Chapter 8. express 모듈

2017.10.14.토

노드 수원

서유진

목차

1. Express 모듈? (기본 서버)
2. 요청&응답 메서드
3. 기본 응답 메서드
4. 기본 요청 메서드
   3.1 요청 헤더의 속성 추출
   3.2 요청 매개변수 추출
5. Express 모듈과 http 모듈의 차이점
6. 미들웨어?
7. Router 미들웨어
8. Static 미들웨어
9. Morgan 미들웨어

Express 모듈?

http 모듈에 여러 기능을 추가하여 웹 서버를 쉽게 개발(서버의 생성 및 실행)할 수 있게 해주는 외부 모듈

```
npm install express@4
```

- 모듈 추출 후 함수 실행하여 서버 생성
- Express 프레임워크에서 다른 모듈을 사용하기 위해서는 http모듈도 함께 사용하여 서버를 실행해야 함

```
1  var http = require('http');
2  var express = require('express');
3  var app = express();
```

- Use() 메서드로 request **이벤트 리스너**를 설정

이벤트 리스너는 쉽게 말해 어떠한 이벤트에 대해 대기중인 상태를 말합니다. 항상 리스닝(들으며) 대기중인거죠.
해당 이벤트가 발생했을 때 등록해놨던 이벤트 리스너가 실행됩니다.

```
5  app.use(function (request, response) {
6    response.writeHead(200, { 'Content-Type': 'text/html' });
7    response.end( '<h1>Hello express</h1>' );
8  });
```

- listen() 메서드로 웹 서버를 설정 및 실행

```
10  app.listen(52273, function(){
11    console.log("server rurnning at http://127.0.0.1:52273");
12  })
```

요청(request) & 응답(response) 메서드

Request 이벤트 리스너의 매개변수에는 request 객체와 response 객체가 들어간다

Request

Properties

req.app
req.baseUrl
req.body
req.cookies
req.fresh
req.hostname
req.ip
req.ips
req.method
req.originalUrl
req.params
req.path
req.protocol
req.query
req.route
req.secure
req.signedCookies
req.stale
req.subdomains
req.xhr

Methods

req.accepts()
req.acceptsCharsets()
req.acceptsEncodings()
req.acceptsLanguages()
req.get()
req.is()
req.param()
req.range()

Response

Properties

res.app
res.headersSent
res.locals

Methods

res.append()
res.attachment()
res.cookie()
res.clearCookie()
res.download()
res.end()
res.format()
res.get()
res.json()
res.jsonp()
res.links()
res.location()
res.redirect()
res.render()
res.send()
res.sendFile()
res.sendStatus()
res.set()
res.status()
res.type()
res.vary()

기본 응답 메서드

Request 이벤트 리스너의 매개변수에는 request 객체와 response 객체가 들어간다

Request

Properties	*Methods*
req.app	req.accepts()
req.baseUrl	req.acceptsCharsets()
req.body	req.acceptsEncodings()
req.cookies	req.acceptsLanguages()
req.fresh	req.get()
req.hostname	req.is()

req.get(field)

Returns the specified HTTP request header field (case-insensitive match). The `Referrer` and `Referer` fields are interchangeable.

```
req.get('Content-Type');
// => "text/plain"

req.get('content-type');
// => "text/plain"

req.get('Something');
// => undefined
```

Aliased as `req.header(field)`.

기본 응답 메서드

send() 메서드

express8-5.js

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

app.use(function (request, response, next) {
  var output = [];
  for (var i = 0; i < 3; i++ ){
    output.push({
      count: i,
      name: 'name -' + i
    });
  }
  response.send(output);
})

app.listen(52273, function(){
  console.log('server rurnning at http://127.0.0.1:52273');
})
```

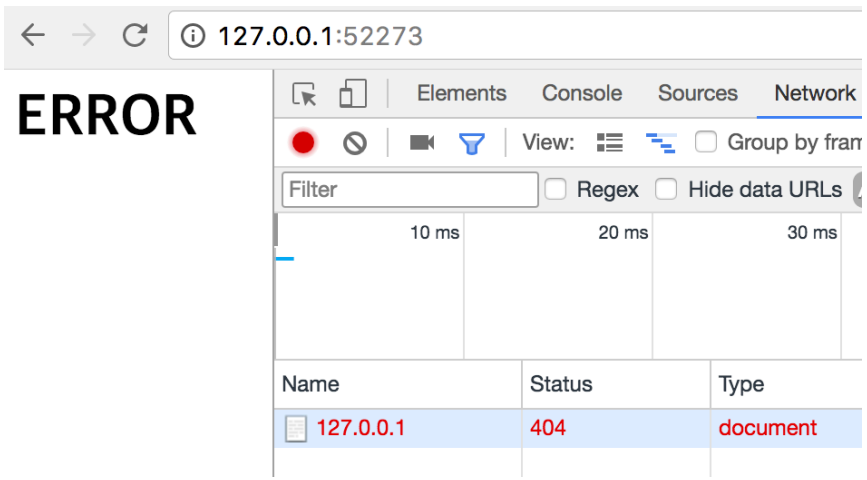
[{"count":0,"name":"name -0"},{"count":1,"name":"name -1"},{"count":2,"name":"name -2"}]

기본 응답 메서드

status() 메서드

express8-6.js

```
1  var express = require('express');
2  var app = express();
3
4  app.use(function (request, response, next) {
5    response.status(404).send('<h1>ERROR</h1>');
6  })
7
8  app.listen(52273, function(){
9    console.log('server rurnning at http://127.0.0.1:52273');
10 })
```



기본 요청 메서드

request.get() 메서드로 User-Agent 속성 추출

express8-7.js

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

app.use(function (request, response) {
  var agent = request.get('User-Agent');
  // var agent = request.header('User-Agent');
  console.log(request.headers);
  console.log(agent);
  response.sendStatus(200);
})

app.listen(52273, function(){
  console.log('server rurnning at http://127.0.0.1:52273');
})
```

```
Starting child process with 'node express8-7'
server rurnning at http://127.0.0.1:52273

{ host: '127.0.0.1:52273',
  connection: 'keep-alive',
  'cache-control': 'max-age=0',
  'upgrade-insecure-requests': '1',
  'user-agent': 'Mozilla/5.0 (Macintosh; Intel Mac OS X 10_12_6) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/61
.0.3163.100 Safari/537.36',
  accept: 'text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,*/*;q=0.8',
  'accept-encoding': 'gzip, deflate, br',
  'accept-language': 'fr-FR,fr;q=0.8,en-US;q=0.6,en;q=0.4',
  'if-none-match': 'W/"2-nOO9QiTIwXgNtWtBJezz8kv3SLc"' }
Mozilla/5.0 (Macintosh; Intel Mac OS X 10_12_6) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/61.0.3163.100 Safar
i/537.36
```

Express 모듈과 http 모듈의 차이점

Request 이벤트 리스너를 연결하는 데 use() 메서드를 사용한다!!!
use() 메서드 안에 미들웨어를 넣어 특정 작업을 수행하는 모듈을 분리 가능, 이로써 재사용이 가능하다는 장점을 가짐

```
app.use(function (request, response, next) {
  console.log("첫 번째 미들웨어");
  next();
})

app.use(function (request, response, next) {
  console.log("두 번째 미들웨어");
  next();
})

app.use(function (request, response, next) {
  console.log("세 번째 미들웨어");
  response.writeHead(200, { 'Content-Type': 'text/html' });
  response.end( '<h1>express basic</h1>' );
})
```

미들웨어?

- 분리 : 요청의 응답을 완료하기 전까지 요청 중간중간에 여러 가지 일을 처리할 수 있음.
  (+) 미들웨어에서 request 객체와 response 객체에 속성 또는 메서드를 추가하면
  다른 미들웨어에서도 추가한 속성과 메서드를 사용할 수 있다.

express.js

```
1  var http = require('http');
2  var express = require('express');
3  var app = express();
4
5  app.use(function (request, response, next) {
6    console.log("첫 번째 미들웨어");
7    next();
8  })
9
10 app.use(function (request, response, next) {
11   console.log("두 번째 미들웨어");
12   request.number = 52;
13   response.number = 273;
14   next();
15 })
16
17 app.use(function (request, response, next) {
18   console.log("세 번째 미들웨어");
19   response.send('<h1>'+request.number+':'+response.number+'</h1>');
20 })
21
22 app.listen(52273, function(){
23   console.log('server rurnning at http://127.0.0.1:52273');
24 })
25
```

```
Starting child process with 'node express.js'
server rurnning at http://127.0.0.1:52273
첫 번째 미들웨어
두 번째 미들웨어
세 번째 미들웨어
첫 번째 미들웨어
두 번째 미들웨어
세 번째 미들웨어
```

Express 모듈과 함께 사용할 수 있는 미들웨어

use() 메서드를 안에 넣을 수 있는 미들웨어들

• Router	페이지 라우트를 수행
• Static	특정 폴더를 서버의 루트 폴더에 올림
• Morgan	로그 정보를 출력
• Cookie parser	쿠키를 분해
• Body parser	POST 요청 매개변수를 추출
• Connect-multiparty	POST 요청 매개변수를 추출
• Express-session	세션 처리 수행
• Csurf	CSRF 보안을 수행
• Error handler	예외처리를 수행
• Limit	POST 요청의 데이터를 제한
• Vhost	가상 호스트를 설정

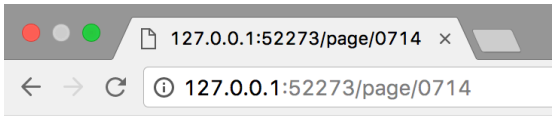
[Express middleware 더 살펴보기]
http://expressjs.com/en/resources/middleware.html

Router 미들웨어

클라이언트 요청에 따라 적절한 페이지를 제공(페이지 라우팅)
http 모듈에서 if문을 사용해서 구현했던 페이지 라우팅을 좀 더 쉽게 구현 가능함

express8-13.js

```
1  var express = require('express');
2  var app = express();
3
4  app.get('/page/:id', function (request, response){
5    var name = request.params.id;
6    response.send('<h1>' + name + ' Page</h1>');
7  });
8
9  app.listen(52273, function(){
10   console.log('Server running at http://127.0.0.1:52273');
11 });
```



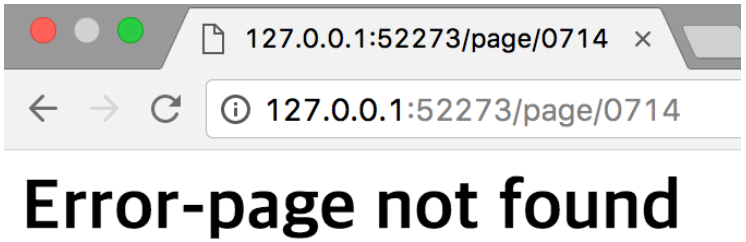
속성 이름	설 명
params	/:id 처럼 ':' 기호를 사용해 지정된 라우팅 매개변수
query	?name=A와 같은 요청 매개변수

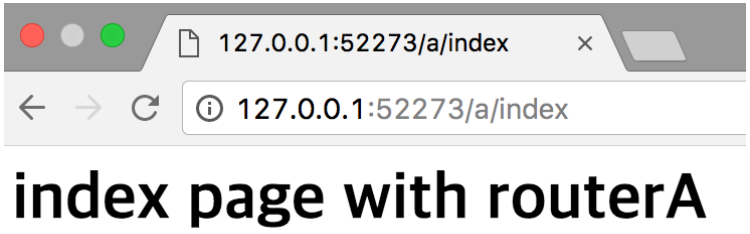
Router 미들웨어

전체 선택자 *
express모듈은 라우터 메서드를 사용한 순서대로 요청을 확인하므로 전체 선택자를 사용하는 경우 반드시 제일 끝에 적어야 함

express8-15.js

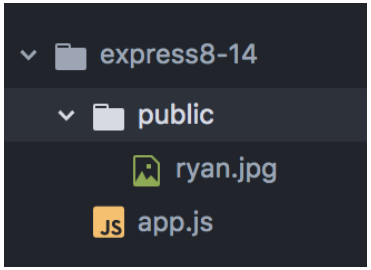
```
var express = require('express');
var app = express();

var routerA = express.Router();
var routerB = express.Router();

routerA.get('/index', function(request, response){
  response.send('<h1>index page with routerA</h1>');
});

routerB.get('/index', function(request, response){
  response.send('<h1>index page with routerB</h1>');
});

app.use('/a', routerA);
app.use('/b', routerB);

app.all('*', function (request, response){
  response.status(404).send('<h1>Error-page not found</h1>');
});

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

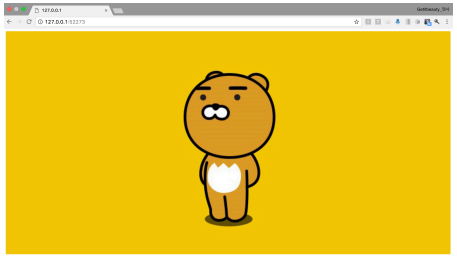




Static 미들웨어

웹서버에 손쉽게 파일을 제공 가능





app.js

```
1  var express = require('express');
2  var app = express();
3
4  app.use(express.static(__dirname + '/public'));
5  app.use(function (request, response){
6    response.writeHead(200, { 'Content-Type': 'text/html' });
7    response.end('<img src="/ryan.jpg" width="100%" />');
8  });
9
10 app.listen(52273, function(){
11     console.log('Server running at http://127.0.0.1:52273');
12 });
```

morgan 미들웨어

[morgan 미들웨어 더 살펴보기]
https://www.npmjs.com/package/morgan

웹 요청이 들어왔을 때 로그(입출력 정보 및 경과 기록)를 출력 `$ npm install morgan`

express8-19.js

```
1  var express = require('express');
2  var morgan = require('morgan');
3  var app = express();
4
5  app.use(morgan('combined'));
6  app.use(function (request, response) {
7    response.send('<h1>express_morgan</h1>');
8  })
9
10 app.listen(52273, function(){
11   console.log('server rurnning at http://127.0.0.1:52273');
12 })
13
```

Combined

```
server rurnning at http://127.0.0.1:52273
::ffff:127.0.0.1 - - [14/Oct/2017:05:45:54 +0000] "GET / HTTP/1.1" 200 23 "-" "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_12_6)
AppleWebKit/537.36 (KHTML, like Gecko) Chrome/61.0.3163.100 Safari/537.36"
```

:remote-addr - :remote-user [:date[clf]] ":method :url HTTP/:http-version"
:status :res[content-length] ":referrer" ":user-agent"

이외에도 common, dev, short, tiny 등의 기본 형식이 존재함.

감사합니다