Express

Express.js

- A minimal and flexible **Node.js web application framework**

- **Core features :**
  - Allows to **set up middlewares** to respond to HTTP Requests.
  - **Defines a routing table** which is used to perform different actions based on HTTP Method and URL.
  - Allows to **dynamically render HTML Pages** based on passing arguments to templates.

Express

- Running App
- Request
- Response
- Basic routing
- Serving static files
- body-parser

Running App

- $ npm install express
  $ npm install express –g

- Starts a server and listens on port 3000

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

app.listen(3000, function() {
   console.log("App listening on port 3000...")
})
```

Response

- **The res object**
  - It represents the HTTP response that an Express app sends when it gets an HTTP request
  - An enhanced version of Node's own response object

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

app.get('/', function (req, res) {
   res.send('Hello World');
})

app.get('/about', function (req, res) {
   res.json({
      name : 'Greg Lim'
   });
})

app.listen(3000, function() {
   console.log("App listening on port 3000...")
})
```

exhello.js

res.set(field [, value])

Sets the response's HTTP header `field` to `value`. To set multiple fields at once, pass an object as the parameter.

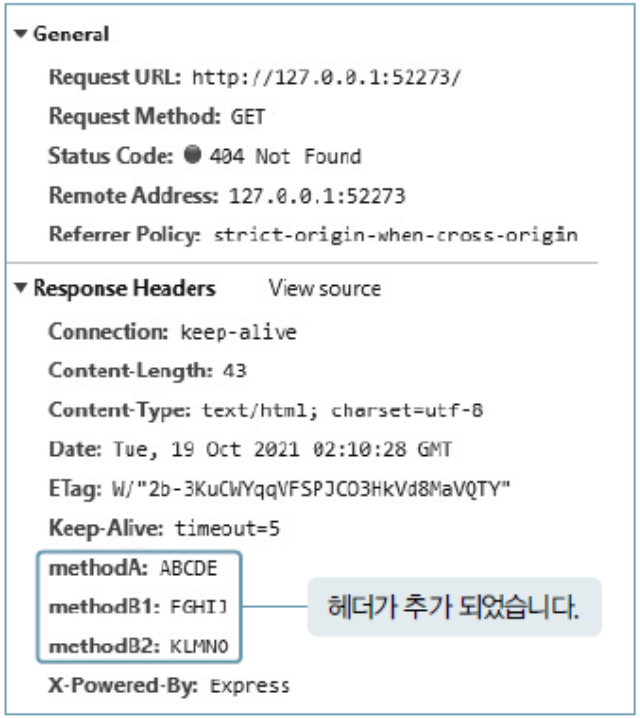
```
res.set('Content-Type', 'text/plain')

res.set({
  'Content-Type': 'text/plain',
  'Content-Length': '123',
  ETag: '12345'
})
```

Aliased as `res.header(field [, value])`.

exResSet.js

```javascript
const express = require('express');

const app = express();

app.get('*', (request, response) => {
    response.status(404);
    response.set('methodA', 'ABCDE');
    response.set({
        'methodB1': 'FGHIJ',
        'methodB2': 'KLMNO'
    });
    response.send('Hello, Again!!');
});

app.listen(52273, () => {
    console.log('Server running at http://127.0.0.1:52273');
});
```

responseBasic.js



개발자 도구 - Network

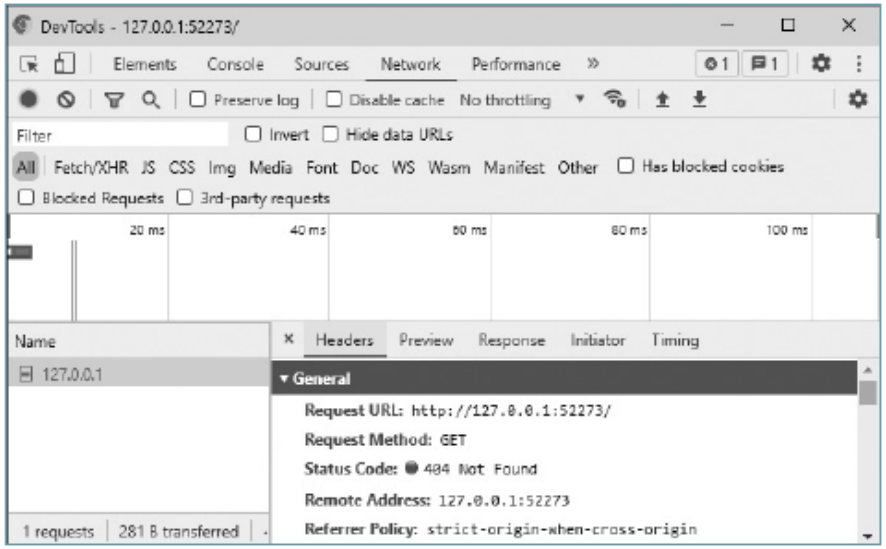
```node
const express = require('express');
const fs = require('fs');
const app = express();

app.get('/image', (request, response) => {
    fs.readFile('image.png', (error, data) => {
        response.type('image/png');
        response.send(data);
    });
});
app.get('/audio', (request, response) => {
    fs.readFile('audio.mp3', (error, data) => {

        response.type('audio/mpeg');
        response.send(data);
    });
});

app.listen(52273, () => {
    console.log('Server running at http://127.0.0.1:52273');
});
```

메소드	설명
type()	Content-Type을 MIME 형식으로 지정합니다.

MIME 형식	설명
text/plain	기본적인 텍스트를 의미합니다.
text/html	html 데이터를 의미합니다.
image/png	png 데이터를 의미합니다.
audio/mpe	MP3 음악 파일
video/mpeg	MPEG 비디오 파일
application/json	json 데이터를 의미합니다.
multipart/form-data	입력 양식 데이터를 의미합니다.

mimeType.js

```
const express = require('express');

const app = express();

app.get('*', (request, response) => {
    response.status(404);
    response.send('해당 경로에는 아무것도 없습니다.');
});

app.listen(52273, () => {
    console.log('Server running at http://127.0.0.1:52273');
});
```



HTTP 상태 코드	설명	예
1XX	처리 중	100 Continue
2XX	성공	200 OK
3XX	리다이렉트	300 Multiple Choices
4XX	클라이언트 오류	400 Bad Request
5XX	서버 오류	500 Internal Server Error

```javascript
const express = require('express');

const app = express();

app.get('*', (request, response) => {
    response.redirect('http://www.knu.ac.kr');
});

app.listen(52273, () => {
    console.log('Server running at http://127.0.0.1:52273');
});
```

redirect.js

Request

- **The req object**
  - It Represents the HTTP request and has properties for the request query string, parameters, body, HTTP headers, and so on.
  - An enhanced version of Node's own request object

basicRoute.js

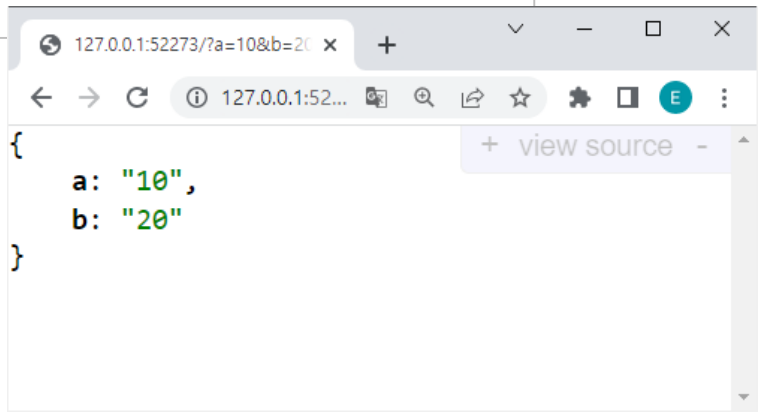
```
const express = require('express');

const app = express();

app.get('/page/:id', (request, response) => {
    const id = request.params.id;
    response.send(`<h1>${id} Page</h1>`);
});

app.listen(52273, () => {
    console.log('Server running at http://127.0.0.1:52273');
});
```

http://127.0.0.1:3000/page/1234

requestQuery.js

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

app.get('*', (request, response) => {
    console.log(request.query);
    response.send(request.query);
});

app.listen(52273, () => {
    console.log('Server running at http://127.0.0.1:52273');
});
```

http://127.0.0.1:52273/?a=10&b=20



Routing with Express

- **Routing** refers to how an application's endpoints (URIs) respond to client requests

- **Basic routing :** app.METHOD(PATH, HANDLER)
  - app is an instance of express.
  - METHOD is an HTTP request method, in lowercase.
  - PATH is a path on the server.
  - HANDLER is the function executed when the route is matched.

```
app.get('/', function (req, res) {
   // --
})
```

메소드	설명
get(path, callback)	GET 요청이 발생했을 때 이벤트 리스너를 지정합니다.
post(path, callback)	POST 요청이 발생했을 때 이벤트 리스너를 지정합니다.
put(path, callback)	PUT 요청이 발생했을 때 이벤트 리스너를 지정합니다.
delete(path, callback)	DELETE 요청이 발생했을 때 이벤트 리스너를 지정합니다.
all(path, callback)	모든 요청이 발생했을 때 이벤트 리스너를 지정합니다.

Route Methods

```
var express = require('express');
var app = express();
// This responds with "Hello World" on the homepage
app.get('/', function (req, res) {
   console.log("Got a GET request for the homepage");
   res.send('Hello GET');
})

// This responds a POST request for the homepage
app.post('/', function (req, res) {
   console.log("Got a POST request for the homepage");
   res.send('Hello POST');
})

// This responds a DELETE request for the /del_user page.
app.delete('/del_user', function (req, res) {
   console.log("Got a DELETE request for /del_user");
   res.send('Hello DELETE');
})

// This responds a GET request for the /list_user page.
app.get('/list_user', function (req, res) {
   console.log("Got a GET request for /list_user");
   res.send('Page Listing');
})

// This responds a GET request for abcd, abxcd, ab123cd, and so on
app.get('/ab*cd', function(req, res) {
   console.log("Got a GET request for /ab*cd");
   res.send('Page Pattern Match');
})

app.listen(3000, function() {
   console.log("App listening on port 3000...")
})
```

exrouting.js

- **app.all()**
  - Executed for requests to the route “/secret” whether using GET, POST, PUT, DELETE, or any other HTTP request method

```
app.all('/secret', function (req, res, next) {
  console.log('Accessing the secret section ...')
  next() // pass control to the next handler
})
```

- **app.use()**
  - To load the middleware function

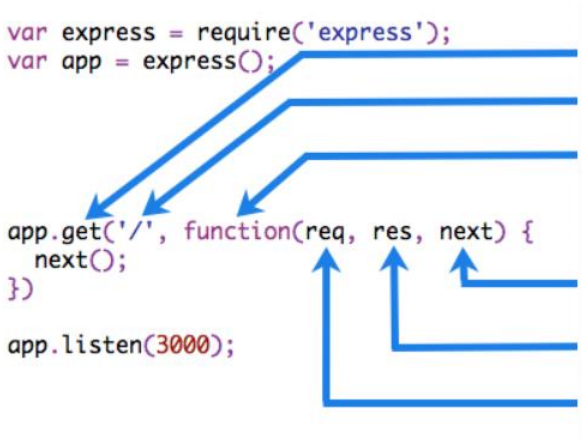
```
var express = require('express')
var app = express()

var myLogger = function (req, res, next) {
  console.log('LOGGED')
  next()
}

app.use(myLogger)

app.get('/', function (req, res) {
  res.send('Hello World!')
})

app.listen(3000)
```

exmiddle.js

Middleware

- **Middleware functions**
  - functions that have access to the request object (**req**), the response object (**res**), and the **next** middleware function

- It can perform the following tasks:
  - Execute any code.
  - Make changes to the request and the response objects.
  - Call the next middleware function in the stack.



Serving Static Files

- A built-in middleware **express.static** to serve static files
  - if you keep your images, CSS, and JavaScript files in a public directory…

For example, use the following code to serve images, CSS files, and JavaScript files in a directory named `public`:

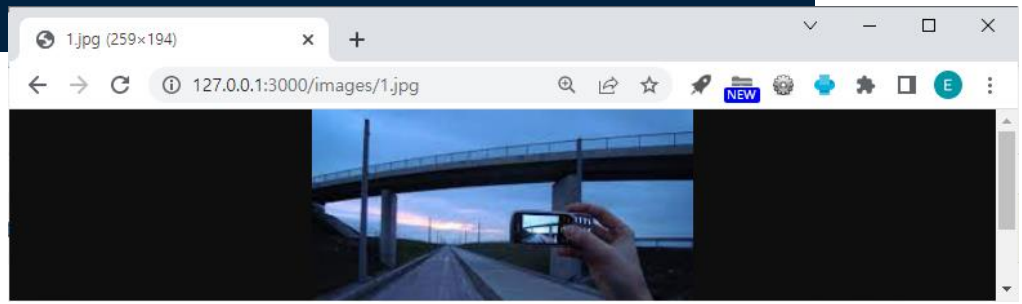
```
app.use(express.static('public'))
```

Now, you can load the files that are in the `public` directory:

```
http://localhost:3000/images/kitten.jpg
http://localhost:3000/css/style.css
http://localhost:3000/js/app.js
http://localhost:3000/images/bg.png
http://localhost:3000/hello.html
```

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

app.use(express.static('public'));

app.get('/', function (req, res) {
   res.send('Hello World');
})

app.listen(3000, function() {
    console.log("App listening on port 3000...")
})
```

exstatic.js

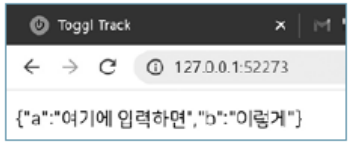


body-parser Middleware

- **express.urlencoded()**
  - A built-in middleware function
  - It parses incoming requests with urlencoded payloads and is based on body-parser.

MIME 형식	설명
application/x-www-form-urlencoded	웹 브라우저에서 입력 양식을 POST, PUT, DELETE 방식 등으로 전달할 때 사용하는 기본적인 요청 형식입니다.
application/json	JSON 데이터로 요청하는 방식입니다.
multipart/form-data	대용량 파일을 전송할 때 사용하는 요청 방식입니다.

```javascript
const express = require('express');
const morgan = require('morgan');
const app = express();
app.use(express.static('public'));
app.use(morgan('combined'));
app.use(express.urlencoded({ extended: false }));

app.get('/', (request, response) => {

    let output = '';
    output += '<form method="post">';
    output += '  <input type="text" name="a" />';
    output += '  <input type="text" name="b" />';
    output += '  <input type="submit" />';
    output += '</form>';

    response.send(output);
});
app.post('/', (request, response) => {
    response.send(request.body);
});
app.listen(52273, () => {
    console.log('Server running at http://127.0.0.1:52273');
});
```



body.js

Express

- Running App
- Request
- Response
- Basic routing
- Serving static files
- body-parser