

Table of Contents

- ◆
- [1. Installation](#)
- [2. Create a Simple Express App \(Manual Setup\)](#)
- [3. Basic Routing](#)
- [4. Middleware](#)
- [5. Static Files](#)
- [6. Handling Form Data \(POST\)](#)
- [7. Error Handling](#)
- [8. Templating \(EJS Setup\)](#)
- [9. JSON Response](#)
- [10. Redirects](#)
- [11. Route Grouping \(Router\)](#)
- [12. File Upload \(Multer\)](#)
- [13. Connect to MongoDB \(Mongoose\)](#)
- [14. API Example \(CRUD Operations\)](#)
- [15. Useful Middleware](#)
- [16. Common Commands](#)

1. Installation

Prerequisites:

- Install Node.js from <https://nodejs.org/>

Install Express Globally:

```
npm install express-generator -g
```

Create Express App:

```
express myapp  
cd myapp  
npm install  
npm start
```

2. Create a Simple Express App (Manual Setup)

1. Initialize Project:

```
mkdir myapp
cd myapp
npm init -y
npm install express
```

2. Create index.js:

```
const express = require('express');
const app = express();
const PORT = 3000;

// Basic Route
app.get('/', (req, res) => {
    res.send('Hello, Express!');
});

// Start Server
app.listen(PORT, () => {
    console.log(`Server running at http://localhost:${PORT}`);
});
```

3. Run App:

```
node index.js
```

- Visit <http://localhost:3000>
-

3. Basic Routing

```
app.get('/', (req, res) => {
    res.send('Home Page');
});
```

```
app.post('/submit', (req, res) => {
  res.send('Form Submitted');
});

app.put('/update', (req, res) => {
  res.send('Data Updated');
});

app.delete('/delete', (req, res) => {
  res.send('Data Deleted');
});
```

- Route Parameters:

```
app.get('/user/:id', (req, res) => {
  res.send(`User ID: ${req.params.id}`);
});
```

- Query Parameters:

```
app.get('/search', (req, res) => {
  res.send(`Query: ${req.query.q}`);
});
```

4. Middleware

Built-in Middleware:

```
app.use(express.json()); // Parse JSON
app.use(express.urlencoded({ extended: true })); // Parse URL-encoded
data
```

Custom Middleware:

```
app.use((req, res, next) => {
  console.log(`${req.method} ${req.url}`);
  next();
});
```

5. Static Files

```
app.use(express.static('public')); // Serve static files from  
"public" folder
```

- Access CSS, JS, images directly by placing them in the public directory.
-

6. Handling Form Data (POST)

```
app.post('/submit', (req, res) => {  
  const { name, email } = req.body;  
  res.send(`Received: ${name}, ${email}`);  
});
```

7. Error Handling

```
app.use((req, res, next) => {  
  res.status(404).send('Page Not Found');  
});  
  
// Custom Error Handler  
app.use((err, req, res, next) => {  
  console.error(err.stack);  
  res.status(500).send('Something broke!');  
});
```

8. Templating (EJS Setup)

1. Install EJS:

```
npm install ejs
```

2. Set EJS as View Engine:

```
app.set('view engine', 'ejs');
```

3. Create views/index.ejs:

```
<!DOCTYPE html>
<html>
<head><title>Express EJS</title></head>
<body>
  <h1>Hello, <%= name %>!</h1>
</body>
</html>
```

4. Render EJS Template:

```
app.get('/', (req, res) => {
  res.render('index', { name: 'Express' });
});
```

9. JSON Response

```
app.get('/api/user', (req, res) => {
  res.json({ id: 1, name: 'John Doe' });
});
```

10. Redirects

```
app.get('/google', (req, res) => {
  res.redirect('https://google.com');
});
```

11. Route Grouping (Router)

```
const userRouter = express.Router();

userRouter.get('/', (req, res) => {
    res.send('User List');
});

userRouter.get('/:id', (req, res) => {
    res.send(`User ID: ${req.params.id}`);
});

app.use('/users', userRouter);
```

12. File Upload (Multer)

1. Install Multer:

```
npm install multer
```

2. Upload File:

```
const multer = require('multer');
const upload = multer({ dest: 'uploads/' });

app.post('/upload', upload.single('file'), (req, res) => {
    res.send(`File uploaded: ${req.file.originalname}`);
});
```

13. Connect to MongoDB (Mongoose)

1. Install Mongoose:

```
npm install mongoose
```

2. Connect to DB:

```

const mongoose = require('mongoose');
mongoose.connect('mongodb://localhost:27017/mydb', { useNewUrlParser: true });

const User = mongoose.model('User', { name: String });
app.get('/add-user', async (req, res) => {
    const user = new User({ name: 'John' });
    await user.save();
    res.send('User added');
});

```

14. API Example (CRUD Operations)

```

let users = [
    { id: 1, name: 'Alice' },
    { id: 2, name: 'Bob' }
];

// GET all users
app.get('/api/users', (req, res) => {
    res.json(users);
});

// GET user by ID
app.get('/api/users/:id', (req, res) => {
    const user = users.find(u => u.id === req.params.id);
    if (user) res.json(user);
    else res.status(404).send('User not found');
});

// POST new user
app.post('/api/users', (req, res) => {
    const newUser = { id: users.length + 1, name: req.body.name };
    users.push(newUser);
    res.status(201).json(newUser);
});

```

```
// DELETE user
app.delete('/api/users/:id', (req, res) => {
  users = users.filter(u => u.id != req.params.id);
  res.status(204).send();
});
```

15. Useful Middleware

```
const cors = require('cors');
app.use(cors()); // Enable CORS

const morgan = require('morgan');
app.use(morgan('dev')); // Logging requests
```

16. Common Commands

```
npm start                      # Start server
nodemon index.js                # Auto-restart on change (install
nodemon globally)
npm install --save express       # Install express
```