

웹응용기술 과제 보고서

(수 7-9)

20201056 문종화



< 목 차 >

1. 프로그램 개요	3
2. 수행 절차	4
2.1 서버 시작	4
2.2 사용방법 속지	5
2.3 inputFile.txt 데이터 입력	5
2.3.1 여러 파일 업로드 시 주의사항	6
2.4 오류 처리	6
2.4.1 차트 선택 및 표시	6
2.5 버튼 동적 생성	7
3. 프로그램 수행 절차 분석	7
3.1 서버 구성	7
3.1.1 서버 구성 절차	7
3.2 데이터 파일 입력	8
3.2.1 데이터 파일 입력 절차	8
3.3 차트 출력 과정	8
3.3.1 차트 출력 과정 설명	8

< 그림 차례 >

<그림 1> 서버 시작 명령어 입력	4
<그림 2> 사용방법	5
<그림 3> inputFile.txt 데이터 입력	5
<그림 4> 파일 정보의 시각화	6
<그림 5> 정보에 따른 버튼 동적 생성	7

1. 프로그램 개요

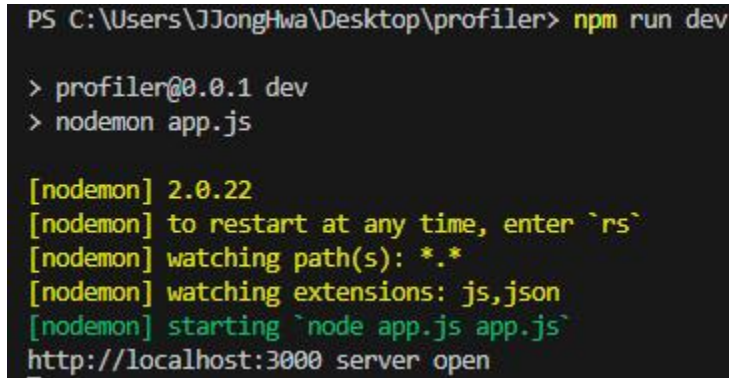
본 프로그램에서는 제공된 예제 파일을 응용하여 시스템을 구현하고자 하였다.

예제 파일에서는 MySQL을 사용하여 구현하였으므로, 본 프로그램에서는 차별성을 두기 위해 MongoDB를 사용하고자 하였다. 그러나 MySQL과 MongoDB를 둘 다 본인의 데스크탑에 설치하는 과정에서 오류가 발생하여, 데이터베이스 연동은 이루어지지 않았다. 이러한 이유로, 여러 파일을 한번에 업로드하고 정보를 읽어오는 기능은 구현되었으나, 정보를 한번에 받아오기 때문에 개별삭제는 불가능하다.

결론은 데이터베이스 연동을 제외한 모든 요구사항은 충족하였다.

2. 수행 절차

2.1 서버 시작



```
PS C:\Users\JJongHwa\Desktop\profiler> npm run dev
> profiler@0.0.1 dev
> nodemon app.js

[nodemon] 2.0.22
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): *.*
[nodemon] watching extensions: js,json
[nodemon] starting `node app.js app.js`
http://localhost:3000 server open
```

<그림 1 서버 시작 명령어 입력>

1) 모듈 설치

프로그램을 사용하기 위해서는 우선 필요한 모듈들을 설치해야 한다. 이를 위해, 터미널에 다음 명령어를 입력한다.

“npm install”

이 명령어를 실행하면, 프로그램 실행에 필요한 모든 모듈이 자동으로 설치된다.

2) 서버 실행

모듈 설치가 완료된 후, 서버를 실행해야 한다. 서버 실행을 위해서는 다음 명령어를 터미널에 입력한다.

“npm run dev”

이 명령어는 개발 서버를 실행시키며, 서버가 성공적으로 실행되었음을 나타내는 메시지가 출력될 것이다.

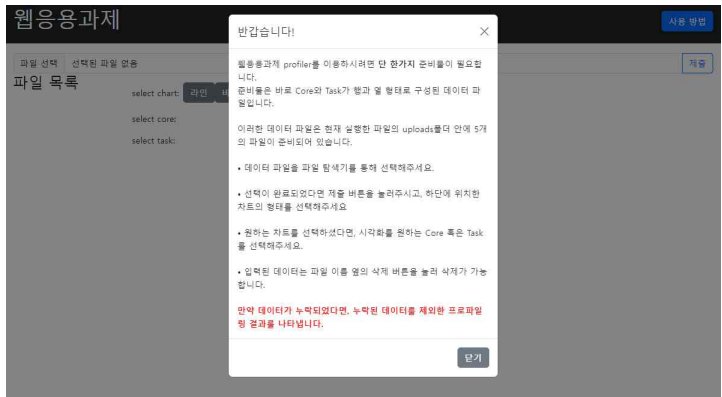
3) 웹 브라우저에서 프로그램 접근

서버가 정상적으로 실행된 후, 프로그램에 접근하기 위해 웹 브라우저를 열고 주소 입력 창에 다음 주소를 입력한다.

“http://localhost:3000/”

또는, Ctrl 키를 누르고 해당 링크를 클릭하면 프로그램이 실행된 창을 바로 열 수 있다.

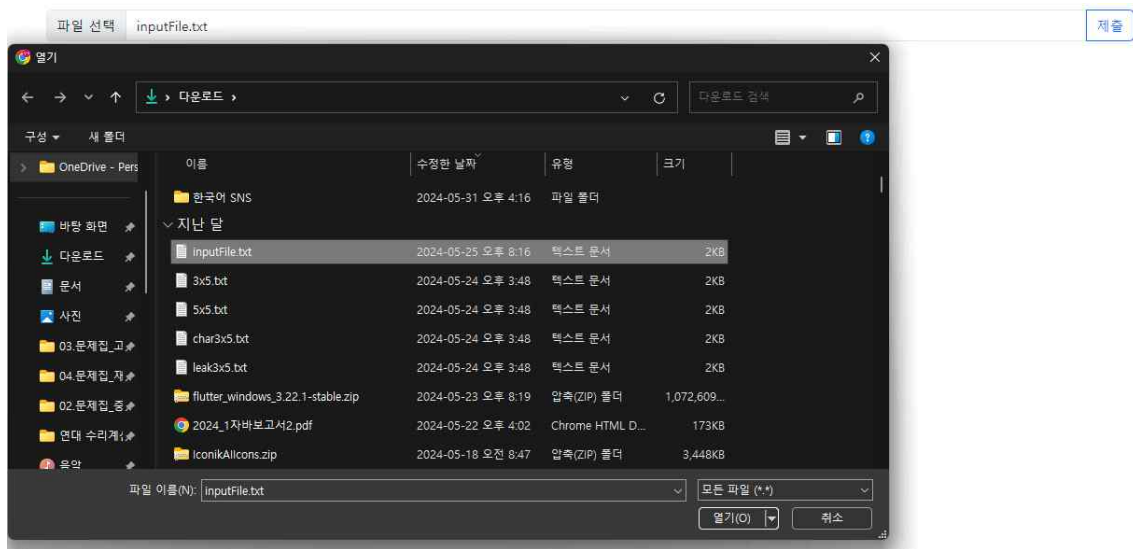
2.2 사용방법 속지



<그림 2 사용방법>

프로그램이 실행된 창에 접속하면, 우측 상단에 위치한 '사용방법' 버튼을 클릭할 수 있다. 이 버튼을 클릭하면 프로그램 사용에 대한 자세한 안내가 나타난다.

2.3 inputFile.txt 데이터 입력



<그림 3 inputFile.txt 데이터 입력>

웹 좌측 상단에 위치한 '파일선택' 버튼을 클릭하여, 프로그램 폴더 안의 uploads 폴더에 저장된 데이터 파일 중 하나를 선택한다. 이 폴더에는 총 5개의 데이터 파일이 존재한다.

- 1) '파일선택' 버튼 클릭
- 2) uploads 폴더에서 원하는 데이터 파일 선택
- 3) '제출' 버튼 클릭

파일을 선택하고 제출 버튼을 누르면 선택한 파일이 프로그램에 업로드된다.

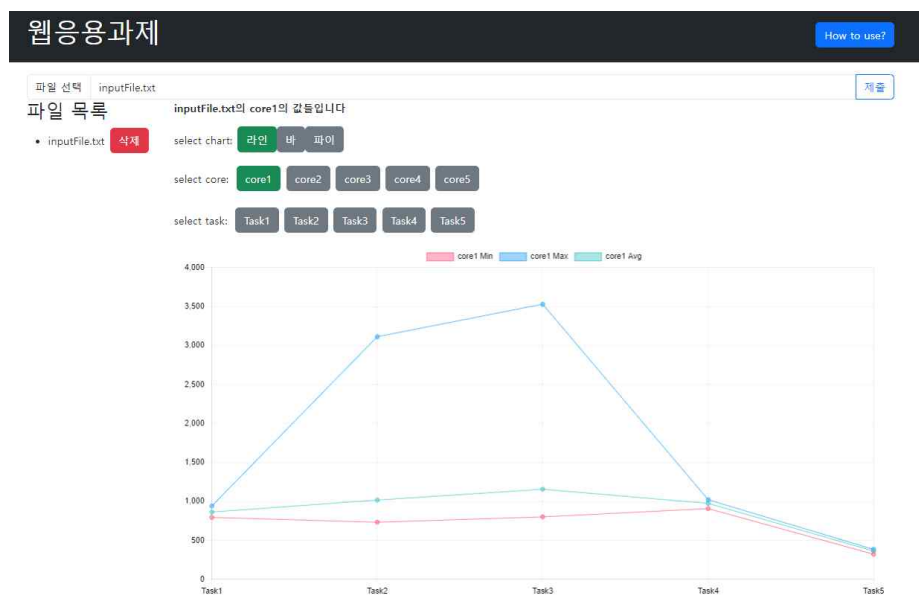
2.3.1 여러 파일 업로드 시 주의사항

여러 파일을 동시에 업로드하는 것도 가능하다. 그러나, 현재 프로그램은 여러 개의 파일을 하나의 파일로 인식한다. 이는 DB 연동이 되어 있지 않기 때문이다. 따라서, 여러 파일을 업로드할 경우 다음 사항에 유의해야 한다.

- 1) 여러 파일을 동시에 업로드할 수 있지만, 한 개의 파일로 처리됨.
- 2) '삭제' 버튼을 누르면 업로드된 모든 파일이 삭제됨.

이 점을 고려하여 필요한 데이터 파일만 선택하고 업로드하는 것이 좋다.

2.4 정보 시각화



<그림 4 파일 정보의 시각화>

파일이 업로드된 후, 화면 왼쪽에 파일 목록이 표시되며, 업로드된 파일의 이름이 목록에 나타난다. 파일 이름 옆에는 '삭제' 버튼이 생기는데, 이 버튼을 클릭하면 파일을 삭제할 수 있다. (여러개를 동시에 올렸다면 모든 파일이 삭제된다.)

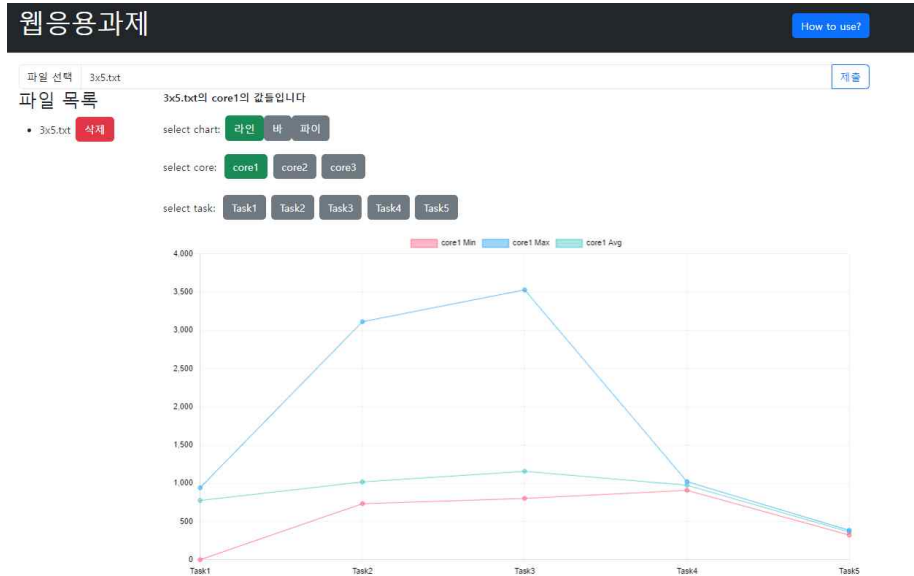
2.4.1 차트 선택 및 표시

파일이 업로드된 후, 원하는 차트를 선택할 수 있다. 차트를 선택하면 선택된 버튼은 초록색으로 표시된다. 그 다음, core 또는 task 버튼 중 하나를 선택하면 해당 차트가 표시된다.

- 1) 원하는 차트 선택 (선택된 모든 버튼은 초록색으로 표시됨)
- 2) core 또는 task 버튼 중 하나 선택
- 3) 선택된 버튼에 맞는 차트가 화면에 표시됨

이 과정을 통해 프로그램을 통해 데이터를 업로드하고, 차트를 선택하고, 데이터를 시각화할 수 있다.

2.5 버튼 동적 생성



<그림 5 정보에 따른 버튼 동적 생성>

예제 파일 중 3X5 파일을 업로드하게 되면, 파일의 내용에 따라 프로그램의 동작이 달라진다. 이 파일의 경우 코어의 개수가 3개이므로, 코어 버튼도 3개가 생성된다.

3. 프로그램 수행 절차 분석

3.1 서버 구성

Node.js 기반의 서버는 'Express.js' 프레임워크를 활용하여 구축되었다. 이 서버는 HTTP 요청을 처리하고 파일 업로드 및 다양한 미들웨어를 활용하여 안정적인 웹 애플리케이션을 제공한다.

3.1.1 서버 구성 절차

1) 필요한 모듈 불러오기:

- 'express', 'morgan', 'path', 'nunjucks', 'multer' 등의 모듈을 사용한다.
- 이를 통해 HTTP 요청 로깅, 정적 파일 제공, 파일 업로드 등의 기능을 구현한다.

2) 서버 인스턴스 생성 및 포트 설정:

- 'Express' 앱 인스턴스를 생성하고, 사용할 포트를 설정한다.

3) 뷰 엔진 설정:

- 'Nunjucks'를 사용하여 템플릿 엔진을 설정한다.
- 이는 HTML 파일을 동적으로 렌더링하는 데 사용된다.

4) 미들웨어 설정:

- 'morgan'을 사용하여 HTTP 요청을 로깅하고, 'express.static'을 통해 정적 파일을 제공한다.
- JSON 및 URL 인코딩된 데이터 처리를 위한 미들웨어를 설정한다.

5) 파일 업로드 설정:

- 'Multer'를 사용하여 파일을 메모리 저장소에 저장하도록 설정한다.

6) 라우터 설정:

- 파일 업로드를 처리하는 라우트를 설정하여 업로드된 파일을 처리하고 응답한다.

7) 기본 경로 설정:

- 기본 경로로 접속 시 템플릿을 렌더링하도록 설정한다.

8) 오류 처리 미들웨어:

- 잘못된 경로 요청 및 서버 오류를 처리하는 미들웨어를 설정한다.

9) 서버 실행:

- 지정된 포트에서 서버를 실행하여 클라이언트의 요청을 처리한다.

3.2 데이터 파일 입력

데이터 파일 입력은 사용자가 업로드한 파일을 서버에서 처리하여 데이터로 변환하고 저장하는 과정을 의미한다. 이 과정은 'Multer'를 통해 파일을 업로드하고, 이를 처리하는 API를 통해 구현되었다.

3.2.1 데이터 파일 입력 절차

1) 파일 업로드 처리:

- 사용자가 업로드한 파일을 받아 메모리에 저장한 후, 이를 처리하여 데이터로 변환한다.
- 파일은 텍스트 파일(.txt) 형식으로 업로드되며, 각 파일의 내용을 파싱하여 데이터로 변환한다.

2) 데이터 저장 및 응답 처리:

- 파싱된 데이터를 메모리에 저장하고, 클라이언트에게 저장된 데이터의 목록을 응답한다.
- 데이터는 JSON 형식으로 변환되어 응답된다.

3.3 차트 출력 과정

차트 출력 과정은 클라이언트에서 데이터를 요청하여 받아온 후, 이를 시각화하여 차트로 출력하는 과정을 의미한다. 이 과정은 'Chart.js' 라이브러리를 사용하여 구현되었다.

3.3.1 차트 출력 과정 설명

1) 클라이언트에서 데이터 요청:

- 클라이언트에서 파일 업로드 API를 호출하여 데이터를 서버로 전송하고, 응답으로 데이터를 받아온다.

2) 데이터 처리 및 통계 계산:

- 받아온 데이터를 처리하여 필요한 통계 정보를 계산한다.
 - 데이터는 그룹화되고, 최소값, 최대값, 평균값 등의 통계가 계산된다.
- 3) 차트 생성 및 렌더링:
- `Chart.js`를 사용하여 데이터를 시각화하고 차트를 생성한다.
 - 차트는 클라이언트의 요청에 따라 동적으로 생성되며, 사용자는 다양한 유형의 차트를 선택하여 볼 수 있다.