



2장 Express

강사 김영석

A top-down view of a wooden desk. On the desk, there is a silver laptop with a black keyboard, a pair of black-rimmed glasses, a white coffee cup filled with dark liquid, and a small green succulent in a pot. The wood grain of the desk is clearly visible.

CONTENT

1 Express 란

2 Express 설치

3 Express 구조

1. Express 란

✓ Express 는 Node 웹 애플리케이션 프레임워크다.

✓ 특징

- 최소화
- 유연함
- 웹 애플리케이션 프레임워크
- 속도
- 다양성과 크기



✓ 최소화

- 웹에 필요한 최소한의 프레임워크만 제공하고 필요한 부분은 설치 하거나 필요 없는 부분은 제거 할 수 있다.

✓ 유연함

- 단순한 것만 처리한다. HTTP 요청을 받고 요청에 응답하는 단순한 기능만을 제공한다.

✓ 웹 애플리케이션 프레임워크

- 서버로써의 역할도 하면서 데이터를 처리할 수 있다.



✓ 속도

- 웹의 트래픽이 많아도 속도 면에서 빠른 성능을 보여준다.

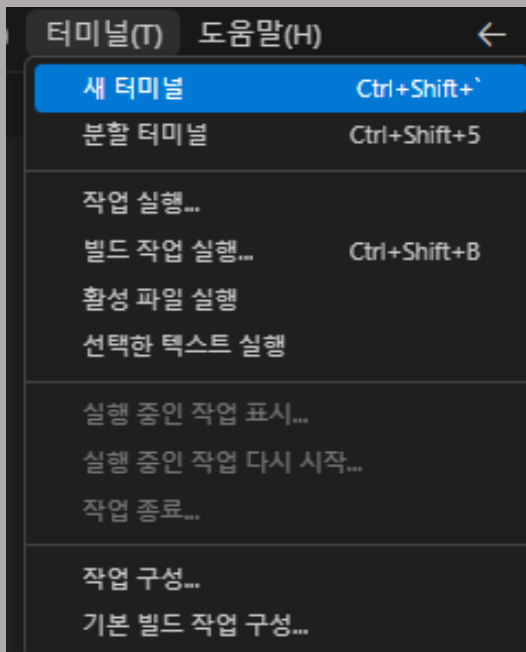
✓ 다양성과 크기

- 여러 커뮤니티를 통해 다양한 생태계가 있으며, 원하는 미들웨어를 추가 삭제 할 수 있도록 하여 크기를 작게 만들 수 있고 성능 또한 좋다.


2. Express 설치

✓ 폴더 생성 및 버전 확인

```
> express_first
```



```
PS F:\node_workspace> cd express_first
PS F:\node_workspace\express_first> node -v
v22.13.1
PS F:\node_workspace\express_first> npm -v
10.9.2
```



```
PS F:\node_workspace\express_first> npm -v
npm : C:\Program Files\nodejs\npm.ps1 파일을 로드할 수 없습니다. C:\Program Files\nodejs\npm.ps1 파일이 디지털 서명되지 않았습니다.
/go.microsoft.co
m/fwlink/?LinkID=135170)를 참조하십시오..
위치 줄:1 문자:1
+ npm -v
+ ~~~~
+ CategoryInfo          : 보안 오류: (:) [], PSSecurityException
+ FullyQualifiedErrorId : UnauthorizedAccess
PS F:\node_workspace\express_first> Set-ExecutionPolicy -Scope CurrentUser -ExecutionPolicy RemoteSigned
PS F:\node_workspace\express_first> npm -v
10.9.2
```

만일 위와 같이 ‘디지털 서명이 되지 않았습니다.’가 출력되면 아래 내용을 추가 한다.
Set-ExecutionPolicy –Scope CurrentUser –ExecutionPolicy RemoteSigned

✓ 초기화

```
● PS F:\node_workspace\express_first> npm init -y
Wrote to F:\node_workspace\express_first\package.json:

{
  "name": "express_first",
  "version": "1.0.0",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "keywords": [],
  "author": "",
  "license": "ISC",
  "description": ""
}
```


✓ Express 설치

```
● PS F:\node_workspace\express_first> npm install express  
  
added 69 packages, and audited 70 packages in 3s  
  
14 packages are looking for funding  
  run `npm fund` for details  
  
found 0 vulnerabilities
```

✓ nodemon 설치

- nodemon 은 파일이 수정 될 때마다 서버를 재시작 하는 걸 자동으로 처리 해 주는 도구
- -g 옵션은 시스템 전체에 설치 하는 옵션
- 만일 개발환경에서만 사용하고 싶다면 `npm install --save-dev nodemon` 을 사용하면 됨

```
● PS F:\node_workspace\express_first> npm install -g nodemon  
  
added 29 packages in 1s  
  
4 packages are looking for funding  
run `npm fund` for details
```

✓ Express 삭제

```
● PS F:\node_workspace\express_first> npm uninstall express  
  
removed 69 packages, and audited 1 package in 523ms  
  
found 0 vulnerabilities
```

3. Express 구조

✓ Express 의 구조를 딱히 정해져 있지는 않지만 아래와 같은 형식으로 사용됨

express_first/

— node_modules/	# 설치된 패키지
— public/	# 정적 파일(css, js, 이미지)
— routes/	# 라우팅 파일 저장 폴더
— index.js	# 기본 라우트
— users.js	# 사용자 관련 라우트
— views/	# 템플릿 파일 (HTML 파일)
— app.js	# 메인 애플리케이션 파일(서버 실행 파일)
— package.json	# 프로젝트 정보 및 의존성 목록
— .env	# 환경 변수 파일


```
✓ express_first
  > node_modules
  JS app.js
  {} package-lock.json
  {} package.json
```

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

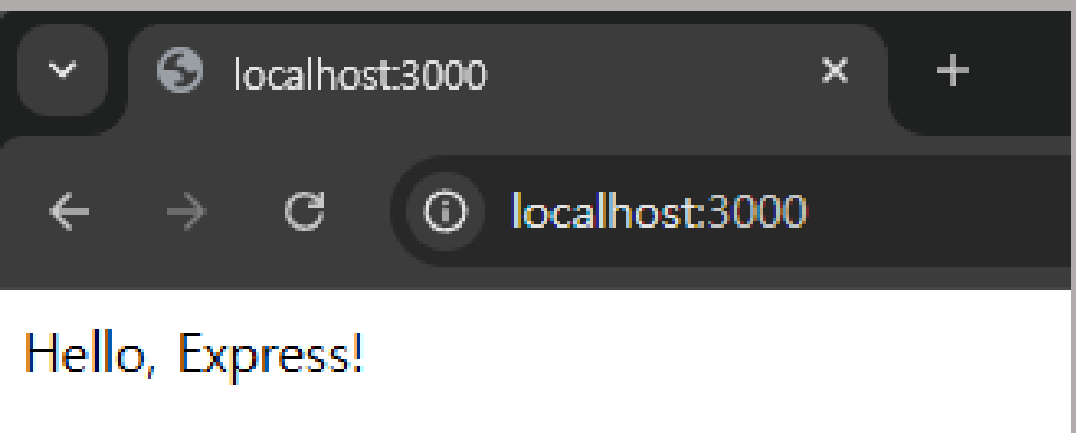
// 미들웨어 설정
app.use(express.json());
app.use(express.urlencoded({ extended: true }));

// 기본 라우트
Tabnine | Edit | Test | Explain | Document
app.get('/', (req, res) => {
  res.send('Hello, Express!');
});

// 서버 실행
Tabnine | Edit | Test | Explain | Document
app.listen(port, () => {
  console.log(`Server is running at http://localhost:\${port}`);
});
```

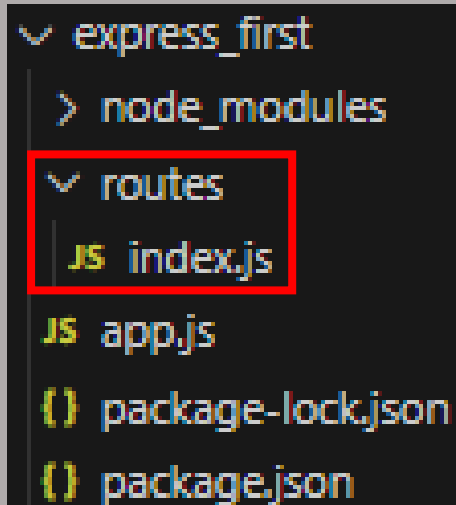



```
PS F:\node_workspace\express_first> nodemon app.js
[nodemon] 3.1.9
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): *.*
[nodemon] watching extensions: js,mjs,cjs,json
[nodemon] starting `node app.js`
Server is running at http://localhost:3000
```



✓ 라우트 분리

```
JS app.js x
express_first > JS app.js > ...
 1  const express = require('express');
 2  const app = express();
 3  const port = 3000;
 4  const indexRouter = require('./routes/index');
 5
 6  // 라우트 분리 - routes/index.js 로 전달
 7  app.use('/', indexRouter);
 8
 9  // 서버 실행
   Tabnine | Edit | Test | Explain | Document
10  app.listen(port, () => {
11    console.log(`Server is running at http://localhost:${port}`);
12  });
```

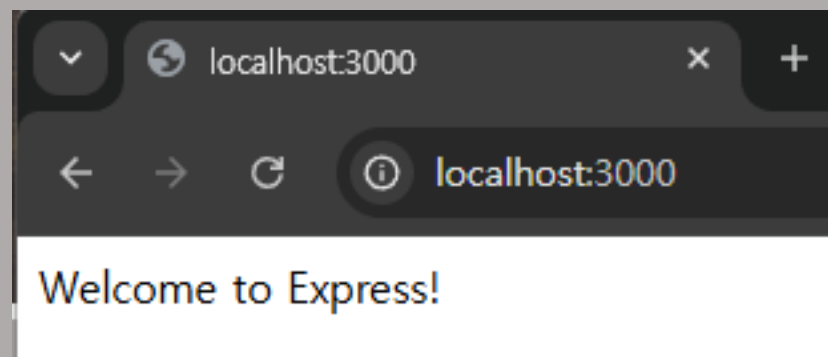


```
const express = require('express');  
const router = express.Router();
```

Tabnine | Edit | Test | Explain | Document

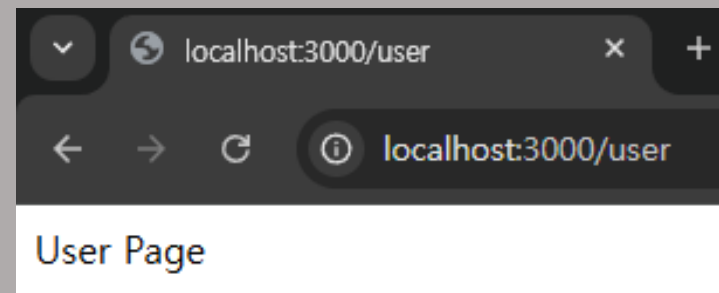
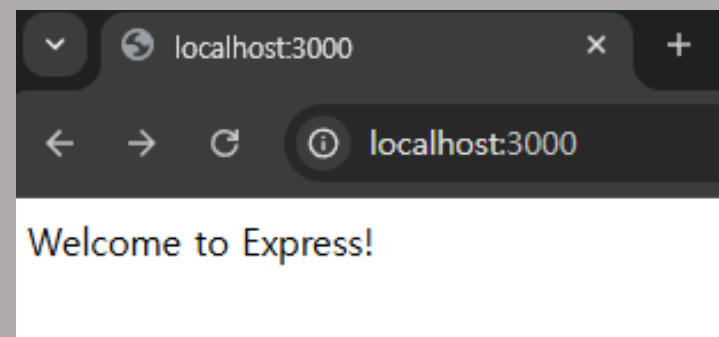
```
router.get('/', (req, res) => {  
  res.send('Welcome to Express!');  
});
```

```
module.exports = router;
```



✓ 경로가 여러 개로 분리

```
JS index.js x
express_first > routes > JS index.js > ...
1  const express = require('express');
2  const router = express.Router();
3
4  // localhost:3000 에서 실행
   Tabnine | Edit | Test | Explain | Document
5  router.get('/', (req, res) => {
6    res.send('Welcome to Express!');
7  });
8
9  // localhost:3000/user 에서 실행
   Tabnine | Edit | Test | Explain | Document
10 router.get('/user', (req, res) => {
11   res.send('User Page');
12 });
13 module.exports = router;
```



✓ 경로를 다른 파일로 수정

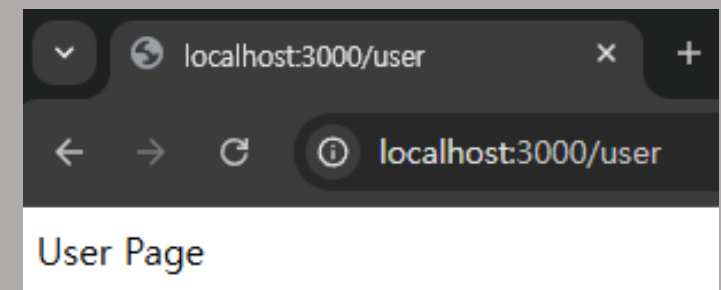
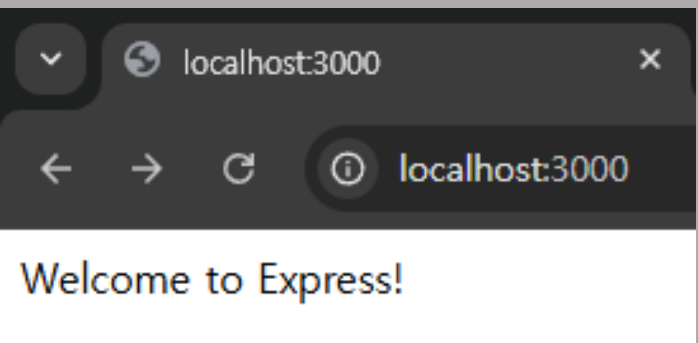
```
JS app.js x
express_first > JS app.js > ...
 1  const express = require('express');
 2  const app = express();
 3  const port = 3000;
 4  const indexRouter = require('./routes/index');
 5  const userRouter = require('./routes/user');
 6
 7  // 라우트 분리 - routes/index.js 로 전달
 8  app.use('/', indexRouter);
 9  // 라우트 분리 - routes/user.js 로 전달
10  app.use('/user', userRouter);
11
12  // 서버 실행
   Tabnine | Edit | Test | Explain | Document
13  app.listen(port, () => {
14    console.log(`Server is running at http://localhost:${port}`);
15  });
```


JS user.js




express_first > routes > JS user.js > ...

```
1 const express = require('express');  
2 const router = express.Router();  
3  
4 // localhost:3000/user 에서 실행  
   Tabnine | Edit | Test | Explain | Document  
5 router.get('/', (req, res) => {  
6   res.send('User Page');  
7 });  
8  
9 module.exports = router;
```



✓ HTML, CSS, JavaScript 파일 이용하는 방법

```
JS app.js  X
express_first > JS app.js > ...
1  const express = require('express');
2  const app = express();
3  const port = 3000;
4  const indexRouter = require('./routes/index');
5  const userRouter = require('./routes/user');
6  const path = require('path');
7
8  // 정적 파일 제공 설정 (HTML, CSS, JS)
9  app.use(express.static(path.join(__dirname, 'public')));
10
```



```
11 // 라우트 분리 - routes/index.js 로 전달
12 app.use('/', indexRouter);
13 // 라우트 분리 - routes/user.js 로 전달
14 app.use('/user', userRouter);
15
16 // 서버 실행
  Tabnine | Edit | Test | Explain | Document
17 app.listen(port, () => {
18   console.log(`Server is running at http://localhost:${port}`);
19 });
```

JS index.js x

express_first > routes > JS index.js > ...

```
1 const express = require('express');
2 const router = express.Router();
3
4 // localhost:3000 에서 실행 - public/index.html 반환
  Tabnine | Edit | Test | Explain | Document
5 router.get('/', (req, res) => {
6   res.sendFile(path.join(__dirname, '../public/index.html'));
7 });
8
9 module.exports = router;
```

JS user.js x

express_first > routes > JS user.js > ...

```
1 const express = require('express');
2 const router = express.Router();
3
4 // localhost:3000/user 에서 실행
  Tabnine | Edit | Test | Explain | Document
5 router.get('/', (req, res) => {
6   res.sendFile(path.join(__dirname, '../public/user.html'));
7 });
8
9 module.exports = router;
```

```

  express_first
    > node_modules
    < public
      index.html
      JS script.js
      # style.css
      user.html
    < routes
      JS index.js
      JS user.js
      JS app.js
      ( ) package-lock.json
      ( ) package.json

```

```

index.html x
express_first > public > index.html > html
1  <!DOCTYPE html>
2  <html lang="en">
3  <head>
4      <meta charset="UTF-8">
5      <meta name="viewport" content="width=device-width, initial-scale=1.0">
6      <title>메인화면</title>
7      <script type="text/javascript" src="script.js"></script>
8      <link rel="stylesheet" href="style.css">
9  </head>
10 <body>
11     index 페이지 입니다.
12     <input type="button" onclick="fn_user()" value="user 페이지 이동">
13 </body>
14 </html>

```


user.html x

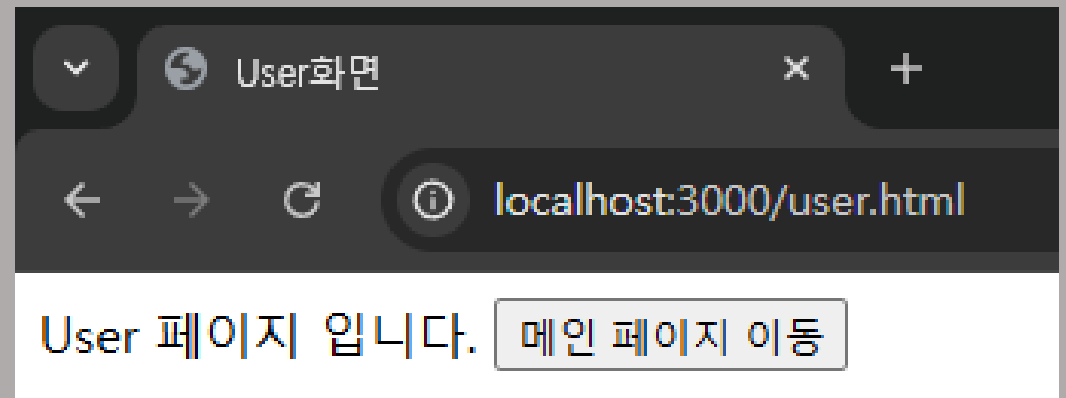
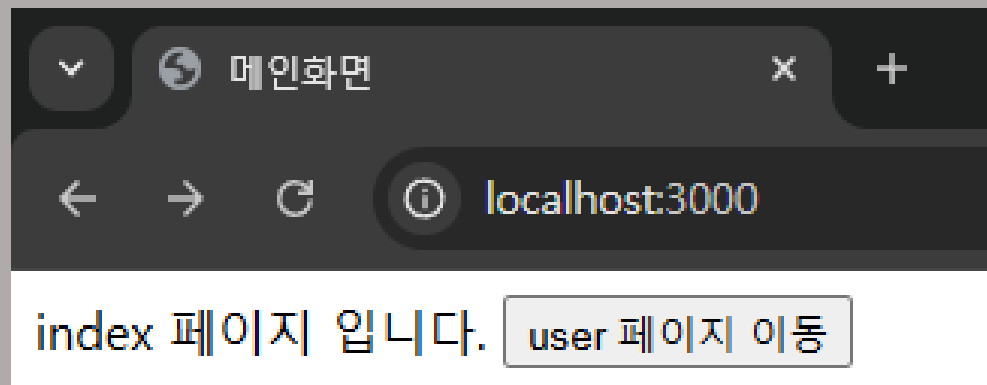
express_first > public > user.html > html

```
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4   <meta charset="UTF-8">
5   <meta name="viewport" content="width=device-width, initial-scale=1.0">
6   <title>User화면</title>
7   <script type="text/javascript" src="script.js"></script>
8   <link rel="stylesheet" href="style.css">
9 </head>
10 <body>
11   User 페이지 입니다.
12   <input type="button" onclick="fn_index()" value="메인 페이지 이동">
13 </body>
14 </html>
```



```
JS script.js x
express_first > public > JS script.js > fn_index
Tabnine | Edit | Test | Explain | Document
1 function fn_user(){
2     // user.html 로 이동
3     location.href = "user.html";
4 }
5
Tabnine | Edit | Test | Explain | Document
6 function fn_index(){
7     location.href = "index.html";
8 }
```

```
# style.css x
express_first > public > # style.css > button
1 button {
2     background-color: burlywood;
3     background-position: center;
4 }
```



A photograph of a server room with rows of server racks on both sides of a central aisle. The racks have glass doors and internal components are visible, with many small blue lights glowing. The ceiling has several long, rectangular light fixtures. The overall atmosphere is dimly lit, emphasizing the blue light from the servers.

수고하셨습니다.