Node.js를 활용한 Profiler 프로그램 분석

웹응용기술 화 4-6교시
20230016 박서윤

< 목 차 >

1. 기존 profiler 프로그램 분석

본 프로그램은 사용자가 업로드한 텍스트 형식의 파일을 분석하여 해당 데이터를 시각화 해준다. 사용자는 Core와 Task 데이터가 들어 있는 텍스트 파일을 업로드하고, 이를 기반으로 동적으로 생성된 테이블을 통해 그래프를 확인해볼 수 있다.

아래는 프로그램의 전체 흐름이다.
1) 서버 구성(app.js)
app.js는 웹 서버를 만들고 라우터를 연결하며 에러 처리도 담당한다. Nunjucks 라는 템플릿 엔진을 연결해 HTML에서도 반복문 같은 문법을 사용할 수 있고, Sequelize를 통해 데이터베이스와 연결할 수 있다.

2) 라우터(routes)
routes 폴더 안의 index.js 파일은 홈페이지에 접속했을 때 보여줄 리스트를 불러오는 역할을 한다.
profiles.js 파일은 파일을 업로드하면 그 내용을 분석해서 데이터베이스에 테이블을 만들어준다. 또한 사용자가 어떤 데이터를 보고 싶을 때 그에 맞는 정보를 가져오고, 삭제도 가능하다.

3) 데이터베이스(models)
models안의 index.js 파일에서는 Sequelize를 통해 실제로 테이블을 만들고 데이터를 넣는 내용이 포함되어 있다.
profilse.js는 각 테이블의 구조를 정의한 파일이다.

4) 화면(Views)과 프론트엔드
views의 index.html 파일은 사용자가 볼 수 있는 페이지로, 파일 업로드 창, 차트와 버튼들이 있다.
sequelize.js는 사용자가 파일을 업로드하거나 차트를 클릭할 때 서버와 연결해서 동작하게 해주는 코드이다. 버튼을 누르면 어떤 차트를 보여줄지, 어떤 데이터를 불러올지 정해주는 역할을 한다.

1-1. 프로그램 수행 절차 분석

1) 사용자가 메인 페이지에서 .txt 파일을 선택하고 제출 버튼을 누른다.
2) 자바스크립트 코드가 동작하여 파일 내용을 읽고, 서버에 JSON 형식으로 전송한다.
3) 서버는 profiles 라우터에서 요청을 받아 전달받은 데이터를 바탕으로 새로운 테이블을 만들고 내용을 저장한다.
4) 데이터가 저장되면 웹 페이지에 테이블 목록이 자동적으로 업데이트된다.
5) 사용자가 테이블 이름을 클릭하면 해당 데이터를 기반으로 차트를 선택하고 시각화할 수 있다.
6) 필요하다면 테이블을 삭제하거나 다른 데이터를 재업로드해서 비교해볼 수도 있다.

1-2. 소스 코드 분석

1) app.js
이 프로젝트의 메인 파일이고, 전체 서버를 구성하는 핵심 역할을 한다. 여기서는 express를 통해 웹 서버를 만들고, 템플릿 엔진과 데이터베이스를 연결한다.
먼저 require 부분에서 이 프로젝트에 필요한 외부 모듈들을 불러온다. 그 다음엔 app.set()으로 서버 포트(3000)와 뷰 엔진을 정해두고, nunjucks.configure()를 통해 nunjucks를 사용할 수 있도록 연결한다.
sequelize.sync() 부분은 데이터베이스랑 연결하는 건데, force: false로 설정해서 기존 테이블은 유지한 채로 연결만 하게 되어 있다. 연결되면 "데이터베이스 연결 성공"이라고 콘솔에 뜨고, 오류가 나면 catch문에서 에러가 출력된다.

2) routes/index.js
이 파일은 홈페이지 첫 화면에서 보여줄 테이블 리스트를 불러오는 역할을 한다.
다음은 소스 일부이다.

```
router.get('/', async (req,res)=>{
    getTableList()
        .then((tableList) => {
            res.render('index', {tableList});
        })
        .catch((error) => {
            console.error('테이블 리스트 조회 중 오류가 발생하였습니다:', error);
        });
});
```

/ 경로로 접속하면 테이블 목록을 가져오는 getTableList() 함수를 실행한다. 결과로 받은 테이블 리스트는 index.html에 넘겨줘서 화면에 표 형태로 보여준다.

3) routes/profiles.js
사용자가 텍스트 파일을 업로드하면 파일을 분석해서 DB 테이블을 만드는 작업을 한다. 만약 DB에 이미 있는 테이블이면 만들지 않는다. 또, 파일 여러 개를 동시에 업로드해도 처리가 가능하다.
router.get('/data/:tableName')을 통해 특정 테이블의 데이터를 불러올 수 있다. 먼저 해당 테이블이 실제 DB에 존재하는지 확인하고, profile_model.initiate()를 통해 해당 테이블에 대한 모델을 초기화한다. 이후 findAll로 테이블의 모든 데이터를 가져오고, 그 안에서 core와 task 값이 어떤 것들이 있는지 가져온다. 가져온 core와 task는 사용자가 차트에서 보고 싶은 항목을 고를 수 있게 화면에 보여주는데 사용이 된다.
router.delete('/drop/:tableName')로 테이블을 삭제할 수도 있다.

4) models/index.js
이 파일은 데이터베이스와 관련된 작업들을 처리해준다. 데이터를 넣고 지우고 불러오고, 테이블을 만들고 삭제하는 등의 핵심 역할을 한다.

다음은 소스 일부이다.
```
const Sequelize = require('sequelize');
const env = process.env.NODE_ENV || 'development';
const config = require('../config/config')[env];
const sequelize = new Sequelize(config.database, config.username, config.password, config);
```
데이터베이스에 접속하기 위한 설정 정보를 불러와서 sequelize객체를 만들어 연결을 준비한다.

createTable(tableName)은 사용자가 업로드한 파일 내용을 바탕으로 새 테이블을 만들어주는 함수이다. 테이블 이름은 tableName이고, core, task, usaged라는 세 개의 열을 갖는다.

dropTable(tableName)은 사용자가 삭제 버튼을 누르면, 해당 테이블을 DB에서 삭제해주는 함수이다.

createDynamicTable(profile)은 텍스트 파일 내용을 파싱해서 테이블을 만들고, 그 테이블에 자동으로 데이터를 채우는 함수이다.

5) models/profile.js
이 파일은 Sequelize를 이용해 Profile라는 데이터 모델을 정의하는 곳이다. DB에 어떤 데이터를 어떤 형식으로 저장할지에 대해 알려준다.
```
class Profile extends Sequelize.Model{
```
에서 Sequelize의 Model 클래스를 확장해서 Profile이라는 클래스를 만든다. (테이블 하나라고 생각해도 된다.)

initiate()는 테이블을 만들기 위해 사용하는 함수이다. core, task, usaged 세 개의 칼럼을 만든다. tableName을 매개변수로, 동적으로 테이블 이름을 바꾸어가며 여러 테이블을 만들 수 있게 설계가 되어 있다.

```
static associations(db){ }
```
이 함수는 나중에 다른 테이블과 관계를 설정할 때 사용할 수 있고, 해당 파일에서는 비어있는 상태이다.

6) views/index.html
이 파일은 사용자가 보는 웹 화면을 구성한다. 코드를 보면 사용자가 처음 접속했을 때 볼 수 있는 구조가 담겨 있고, 데이터 파일을 업로드하고, 차트를 통해 시각화할 수 있도록 구성되어 있다.
상단에는 Bootstrap navbar가 있고, 오른쪽에는 How to use? 버튼이 있다. 이 버튼을 누르면 사용법 안내를 볼 수 있다.
그 아래에는 파일 업로드를 위한 입력창과 제출 버튼이 있다.
화면은 좌우로 나뉘고, 왼쪽은 테이블 목록이, 오른쪽은 차트를 시각화하여 볼 수 있는 영역이 있다.
차트 영역에는 line, bar, polarArea 버튼이 있는데, 이 버튼들은 사용자가 원하는 방식으로 차트를 그릴 수 있게 구성 되어 있다.
이 파일은 외부 자바스크립트(/sequelize.js)와 연동되어 있다.

7) public/sequelize.js
이 파일은 사용자가 차트를 고르거나 버튼을 눌렀을 때 실행되는 코드이다. 사용자의 행동을 감지해서 백엔드에 요청을 보내고, 받아온 데이터를 차트로 시각화하는 역할을 한다.
사용자가 텍스트 파일을 업로드하면 FileReader로 파일 내용을 읽는다. 이 데이터를 JSON 형식으로 바꿔서 /profiles 주소로 POST 요청을 보낸다. 요청이 성공하면 서버에서 응답으로 메시지를 보내주고, 그걸 alert로 띄워준다.
그리고 왼쪽 테이블 목록에서 파일명을 클릭하면, 해당 파일이 선택되고 getdata() 함수가 실행된다. 이 함수는 core와 task 데이터를 받아오고 각각 버튼을 만들어서 화면에 보여주도록 한다.

2. 새로운 기능 추가

2-1. 프로그램 개요

기존 제공된 프로그램은 사용자가 웹 브라우저를 통해 텍스트 파일을 업로드하면, 서버에서 그 데이터를 분석해 core와 task별로 최솟값(MIN), 최댓값(MAX), 평균(AVG)을 계산하고, 이를 시각적으로 볼 수 있는 기능을 한다.
이 프로그램은 Node.js와 Express를 이용해서 만든 웹 서버 위에서 동작한다. 사용자가 텍스트 파일을 올리면, 서버가 그 파일을 읽어서 필요한 데이터를 뽑아낼 수 있다. 계산된 결과는 웹 페이지에 보내지고, 사용자는 그 결과를 차트로 확인할 수 있다.
또한 본인은 기능 확장을 위해, 차트를 이미지 파일로 저장할 수 있는 기능을 구현하였다. 사용자가 버튼을 클릭하면 현재 화면에 표시된 차트를 png 형식으로 저장할 수 있다.
이를 통해 사용자는 데이터 분석 결과를 로컬에 저장하거나 다른 문서에 필요하면 가져와서 편리하게 쓸 수 있게 된다.

2-2. 프로그램 수행 절차 분석

이 프로그램은 위에서 언급했듯이, 사용자가 텍스트 파일을 업로드하면, 그 파일에 들어있는 데이터를 분석해서 결과를 보여주는 방식으로 작동한다. 전체적인 흐름은 아래와 같다.

1) 파일 업로드
사용자가 웹 브라우저에서 inputFile.txt 파일을 선택해서 업로드한다.

2) 서버에서 파일 읽기
업로드된 파일은 Node.js의 파일 시스템 모듈을 통해 서버에서 읽혀진다. 서버는 파일의 내용을 줄 단위로 잘라서 데이터를 분석한다.

3) 데이터 파싱과 정리
각 줄에서 필요한 값들을 뽑아내어(Task 번호, Core 번호, Time 값), 뽑아낸 값들을 그룹별로 묶는다. 묶은 각 그룹마다 수학적으로 분석한다.

4) 결과 시각화
계산된 결과는 웹 페이지로 전달되며, 사용자 화면에서는 분석된 결과를 차트를 통해 볼 수 있다. Task나 Core별로 버튼을 누르면 해당 데이터에 대한 차트를 확인해볼 수 있다.

5) (새로운 기능) 차트 이미지 저장 기능
사용자가 버튼을 클릭하면 화면에 표시되어있는 차트를 이미지(png) 파일로 저장할 수 있다. 이를 이용하면 차트 분석 결과를 다운 받아 볼 수도 있고, 다른 문서에도 편리하게 가져와 쓸 수 있게 된다.

2-3. 소스 코드

차트 이미지 저장 기능을 추가하였다. 기존 제공된 프로그램에서 두 개의 파일을 수정하면 이미지 저장 기능을 추가로 수행할 수 있게 된다. 수정한 소스 코드는 아래와 같으며, 개인 깃허브 링크(https://github.com/Seoyun12380/Node.js-subject)에도 기재하였다.

수정한 파일:
1) public/sequelize.js
2) views/index.html

1) public/sequelize.js

파일의 맨 아래에 다음 코드를 추가하였다. 사용자가 버튼을 클릭하면 현재 화면의 차트를 이미지로 저장할 수 있도록 한다.

```
// 차트 이미지 저장하는 버튼 요소를 가져온다.
const saveBtn = document.getElementById("saveChartBtn");

if (saveBtn) {
    saveBtn.addEventListener("click", function () {
        if (chart) {
            const image = chart.toBase64Image();  //차트를 이미지로 변환
            const link = document.createElement("a");  //다운로드용 링크 생성
            link.href = image;
            link.download = "chart.png";
            link.click();
        } else {
            alert("저장할 차트가 없습니다.");
        }
    });
}
```

2) views/index.html

line, bar, polarArea 버튼을 생성하는 코드 아래에 다음 코드를 추가하였다.
이를 통해 사용자가 차트 이미지를 저장할 수 있는 버튼이 웹 페이지에 표시되도록 하였다.

```
<button id="saveChartBtn" class="btn btn-outline-secondary mt-3">
  차트 이미지 저장
</button>
```

마무리&느낀점

이번 과제를 통해 사용자가 텍스트 파일을 업로드한 데이터를 분석하고 시각화하는 프로그램이 어떻게 작동하는지를 전체적으로 이해할 수 있었다.
특히 각 폴더와 파일들이 어떤 역할을 하고 어떻게 서로 연결되어 동작하는지를 직접 확인하며, 단순한 이론보다 훨씬 쉽게 구조를 익힐 수 있었다.
이번에 새롭게 추가한 기능을 통해 사용자는 차트 분석 결과를 이미지로 저장할 수 있게 되었고, 이를 통해 결과를 보관하거나 다른 문서에 활용하는 것이 훨씬 간편해졌다.
profiler 프로그램을 구성하는 파일인 app.js, sequelize.js, view의 index.html 등의 소스 코드들을 직접 분석하면서, 파일 구성과 동작 원리를 실습을 통해 더 확실하게 이해할 수 있었다.

깃허브 링크
https://github.com/Seoyun12380/Node.js-subject