
</div>
<% } else { %>
<div class="card" style="width: 18rem;">
  <div class="card-body">
    <h5 class="card-title"><%= contact.nama %></h5>
    <h6 class="card-subtitle mb-2 text-muted"><%= contact.nohp %></h6>
    <p class="card-text"><%= contact.email %></p>
    <a href="/contact" class="card-link">kembali</a>
  </div>
</div>
<% } %>
```

- **Update**

1. Routing di app.js

```
// halaman form ubah contact (update)
app.get('/contact/edit/:nama', async (req, res) => {
  const contact = await Contact.findOne({ nama: req.params.nama });

  res.render('edit-contact', {
    title: 'Form Ubah Contact',
    contact
  });
});

// proses ubah data contact (update)
app.put('/contact', [
  // rules validation
  body('nama').custom(async (value, { req }) => {
    const duplikat = await Contact.findOne({ nama: value });
    if(duplikat && value !== req.body.oldNama) {
      throw new Error('Nama contact sudah digunakan.');
    }

    return true;
  }),
  check('email', 'Email tidak valid.').isEmail(),
  check('nohp', 'No HP tidak valid.').isMobilePhone('id-ID'),
], (req, res) => {
  const errors = validationResult(req);
  if(!errors.isEmpty()) {
    res.render('edit-contact', {
      title: 'Form Ubah Contact',
      errors: errors.array(),
      contact: req.body
    });
  } else {
    Contact.updateOne(
      { _id: req.body._id },
      {
        $set: {
          nama: req.body.nama,
          nohp: req.body.nohp,
          email: req.body.email,
        }
      }
    ).then((result) => {
      req.flash('msg', 'Data contact berhasil diubah.');
      res.redirect('/contact');
    });
  }
});
```

2. Penggunaan di view

```
<h2>Form Ubah Data Contact</h2>

<% if (typeof errors != 'undefined') { %>
<div class="alert alert-danger" role="alert">
  <ul>
    <% errors.forEach(error => { %>
      <li><%= error.msg %></li>
    <% }) %>
  </ul>
</div>
<% } %>

<form method="post" action="/contact?_method=PUT">
  <input type="hidden" name="_id" value="<%= contact._id %>">
  <input type="hidden" name="oldNama" value="<%= contact.oldNama || contact.nama %>">
  <div class="mb-3">
    <label for="nama" class="form-label">Nama</label>
    <input type="text" class="form-control" id="nama" name="nama" value="<%= contact.nama %>"
required>
  </div>
  <div class="mb-3">
    <label for="email" class="form-label">Email</label>
    <input type="email" class="form-control" id="email" name="email" value="<%= contact.email %>"
required>
  </div>
  <div class="mb-3">
    <label for="nohp" class="form-label">No HP</label>
    <input type="text" class="form-control" id="nohp" name="nohp" value="<%= contact.nohp %>"
required>
  </div>
  <button type="submit" class="btn btn-primary">Ubah Data</button>
</form>
```

- **Delete**

1. Routing di app.js

```
// proses hapus contact (delete)
app.delete('/contact', (req, res) => {
 Contact.deleteOne({ nama: req.body.nama }).then((result) => {
  req.flash('msg', 'Data contact berhasil dihapus.');
  res.redirect('/contact');
 });
});
```

2. Penggunaan di view

```
<!-- DETAIL -->
<a href="/contact/<%= contact.nama %>" class="btn btn-success badge rounded-pills">
  <i class="bi bi-info-circle"></i>
</a>
<!-- UPDATE -->
<a href="/contact/edit/<%= contact.nama %>" class="btn btn-warning badge rounded-pills">
  <i class="bi bi-pencil"></i>
</a>
<!-- DELETE -->
<form action="/contact?_method=DELETE" method="post" class="d-inline">
  <input type="hidden" name="nama" value="<%= contact.nama %>">
  <button type="submit" class="btn btn-danger badge rounded-pills" onclick="return confirm('yakin?')">
    <i class="bi bi-trash"></i>
  </button>
</form>
```