

기본 서버 설정 (Basic Server Setup)

Express는 빠르고 유연하며 미니멀한 Node.js 웹 및 API 애플리케이션 프레임워크입니다.

```
import express from 'express';
const app = express();
const PORT = process.env.PORT || 3000;
```

```
// 내장 미들웨어
app.use(express.json()); // JSON 바디 파싱
app.use(express.urlencoded({ extended: true }));
app.use(express.static('public')); // 정적 파일 제공
```

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

```
app.listen(PORT, () => {
  console.log(`Server running on port ${PORT}`);
});
```

라우팅 (Routing & HTTP Methods)

기본 라우트

```
• app.get('/path', (req, res) => { ... })
• app.post('/path', (req, res) => { ... })
• app.put('/path', (req, res) => { ... })
• app.patch('/path', (req, res) => { ... })
• app.delete('/path', (req, res) => { ... })
• app.all('/path', (req, res) => { ... })
  (모든 HTTP 메서드 처리)
```

모듈식 라우터 (express.Router)

```
// routes/users.js
import { Router } from 'express';
const router = Router();
```

```
router.get('/', (req, res) => {
  res.json({ users: [] });
});
```

```
router.get('/:id', (req, res) => {
  const { id } = req.params;
  res.json({ id, name: 'Alice' });
});
```

```
export default router;
```

```
// app.js
import userRouter from './routes/users.js';
app.use('/api/users', userRouter);
```

요청 및 응답 객체 (Request & Response)

Request (req)

- **req.params**: URL 경로 파라미터 (`/users/:id` - `> req.params.id`)
- **req.query**: URL 쿼리 스트링 (`?page=2` - `> req.query.page`)
- **req.body**: 파싱된 요청 본문 데이터 (JSON 등)
- **req.headers**: 요청 헤더 객체
- **req.cookies**: 파싱된 쿠키 (`cookie-parser` 필요)
- **req.ip / req.path / req.method**: 클라이언트 IP, 경로, 메서드

Response (res)

- **res.status(200)**: HTTP 상태 코드 설정
- **res.json({ success: true })**: JSON 응답 전송
- **res.send('Text')**: 문자열/버퍼 전송
- **res.sendStatus(404)**: 상태 코드와 기본 메시지 즉시 전송
- **res.redirect('/new-path')**: 리다이렉트
- **res.sendFile(path.resolve('file.pdf'))**: 파일 다운로드 전송
- **res.setHeader('X-Custom', 'val')**: 응답 헤더 설정

미들웨어 (Middleware Architecture)

요청-응답 주기(Cycle) 중간에서 실행되는 함수 체인 (`req, res, next`).

애플리케이션 / 라우터 레벨 미들웨어

```
// 로깅 미들웨어
const logger = (req, res, next) => {
```

```
  console.log(`${req.method} ${req.url} - ${Date.now()}`);
  next(); // 다음 미들웨어로 제어 전달
};
app.use(logger);
```

```
// 특정 라우트에 미들웨어 적용
```

```
const requireAuth = (req, res, next) => {
  if (!req.headers.authorization) {
    return res.status(401).json({ error: '인증 필요' });
  }
  next();
};
app.get('/api/secret', requireAuth, (req, res) => {
  res.json({ secret: 'data' });
});
```

에러 핸들링 미들웨어 (4개의 매개변수)

```
app.use((err, req, res, next) => {
  console.error(err.stack);
  res.status(err.status || 500).json({
    error: err.message || '내부 서버 오류',
  });
});
```

필수 서드파티 미들웨어

- **cors**: Cross-Origin Resource Sharing 활성화


```
import cors from 'cors';
app.use(cors({ origin: 'http://localhost:3000' }));
```
- **helmet**: 보안 관련 HTTP 헤더 자동 설정


```
import helmet from 'helmet';
app.use(helmet());
```
- **morgan**: HTTP 요청 로거


```
import morgan from 'morgan';
app.use(morgan('dev'));
```
- **cookie-parser**: 쿠키 파싱


```
import cookieParser from 'cookie-parser';
app.use(cookieParser('secret-key'));
```