Node.js test

서버 프로그램 구현

포트폴리오

서지영

모듈 참조 / Express 객체 생성

모듈 참조

```
/*-------------------------------------------------------
 ¦ 1) 모듈참조
 -------------------------------------------------------*/
// 직접 구현한 모듈
const config = require("../helper/_config");
const logger = require("../helper/LogHelper");
const util = require("../helper/UtilHelper");
const fileHelper = require("../helper/FileHelper");
// 내장 모듈
const url = require("url");
const path = require("path");
// 설치가 필요한 모듈
const express = require("express"); // express 본체
const useragent = require("express-useragent"); // 클라이언트의 정보를 조회할 수
있는 기능
```

Express 객체 생성

```
/*-------------------------------------------------------
 ¦ 2) Express 객체 생성
 -------------------------------------------------------*/
// 여기서 생성한 app 객체의 use() 함수를 사용해서
// 각종 외부 기능, 설정 내용, URL을 계쏙해서 확장하는 형태로 구현이 진행된다.
const app = express();
```

클라이언트 접속시 초기화 / 접속한 클라이언트의 정보 파악

```
/*-------------------------------------------------------
 ¦ 3) 클라이언트의 접속시 초기화 -> 접속한 클라이언트의 정보 파악
 -------------------------------------------------------*/
/** app 객체에 UserAgent 모듈을 탑재 */
// -> Express객체(app)에 추가되는 확장 기능들을 Express에서는 미들웨어라고 부른다.
// -> 초기화 콜백함수에 전달되는 req, res객체를 확장하기 때문에 다른 모듈들보다 먼저 설정되어야 한다.
app.use(useragent.express());

// 초기화 - 클라이언트의 접속을 감지
app.use((req, res, next) => {
  logger.debug("클라이언트가 접속했습니다.");

  // 클라이언트가 접속한 시간
  const beginTime = Date.now();

  // 클라이언트의 IP주소
  const ip =
    req.headers["x-forwarded-for"] ¦¦
    req.~~connection~~.remoteAddress ¦¦
    req.socket.remoteAddress ¦¦
    req.~~connection~~.socket.remoteAddress;
```

```
// 클라이언트의 디바이스 정보 기록 (UserAgent 사용)
logger.debug(
  "[client]" +
    ip +
    "/" +
    req.useragent.os +
    "/" +
    req.useragent.browser +
    "(" +
    req.useragent.version +
    ") / " +
    req.useragent.platform
);

// 클라이언트가 요청한 페이지 URL
// 콜백함수에 전달되는 req 파라미터는(객체는) 클라이언트가 요청한 URL의 각 부분을 변수로 담고 있다.
// url.format()의 인자로 입력한 객체의 프로퍼티들을 하나의 url로 리턴
const curretn_url = url.~~format~~({
  protocol: req.protocol, // ex) http://
  host: req.get("host"), // ex) 172.16.141.1
  port: req.port, // ex) 3000
  pathname: req.originalUrl, // ex) /page1.html
});
```

```
  logger.debug("[" + req.method + "]" + decodeURIComponent(curretn_url));

  // 클라이언트의 접속이 종료된 경우의 이벤트
  res.on("finish", () => {
    // 접속 종료시간
    const endTime = Date.now();

    // 이번 접속에서 클라이언트가 머문 시간 = 백엔드가 실행하는데 걸린 시간
    const time = endTime - beginTime;
    logger.debug(
      "클라이언트의 접속이 종료되었습니다. ::: [runtime] " + time + "ms"
    );
    logger.debug("-------------------------------------------------------");
  });

  // 이 콜백함수를 종료하고 요청 URL에 연결된 기능으로 제어를 넘김
  next();
});
```

Express 객체의 추가 설정 / URL 모듈 참조

Express 객체의 추가 설정

```
/*-------------------------------------------------------
 ¦ 4) Express 객체의 추가 설정
 -------------------------------------------------------*/
/** 라우터(URL 분배기) 객체 설정  -> 맨 마지막에 설정 */
const router = express.Router();
// 라우터를 express에 등록
app.use("/", router);
```

코로나 확진자 URL 모듈 참조

```
/*-------------------------------------------------------
 ¦ 5) URL 모듈화
 -------------------------------------------------------*/
app.use(require("./route/test")(app));
```

모듈화한 코로나 확진자 URL 코드

모듈 참조

```
module.exports = (app) => {
  const router = require("express").Router();
  const logger = require("../../helper/LogHelper");
  const config = require("../../helper/_config");
  const axios = require("axios");
```

프론트엔드로부터 전달받은 URL 파라미터 확인

```
/** URL 파라미터를 처리하기 위한 라우터 등록 */
// http://<hostname>:<port>/페이지이름/변수1/변수2
router.route("/today_covid19/:region").get((req, res, next) => {
  // URL 파라미터들은 req.params 객체의 하위 데이터로 저장된다.
  // 전달받은 URL 파라미터는 GET 파라미터와 같은 방법으로 사용 가능함.
  const str = `프론트엔드로부터 전달받는 변수 : ${req.params.region}`;
  logger.debug(str);
```

모듈화한 코로나 확진자 URL 코드

데이터 API URL 참조 / 응답 받은 데이터 처리

```
const covidUrl = "http://itpaper.co.kr/demo/covid19/now.php";

let html = "";
let confirmed = "";

(async () => {
  let json = null;
  try {
    // axios를 활용하여 json 데이터 요청
    const response = await axios.get(covidUrl);
    json = response.data;
  } catch (error) {
    const errorMessage = `[${error.response.status}] ${error.response.statusText}
    `;
    logger.error(errorMessage);
    return;
  }
```

모듈화한 코로나 확진자 URL 코드

응답 받은 데이터와 URL 파라미터 비교 / 브라우저에 결과 전송

```
      json.state.map((v, i) => {
        if (v.region === req.params.region) {
          confirmed = v.confirmed - v.confirmed_prev;
          logger.debug(`{'확진': ${confirmed}}`);
        }
      });

      html = `{'확진': ${confirmed}}`;

      // 브라우저에게 전달할 결과 코드
      res.writeHead(200);
      res.write(html);
      res.end();
    })();
  });

  return router;
};
```

실행 결과

서버 실행

```
node covid_app.js
```

URL 요청

```
GET ▾ http://192.168.35.134:3000/today_covid19/부산
```

실행 결과

요청 클라이언트 정보 log 기록

```
2021-12-27 14:42:23 || [debug]: ------------------------------------------------------------
2021-12-27 14:42:23 || [debug]: |                        start express server                |
2021-12-27 14:42:23 || [debug]: ------------------------------------------------------------
2021-12-27 14:42:23 || [debug]: server address => http://192.168.35.134:3000
2021-12-27 14:42:23 || [debug]: ------------------------------------------------------------
2021-12-27 14:42:26 || [debug]: 클라이언트가 접속했습니다.
2021-12-27 14:42:26 || [debug]: [client]::ffff:192.168.35.134/unknown/insomnia(2021.7.2) / unknown
2021-12-27 14:42:26 || [debug]: [GET]http://192.168.35.134:3000/today_covid19/부산
2021-12-27 14:42:26 || [debug]: 프론트엔드로부터 전달받는 변수 : 부산
2021-12-27 14:42:26 || [debug]: {'확진': 270}
2021-12-27 14:42:26 || [debug]: 클라이언트의 접속이 종료되었습니다. ::: [runtime] 39ms
2021-12-27 14:42:26 || [debug]: ------------------------------------------------------
```

응답 결과

Thank you!

서지영