웹응용기술 과제

Node.js를 활용한 Profiler 프로그램 분석

성결대학교

20210854 오현영

**<목차>**

**1. 소스 코드 분석**

1.1데이터베이스 연결 설정(config/config.json) 2

1.2 models

1.2.1 index.js 2

1.2.2 profile.js 6

1.3 public

1.3.1 sequelize.js 7

1.4 route

1.4.1 index.js 16

1.4.2 profiles.js 17

1.5 app.js 20

**2. 새로운 Profiler 프로그램 제안**

2.1프로그램 개요

2.2프로그램 수행 절차

2.3프로그램 사용 방법 안내

2.4프로그램 시작

2.5데이터 입력

2.6오류 처리

2.7버튼 동적 생성

2.8데이터 삭제

2.9서버 수행 절차 및 로직

2.10 Node 서버 구성

2.11데이터 파일 입력

2.12차트 출력 과정

2.13프론트엔드 시각화 및 로직

2.14버튼 동적 생성 로직

2.15후기

**<그림 목차>**

[그림 1] 데이터베이스 연결 설정 3

[그림 2] index.js 모듈 불러오기 3

[그림 3] 시퀄라이저Model 정의 4

[그림 4] 테이블 삭제 함수 5

[그림 5]동적 테이블 생성 함수 6

[그림 6] 테이블 조회 함수 7

[그림 7] Profile 테이블 8

[그림 8] ProfileList 이벤트 리스너 9

[그림 9] txt파일 처리 로직 10

[그림 10] 파일 error 처리 10

[그림 11] getList() 함수 코드 11

**[**그림 12] deleteTable() 함수 코드 12

**1. (제공된) 과거 수행한 소스 코드 분석**

**1.1 데이터베이스 연결 설정(config/config.json)**

텍스트, 멀티미디어 소프트웨어, 소프트웨어, 스크린샷이(가) 표시된 사진

AI가 생성한 콘텐츠는 부정확할 수 있습니다.

[그림1] 데이터베이스 연결 설정

위 파일은 데이터베이스 연동을 위해 필요한 사용자의 비밀번호, 데이터베이스명, 호스트, DB 종류 등 중요한 설정 정보를 환경별로(ex. development) 미리 정의해두는 파일이다.

**1.2 models**

**1.2.1 index.js**

텍스트, 소프트웨어, 멀티미디어 소프트웨어, 스크린샷이(가) 표시된 사진

AI가 생성한 콘텐츠는 부정확할 수 있습니다.

[그림 2] index.js 모듈 불러오기

먼저 “sequelize”를 활용하기 위해 불러옵니다. 현재 작업이나 데이터베이스 환경 등을 미리 작성한 환경설정 파일과 “config” 파일의 정보를 통해 불러온다. 그 후에 불러온 설정 정보를 바탕으로 Sequelize 객체를 생성한다.

텍스트, 스크린샷, 소프트웨어, 멀티미디어 소프트웨어이(가) 표시된 사진

AI가 생성한 콘텐츠는 부정확할 수 있습니다.

[그림 3] 시퀄라이저Model 정의

테이블 생성 함수(“createTable”)는 tableName을 매개변수로 받아 데이터베이스에서 새로운 테이블을 동적으로 생성한다.

1. core; STRING(20)타입으로, allowNull: false를 통해 Null값을 허용하지 않는다.
2. task: STRING(20)타입으로, 마찬가지로 Null값을 허용하지 않는다.
3. Usaged: INTEGER.UNSIGNED타입으로, 마찬가지로 Null값을 허용하지 않는다.

“await Model.sync()”를 통해 정의된 “Model”에 따라 실제 데이터베이스에 해당 테이블을 생성하거나, 이미 존재하는 경우 스키마를 동기화하여 비동기적으로 처리한다.

텍스트, 스크린샷, 폰트, 멀티미디어 소프트웨어이(가) 표시된 사진

AI가 생성한 콘텐츠는 부정확할 수 있습니다.

[그림 4] 테이블 삭제 함수

테이블 삭제 함수(dropTable)은 주어진 “tableName”에 해당하는 테이블을 데이터베이스에서 삭제하는 함수이다. 만약 해당 “tableName”의 테이블이 데이터베이스에 존재하는 경우 삭제를 하고, 그렇지 않은 경우엔 오류 메시지를 출력하도록 하였다.

텍스트, 스크린샷, 소프트웨어이(가) 표시된 사진

AI가 생성한 콘텐츠는 부정확할 수 있습니다.

[그림 5]동적 테이블 생성 함수

동적 테이블 생성 함수(createDynamicTable)는 “profile”이라는 2차원 배열을 매개변수로 받아 동적으로 테이블을 생성하고 데이터를 삽입한다. 테이블 이름은 “profile”배열의 첫 번째 행, 첫 번째 열의 값으로 정의한다. 정의된 “tableName”을 사용하여 미리 정의한 ‘createTable’함수로 해당 테이블에 대한 Sequelize 모델을 얻는다. “core\_row”는 데이터의 기준 행 인덱스이며, 배열의 두 번째 행부터 profile 배열의 길이만큼 반복을 진행한다. 이때 “core\_row”의 값이 -1이면, 즉 초기 상태이거나 이전 데이터 블록이 끝났을 경우, “core\_row”변수에 현재 row값을 업데이트하고 새로운 데이터 블록의 시작임을 알린다.

만약 해당 행의 profile의 크기가 1이라면, 다시 “core\_row”변수를 초기화하여 새로운 행의 컬럼들을 받을 준비를 한다. 모두 해당하지 않는 경우 각 행의 컬럼들을 순회하며

각 데이터 포인트를 “DynamicModel”을 통해 데이터베이스에 새 레코드로 삽입한다.

텍스트, 스크린샷, 소프트웨어, 멀티미디어 소프트웨어이(가) 표시된 사진

AI가 생성한 콘텐츠는 부정확할 수 있습니다.

[그림 6] 테이블 조회 함수

테이블 목록 조회 함수(getTableList)함수는 현재 데이터베이스에 존재하는 모든 테이블의 목록을 조회한다. “SHOW TABLES” 쿼리는 MySQL 데이터베이스에서의 표준 쿼리이므로 다른 데이터베이스에 연동하는 경우 수정해줄 필요가 있다. “sequelaize”인스턴스의 “query”메서드를 통해 SQL 쿼리를 실행하고 실제 테이블 이름만 추출하여 배열로 반환한다. 마지막으로 “db”객체에 “sequelize”인스턴스를 할당하여 외부에서로 데이터베이스 연결 객체에 접근할 수 있도록 하고, 다른 함수 또한 외부 파일에서 사용할 수 있도록 “exports’한다.

**1.2.2 Profile.js**

텍스트, 스크린샷, 소프트웨어, 멀티미디어 소프트웨어이(가) 표시된 사진

AI가 생성한 콘텐츠는 부정확할 수 있습니다.

[그림 7] Profile 테이블

이 코드는 Sequelize ORM을 사용하여 데이터베이스 테이블의 구조를 정의한다. “Sequelize.Model” 클래스를 상속받아 데이터베이스와 매핑된 새로운 “Profile”클래스를 생성한다. 정적 메서드는 “Profile”모델을 초기화하고 모델을 데이터베이스와 연결하고 스키마를 정의한다. “associations”정적 메서드는 다른 모델과의 관계를 정의하는 데 사용한다. 현재 코드에서는 사용되지 않는다.

**1.3. public**

**1.3.1 sequelize.js**

텍스트, 스크린샷, 소프트웨어이(가) 표시된 사진

AI가 생성한 콘텐츠는 부정확할 수 있습니다.

[그림 8] ProfileList 이벤트 리스너

먼저 “profile\_list” id를 가진 테이블 내 모든 행의 가장 첫 번째 <td>요소를 선택하여 “profileList”에 저장한다. 각 항목을 처리하며 사용자가 목록의 항목을 클릭하는 경우 그 항목의 텍스트 콘텐츠를 ‘fineName’으로 저장한다. 모든 프로필 목록 항목의 배경색을 흰색으로 설정하고 선택된 항목에 대해서만 회색으로 표현하여 선택된 항목을 표현하였다. “select’는 “core”나 “task” 선택 변수인데 현재는 새로운 항목을 선택했으므로 초기화하고 만약 이미 차트가 존재하는 상황이었다면 차트를 제거하고 선택된 파일명에 해당하는 데이터를 백엔드로부터 불러온다.

텍스트, 스크린샷, 소프트웨어, 디스플레이이(가) 표시된 사진

AI가 생성한 콘텐츠는 부정확할 수 있습니다.

[그림 9] txt파일 처리 로직

Id가 “profile\_form”인 태그에 submit 이벤트에 대한 비동기 함수이다. Id가 “Input\_profile”인 태그의 파일들을 “files”변수에 저장한다. files 변수가 존재하지 않는다면, 즉 파일이 없다면 메시지를 출력한다. 파일명을 전처리하여 확장자를 추출한다 만약 “txt”라면 프로미스를 사용하여 해당 파일의 내용을 읽고 미리 정의해둔 “profiles” 배열에 업데이트 한 후에 프로미스를 해결한다.

텍스트, 스크린샷, 소프트웨어, 멀티미디어 소프트웨어이(가) 표시된 사진

AI가 생성한 콘텐츠는 부정확할 수 있습니다.

[그림 10] 파일 error 처리

파일을 읽는 과정에서 오류가 없었다면 “profiles”를 json으로 직렬화하여 백엔드의 “/profiles”로 POST요청을 보낸다. 응답 코드가 성공인 경우, JSON 메시지를 alert 창으로 표시하고 “getList()”함수를 호출하여 프로필 목록을 갱신한다. 그렇지 않은 경우 콘솔에 오류 메시지를 출력한다.

텍스트, 스크린샷, 소프트웨어이(가) 표시된 사진

AI가 생성한 콘텐츠는 부정확할 수 있습니다.

[그림 11] getList() 함수 코드

텍스트, 스크린샷, 폰트, 소프트웨어이(가) 표시된 사진

AI가 생성한 콘텐츠는 부정확할 수 있습니다.

[그림 12] deleteTable() 함수 코드

“getList” 함수는 백엔드에서 저장된 테이블 목록을 비동기적으로 불러와 웹 페이지에 렌더링하는 함수이다. Axios 라이브러리를 사용하여 profiles 앤드포인트에 GET을 요청하고 그에 대한 응답 값을 res에 저장한다. “profiles” 변수에 응답의 데이터를 저장한다. 만족하는 Id의 <tbody>를 tbody에 저장한 후에 빈 문자열로 초기화한다. 그런 다음 각 프로파일에 대해 <tr>과 <td>를 동적으로 생성하고, 이름을 표시한다. 항목 선택 표시는 위에서 정의했던 이벤트 리스너를 적용하였으며, 현재 선택된 것이 있다면 배경색을 유지한다. 여기서 삭제 버튼을 하나 추가하여 사용자가 해당 버튼을 클릭한 경우, “deleteTable()” 함수를 호출하여 해당 테이블을 삭제하도록 한다.

텍스트, 스크린샷, 소프트웨어이(가) 표시된 사진

AI가 생성한 콘텐츠는 부정확할 수 있습니다.

현재 선택된 fileName에 해당하는 데이터를 백엔드에서 불러와 “core”및 “task” 선택 버튼을 렌더링하는 비동기 함수이다. “core” 버튼 렌더링의 경우 “#core” id를 가진 <div>요소의 내용을 초기화하고 클래스 이름을 정의한다. 버튼 클릭시 task.core에 대한 차트를 업데이트하고 선택 상태를 시각적으로 표현한다. 아래 “task”에 대한 로직도 동일하다.

텍스트, 스크린샷, 소프트웨어, 멀티미디어 소프트웨어이(가) 표시된 사진

AI가 생성한 콘텐츠는 부정확할 수 있습니다.

이는 텍스트 파일을 읽는 함수이다. “FileReader” 객체를 생성하여 파일 내용을 비동기적으로 읽는다. 파일을 성공적으로 읽은 경우, 읽은 파일의 텍스트 내용을 “contents’에 저장하고 줄바꿈을 기준으로 분리한다. 파싱된 데이터를 2차원 배열에 저장을 하기 위해 파일의 이름을 첫 요소로 하여 초기화한다. 반복을 통해 데이터를 파싱하고 파싱된 2차원 배열 데이터를 저장한다. 파일 읽기 중 오류 발생 시 콘솔에 에러 메시지를 출력한다.

텍스트, 스크린샷, 소프트웨어, 멀티미디어 소프트웨어이(가) 표시된 사진

AI가 생성한 콘텐츠는 부정확할 수 있습니다.

이 함수는 차트 업데이트 함수이다. 우선 차트가 그려질 <canvas> 요소의 2D 렌더링 컨텍스트를 불러온다. 이미 차트가 존재한다면 해당 차트를 제거한다. 차트의 타입이 “core”인 경우 백엔드에서 task별 레이블, 최솟값, 최댓값, 평균값 데이터를 불러와서 각각의 배열에 저장한다. “task”인 경우에도 동일하게 작동하며, 차트를 그리기 위한 필수 정보가 없다면 함수를 종료한다.

텍스트, 스크린샷, 소프트웨어, 멀티미디어 소프트웨어이(가) 표시된 사진

AI가 생성한 콘텐츠는 부정확할 수 있습니다.

마지막으로 차트를 생성하며 사용자가 선택한 타입으로 차트를 그린다. 각 데이터를 활용하여 차트 데이터를 구성하고, 각 데이터는 고유한 색상과 레이블을 가진다.

**1.4 Routes**

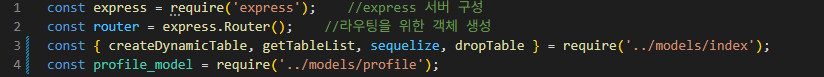
**1.4.1 index.js**

텍스트, 스크린샷, 디스플레이, 소프트웨어이(가) 표시된 사진

AI가 생성한 콘텐츠는 부정확할 수 있습니다.

이 코드는 Express라우터의 한 부분으로 Routes/index.js에 앞서 정의한 데이터베이스 조회 함수인 “getTableList()”를 불러온다. Express의 Router 객체를 생성하여 특정 경로에 대한 요청을 처리하는 미들웨어와 라우트 핸들러를 정의하는데 사용한다. 루트 경로로 GET 요청이 들어왔을 때 Promise가 정상적으로 해결되면 테이블리스트를 이용하여 웹페이지를 렌더링하고 응답을 한다. 오류가 발생하는 경우 콘솔에 오류 메시지를 출력한다.

**1.4.2 profile.js**



“profile”에 대한 미들웨어와 라우트 핸들러 정의하는 코드이다. “/models/index”에서 정의한 여러가지 DB 함수와 sequelize를 불러온다.

텍스트, 스크린샷, 소프트웨어이(가) 표시된 사진

AI가 생성한 콘텐츠는 부정확할 수 있습니다.

루트 경로에서 POST 요청 시 응답의 데이터 부분을 profiles에 저장한다. 데이터베이스에 존재하는 모든 테이블의 이름 목록을 비동기적으로 가져오는데, 이는 새로 업로드 될 파일 이름과의 중복 여부를 확인하는데 사용된다. 파일 이름을 가져온 후 정제한다. 그런 다음 중복 테이블을 확인해서 이미 존재하는 경우 콘솔에 메시지를 출력한다. 그렇지 않은 경우 core, task의 개수에 따라 동적 테이블을 생성한다. Count의 값에 따라 메시지를 JSON 형태로 응답한다.

텍스트, 스크린샷이(가) 표시된 사진

AI가 생성한 콘텐츠는 부정확할 수 있습니다.

테이블 이름에 대한 GET 라우터 정의이다. 응답의 파라미터 값을 추출하여 “tableName”에 저장한다. 그런 현재 데이터베이스에 존재하는 모든 테이블의 이름 목록을 비동기적으로 가져와서 해당 테이블이 데이터베이스에 존재하지 않으면 오류로 처리한다. 다음으로 Profile.js에서 정의된 Profile 모델 클래스의 정적 메서드를 호출하여 해당 테이블에 대한 모델을 초기화한다. 테이블의 모든 데이터를 가져와서 “data”변수에 저장하고 ‘core”컬럼의 고유한 값들을 조회하여 tasks 변수에 저장한다. 마찬가지로 “task”컬럼의 고유한 값들을 조회하여 core 변수에 저장한다. 모든 조회된 데이터들은 JSON 문서 형태로 응답한다.

**1.5 models**

**1.5.1 app.js**

App.js는 express 모듈을 불러와 웹 서버 인스턴스를 초기화하고 포트 번호를 설정한다. 웹을 동적으로 생성하기 위해 Nunjucks 템플릿 엔진을 설정하고 렌더링할 수 있도록 뷰 엔진을 지정한다. Models/index.js에서 불러온 sequelize 인스턴스를 통해 데이터베이스 동기화 등의 작업을 한다. App.use()를 사용하여 다양한 미들웨어를 장착한다. 해당 app.js에서는 정적 파일 제공, 본문 파싱 등이 포함된다. 라우터를 연결하여 각 경로에 대한 요청을 처리한다.