요청 수신

라우팅

- 라우팅이란 클라이언트의 요청에 따라 App(서버)이 어떻게 반응(응답)하는지 정하는 방식이다.

- Node.js에서는 클라이언트의 요청을 콜백함수로 처리한다.

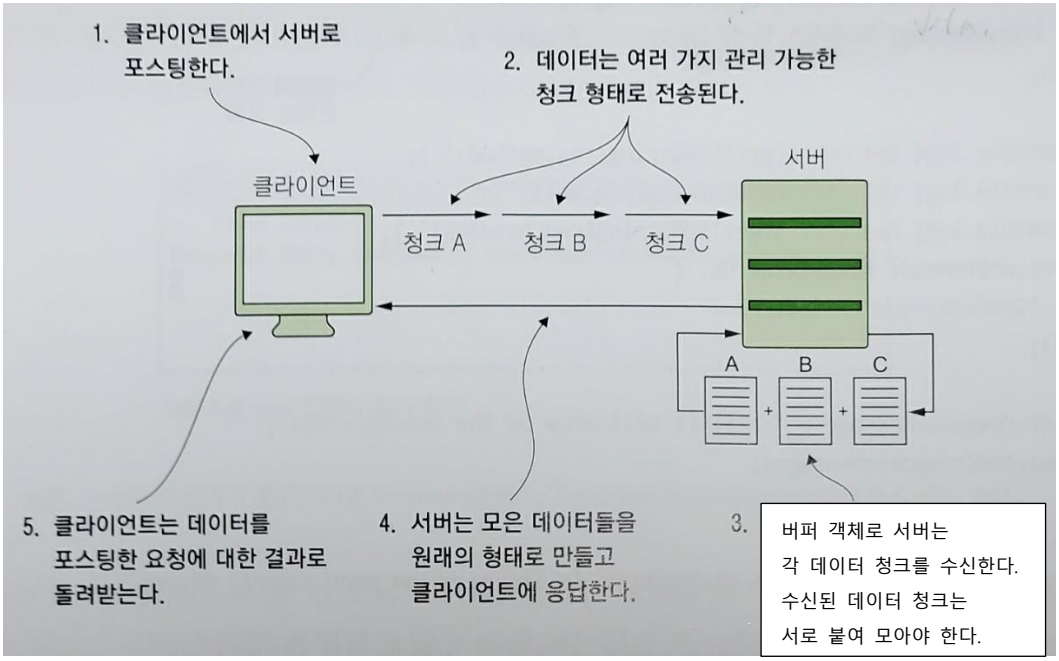
```
ex)
...
app = http.createServer();
app.on("request", function(request, response){ // 요청데이터 처리
        ...
});
...
```

  - app객체는 http.createServer()의 리턴으로 인해 HTTP Server의 역할을 수행하는 객체이다.
  - 4번째 줄의 app.on()에서 Client의 Request를 받을 때 응답하는 콜백함수를 정의한다.

요청 메소드

- GET
  : URL에 Data를 붙여서 요청하는 메소드이다.
  : Ex) http://localhost:3000/get_request의 URL에서 /get_request가 GET요청 Data이다.

- POST
  : HTTP body에 특정 형식(폼)으로 Data를 붙여서 요청하는 메소드이다.
  : POST는 주로 JSON폼을 사용한다.
  : POST는 데이터 청크(Data Chunk)로 구성되어 조각들로 요청된다.



예제1) HTTP GET 처리

1. 전 챕터(4)에서 만들었던 HTTP 웹 서버의 루트 디렉터리를 복사하여 새로운 디렉터리 생성

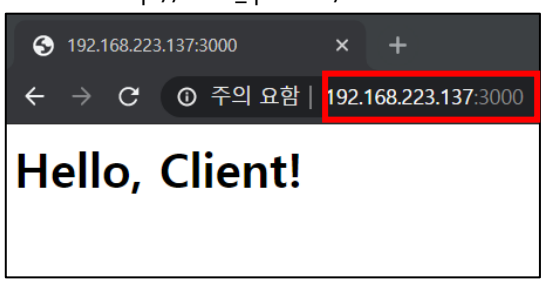
2. 복사된 디렉터리내 main.js의 이름을 main_get.js로 변경
   : mv main.js main_get.js

3. main_get.js에 아래의 내용과 같이 입력

```
"use strict";

const getJSONString = function(obj){ // JavaScript String -> JSON format
        return JSON.stringify(obj, null, 2);
};

const port = 3000;
const http = require("http");
const httpStatus = require("http-status-codes");
const app = http.createServer();

app.on("request", function(request, response){
        console.log("Method : ", request.method);
        console.log("URL : ", request.url);
        console.log("Header : ", getJSONString(request.headers));

        response.writeHead(httpStatus.OK, {
                "Content-Type" : "text/html"
        });

        let resMsg = "<h1>Hello, Client</h1>";
        response.end(resMsg);
});
```

4. main_get.js 실행
   : node main_get.js

✓ 1) 클라이언트 웹 브라우저 실행화면
URL >> http://host_ip:3000/

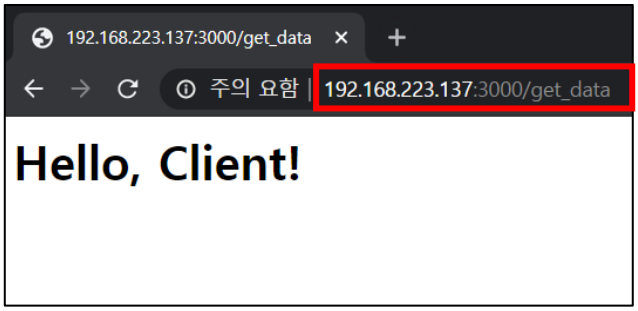


✓ 1) 서버 콘솔 실행화면

```
wlgns12www@nodejs-test:~/nodejs/4_receive/simple_server$ ls
main_get.js  main_post.js  node_modules  package-lock.json  package.json
wlgns12www@nodejs-test:~/nodejs/4_receive/simple_server$ node main_get.js
The Server has started and is listening on port number : 3000
Received the Client_Request
Method :  GET ──▶요청메소드
URL :  / ──▶ GET URL
Header :  {
  "host": "192.168.223.137:3000",
  "connection": "keep-alive",
  "upgrade-insecure-requests": "1",
  "user-agent": "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (K
  "accept": "text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,ir
  "accept-encoding": "gzip, deflate",
  "accept-language": "ko-KR,ko;q=0.9,en-US;q=0.8,en;q=0.7"
}
```

✓ 2) 클라이언트 웹 브라우저 실행화면

URL >> http://host_ip/get_data



✓ 2) 서버 콘솔 실행화면

```
wlgns12www@nodejs-test:~/nodejs/4_receive/simple_server$ node main_get.js
The Server has started and is listening on port number : 3000
Received the Client_Request
Method :  GET ──▶요청메소드
URL :  /get_data ──▶ GET URL
Header :  {
  "host": "192.168.223.137:3000",
  "connection": "keep-alive",
  "cache-control": "max-age=0",
  "upgrade-insecure-requests": "1",
  "user-agent": "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.
  "accept": "text/html,application/xhtml+xml,application/xml;q=0.9,image/we
  "accept-encoding": "gzip, deflate",
  "accept-language": "ko-KR,ko;q=0.9,en-US;q=0.8,en;q=0.7"
}
```

❖ 서버 콘솔 실행화면 1,2)에서 보다시피 GET URL이 다름에도 불구하고 클라이언트에게 응답되는 결과는 같은 것을 확인할 수 있다.

예제2) HTTP POST 처리

1. 예제1)에서 수행한 디렉터리에 main_post.js 생성

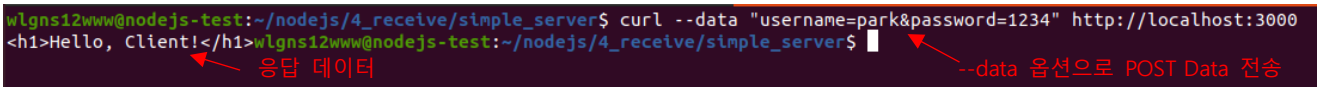
2. main_post.js에 아래와 같이 입력

```
"use strict";

const getJSONString = function(obj){ // JavaScript String -> JSON format
        return JSON.stringify(obj, null, 2);
};

const port = 3000;
const http = require("http");
const httpStatus = require("http-status-codes");
const app = http.createServer();

app.on("request", function(req, res){
        console.log("Request Method : ", req.method);
        console.log("Request URL : ", req.url);
        console.log("Request Header : ", getJSONString(req.headers);

        var body = ""; // HTTP body (POST DATA)
        req.on("data", function(dataChunk){ // 데이터 청크가 도착할 때 마다 event
                body += dataChunk; // 데이터 청크를 body에 합치기
        });
        req.on("end", function(){ // 모든 데이터 청크가 도착했을 시 event
                console.log("Request body content : ", body); // body 출력
        });

        res.writeHead(httpStatus.OK, {
                "Content-Type" : "text/html"
        });

        res.end("<h1>Hello, Client</h1>");
});

app.listen(port);
console.log("The Server has started and is listening on port number : %d", port);
```

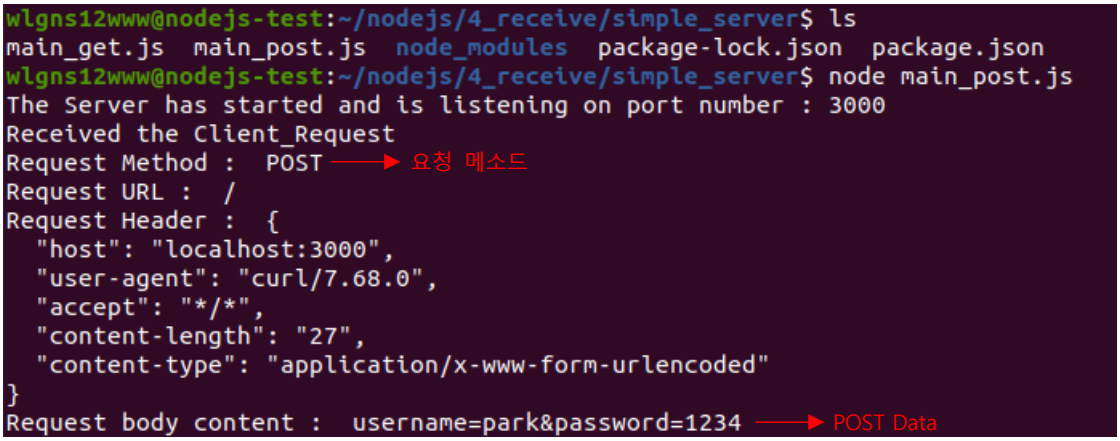
3. main_post.js를 실행
   : node main_post.js

4. curl 명령어를 통해 HTTP POST 전송
   : curl –data "username=Park&password=1234" http://host_ip:3000

✓ 클라이언트 curl 명령어 실행화면



✓ 서버 콘솔 실행화면



참고문헌

- 조나단 웩슬러, 김성준. (2020.01.31). NODEJS로 프로그래밍 시작하기. 90-95.