# MODUL PERKULIAHAN

**Mata Kuliah** : Pengembangan Aplikasi Web  
**Materi** : Pengelolaan Routing dan Middleware di Express.js  
**Pertemuan** : 03  
**Fakultas** : Teknik  
**Program Studi** : Teknologi Informasi  
**Kode MK** : AntaraTI501P  
**Disusun Oleh** : Ir.Asroni, S.T., M.Eng dan Rama Soni Ravictor

## DAFTAR ISI

1. [PEMBAHASAN](#pembahasan)
2. [PREREQUISITES](#prerequisites)
3. [PRAKTIKUM](#praktikum)
   * [A. Tujuan Praktikum](#a-tujuan-praktikum)
   * [B. Materi](#b-materi)
   * [C. Metode Praktikum](#c-metode-praktikum)
   * [D. Langkah-langkah Praktikum](#d-langkah-langkah-praktikum)
4. [TUGAS](#tugas)

## PEMBAHASAN

1. Pendalaman Routing di Express.js:
   * Basic routing
   * Route parameters
   * Route handlers
   * Response methods
2. Advanced Routing:
   * Router-level middleware
   * Express.Router
   * Chaining route handlers
3. Middleware di Express.js:
   * Konsep dan fungsi middleware
   * Application-level middleware
   * Router-level middleware
   * Error-handling middleware
4. Built-in Middleware:
   * express.static
   * express.json
   * express.urlencoded
5. Custom Middleware:
   * Membuat middleware sendiri
   * Menggunakan middleware di route spesifik
6. Third-party Middleware:
   * cors
   * helmet
   * morgan
7. Error Handling:
   * Menangani error di middleware dan route handlers
   * Custom error handling middleware
8. Best Practices:
   * Strukturisasi aplikasi Express.js
   * Pemisahan concerns (routing, middleware, controllers)

## PREREQUISITES

Beberapa perangkat lunak yang digunakan untuk menunjang proses pembelajaran:

1. Node.js (versi LTS terbaru)
2. NPM (Node Package Manager)
3. Text editor (Visual Studio Code, Sublime Text, atau Atom)
4. Postman atau tools API testing lainnya

## PRAKTIKUM

### A. Tujuan Praktikum

Praktikum ini bertujuan untuk memperdalam pemahaman mahasiswa tentang routing dan middleware di Express.js. Mahasiswa akan mempelajari cara mengelola rute yang kompleks, membuat dan menggunakan middleware, serta menerapkan best practices dalam pengembangan aplikasi Express.js.

### B. Materi

1. Advanced routing di Express.js
2. Pembuatan dan penggunaan middleware
3. Penanganan error dengan middleware
4. Penggunaan third-party middleware
5. Strukturisasi aplikasi Express.js

### C. Metode Praktikum

1. Penjelasan materi oleh instruktur
2. Demonstrasi langsung penggunaan routing dan middleware
3. Praktik mandiri oleh mahasiswa
4. Diskusi dan tanya jawab

### D. Langkah-langkah Praktikum

1. Inisialisasi & Struktur Folder

Lanjutkan project node server sebelumnya. Buat struktur folder berikut di dalam project Anda:

A screenshot of a black screen

AI-generated content may be incorrect.

Desain Endpoint API

Attendance:

* POST /api/presensi/check-in
* POST /api/presensi/check-out

Report (Khusus Admin):

* GET /api/reports/daily

1. Buat penyimpanan data, isi dengan kode berikut pada file data/presensiData.js:

const presensiRecords = [

// Kita bisa mulai dengan data contoh

{

userId: 456,

nama: 'User Karyawan',

checkIn: new Date('2025-10-14T08:05:00'),

checkOut: new Date('2025-10-14T17:00:00')

}

];

module.exports = presensiRecords;

1. Implementasi Middleware
   * kita akan buat middleware untuk simulasi izin (permission) dan untuk menambahkan data karyawan ke request:
   * exports.addUserData = (req, res, next) => {
   * console.log('Middleware: Menambahkan data user dummy...');
   * req.user = {
   * id: 123,
   * nama: User Karyawan',
   * role: 'karyawan'
   * };
   * next();
   * };
   * exports.isAdmin = (req, res, next) => {
   * if (req.user && req.user.role === 'admin') {
   * console.log('Middleware: Izin admin diberikan.');
   * next();
   * } else {
   * console.log('Middleware: Gagal! Pengguna bukan admin.');
   * return res.status(403).json({ message: 'Akses ditolak: Hanya untuk admin'});
   * }
   * };
2. Gunakan library seperti date-fns-tz, untuk membuat manipulasi zona waktu, kita akan ubah menjadi zona waktu Jakarta.

Buka terminal di folder proyek Anda dan instal *library* yang dibutuhkan:

npm install date-fns date-fns-tz

1. Implementasi Controllers
   * **controllers/presensiController.js:**

*const* presensiRecords = require("../data/presensiData");

*const* { format } = require("date-fns-tz");

*const* timeZone = "Asia/Jakarta";

exports.CheckIn = (*req*, *res*) => {

*const* { id: userId, nama: userName } = req.user;

*const* waktuSekarang = new Date();

*const* existingRecord = presensiRecords.find(

(*record*) => record.userId === userId && record.checkOut === null

);

if (existingRecord) {

return res

.status(400)

.json({ *message:* "Anda sudah melakukan check-in hari ini." });

}

*const* newRecord = {

userId,

*nama:* userName,

*checkIn:* waktuSekarang,

*checkOut:* null,

};

presensiRecords.push(newRecord);

*const* formattedData = {

...newRecord,

*checkIn:* format(newRecord.checkIn, "yyyy-MM-dd HH:mm:ssXXX", { timeZone }),

};

console.log(

`DATA TERUPDATE: Karyawan ${userName} (ID: ${userId}) melakukan check-in.`

);

res.status(201).json({

*message:* `Halo ${userName}, check-in Anda berhasil pada pukul ${format(

waktuSekarang,

"HH:mm:ss",

{ timeZone }

)} WIB`,

*data:* formattedData,

});

};

exports.CheckOut = (*req*, *res*) => {

*const* { id: userId, nama: userName } = req.user;

*const* waktuSekarang = new Date();

*const* recordToUpdate = presensiRecords.find(

(*record*) => record.userId === userId && record.checkOut === null

);

if (!recordToUpdate) {

return res.status(404).json({

*message:* "Tidak ditemukan catatan check-in yang aktif untuk Anda.",

});

}

recordToUpdate.checkOut = waktuSekarang;

*const* formattedData = {

...recordToUpdate,

*checkIn:* format(recordToUpdate.checkIn, "yyyy-MM-dd HH:mm:ssXXX", {

timeZone,

}),

*checkOut:* format(recordToUpdate.checkOut, "yyyy-MM-dd HH:mm:ssXXX", {

timeZone,

}),

};

console.log(

`DATA TERUPDATE: Karyawan ${userName} (ID: ${userId}) melakukan check-out.`

);

res.json({

*message:* `Selamat jalan ${userName}, check-out Anda berhasil pada pukul ${format(

waktuSekarang,

"HH:mm:ss",

{ timeZone }

)} WIB`,

*data:* formattedData,

});

};

* + **controllers/reportController.js:**

*const* presensiRecords = require("../data/presensiData");

exports.getDailyReport = (*req*, *res*) => {

console.log("Controller: Mengambil data laporan harian dari array...");

res.json({

*reportDate:* new Date().toLocaleDateString(),

*data:* presensiRecords,

});

};

1. Menggabungkan Semuanya di Router
   * **routes/presensi.js:**

const express = require('express');

const router = express.Router();

const presensiController = require('../controllers/presensiController');

const { addUserData } = require('../middleware/permissionMiddleware');

router.use(addUserData);

router.post('/check-in', presensiController.CheckIn);

router.post('/check-out', presensiController.CheckOut);

module.exports = router;

* + **routes/reports.js:**  
    const express = require('express');

const router = express.Router();

* + const reportController = require('../controllers/reportController');
  + const { addUserData, isAdmin } = require('../middleware/permissionMiddleware');
  + router.get('/daily', [addUserData, isAdmin], reportController.getDailyReport);
  + module.exports = router;

1. Konfigurasi Akhir di server.js
   * Install necessary packages untuk Third-party middleware:

npm install cors helmet morgan

* + Modifikasi server.js:

*const* express = require("express");

*const* cors = require("cors");

*const* app = express();

*const* PORT = 3001;

*const* morgan = require("morgan");

*// Impor router*

*const* presensiRoutes = require("./routes/presensi");

*const* reportRoutes = require("./routes/reports");

*// Middleware*

app.use(cors());

app.use(express.json());

app.use(morgan("dev"));

app.use((*req*, *res*, *next*) => {

console.log(`${new Date().toISOString()} - ${req.method} ${req.url}`);

next();

});

app.get("/", (*req*, *res*) => {

res.send("Home Page for API");

});

*const* ruteBuku = require("./routes/books");

app.use("/api/books", ruteBuku);

app.use("/api/presensi", presensiRoutes);

app.use("/api/reports", reportRoutes);

app.listen(PORT, () => {

console.log(`Express server running at http://localhost:${PORT}/`);

});