**Most Used Express.js Third-Party Middleware Packages**

Express has many powerful third-party middleware to enhance your application. Here are some of the most commonly used ones:

**1. cors (Cross-Origin Resource Sharing)**

Used to enable cross-origin requests.

* **Installation:**

npm install cors

* **Usage:**

const cors = require('cors');

app.use(cors()); // Allows all origins

* **Restrict to Specific Origin:**

app.use(cors({ origin: 'https://example.com' })); // Allows only example.com

**2. morgan (HTTP Request Logger)**

Logs incoming requests in the console.

* **Installation:**

npm install morgan

* **Usage:**

const morgan = require('morgan');

app.use(morgan('dev')); // Logs requests in a readable format

📌 **Example Log:**

GET /home 200 12ms - 1.5kb

**3. helmet (Security Headers)**

Adds security-related HTTP headers to protect the app.

* **Installation:**

npm install helmet

* **Usage:**

const helmet = require('helmet');

app.use(helmet());

* **Security Headers Added:**
* X-Frame-Options
* X-XSS-Protection
* Content-Security-Policy

**4. express-rate-limit (Rate Limiting)**

Limits requests from the same IP to prevent **brute-force attacks**.

* **Installation:**

npm install express-rate-limit

npm install express-rate-limit

* **Usage:**

const rateLimit = require('express-rate-limit');

const limiter = rateLimit({

    windowMs: 15 \* 60 \* 1000, // 15 minutes

    max: 100, // Limit each IP to 100 requests per window

    message: 'Too many requests, please try again later.'

});

app.use(limiter);

**5. express-validator (Request Validation)**

Validates request data to prevent invalid input.

* **Installation:**

npm install express-validator

* **Usage:**

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

app.post('/register', [

    body('email').isEmail().withMessage('Invalid email'),

    body('password').isLength({ min: 6 }).withMessage('Password must be at least 6 characters')

], (req, res) => {

    const errors = validationResult(req);

    if (!errors.isEmpty()) {

        return res.status(400).json({ errors: errors.array() });

    }

    res.send('User registered');

});

**6. compression (Gzip Compression)**

Compresses responses to reduce bandwidth usage.

* **Installation:**

npm install compression

* **Usage:**

const compression = require('compression');

app.use(compression());

**7. cookie-parser (Parse Cookies)**

Parses cookies from incoming requests.

* **Installation:**

npm install cookie-parser

* **Usage:**

const cookieParser = require('cookie-parser');

app.use(cookieParser());

app.get('/cookies', (req, res) => {

    res.json(req.cookies);

});

**8. jsonwebtoken (JWT Authentication)**

Used to verify and generate JWT tokens.

* **Installation:**

npm install jsonwebtoken

* **Usage:**

const jwt = require('jsonwebtoken');

app.post('/login', (req, res) => {

    const user = { id: 1, username: 'admin' };

    const token = jwt.sign(user, 'secretkey', { expiresIn: '1h' });

    res.json({ token });

});

* **Verify Token:**

const verifyToken = (req, res, next) => {

    const token = req.headers['authorization'];

    if (!token) return res.status(403).send('No token provided');

    jwt.verify(token, 'secretkey', (err, decoded) => {

        if (err) return res.status(401).send('Invalid token');

        req.user = decoded;

        next();

    });

};

app.get('/protected', verifyToken, (req, res) => {

    res.send('You are authorized');

});

**9. express-session (Session Management)**

Stores session data on the server.

* **Installation:**

npm install express-session

* **Usage:**

const session = require('express-session');

app.use(session({

    secret: 'mysecret',

    resave: false,

    saveUninitialized: true,

    cookie: { secure: false }

}));

app.get('/session', (req, res) => {

    req.session.user = 'John Doe';

    res.send('Session stored');

});

**10. multer (File Uploads)**

Handles file uploads in Express.

* **Installation:**

npm install multer

* **Usage:**

const session = require('express-session');

const multer = require('multer');

const storage = multer.diskStorage({

    destination: (req, file, cb) => cb(null, 'uploads/'),

    filename: (req, file, cb) => cb(null, Date.now() + '-' + file.originalname)

});

const upload = multer({ storage });

app.post('/upload', upload.single('file'), (req, res) => {

    res.send('File uploaded: ' + req.file.filename);

});

**Summary Table**

| **Middleware** | **Purpose** |
| --- | --- |
| **cors** | Enables Cross-Origin requests |
| **morgan** | Logs incoming requests |
| **helmet** | Adds security headers |
| **express-rate-limit** | Prevents excessive requests (DDoS protection) |
| **express-validator** | Validates request data |
| **compression** | Compresses responses |
| **cookie-parser** | Parses cookies from requests |
| **jsonwebtoken** | Handles JWT authentication |
| **express-session** | Manages user sessions |
| **multer** | Handles file uploads |

Would you like an example for a specific middleware?