Belajar Express JS

SEPTEMBER 20, 2015

Reading time ~10 minutes

Pada tutorial sebelumnya penulis telah membahas bagaimana cara instalasi untuk coding node js. Kali ini kita akan belajar bagaimana membagun sebuah aplikasi berbasis node js.

Berikut merupakan tahapan untuk membuat sebuah aplikasi dengan node js :

- Initialisasi Project Node JS
- Initialisasi dependency Library dengan bower
- Membuat model
- Membuat Router
- Membuat view
- Uji Coba

Initialisasi Project Node JS

Kita mulai dengan membuat sebuah folder aplikasi, disini penulis memberi nama folder dengan `Belajar-ExpressJS` kemudian masuk ke folder dengan menggunakan terminal. Jalankan perintah `npm init` kemudian masukkan inputan sesuai dengan anda inginkan. Jika berhasil maka akan dibuatkan sebuah file yaitu `package.json` sebagai konfigurasi project node js. Langkah selanjutnya adalah kita memerlukan banyak dependency library diantaranya seperti express, mongoose, jade dan lain - lain. Untuk melakukan instalasi dependency tersebut jalankan perintah berikut :

```
npm install body-parser cookie-parser csurf errorhandler express express-session jade method-override m
```

Berikut merupakan list dari library diatas :

- body-parser : digunakan untuk parsing sebuah url dan json
- cookie-parser : digunakan untuk konfigurasi cookie

- csurf : digunakan untuk konfigurasi CSRF(Cross Site Request Forgery), untuk pembahasan CSRF akan dibahas pada tutorial selanjutnya :)
- errorhandler : digunakan untuk handle error pada node js
- express : merupakan framework node js untuk membangun sebuah aplikasi web
- express-session : digunakan sebagai session pada express
- jade : digunakan sebagai template engine pada node js, sebenarnya masih ada banyak template engine lain seperti EJS akan tetapi kita akan menggunakan template jade dikarenakan jade termasuk template yang bagus terutama support terhadap extends template
- method-override : biasnya digunakan untuk membangun protokol HTTP method
- mongoose : sebagai framework ODM (Object Document Mapping) yang akan melakukan akses ke mongoDB. MongoDB adalah database yang berbasis NoSQL, silahkan install MongoDB berikut dokumentasi cara instalasi MongoDB
- morgan : digunakan sebagai log aplikasi
- node-uuid : digunakan untuk generate otomatis UUID, UUID merupakan key yang unik, biasanya digunakan untuk generate ID pada sebuah database yang bersifat primary key
- serve-favicon : digunakan untuk favicon atau icon sebuah web
- winston : digunakan sebagai library log aplikasi

Initialisasi dependency Library dengan bower

Setelah konfigurasi project node js, selanjutnya kita ingin download dependency library untuk front end. Jalanakn perintah `bower init` lalu masukkan inputan lagi dan akan digenerate sebuah file `bower.json` yang merupakan file konfigurasi bower. Jalankan perintah berikut untuk instalasi dependency library bootstrap dan jquery :

```
bower install jquery bootstrap --save
```

Membuat model

Framework express mendukung pattern MVC (model view controller) sama halnya seperti framework lain seperti Ruby On Rails dan Spring Framework. Buat sebuah folder models lalu buat sebuah file `Pegawai.js`. Berikut merupakan codingan dari file `Pegawai.js`.

```
(function() {
  'use strict';

  var pegawai,
    mongoose = require('mongoose'),
    Schema = mongoose.Schema;

  pegawai = new Schema({
    idPegawai: {
      type: 'String',
      required: true
    },
    namaPegawai: {
      type: 'String',
      required: true
    },
    alamat: {
      type: 'String'
    },
    tanggalLahir: {
      type: 'Date',
      required: true
    }
  }, {
    collection: 'tb_pegawai'
  });

  module.exports = mongoose.model('Pegawai', pegawai);

}).call(this);
```

Berikut penjelasan singkat tentang codingan diatas :

- `var pegawai, mongoose;` merupakan deklarasi dari masing - masing variabel. variabel pegawai merupakan variabel kosong, variabel monggose merupakan variabel dengan type data mongoose dan variabel Schema merupakan variabel dari Schema mongoose. Tidak seperti bahasa pemrograman lain, javascript dapat langsung mendeklarasikan variabelnya tanpa type data. require('monggose') artinya kita melakukan import dari library monggose, ini sama halnya ketika kita melakukan import pada bahasa pemrograman java.
- codingan selanjutnya merupakan pendeklarasian dari pada column - column yang ada pada sebuah database. Konsep ODM dan ORM memiliki kemiripan akan tetapi terdapat perbedaan yaitu pada ODM terdapat document dan tidak ada relasi sedangkan pada ORM terdapat table dan terdapat relasi.
- collection : merupakan definisi dari pada collection pada mongodb, jika pada database RDBMS (SQL) kita mengenalnya dengan table. Silahkan pelajari lebih lanjut mengenai database MongoDB :).

- module.exports : merupakan kodingan untuk melakukan export variabel pegawai, sehingga variabel pegawai dapat diakses oleh file javascript yang lainnya.

Membuat Router

Router merupakan sebuah controller, hanya saja dari penamaannya saja. Developer express lebih suka menyebutkan router sehingga disini penulis menggunakan dengan nama router. Sebelum kita membuat router, penulis membuat sebuah konfigurasi untuk log sebuah aplikasi. Buat 3 folder dengan nama `routes` , `utils` dan `logs` dan buat sebuah file dengan nama `logger.js` di dalam folder `utils` . Berikut adalah kodingan dari `logger.js` .

```
var winston = require('winston');
winston.emitErrs = true;

var logger = new winston.Logger({
  transports: [
    new winston.transports.File({
      level: 'debug',
      filename: './logs/logs-aplikasi.log',
      handleExceptions: true,
      json: true,
      maxsize: 5242880,
      maxFiles: 5,
      colorize: false
    }),
    new winston.transports.Console({
      level: 'debug',
      handleExceptions: true,
      json: false,
      colorize: true
    })
  ],
  exitOnError: false
});

module.exports = logger;
module.exports.stream = {
  write: function(message, encoding) {
    logger.info(message);
  }
};
```

Untuk codingan diatas akan dibahas pada tutorial selanjutnya mengenai log sebuah aplikasi :). Oke selanjutnya buat sebuah file `PegawaiRoute.js` di dalam folder `routes` . Berikut codingannya.

```
(function() {
  'use strict';

  var express = require('express'),
    Pegawai = require('../models/Pegawai'),
    logger = require('../utils/logger'),
    csrf = require('csurf'),
    router = express.Router(),
    csrfProtection = csrf({
      cookie: true
    }),
    uuid = require('node-uuid');

  router.get('/', function(req, res) {
    return Pegawai.find(function(err, pegawais) {

      if (err) {
        logger.error('error bung ', err);
        return res.render('500');
      }

      logger.debug('Berhasil ngambil data bung ', pegawais);
      return res.render('index', {
        pegawais: pegawais
      });

    });
  });

  router.get('/tambah/pegawai', csrfProtection, function(req, res) {
    logger.info('render page tambah');
    return res.render('tambah', {
      csrfToken: req.csrfToken()
    });
  });

  router.post('/save/pegawai', csrfProtection, function(req, res) {
    var pegawai = new Pegawai({
      idPegawai: uuid.v4(),
      namaPegawai: req.body.namaPegawai,
      alamat: req.body.alamat,
      tanggalLahir: req.body.tanggalLahir
    });

    return pegawai.save(function(err, pegawai) {
      if (err) {
        logger.error('error bung ', err);
        return res.render('500');
      }

      logger.debug('data tersimpan bung ', pegawai);
      logger.info('render page awal');
      return res.redirect('/');
    });
```

```
});

router.get('/edit/pegawai/:idPegawai', csrfProtection, function(req, res) {
  logger.info('edit pegawai');

  var idPegawai = req.params.idPegawai;

  Pegawai.findOne({
    idPegawai: idPegawai
  }, function(err, pegawai) {

    if (err) {
      logger.error('error bung ', err);
      return res.render('500');
    }

    logger.debug('data tersedia ', pegawai);
    logger.info('render page edit');
    return res.render('edit', {
      pegawai: pegawai,
      csrfToken: req.csrfToken()
    });
  });
});

router.post('/update/pegawai/:idPegawai', csrfProtection, function(req, res) {

  var idPegawai = req.params.idPegawai;

  Pegawai.findOne({
    idPegawai: idPegawai
  }, function(err, pegawai) {

    if (err) {
      logger.error('error bung ', err);
      return res.render('500');
    }

    pegawai.namaPegawai = req.body.namaPegawai;
    pegawai.alamat = req.body.alamat;
    pegawai.tanggalLahir = req.body.tanggalLahir;
    pegawai.save();

    logger.debug('data tersimpan bung ', pegawai);
    logger.info('render page awal');

    res.redirect('/');
  });
});

router.get('/delete/pegawai/:idPegawai', function(req, res) {
  Pegawai.remove({
    idPegawai: req.params.idPegawai
```

```
    }, function(err) {
      if (err) {
        logger.error('error bung ', err);
        return res.render('500');
      }

      logger.debug('data dihapus');
      logger.info('render page awal');
      res.redirect('/');
    });
  });

  module.exports = router;

}).call(this);
```

Berikut merupakan penjelasan singkat :

- `var express = require('express')` merupakan codingan untuk melakukan import file javascript.
- dimulai dari `router.get('/', function(req, res)` merupakan sebuah method untuk mengambil data lalu data tersebut di lempar ke view sekaligus melakukan render terhadap sebuah page jade. Disana terdapat kodingan `Pegawai.find` yang berfungsi untuk mengambil semua data pegawai, Lalu bagaimana melempar ke view ? pada bagian `res.render('index', {pegawais: pegawais})` merupakan aksi untuk melakukan render dengan nama page `index` lalu mengirim data dengan parameter `pegawais`.
- `router.get('/tambah/pegawai')` merupakan kodingan untuk menampilkan halaman tambah data pegawai, disini penulis menggunakan token CSRF :).
- `router.post('/save/pegawai')` merupakan method untuk menyimpan data pegawai. Untuk menyimpan, kita menggunakan Model yaitu Pegawai akan tetapi kita diharuskan membuat object baru, dapat dilihat di `var pegawai = new Pegawai`. Untuk idPegawai kita membuatnya dengan menggunakan bantuan `node-uuid` sedangkan data lain berdasarkan dari pada form, untuk mengambil data pada form kita menggunakan `req.body.namaTextField`. Jika data tersimpan maka kita akan melakukan redirect ke page awal.
- method selanjutnya adalah `router.get('/edit/pegawai/:idPegawai'` yang berfungsi untuk melalukan render halaman edit, sebelum melakukan render, kita mengambil data terlebih dahulu berdasarkan idPegawai dapat diliha pada codingan `router.get('/edit/pegawai/:idPegawai'` lalu melakukan render ke page edit.
- `router.post('/update/pegawai/:idPegawai'` sama seperti method save sebelumya bedanya adalah, disini kita melakukan query terlebih dahulu terhadap Pegawai kemudian kita melakukan update data, dapat dilihat pada codingan `router.post('/update/pegawai/:idPegawai')`.

- `router.get('/delete/pegawai/:idPegawai)'` merupakan method yang terakhir untuk menghapus sebuah data. Untuk menghapus data dapat dilihat pada `Pegawai.remove({})`.

Membuat view

Tahap selanjutnya adalah membuat view, silahkan buat folder `views` kemudian buat 5 file di dalam folder tersebut yaitu `layout.jade`, `index.jade`, `tambah.jade`, `edit.jade` dan `500.jade`. Berikut adalah kodingan untuk `layout.jade`.

```
doctype html
html(lang='en')
    head
      meta(charset='utf-8')
      meta(name='viewport', content='width=device-width, initial-scale=1, maximum-scale=1, user-scalabl
      meta(name='description', content='description')
      meta(name='keywords', content='keywords')
      meta(name='author', content='author')
      meta(name="robots", content="index, follow")

      //- ----- link website ----- //
      link(rel='icon' href='../favicon.ico' type='image/x-icon')

      //- ----- CSS Files ----- //
      link(rel='stylesheet', href='/bootstrap/dist/css/bootstrap.min.css')

      block title
        title Default title

    body

      script(type='text/javascript', src='/jquery/dist/jquery.min.js')
      script(type='text/javascript', src='/bootstrap/dist/js/bootstrap.min.js')

      block content
```

codingan diatas berfungsi untuk mendeklarasikan layout untuk semua page, page lain akan melakukan extend terhadap layout ini sehingga kita hanya perlu melakukan import library jquery dan bootstrap di layout ini. Sama seperti html biasa hanya saja pada page engine `jade` kita tidak menggunakan penutup tag, akan tetapi secara otomatis akan dibuatkan oleh jade, untuk lebih lengkap silahkan baca dokumentasi dari jade. Selanjutnya kita ke page `index.jade` dan berikut codingannya.

```
extends ./layout.jade

block title
```

```
  title Data Pegawai
block content
  div.container
    div.row
      div.col-xs-6.col-md-1
      div.col-xs-6.col-md-10
        p
          a.class(href='/tambah/pegawai')
            button(type="button").btn.btn-primary Tambah
        p
        table.table.table-bordered.table-striped.table-hover
          thead
            tr
              th.text-center Id Pegawai
              th.text-center Nama Pegawai
              th.text-center Alamat
              th.text-center Tanggal Lahir
              th.text-center Aksi
          tbody
            each pegawai in pegawais
              tr
                td #{pegawai.idPegawai}
                td #{pegawai.namaPegawai}
                td #{pegawai.alamat}
                td #{pegawai.tanggalLahir}
                td.text-center
                  a.class(href='/edit/pegawai/#{pegawai.idPegawai}')
                    button(type="button").btn.btn-success Edit
                  a.class(href='/delete/pegawai/#{pegawai.idPegawai}', onclick='return confirm("Apakah
                    button(type="button").btn.btn-danger Delete
      div.col-xs-6.col-md-4
```

Bisa dilihat pada kodingan `each pegawai in pegawais` kita melakukan render data berdasarkan data pegawais yang tadinya kita kirim melalui router. Data yang dikirim berupa array sehingga kita dapat melakukan render data tersebut ke dalam sebuah table. Jade juga mendukung penggunaan javascript client, contohnya ada pada button delete. Selanjutnya kita beralih ke page `tambah.jade` dengan kodingan.

```
extends ./layout.jade

block title
  title Tambah Data Pegawai
block content
  div.container
    div.row
      div.col-xs-6.col-md-4
      div.col-xs-6.col-md-4
        form(id="form", method="post", action="/save/pegawai")
          .form-group
```

```
            label Nama Pegawai
            input(type="text", placeholder="Nama Pegawai", name="namaPegawai").form-control
          .form-group
            label Alamat
            textarea(placeholder="Alamat", name="alamat").form-control
          .form-group
            label Tanggal Lahir
            input(type="date", placeholder="Tanggal Lahir", name="tanggalLahir").form-control
          input(type="hidden", name="_csrf" value="#{csrfToken}")
          button(type="submit").btn.btn-primary Save
      div.col-xs-6.col-md-4
```

Sama seperti kodingan form biasa, disana kita deklarasikan nama masing - masing input agar dapat dibaca oleh server lalu datanya akan disimpan, disini penulis juga menggunakan CSRF dapat dilihat pada bagian

```
input(type="hidden", name="_csrf" value="#{csrfToken}")
```

Selanjutnya adalah page `edit.jade`.

```
extends ./layout.jade

block title
  title Edit Data Pegawai
block content
  div.container
    div.row
      div.col-xs-6.col-md-4
      div.col-xs-6.col-md-4
        form(id="form", method="post", action="/update/pegawai/#{pegawai.idPegawai}")
          .form-group
            label Id Pegawai
            input(type="text", placeholder="Id Pegawai", name="idPegawai", value="#{pegawai.idPegawai}"
          .form-group
            label Nama Pegawai
            input(type="text", placeholder="Nama Pegawai", name="namaPegawai", value="#{pegawai.namaPeg
          .form-group
            label Alamat
            textarea(type="text",placeholder="Alamat", name="alamat").form-control
              | #{pegawai.alamat}
          .form-group
            label Tanggal Lahir
            input(type="text", placeholder="Tanggal Lahir", name="tanggalLahir", value="#{pegawai.tangg
          input(type="hidden", name="_csrf" value="#{csrfToken}")
          button(type="submit").btn.btn-primary Save
      div.col-xs-6.col-md-4
```

Sama seperti page `tambah.jade` hanya saja disini kita langsung memberikan value, dikarenakan user dapat melihat value sebelumnya, dan disini penulis mendisable untuk inputan idPegawai dikarenakan idPegawai bersifat unik. Dan page terakhir adalah `500.jade`, berikut kodingannya .

```
extends ./layout.jade

block title
  title Edit Data Pegawai
block content
  h1 Error 500 bung ... silahkan coba sesaat lagi :)
```

Page `500.jade` hanya sebagai page yang digunakan jika terjadi error pada aplikasi web node js ini. Tahap terakhir adalah kita membuat konfigurasi server agar dapat dijalankan. Buatlah sebuah file `app.js` pada root folder. Kemudian masukkan codingan berikut.

```
(function() {
  'use strict';

  var http = require('http'),
    express = require('express'), //express
    path = require('path'), //untuk path folder
    favicon = require('serve-favicon'), //untuk favicon
    morgan = require('morgan'), //untuk log aplikasi
    methodOverride = require('method-override'),
    session = require('express-session'), //session
    bodyParser = require('body-parser'), //handle rest
    errorhandler = require('errorhandler'), //error
    logger = require('./utils/logger'), //loging
    mongoose = require('mongoose'), //mongoose
    cookieParser = require('cookie-parser'),
    PegawaiRoute = require('./routes/PegawaiRoute'),
    app = express();

  app.set('port', process.env.PORT || 3000);
  app.set('views', path.join(__dirname, 'views'));
  app.set('view engine', 'jade');

  app.use(cookieParser());
  app.use(favicon(__dirname + '/public/favicon.ico'));
  app.use(morgan('combined', {
    stream: logger.stream
  }));
  app.use(methodOverride());
  app.use(session({
    resave: true,
    saveUninitialized: true,
```

```
    secret: 'uwotm8'
  }));
  app.use(bodyParser.urlencoded({
    extended: true
  }));
  app.use(express.static(path.join(__dirname, 'public')));
  app.use(express.static(path.join(__dirname, 'bower_components')));

  app.use('/', PegawaiRoute);

  mongoose.connect('mongodb://localhost/BelajarExpressJS', function(err, res) {
    if (err) {
      return logger.error('koneksi mongodb gagal bung', err);
    } else {
      return logger.info('koneksi mongodb berhasil bung');
    }
  });

  if ('development' === app.get('env')) {
    app.use(errorhandler());
  }

  app.use(function(err, req, res, next) {
    if (err.code !== 'EBADCSRFTOKEN') return next(err)

    // handle CSRF token errors here
    res.status(403)
    res.send('csrf token tidak tersedia bung');
  });

  var server = http.createServer(app);
  server.listen(app.get('port'), function() {
    return console.log('Server jalan pada port ' + app.get('port'));
  });

}).call(this);
```

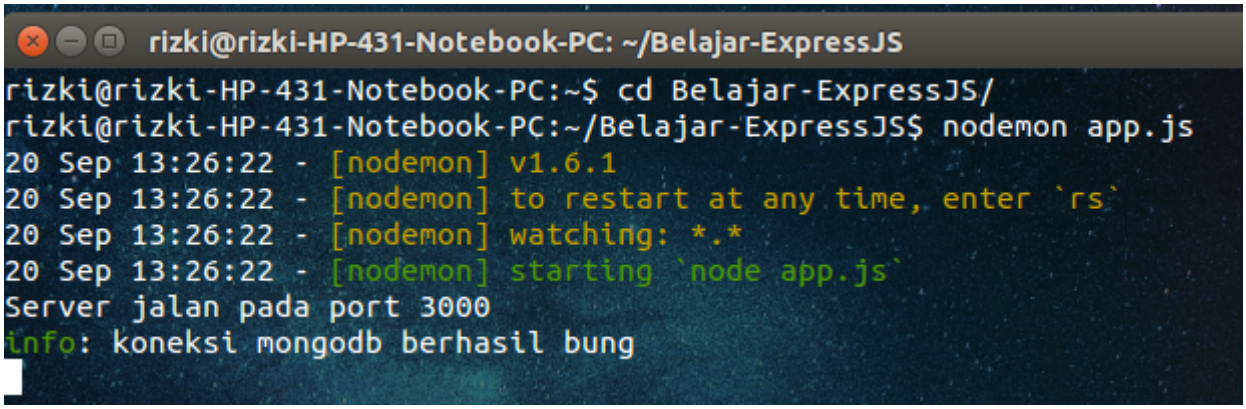
Lumayan panjang :D baiklah, penulis menjelaskan beberapa hal saja.

- `app.set('port');` berfungsi untuk mendeklarasikan aplikasi akan jalan pada port 3000.
- `app.set('views');` berfungsi untuk mendeklarasikan folder view yang akan kita gunakan
- `app.set('view engine', 'jade');` berfungsi untuk mendeklarasikan bahwa kita menggunakan jade sebagai template enginenya.
- `mongoose.connect()` untuk melakukan koneksi ke database mongodb
- `var server = http.createServer(app);` berfungsi untuk membuat sebuah server lalu menjalankannya. berbeda dengan php, node js dapat dijalankan dengan menggunakan server yang bersifat embedded artinya ketika menjalankan sebuah server node js, kita tidak perlu

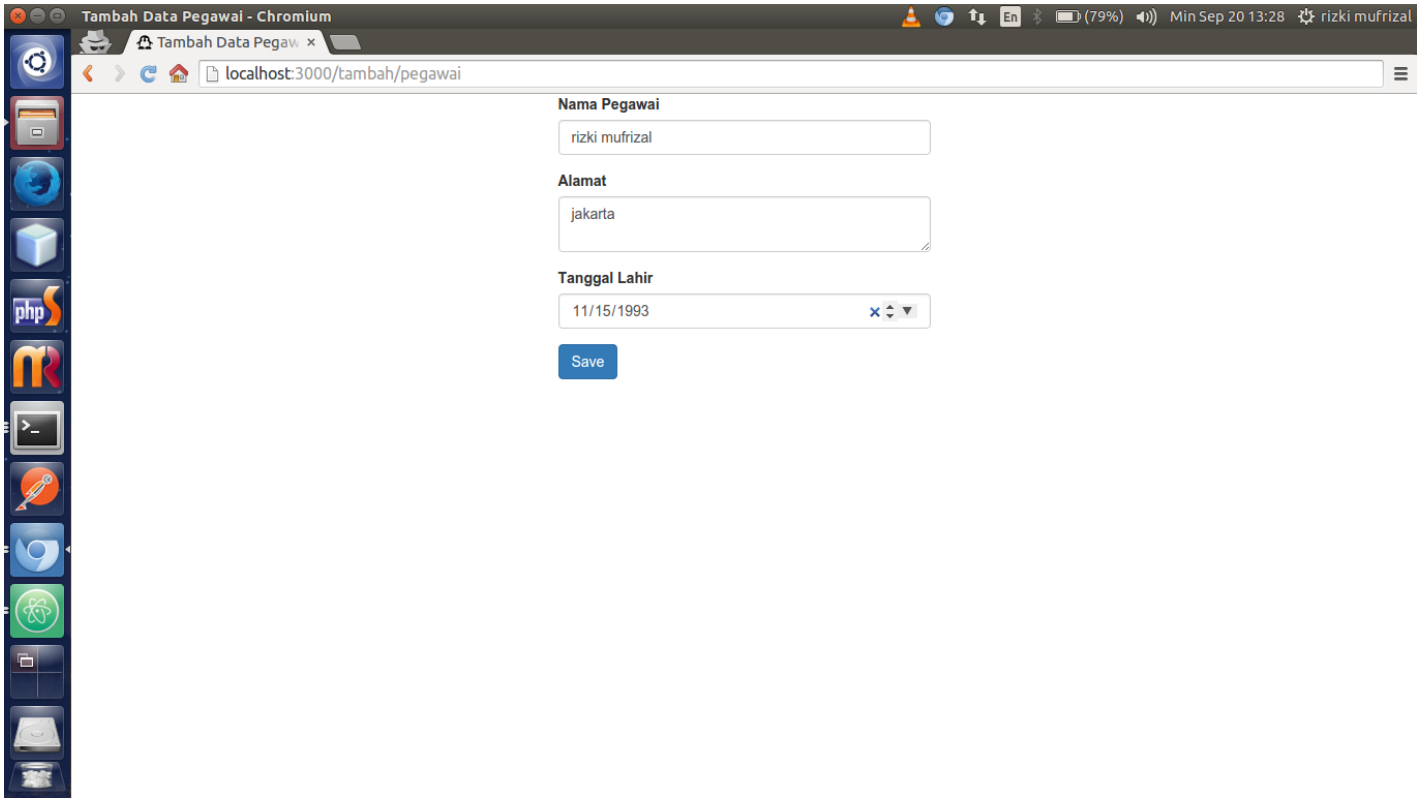
mencopy project ke folder htdoc untuk dijalankan akan tetapi cukup menjalankannya di folder project maka aplikasi siap digunakan, konsep ini sama seperti pemrograman web pada java, kita dapat menjalankan web server baik jetty, tomcat maupun wildfly pada folder project dengan bantuan maven plugin.

Uji Coba

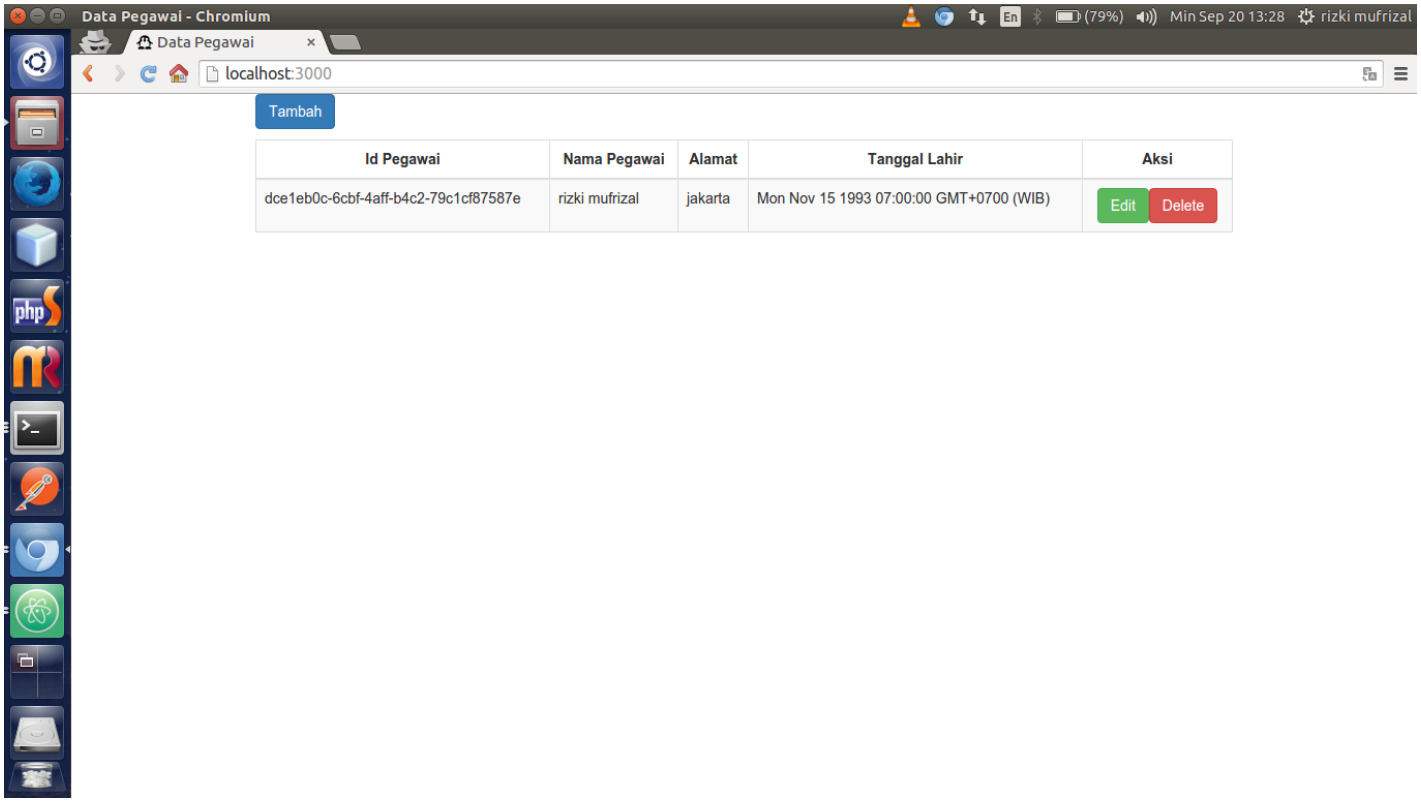
Akhirnya selesai juga, kita lakukan uji coba, jalankan aplikasi dengan menggunakan nodemon dengan perintah `nodemon app.js` maka hasilnya seperti ini.



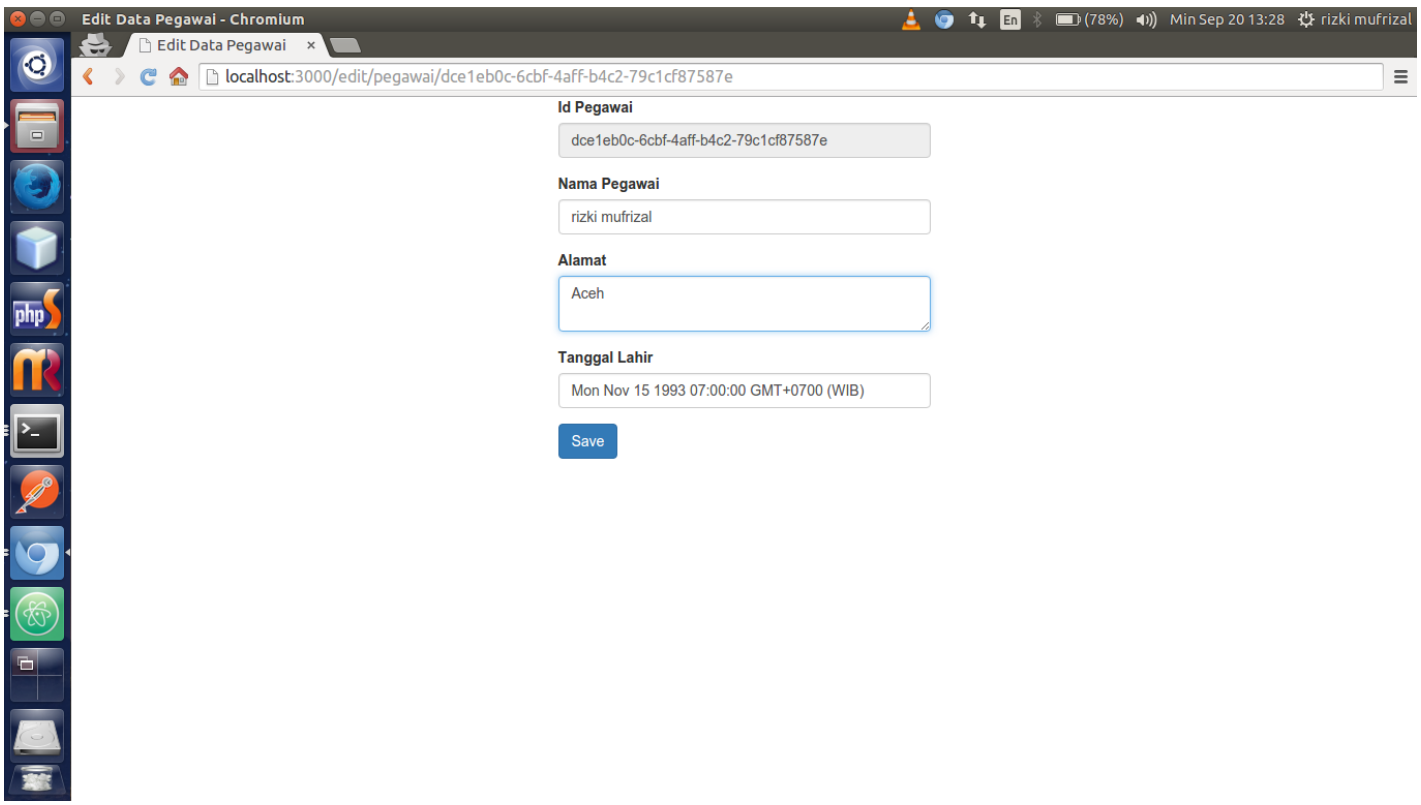
Kemudian hit ke browser dengan url `http://localhost:3000/` kemudian tambah data dengan menekan tombol tambah kemudian masukkan data anda, maka hasilnya seperti ini.



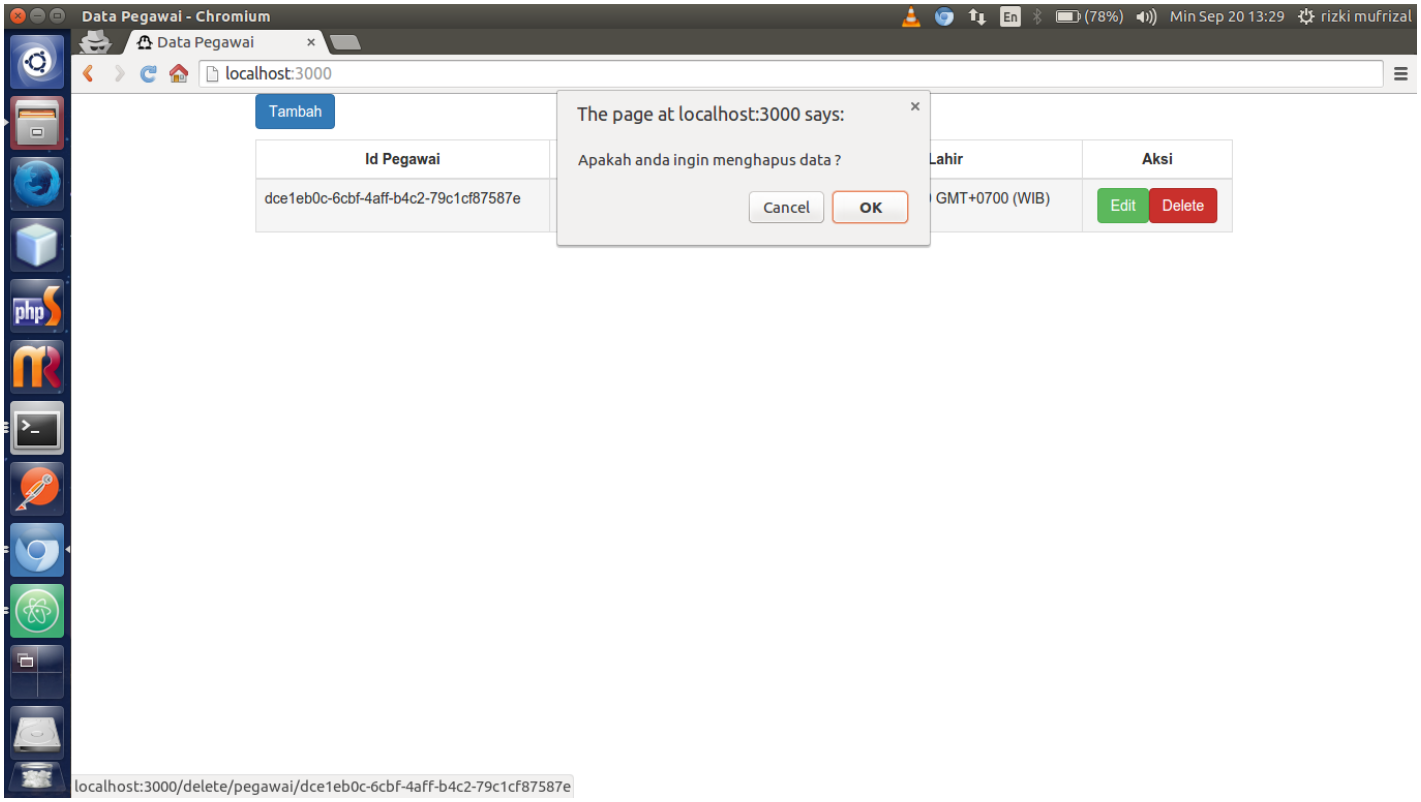
Data yang telah disimpan dapat di lihat pada index page, berikut hasilnya.



Untuk melakukan edit data, kita dapat menekan tombol edit, dah hasilnya seperti ini.



Untuk menghapus, kita dapat menekan tombol hapus dan terdapat alert javascript untuk memilik ya atau tidak dan hasilnya seperti ini.



Sekian tutorial belajar express dan Terima kasih :). Untuk source code lengkap, penulis publish di Belajar Express.

EXPRESS JS BELAJAR EXPRESS JS MONGODB MONGOOSE JADE

LIKE TWEET +1

Wahyudin Marsaoly • 8 months ago

terimakasih sudah share ilmunya mas,

saya sudah coba tutorial di atass saya ada kendala untuk koneksi mongodb

saat saya jalankan perintah nodemon app.js

koneksi monggodb gagal dan keluar pesan error seperti berikut :

Error: connect ECONNREFUSED 127.0.0.1:27017

at Object.exports._errnoException (util.js:870:11)

at exports._exceptionWithHostPort (util.js:893:20)

at TCPConnectWrap.afterConnect [as oncomplete] (net.js:1061:14)

rizki mufrizal Mod → Wahyudin Marsaoly • 8 months ago

mongodb nya udh dijalankan ?

Read More

Membuat RESTful Web Service Dengan Framework Spring Boot

membuat RESTful Web Service dengan Framework Spring Boot Continue reading

Belajar Load Balancing Dengan HAProxy, Docker Dan Spring Boot

Published on September 03, 2016

Pengalaman Sidang Komprehensif Paket 1 Teknik Informatika Universitas Gunadarma 2016

Published on September 01, 2016