

MEAN 스택 활용 웹 개발

Express 개요



2016년도 2학기 2학년 방과후학교

Express 소개

- Express 프레임워크는 경량의 웹 프레임워크로 HTTP 기반의 서비스를 쉽고 간단하게 작성할 수 있다.
- HTTP 요청 분석과 응답 메시지를 쉽고 간단하게 만들 수 있다.
- Express는 미들웨어라는 모듈을 이용해서 동작한다.
- 템플릿 엔진도 제공해서 뷰와 제어 코드를 분리해서 HTML을 쉽게 작성할 수 있다.
- `npm install express`

Express
Fast, unopinionated,
minimalist web framework
for [Node.js](https://nodejs.org/)

Express 소개

■ expressjs.com

Express

[Home](#) [Getting started](#) [Guide](#) [API reference](#) [Advanced topics](#) [Resources](#)

Express

Fast, unopinionated,
minimalist web framework
for **Node.js**

```
$ npm install express --save
```

Web Applications

Express is a minimal and flexible Node.js web application framework that provides a robust set of features for web and mobile applications.

APIs


With a myriad of HTTP utility methods and middleware at your disposal, creating a robust API is quick and easy.

Performance

Express provides a thin layer of fundamental web application features, without obscuring Node.js features that you know and love.

Frameworks

Many [popular frameworks](#) are based on Express.

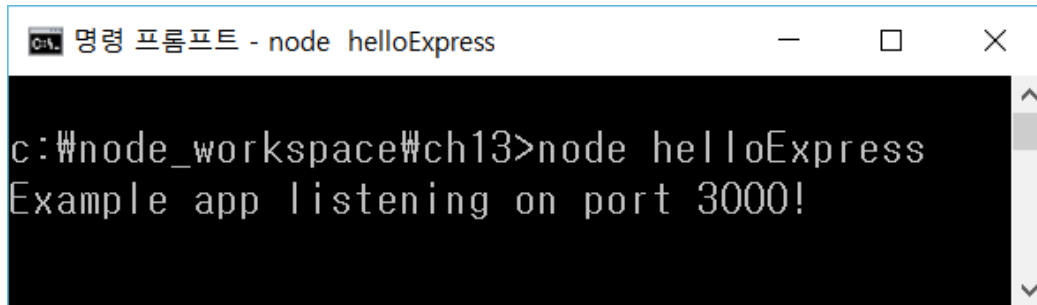
 June 16, 2016

Express 4.14.0 released

Express 4.14.0 is a bunch of bug fixes, security update, performance improvements, and minor feature additions. For details refer to the [4.14.0 changelog](#).

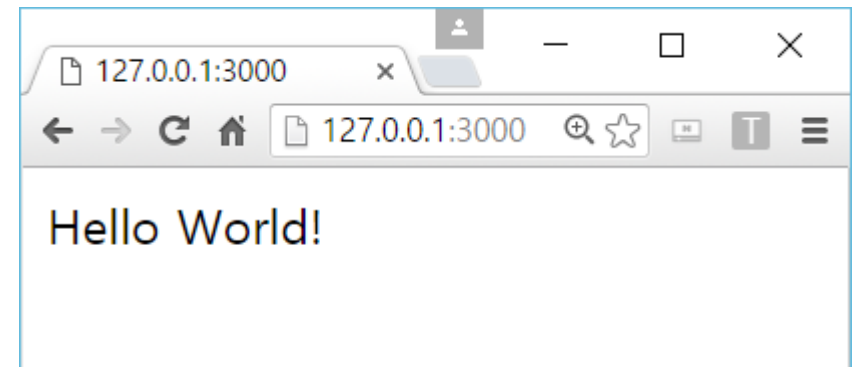
Hello, World

```
var express = require('express');  
var app = express();  
  
app.get('/', function (req, res) {  
  res.send('Hello World!');  
});  
  
app.listen(3000, function () {  
  console.log('Example app listening on port 3000!');  
});
```



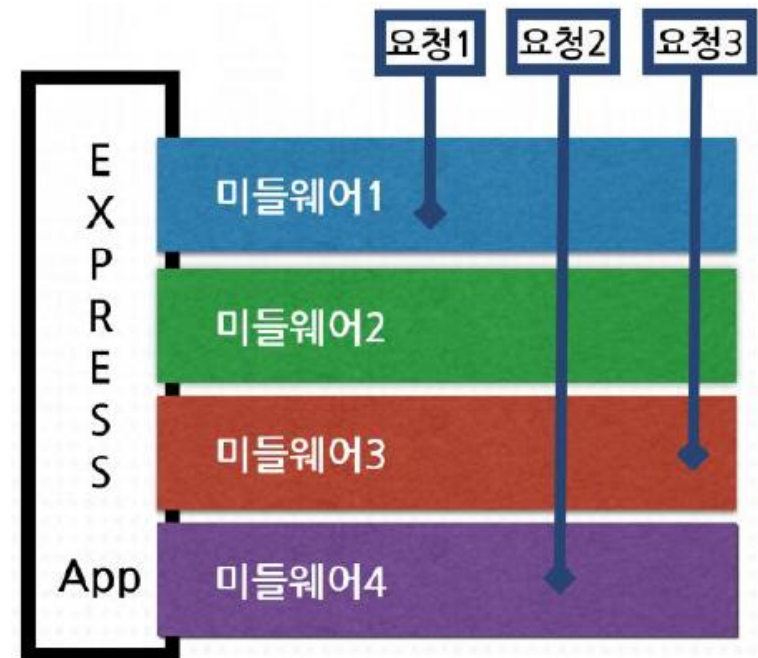
A terminal window titled '명령 프롬프트 - node helloExpress' showing the command 'node helloExpress' being executed in the directory 'c:\#node_workspace#ch13'. The output is 'Example app listening on port 3000!'.

```
C:\#node_workspace#ch13>node helloExpress  
Example app listening on port 3000!
```



Express 미들웨어란?

- 미들웨어는 요청을 분석하고 응답을 하는 작은 모듈로 Express는 여러 개의 미들웨어를 조합해서 동작한다.
- 하나의 요청은 Express에 설정(mount)된 미들웨어 중 하나의 미들웨어에서 요청 분석과 응답이 끝난다.
- 또한 하나의 요청은 여러 미들웨어의 협력으로 요청 분석과 응답 과정이 끝나기도 한다.



Express 미들웨어 사용 설정

- `app.use([미들웨어])`
- Express에서 미들웨어를 설정(mount)하는 함수는 `use()`다.
- 미들웨어는 요청과 응답을 파라미터로 하는 함수 형태다.

```
var express = require('express');
var app = express();

app.use(function (req, res) {
  res.send('Hello World!');
});

app.listen(3000, function () {
  console.log('Example app listening on port 3000!');
});
```

Express 미들웨어 연속 사용

- 하나의 미들웨어에서 다음 미들웨어가 실행될 수 있도록 하려면 미들웨어의 3번째 파라미터로 next를 사용한다.
- next();를 호출하면 다음 미들웨어가 실행된다.

```
// 로그 남기는 미들웨어
app.use(function (req, res, next) {
  console.log('url:', req.url);
  next(); // 다음 미들웨어 실행
});
```

```
app.use(function (req, res) {
  res.send('Hello, Express!');
});
```

Express 요청 분석

- Express를 사용하면서 얻게 되는 장점 중 하나는 요청 분석이 쉬워진다.
 - ✓ req.query : 쿼리 문자열
 - ✓ req.path : 요청 URL 경로
 - ✓ req.params : URL의 파라미터
 - ✓ req.cookie : 요청 메시지 내 쿠키 (cookie parser 필요)
 - ✓ req.body : 요청 메시지 바디 (body parser 필요)

Express 응답 처리

- Express를 사용하면서 다양한 형태의 응답 처리를 손쉽게 할 수 있다.
 - ✓ `res.json()` : JSON 응답 메시지 전송
 - ✓ `res.redirect()` : 리다이렉션 응답 전송
 - ✓ `res.render()` : 템플릿으로 렌더링
 - ✓ `res.send()` : JSON, HTML, Buffer 전송, 메시지 헤더에 Content-Type 자동 설정
 - ✓ `res.sendStatus()` : 상태 코드와 상태 메시지 전송
 - ✓ `res.status()` : 상태 코드 설정
 - ✓ `res.download()` : 파일 다운로드

Express 미들웨어 동작 순서

- 미들웨어의 조합으로 동작 시 미들웨어 결합 순서가 중요하다.
- 보통 미들웨어는 파비콘을 처리하는 미들웨어와 로그를 남기는 미들웨어가 먼저 동작하도록 작성한다.
- 그리고 정적 파일 요청을 담당하는 미들웨어를 설정하고 이후에는 서비스 동작 미들웨어가 동작하도록 작성한다.

Express 미들웨어 종류

- Express 내장 미들웨어
 - ✓ 예) `express.static` : 정적 파일 요청 처리 미들웨어
- 모듈 설치와 모듈 로딩 과정이 필요한 써드파티 미들웨어
 - ✓ 예) 쿠키 분석 용 미들웨어

```
var cookieParser = require('cookie-parser');  
  
app.use(cookieParser());
```

정적 파일 처리 미들웨어

- `express.static(root [, options])` // root : 경로 정보 필수
- `app.use(express.static('images'));`
- `app.use(express.static('files'));` // 여러 개 설정 가능
- 정적 파일에 대한 요청이 오면 images 폴더에서 파일을 찾아서 응답한다.
- images 폴더에 해당 이미지가 없으면 files 폴더에서 찾는다.
 - ✓ `http://127.0.0.1/cat.jpg -> ./images/cat.jpg`
 - ✓ `http://127.0.0.1/dog.jpg -> ./images/dog.jpg`

정적 파일 처리 미들웨어

- 정적 파일 처리 미들웨어 설정 시 가상 경로 설정이 가능하다.
- `app.use('/static', express.static('images'));`
 - ✓ `http://127.0.0.1/static/cat.jpg` -> `./images/cat.jpg`
 - ✓ `http://127.0.0.1/static/dog.jpg` -> `./images/dog.jpg`

Body Parser

- 바디 파서(body-parser) 미들웨어는 HTTP 메시지 바디를 분석해주는 미들웨어다.
- 요청 분석을 도와주는 다양한 미들웨어를 이용하면 요청을 분석하는 긴 코드를 작성하지 않아도 된다.
- `npm install body-parser`

```
// 바디 메시지가 JSON인 요청을 분석하는 바디파서 설정
app.use(bodyParser.json());
```

```
// URLEncoded 방식으로 전달되는 메시지의 바디를 분석
// extended: false이면 querystring으로 파싱
app.use(bodyParser.urlencoded({ extended: false }));
```

```
// bodyParser에 의해 분석된 데이터 사용
function task(req, res) {
  var name = req.body.name;
  ...
}
```

동적 파라미터 처리

- `http://127.0.0.1:3000/music/1`, `http://127.0.0.1:3000/music/2`
- URL 경로에서 동적으로 변경되는 부분을 처리하기 위해 콜론을 사용한다.
- `app.get('/music/:id', callback);`
- `req.params.id`로 1, 2를 얻어올 수 있다.

[실습] 영화 관리 어플리케이션

- 지금까지의 movie.js 파일을 Express를 사용하도록 수정해보자.
- postman을 이용하여 CRUD 테스트를 수행한다.
- 파일명 : 21_basicWmovie.js

