**express-session Middleware in Express.js**

express-session is a middleware that enables **session management** in Express.js. It allows storing user data (like authentication state) across multiple requests.

**1. Installing express-session**

Before using it, install the package:

npm install express-session

**2. Basic Usage**

Import and configure express-session:

const express = require('express');

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

const app = express();

// Use session middleware

app.use(session({

secret: 'mySecretKey', // Secret key to sign the session ID cookie

resave: false, // Prevents resaving sessions that haven't changed

saveUninitialized: true, // Saves uninitialized sessions (use false for login-required sites)

cookie: { secure: false } // Set to true if using HTTPS

}));

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

res.send('Welcome! Use /set-session and /get-session routes.');

});

// Set a session variable

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

req.session.username = 'JohnDoe'; // Store session data

res.send('Session data has been set!');

});

// Retrieve session data

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

res.json({ sessionData: req.session });

});

// Destroy session

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

req.session.destroy((err) => {

if (err) {

return res.status(500).send('Error destroying session');

}

res.send('Session destroyed!');

});

});

app.listen(3000, () => console.log('Server running on port 3000'));

**3. Understanding Options**

| **Option** | **Description** |
| --- | --- |
| secret | A secret key used to sign the session ID cookie (Required) |
| resave | Forces session to be saved even if unmodified (default: true, but set to false for performance) |
| saveUninitialized | Saves **uninitialized** sessions (should be false for login-required apps) |
| cookie.secure | Ensures cookies are only sent over HTTPS (true for production) |

**4. Configuring Session Expiry**

You can configure the session expiration by setting cookie.maxAge:

app.use(session({

secret: 'mySecretKey',

resave: false,

saveUninitialized: true,

cookie: { maxAge: 60000 } // Session expires in 60 seconds

}));

📌 **After 60 seconds**, the session will be automatically removed.

**5. Storing Sessions in a Database**

By default, sessions are stored **in-memory**, which is **not ideal for production**.  
Use **connect-mongo** to store sessions in MongoDB.

📌 **Installation:**

npm install connect-mongo

📌 **Usage with MongoDB:**

const MongoStore = require('connect-mongo');

app.use(session({

secret: 'mySecretKey',

resave: false,

saveUninitialized: true,

store: MongoStore.create({ mongoUrl: 'mongodb://localhost:27017/sessions' }),

cookie: { maxAge: 60000 }

}));

Other session stores:

* **Redis** → connect-redis
* **MySQL/PostgreSQL** → connect-session-sequelize
* **MemoryStore** (default, not recommended for production)

**6. Checking if a Session Exists**

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

if (req.session.username) {

res.send(`Session active for: ${req.session.username}`);

} else {

res.send('No active session');

}

});

**7. Destroying a Session**

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

req.session.destroy(() => {

res.send('User logged out, session destroyed');

});

});

**8. Summary**

| **Feature** | **Example** |
| --- | --- |
| **Install express-session** | npm install express-session |
| **Set session data** | req.session.username = 'JohnDoe' |
| **Retrieve session data** | req.session |
| **Destroy session** | req.session.destroy() |
| **Set session expiry** | { cookie: { maxAge: 60000 } } |
| **Store sessions in MongoDB** | connect-mongo |

Would you like an example with authentication? 🚀