2022-10-17 2반 실습

Express로 URL Routing 하기

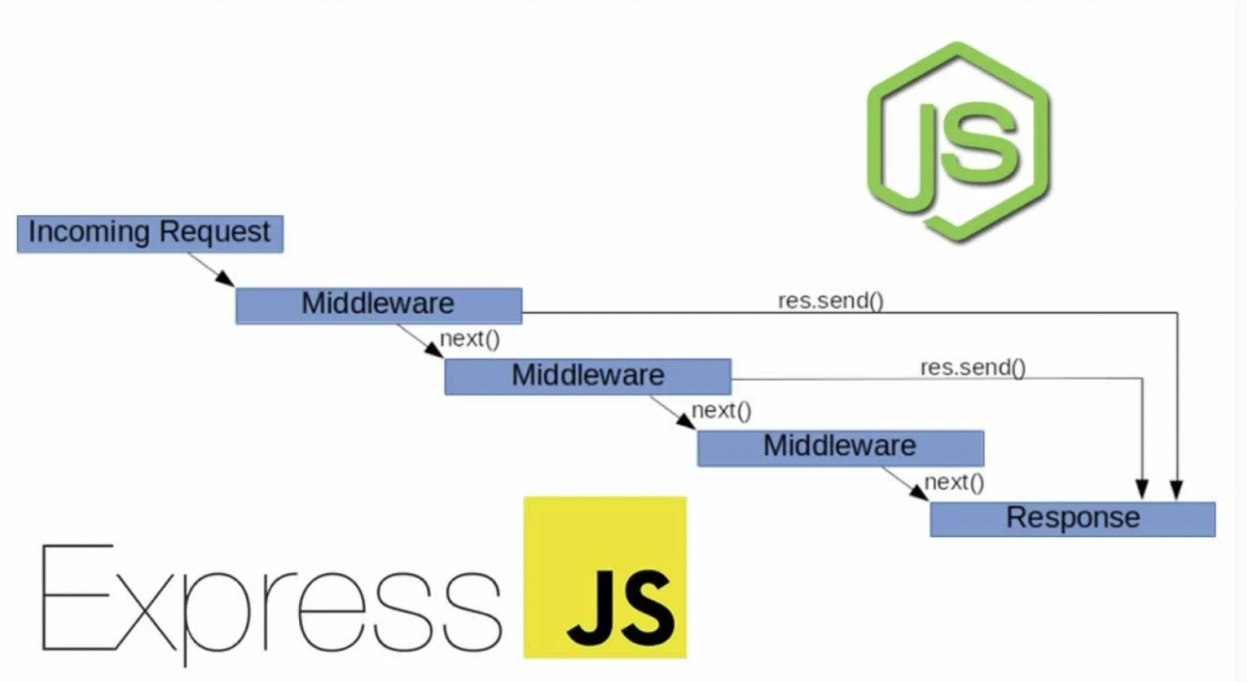
Express

- node.js위에서 동작하는 웹 프레임워크이다.
- 웹 프레임워크 : 웹서비스를 개발, 제공하는 과정에서 반복적으로 처리하는 작업의 효율적인 자동화를 위한 클래스와 라이브러리의 모음이다. (Spring, Django, Express.js, Angular JS, Vue.js 등)
- node.js 개발 시 개발을 빠르고 손쉽게 할수록 도와주는 역할을 한다. 이것은, 미들웨어 구조 때문에 가능한 것이다. 자바스크립트 코드로 작성된 다양한 기능의 미들웨어는 개발자가 필요한 것만 선택하여 express와 결합해 사용할 수 있다.
- 프레임워크이므로 웹 애플리케이션을 만들기 위한 각종 라이브러리와 미들웨어 등이 내장돼 있어 개발하기 편하고, 수많은 개발자들에게 개발 규칙을 강제하여 코드 및 구조의 통일성을 향상시킬 수 있다.

NodesJS를 사용하여 쉽게 서버를 구성할 수 있게 만든 라이브러리

Middleware

- 요청에 대한 응답 과정 중간에 껴서 어떠한 동작을 해주는 프로그램(요청을 proccessing(처리) 해주는 함수, 과정 등을 총칭)이다.



- Express.js는 요청이 들어올 때 그에 따른 응답을 보내주는데, 응답을 보내주기 전에 미들웨어가 지정한 동작을 수행한다.
- Express에서는 함수로 미들웨어를 구현이 가능하다.
- Express에서 미들웨어를 사용하는 과정은 미들웨어 함수를 호출하여 사용한다.

미들웨어 함수 구현, 사용 방법

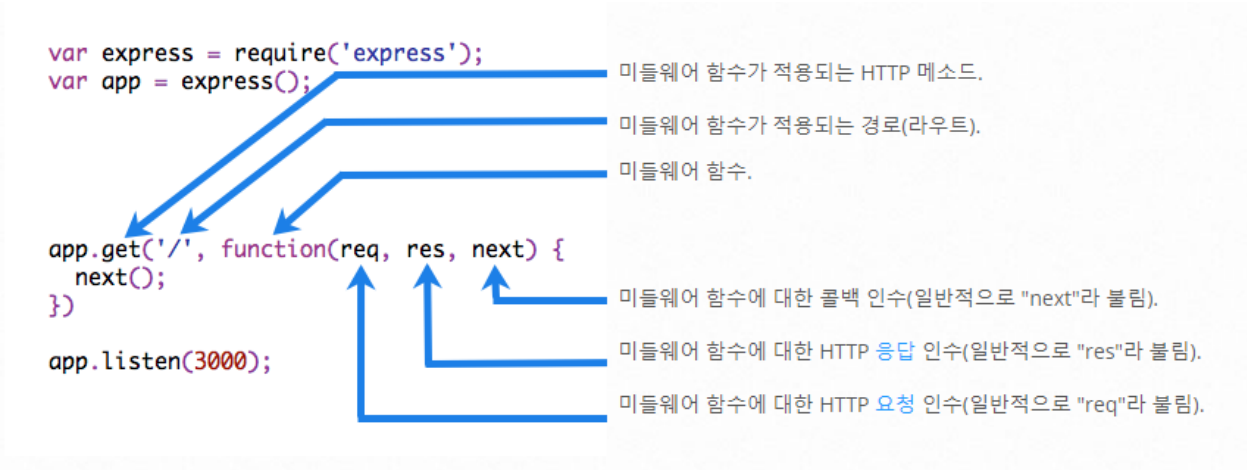
```
//미들웨어 함수 구현
var functionName = function (req, res, next) {
 //do something
 next();
};

//미들웨어 실행
app.use('/', functionName);
```

```
//()=> arrow function을 사용하여 구현한 미들웨어
let functionName = (request, response, next) => {
  //do something
  next();
}

//미들웨어 실행
app.use('/', functionName)
```

```
app.use((req, res, next)=> {
  //do something
  next();
};
```



미들웨어의 구조

function(req, res, next) {}

Routing

- URL(URI) 요청에 따라 어플리케이션이 응답하는 방법을 결정하는 것이다.
- 클라이언트의 특정 요청 method와 엔드포인트에 따라 에 따라 다른 응답 처리 하는 것을 의미한다.
- 클라이언트의 요청 경로(path)를 보고 이 요청을 처리할 수 있는 곳으로 기능을 전달해주는 역할을 한다.

요청을 처리하는 과정에서 경로에 해당하는 작업을 처리하는 것

라우팅 코드 예제1

```
// 라우터 객체 참조
const router = express.Router();

// 라우팅 함수 등록
router.route('/process/login').get(...);
router.route('/process/login').post(...);
//www.alice.io/asd -> post

...
//router.route(요청 경로).get(실행될 함수);
//router.route(요청 경로).post(실행될 함수);
//get(callback), post(callback), put(callback),delete(callback),all(callback) 등 가능
// 라우터 객체를 app 객체에 등록
app.use('/', router);
```

라우팅 코드 예제2

- Express/server.js

```
const indexRouter = require('./routes/index');
const userRouter = require('./routes/user');

app.use('/', indexRouter);
app.use('/user', userRouter);
```

- Express/routes/index.js

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

router.get("/", (req, res) => {
...
});

module.exports = router;
```

- Express/routes/user.js

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

router.get("/", (req, res) => {
...
```

```
});

router.get("/:userId", (req, res) => {
...
});

...

app.route('/books')
  .get(function(req, res) {
    res.send('Get a random book');
  })
  .post(function(req, res) {
    res.send('Add a book');
  })
  .put(function(req, res) {
    res.send('Update the book');
  });

module.exports = router;
```

참조 : https://cotak.tistory.com/85

URL 파라미터 사용하기

- **Route parameters**
  - • ex) GET /artists/1, GET /artists/1/company/entertainment

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

router.get('/artists/:id', function (req, res) {
  console.log("id는 " + req.params.id + " 입니다")
  res.send("id : " + req.params.id)
});

// 여러개도 가능
router.get('/artists/:id/company/:company', function (req, res) {
  res.send("id : " + req.params.id + " 회사 : " + req.params.company)
});
```

- **Query string**
  - • ex) GET /artists?name=hello

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

router.get('/artists', function (req, res) {
  console.log("이름은 " + req.query.name + " 입니다")
  res.send("name : " + req.query.name)
});
```

미들웨어의 작성

이전 미들웨어 참조

Express의 body-parser 사용하기

body-parser

- parsing : 어떤 데이터를 특정 기준으로 가공하는 작업
- parser : parsing을 수행하는 모듈 또는 메소드
- 요청의 본문(body)를 지정한 형태로 파싱해주는 미들웨어이다.
- 기본적으로 요청(request 혹의 req 파라미터)의 req.body를 출력해보면 undefined로 나타난다.
- body-parser의 기능
  1. JSON body parser : body를 JSON으로 parsing, header의 Content-Type 속성 값이 “application/json” 이 아닐 경우에는 parsing 을 하지 않는다.
  2. Raw body parser : body를 RAW로 parsing
  3. Text body Parser : body를 Text로 parsing
  4. URL-encoded from body parser : body를 x-www-form-urlencoded 으로 parsing, header의 Content-Type 속성 값이 “x-www-form-urlencoded” 이 아닐 경우에는 parsing 을 하지 않는다.
- body-parser 사용하기

body-parser는 request의 body를 JSON 형식으로 파싱하는 미들웨어이다.

참조 : https://www.npmjs.com/package/body-parser

- JSON(JavaScript Object Notation) : Javascript 객체 문법으로 구조화된 데이터를 표현하기 위한 문자 기반의 표준 포맷이다.
  - 지원하는 타입 : String, Number, Boolean, Null, Object, Array

```
{
  { "name" : "Jones" },
  {
    "number_1" : 210,
    "number_2" : 215,
    "number_3" : 21.05,
    "number_4" : 10.05
  },
  { "AllowPartialShipment" : false },
  { "Special Instructions" : null },
  {
    "Influencer" :   { "name" : "Jaxon" ,  "age" : "42" ,  "city" ,  "New York" }
  },
  {
    "Influencers" :   [
      {
```

```
        "name" : "Jaxon",
        "age" : 42,
        "Works At" : "Tech News"
       },
       {
        "name" : "Miller",
        "age" : 35
        "Works At" : "IT Day"
       }
     ]
   }
}
```

- 모튤 설치 → 모듈 불러오기 → 모듈 사용

1. 모듈 설치 (이미 설치된 경우 과정 생략 가능)

```
npm install body-parser
```

2. 모듈 불러오기

```
//body-parser 모듈 불러오기
const bodyParser = require("body-parser");
```

3. 모듈 사용

```
app.use(bodyParser.json());    //요청 본문을 json 형태로 파싱
app.use(bodyParser.urlencoded({extended: false}));  //true를 하면 qs 모듈을 사용하고 false면 query-string 모듈을 사용
```