PWL - Node.JS - Sequelize (Edisi ke-3)

> **PWL - Node.JS + Pug + Sequelize Notes**
>
> By 2472008
> Update 23 Juni 2026

Instruksi Instalasi & Menjalankan Server

Sebelum memulai, silahkan menginisialisasi proyek dengan

```
npm init
```

Untuk menginstall express & nodemon, gunakan perintah

```
npm install express
npm install nodemon
npm install ejs
```

Lalu untuk menjalankan server

```
npx nodemon <filename.js>
```

Instruksi Membuat Routes, dan Views

Dengan struktur direktori sebagai berikut

```
my-project/
    server.js
    routes/
        routes.js
    views/
        index.ejs
        forms.ejs
        received.ejs
```

File untuk `server.js`

```
const express = require('express')
const app = express()
const router = require('./routes/routes.js')

app.set('view engine', 'ejs')
app.use(router)

const PORT = 5000

app.listen(PORT, () => {
```

```
 console.log(`Server is running at port ${PORT}`)
})
```

Lalu file untuk `routes.js`

```
const express = require('express')
const router = express.Router()

router.use(express.urlencoded({extended: false}))

router.post('/receiver', (req, res) => {
    const name = req.body.name
    const email = req.body.email
    res.status(200).render('received', {name, email})
})

router.get('/form', (req, res) => {
    res.status(200).render('forms')
})

router.get('/', (req, res) => {
    res.status(200).render('index')
})

module.exports = router
```

Dengan begitu, pada views, untuk menampilkan data, dapat digunakan sintaks sebagai berikut:

```
<h1>Data Received!!</h1>
<p><b>Name</b>: <%= name %></p>
<p><b>Email</b>: <%= email %></p>
```

Snippet code ini berada pada file `received.ejs`
Silahkan membuat HTML seperti biasanya pada file `.ejs` tersebut

---

Chapter 2: PUG

PUG Templating Engine

Setup:

```
npm install pug
```

Untuk membuat views pug, jika memiliki template dalam format HTML, maka HTML harus dikonversi ke PUG dengan html-to-pug.com

Dengan struktur direktori berikut

```
my-project/
    server.js
    routes/
        routes.js
    views/
        layouts/
            master.pug
            header.pug
            sidebar.pug
            footer.pug
        starter.pug
```

Lalu lakukan sedikit penyesuaian pada `server.js`

```
const express = require('express')
const app = express()
const router = require('./routes/route')

app.set('view engine', 'pug')
app.use(express.urlencoded({extended: false}))
app.use(router)

const PORT = 5000
app.listen(PORT, () => {
 console.log(`Server is running at port ${PORT}`)
})
```

Lakukan juga penyesuaian pada `routes.js`

```
const express = require('express')
const router = express.Router()

router.use(express.urlencoded({extended: false}))

router.get('/', (req, res) => {
    res.status(200).render('starter')
})

module.exports = router
```

Maka dapat dibuat file `master.pug` dengan memiliki baris-baris berikut

```
doctype html
head
  meta(charset='utf-8')
  meta(name='viewport' content='width=device-width, initial-scale=1')
  title AdminLTE 3 | Starter Page
```

```
  link(rel='stylesheet' href='https://fonts.googleapis.com/css?
family=Source+Sans+Pro:300,400,400i,700&display=fallback')
  link(rel='stylesheet' href='https://cdnjs.cloudflare.com/ajax/libs/font-
awesome/5.15.4/css/all.min.css')
  link(rel='stylesheet' href='https://cdn.jsdelivr.net/npm/admin-
lte@3.2/dist/css/adminlte.min.css')
body.hold-transition.sidebar-mini
  .wrapper
        // header
        include header

        // sidebar
        include sidebar

        .content-wrapper
            .content-header
                .container-fluid
                    .row.mb-2
                        .col-sm-6
                            h1.m-0 Starter Page
         .content
            // content
           block content
        // footer
        include footer
  script(src='https://cdnjs.cloudflare.com/ajax/libs/jquery/3.6.0/jquery.min.js')

script(src='https://cdnjs.cloudflare.com/ajax/libs/bootstrap/4.6.1/js/bootstrap.bun
dle.min.js')
  script(src='https://cdn.jsdelivr.net/npm/admin-lte@3.2/dist/js/adminlte.min.js')
```

juga dapat dibuat `starter.pug` demikian:

```
extends ./layouts/master

block content
  .container-fluid
        .row
          .col-lg-6
            .card.card-primary.card-outline
                .card-body
                    h5.card-title Selamat Datang!
                   p.card-text
                     | Ini adalah halaman starter menggunakan AdminLTE 3 via CDN.
                   a.btn.btn-primary(href='#') Go somewhere
```

Chapter 3: MVC, MySQL + Sequelize

Sequelize

Setup & EDA (Exploratory Data Analysis) Struktur Project

Setup:

```
npm install bcrypt dotenv sequelize
```

Tetapi apabila kita telah menerima proyek, maka kita hanya perlu lakukan

```
npm install
```

Lalu kita berdoa bahwa pembuat proyek sebelumnya telah menyertakan *dependency* yang diperlukan dalam file `package.json` dengan baik dan benar

berikut adalah contoh file `package.json` yang sudah dikonfigurasi dengan baik dan benar oleh Mr Robby Tan

```
{
  "name": "learn_node",
  "version": "1.0.0",
  "description": "Repository for Node.js Tutorial",
  "keywords": [
    "Node"
  ],
  "license": "MIT",
  "author": "Robby Tan",
  "type": "commonjs",
  "main": "app.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "devDependencies": {
    "nodemon": "^3.1.14"
  },
  "dependencies": {
    "bcrypt": "^6.0.0",
    "dotenv": "^17.4.2",
    "ejs": "^5.0.2",
    "express": "^5.2.1",
    "express-session": "^1.19.0",
    "method-override": "^3.0.0",
    "mysql2": "^3.22.3",
    "pug": "^3.0.4",
    "sequelize": "^6.37.8"
  }
}
```

Dengan demikian, apabila `server.js` atau `app.js` belum ada, maka kita harus membuat `server.js` pada direktori root project seperti berikut:

```
require('dotenv').config()
const express = require('express')
const methodOverride = require('method-override')
const app = express()
const route = require('./routes/route')
const sequelize = require('./config/database')
const session = require('express-session')

app.set('view engine', 'pug')
app.use(express.urlencoded({extended: false}))
app.use(methodOverride('_method'));
app.use(session({
  secret: process.env.SESSION_SECRET || 'yoursecret',
  resave: false,
  saveUninitialized: false,
  cookie: {
    maxAge: 1000 * 60 * 60 * 2
  }
}))
app.use(route)

const PORT = process.env.PORT || 3000
sequelize.sync({alter: false})
  .then(() => {
    app.listen(PORT, () => {
      console.log(`Server is running at port ${PORT}`)
    })
  })
  .catch(err => console.log(err))
```

Kita mungkin menyadari bahwa implementasi `server.js` milik Mr Robby Tan ini adalah cara lama, sebab cara import yang baru adalah demikian:

```
import express from 'express'
import session from 'express-session'
import router from './routes/route.js'
```

Tetapi berhubung kita akan melakukan UAS, dan yang menilai UAS ini adalah Mr Robby Tan, maka kita tetap akan menggunakan cara lama

Kita perhatikan bahwa struktur direktori proyek adalah sebagai berikut:

```
chelsea-sequelize-projects/
    config/
        database.js
    controllers/
        AuthController.js
        CategoryController.js
    middlewares/
        AuthMiddleware.js
```

```
    models/
        Category.js
        User.js
    public/dist/img/
        AdminLTELogo.png
        avatar5.png
    routes/
        route.js
    views/
        auth/
            login.pug
        category/
            create.pug
            edit.pug
            index.pug
        layouts/
            footer.pug
            header.pug
            master.pug
            sidebar.pug
        starter.pug
    .env
    .env.example
    server.js
    package.json
```

Konfigurasi Sequelize

Hal yang menarik adalah file `config/database.js`, ini adalah hal baru untuk sequelize ini

```
const {Sequelize} = require('sequelize');

const sequelize = new Sequelize(
  process.env.DB_NAME,
  process.env.DB_USER,
  process.env.DB_PASSWORD,
  {
    host: process.env.DB_HOST,
    // port: process.env.DB_PORT || 3306,
    dialect: 'mysql',
    logging: false, // jadikan "console.log" apabila ingin debugging
    // define: {
     //    underscored: true // kolom otomatis snake_case
    // }
  }
);

module.exports = sequelize;
```

terlihat bahwa kita perlu konfigurasi file `.env`, berikut adalah contoh konfigurasi `.env`

```
PORT=5000
```

```
DB_HOST=localhost
DB_USER=root
DB_PASSWORD=root
DB_NAME=node_lib_db
DB_PORT=3306

SESSION_SECRET=key_for_sessions
```

MVC - Models:

Setelah kita melakukan konfigurasi yang demikian, kita sekarang ke bagian MVC yang pertama, yaitu Model, Model adalah bagian yang berkaitan erat dengan Basis Data, kita sebelumnya telah melakukan konfigurasi basis data, maka sekarang kita perlu melakukan modelling, alias memodelkan skema basis data kita pada aplikasi, kita kembali melihat struktur project kita, yaitu demikian:

```
models/
     Category.js
     User.js
```

berikut adalah contoh untuk `models/User.js:

```
const { DataTypes } = require('sequelize');
const sequelize = require('../config/database');
const bcrypt = require('bcrypt');

const User = sequelize.define('User', {
  id: {
    type: DataTypes.BIGINT,
    autoIncrement: true,
    primaryKey: true,
    unsigned: true,
  },
  name: {
    type: DataTypes.STRING,
    allowNull: false,
    maxlength: 255,
  },
  email: {
    type: DataTypes.STRING,
    allowNull: false,
    maxlength: 255,
    unique: true,
    validate: {
      isEmail: true
    }
  },
  password: {
    type: DataTypes.STRING,
    allowNull: false
  }
}, {
  tableName: 'users',
  timestamps: true,
```

```
  hooks: {
    beforeCreate: async (user) => {
      if (user.password) {
        const salt = await bcrypt.genSalt(10);
        user.password = await bcrypt.hash(user.password, salt);
      }
    }
  }
});

module.exports = User;
```

lalu untuk `models/Category.js` adalah sebagai berikut:

```
const {DataTypes} = require('sequelize')
const sequelize = require('../config/database')

const Category = sequelize.define('Category', {
    id: {
      type: DataTypes.INTEGER,
      autoIncrement: true,
      primaryKey: true,
      unsigned: true,
    },
    name: {
      type: DataTypes.STRING,
      allowNull: false,
      maxlength: 60,
    },
    description: {
      type: DataTypes.STRING,
      allowNull: true,
      maxlength: 150,
    }
  },
  {
    tableName: 'category',
    timestamps: true,
  }
)

module.exports = Category;
```

MVC - Controller, Middleware, Routes

Kita ke bagian MVC yang berikutnya, yaitu `Controller`, kita perhatikan strukturnya sebagai berikut:

```
controllers/
     AuthController.js
     CategoryController.js
middlewares/
     AuthMiddleware.js
```

```
routes/
    route.js
```

Controller

Kita melihat `controllers/AuthController.js` sebagai berikut:

```
const User = require('../models/User')
const bcrypt = require('bcrypt')

const AuthController = {

  showLogin: (req, res) => {
    res.render('auth/login', { currentPage: 'login' })
  },

  login: async (req, res) => {
    try {
      const { email, password } = req.body
      if (!email || !password) {
        return res.status(400).send('Email and password are required')
      }
      const user = await User.findOne({ where: { email } })
      if (!user) {
        return res.status(401).send('Invalid email or password')
      }
      const isPasswordValid = await bcrypt.compare(password, user.password)
      if (!isPasswordValid) {
        return res.status(401).send('Invalid email or password')
      }

      req.session.userId = user.id
      req.session.userName = user.name
      req.session.userEmail = user.email
      res.redirect('/')
    } catch (error) {
      res.status(500).send(`Error server: ${error.message}`)
    }
  },

  logout: async (req, res) => {
    req.session.destroy((err) => {
      if (err) {
        return res.status(500).send(`Error server: ${err}`)
      }
      res.clearCookie('connect.sid')
      res.redirect('/login')
    })
  }
}

module.exports = AuthController
```

Lalu `controllers/CategoryController.js` sebagai berikut:

```
const Category = require('../models/Category')

const CategoryController = {
  index: async (req, res) => {
    try {
      const data = await Category.findAll()
      res.status(200).render('category/index', {
        categories: data,
        currentPage: 'category',
      });
    } catch (error) {
      res.status(500).send(`Error server: ${error.message}`);
    }
  },

  create: (req, res) => {
    try {
      res.status(200).render('category/create', {
        currentPage: 'category',
      })
    } catch (error) {
      res.status(500).send(`Error server: ${error.message}`)
    }
  },

  store: async (req, res) => {
    try {
      const { name } = req.body;
      if (!name) {
        return res.status(400).send('Name is required');
      }

      await Category.create({ name });
      res.redirect(`/category`)
    } catch (error) {
      res.status(500).send(`Error server: ${error.message}`)
    }
  },

  edit: async (req, res) => {
    try {
      const { id } = req.params;
      const category = await Category.findByPk(id);

      if (!category) {
        return res.status(404).send('Category not found');
      }

      res.status(200).render('category/edit', {
        category: category,
        currentPage: 'category',
      })
    } catch (error) {
      res.status(500).send(`Error server: ${error.message}`)
```

```
    }
  },

  update: async (req, res) => {
    try {
      const { id } = req.params;
      const { name } = req.body;

      if (!name) {
        return res.status(400).send('Name is required');
      }

      await Category.update({ name }, { where: { id } });
      res.redirect('/category')
    } catch (error) {
      res.status(500).send(`Error server: ${error.message}`)
    }
  },

  destroy: async (req, res) => {
    try {
      const { id } = req.params;

      await Category.destroy({ where: { id } });
      res.redirect('/category')
    } catch (error) {
      res.status(500).send(`Error server: ${error.message}`)
    }
  }
}

module.exports = CategoryController;
```

Middleware

Untuk melakukan pembatasan akses pada laman tertentu, misalnya memerlukan otentikasi terlebih dahulu sebelum mengakses maka kita gunakan middleware untuk melakukan hal tersebut, berikut adalah contoh untuk file `AuthMiddleware.js`:

```
const authMiddleware = (req, res, next) => {
  if (req.session && req.session.userId) {
    return next();
  }
  res.redirect('/login');
};

module.exports = authMiddleware;
```

Routes

Berikutnya, untuk web dapat diakses, kita memerlukan perutean, yaitu `routes/Route.js`:

```
const express = require('express')
const router = express.Router()
const AuthController = require('../controllers/AuthController')
const CategoryController = require('../controllers/CategoryController')

const authMiddleware = require('../middlewares/AuthMiddleware')

router.use(express.static('public'))

router.get('/login', AuthController.showLogin);
router.post('/login', AuthController.login);
router.get('/logout', authMiddleware, AuthController.logout);

router.get('/category/:id/edit', authMiddleware, CategoryController.edit)
router.put('/category/:id', authMiddleware, CategoryController.update)
router.delete('/category/:id', authMiddleware, CategoryController.destroy)
router.get('/category/create', authMiddleware, CategoryController.create)
router.post('/category', authMiddleware, CategoryController.store)
router.get('/category', authMiddleware, CategoryController.index)

router.get('/', (req, res) => {
  res.render('starter', {
    currentPage: 'starter',
  })
})

module.exports = router
```

MVC - Views & Public

Kita ke bagian MVC yang terakhir, yaitu Views, di sini kita bermain dengan *front-end*, alias tampilan

```
views/
     auth/
          login.pug
     category/
          create.pug
          edit.pug
          index.pug
     layouts/
          footer.pug
          header.pug
          master.pug
          sidebar.pug
     starter.pug
public/dist/img/
     AdminLTELogo.png
     avatar5.png
```

Template Utama

Kita melihat bahwa tampilan dimulai dari `starter.pug`

```
extends layouts/master

block content
  .content-wrapper
      // Content Header
      section.content-header
        .container-fluid
          .row.mb-2
            .col-sm-6
              h1 Dashboard
      // Main content
      section.content
        .container-fluid
          .card
            .card-body
              | Welcome to Dashboard AdminLTE 3!
```

Layouts

Berikutnya, layouts utama dari project adalah sebagai berikut

Hal yang terutama adalah `views/layouts/master.pug` sebagai berikut:

```
doctype html
head
  meta(charset='UTF-8')
  meta(name='viewport' content='width=device-width, initial-scale=1')
  title AdminLTE 3 Starter
  // Google Font: Source Sans Pro
  link(rel='stylesheet' href='https://fonts.googleapis.com/css?
family=Source+Sans+Pro:300,400,400i,700&display=fallback')
  // Font Awesome Icons
  link(rel='stylesheet' href='https://cdnjs.cloudflare.com/ajax/libs/font-
awesome/6.4.0/css/all.min.css' integrity='sha512-...' crossorigin='anonymous'
referrerpolicy='no-referrer')
  // AdminLTE CSS
  link(rel='stylesheet' href='https://cdn.jsdelivr.net/npm/admin-
lte@3.2/dist/css/adminlte.min.css')
  block CSS

body.hold-transition.sidebar-mini
  .wrapper
         // Navbar
        include header

        // Main Sidebar Container
        include sidebar

        // Content Wrapper
        block content

        // Footer
        include footer
```

```
    // jQuery
    script(src='https://cdnjs.cloudflare.com/ajax/libs/jquery/3.7.0/jquery.min.js'
integrity='sha512-...' crossorigin='anonymous' referrerpolicy='no-referrer')
    // Bootstrap 4
    script(src='https://cdnjs.cloudflare.com/ajax/libs/twitter-
bootstrap/4.6.2/js/bootstrap.bundle.min.js' integrity='sha512-...'
crossorigin='anonymous' referrerpolicy='no-referrer')
    // AdminLTE App
    script(src='https://cdn.jsdelivr.net/npm/admin-
lte@3.2/dist/js/adminlte.min.js')

    block JS
```

views/layouts/footer.pug

```
footer.main-footer
    strong
        | &copy; 2026
        a(href='#') AdminLTE.io
        | .
    |  All rights reserved.
```

views/layouts/header.pug

```
nav.main-header.navbar.navbar-expand.navbar-white.navbar-light
  // Left navbar links
  ul.navbar-nav
    li.nav-item
      a.nav-link(data-widget='pushmenu' href='#' role='button')
        i.fas.fa-bars
    li.nav-item.d-none.d-sm-inline-block
      a.nav-link(href='#') Home
```

views/layouts/sidebar.pug

```
aside.main-sidebar.sidebar-dark-primary.elevation-4
  // Brand Logo
  a.brand-link(href='#')
    span.brand-text.font-weight-light AdminLTE 3

  // Sidebar
  .sidebar
    // Sidebar user panel
    .user-panel.mt-3.pb-3.mb-3.d-flex
      .image
        img.img-circle.elevation-2(src='/dist/img/avatar5.png' alt='User Image')
      .info
        a.d-block(href='#') Robby Tan

    // Sidebar Menu
    nav.mt-2
      ul.nav.nav-pills.nav-sidebar.flex-column(data-widget='treeview' role='menu')
        li.nav-item
```

```
          a.nav-link(href='/' class=(currentPage === 'starter' ? 'active' : ''))
            i.nav-icon.fas.fa-tachometer-alt
            p Dashboard
        li.nav-item
          a.nav-link(href='/category' class=(currentPage === 'category' ? 'active'
: ''))
            i.nav-icon.fas.fa-th
            p Category Mgmt
        li.nav-item
          a.nav-link(href='/logout')
            i.nav-icon.fas.fa-sign-out-alt
            p Logout
```

Direktori Auth

Ini adalah tampilan untuk laman login, yaitu `views/auth/login.pug`

```
doctype html
head
  meta(charset='UTF-8')
  meta(name='viewport' content='width=device-width, initial-scale=1.0')
  title Login
  // Bootstrap 5

link(href='https://cdn.jsdelivr.net/npm/bootstrap@5.3.2/dist/css/bootstrap.min.css'
rel='stylesheet')
  // Bootstrap Icons
  link(href='https://cdn.jsdelivr.net/npm/bootstrap-icons@1.11.1/font/bootstrap-
icons.css' rel='stylesheet')
  style.
      body {
          background: #f4f6f9;
          height: 100vh;
      }

      .login-container {
          height: 100vh;
      }

      .toggle-password {
          cursor: pointer;
      }
body
  .container.login-container.d-flex.justify-content-center.align-items-center
    .col-md-4
      .card.login-card.p-4
        .text-center.mb-4
          h4.login-title Login Form
        form(method='POST' action='/login')
          .mb-3
            label.form-label(for='email') Email
            input#email.form-control(name='email' type='email' placeholder='Input
email' required autofocus)
          .mb-3
```

```
            label.form-label(for='password') Password
            .input-group
              input#password.form-control(type='password' name='password'
placeholder='Input password' required)
              span.input-group-text.toggle-password(onclick='togglePassword()')
                i#icon.bi.bi-eye
          .d-grid
            button.btn.btn-primary(type='submit') Login
  script.
    function togglePassword() {
      const password = document.getElementById("password");
      const icon = document.getElementById("icon");
      if (password.type === "password") {
        password.type = "text";
        icon.classList.remove("bi-eye");
        icon.classList.add("bi-eye-slash");
      } else {
        password.type = "password";
        icon.classList.remove("bi-eye-slash");
        icon.classList.add("bi-eye");
      }
    }
```

Direktori Category

Dengan menggunakan layouts pada direktori `layouts`, demikian adalah views-views pada bagian category

Pada direktori Category, hal terutama adalah pada bagian `views/category/index.pug`

```
extends ../layouts/master

block content
  .content-wrapper
      // Content Header
      section.content-header
        .container-fluid
          .row.mb-2
            .col-sm-6
              h1 Dashboard
      // Main content
      section.content
        .container-fluid
          .card
            .card-header
              h3.card-title Category List
              a.btn.btn-primary.btn-sm.float-right(href='/category/create') Create
New Category
            .card-body
              table.table.table-bordered
                thead
                  tr
                    th(scope='col') ID
```

```
                    th(scope='col') Name
                    th(scope='col') Action
                tbody
                  each category in categories
                    tr
                      td= category.id
                      td= category.name
                      td
                        a.btn.btn-warning.btn-
sm(href=`/category/${category.id}/edit`) Edit
                        form(method='POST', action=`/category/${category.id}?
_method=DELETE`, style='display:inline', onsubmit='return confirm("Are you sure you
want to delete this category?");')
                          button.btn.btn-danger.btn-sm(type='submit') Delete
```

views/category/create.pug

```
extends ../layouts/master

block content
  .content-wrapper
      // Content Header
      section.content-header
        .container-fluid
          .row.mb-2
            .col-sm-6
              h1 Dashboard
      // Main content
      section.content
        .container-fluid
          .card
            .card-body
              form(method='POST', action='/category')
                .form-group
                  label(for='name') Name
                  input#name.form-control(type='text', name='name', required,
autofocus)
                button.btn.btn-primary(type='submit') Submit
```

views/category/edit.pug

```
extends ../layouts/master

block content
  .content-wrapper
      // Content Header
      section.content-header
        .container-fluid
          .row.mb-2
            .col-sm-6
              h1 Dashboard
      // Main content
      section.content
        .container-fluid
```

```
          .card
            .card-body
              form(method='POST', action=`/category/${category.id}?_method=PUT`)
                .form-group
                  label(for='name') Name
                  input#name.form-control(type='text', name='name',
value=`${category.name}`, required)
                button.btn.btn-primary(type='submit') Submit
```

Akhir Kata

Dengan demikian, kita telah berhasil membuat proyek sederhana yang mengakses basis data `MySQL` dengan menggunakan `Sequelize ORM`

Kiranya proyek sederhana dari Mr Robby Tan ini dapat membantu kita semua untuk menyelesaikan permasalahan yang muncul dalam UAS

Semangat, semoga sukses UAS-nya!! 🔥🔥