기존 Profiler 프로그램 분석

20200871 정세준

<제목 차례>

<표 차례>

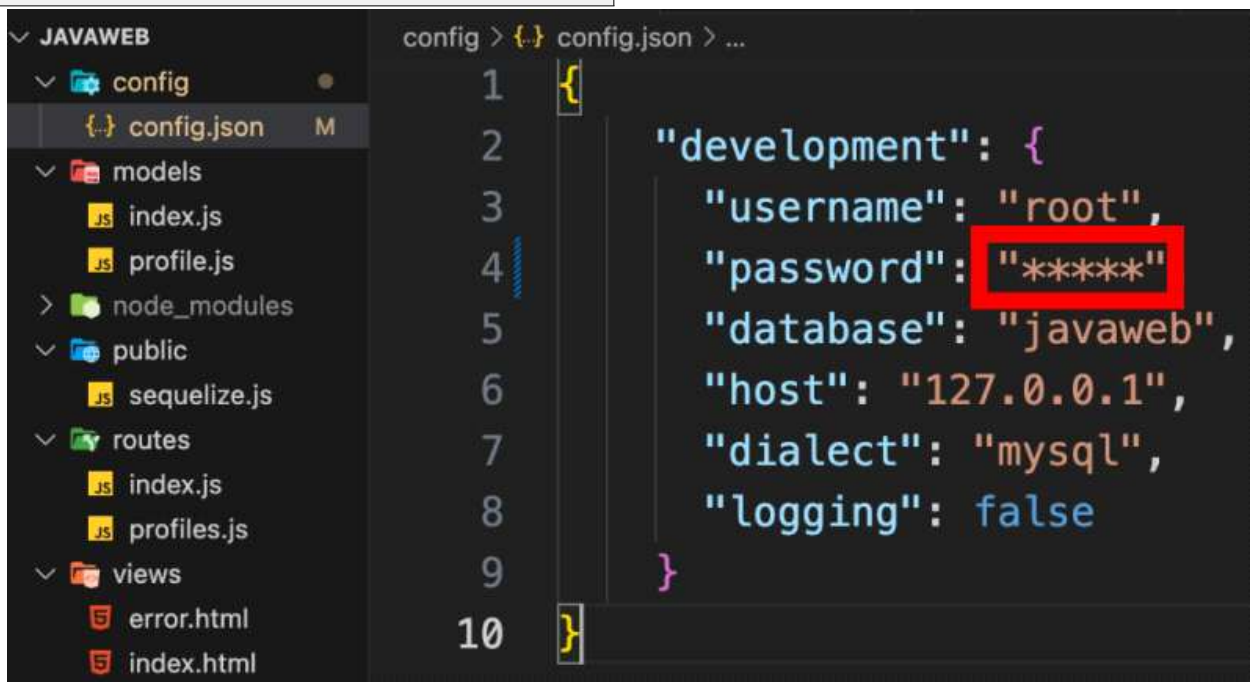
<그림 차례>

<프로그램 분석>

MySQL 오류로 프로그램이 실행되지 않아 기존 보고서에 있는 사진을 그대로 사용하였습니다.

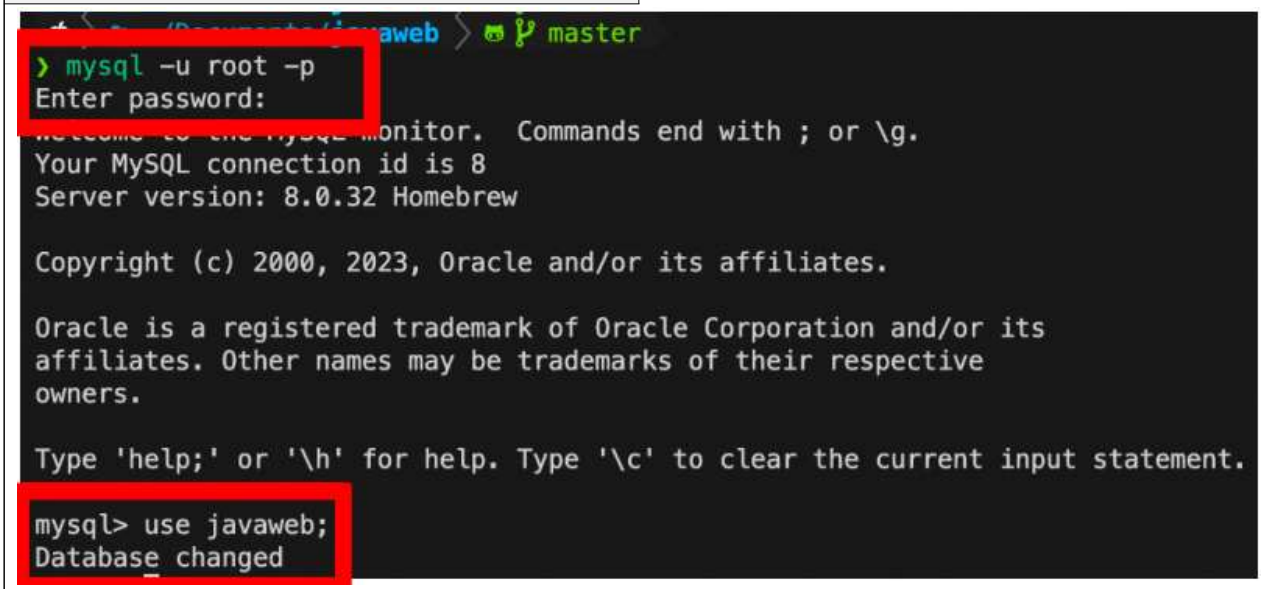
1. 프로그램 수행 절차 분석

1-1. MySQL 비밀번호 변경 (config.json)



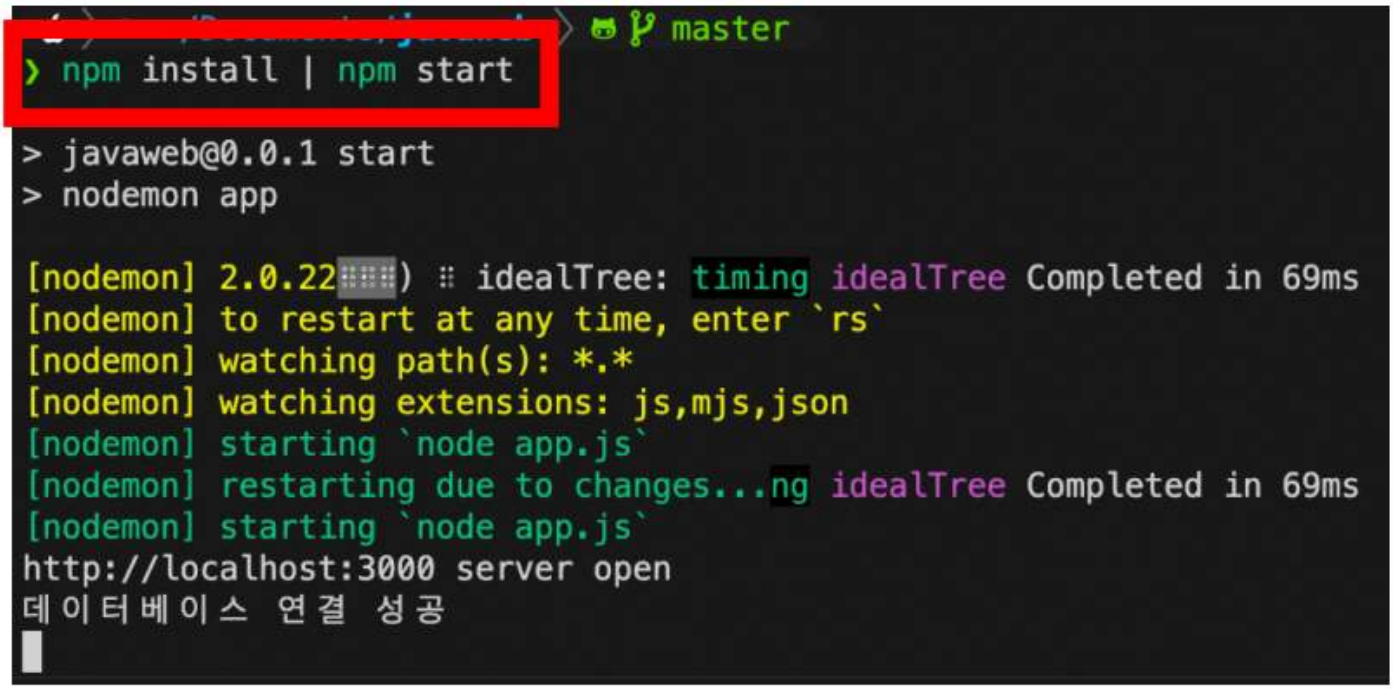
MySQL의 접속을 위한 파일로 해당 패스워드를 자신의 컴퓨터의 MySQL 비밀번호로 변경한다.

1-2. DB 생성 및 선택 (터미널)



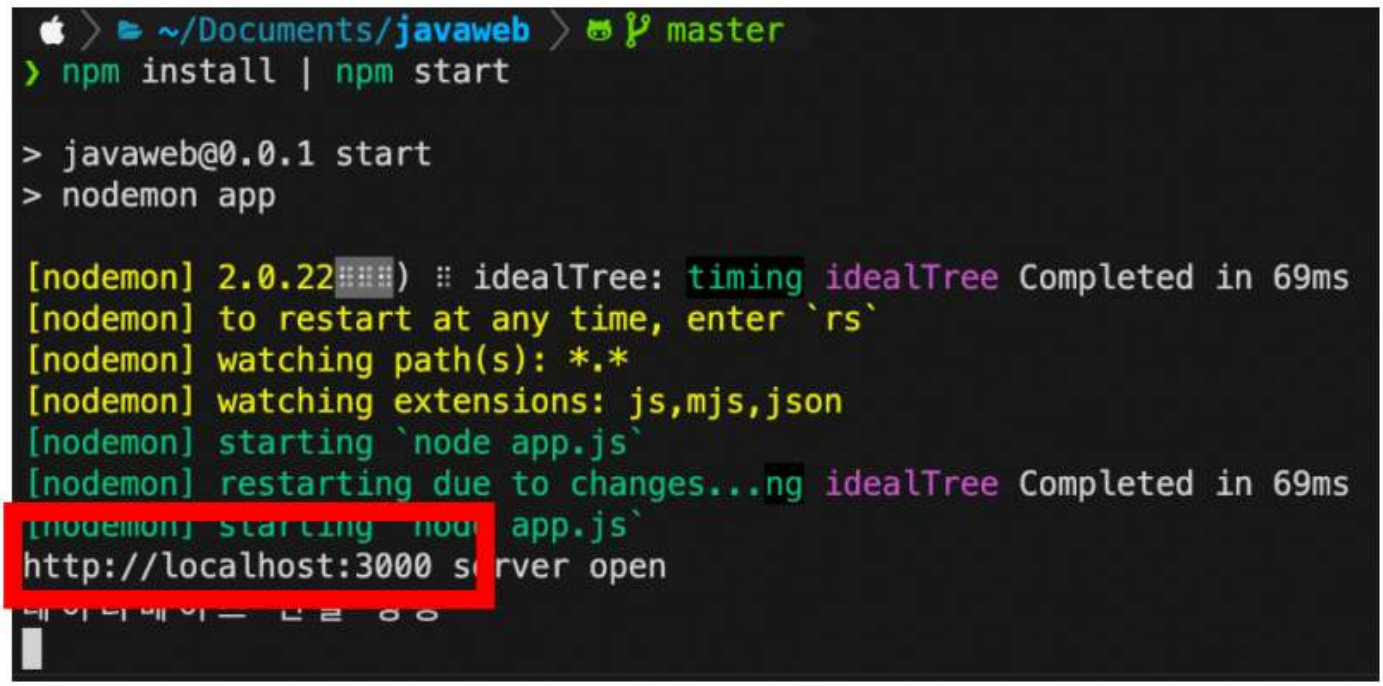
터미널에서 MySQL에 접속해 JavaWeb DB를 생성한다.

1-3. npm 설치 및 시작 (터미널)



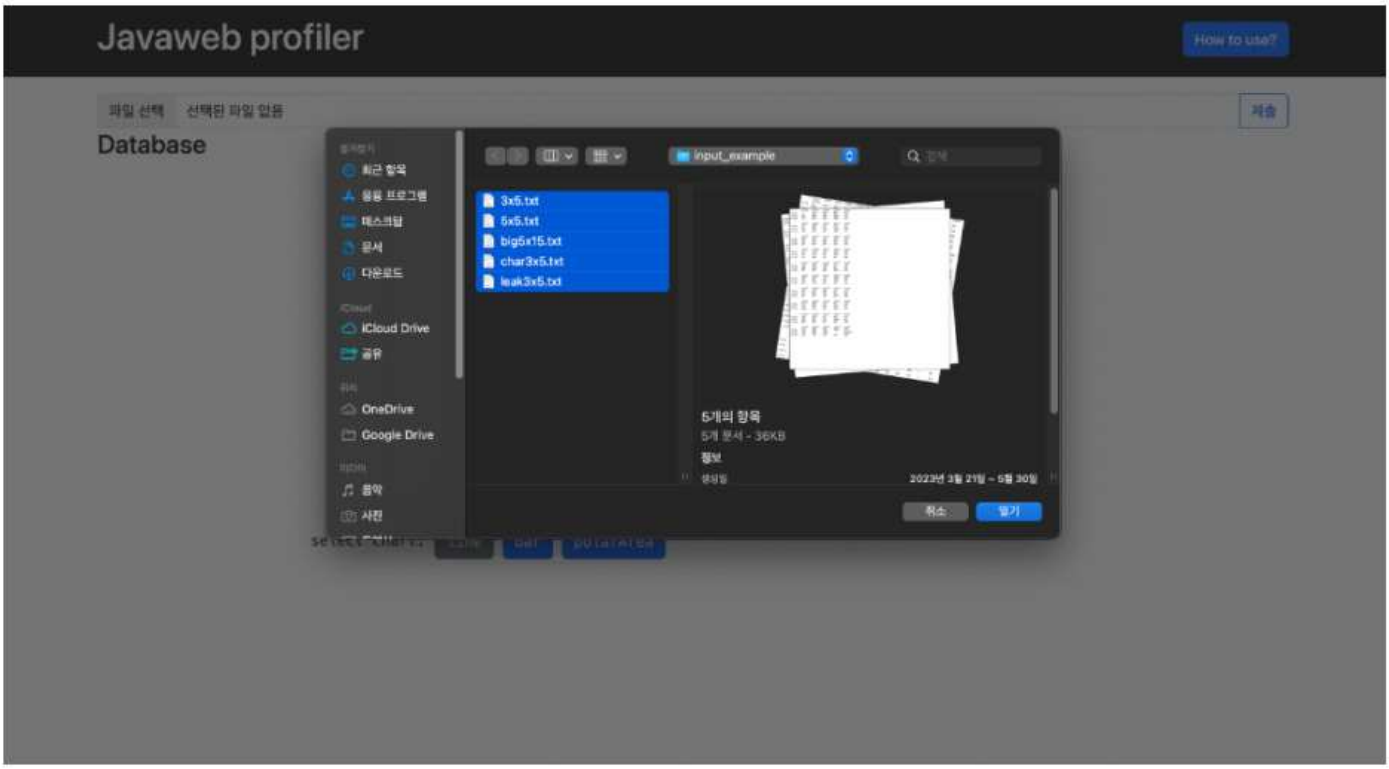
npm 설치를 통해 모듈을 사용한다.

1-4. URL 접속 (터미널)



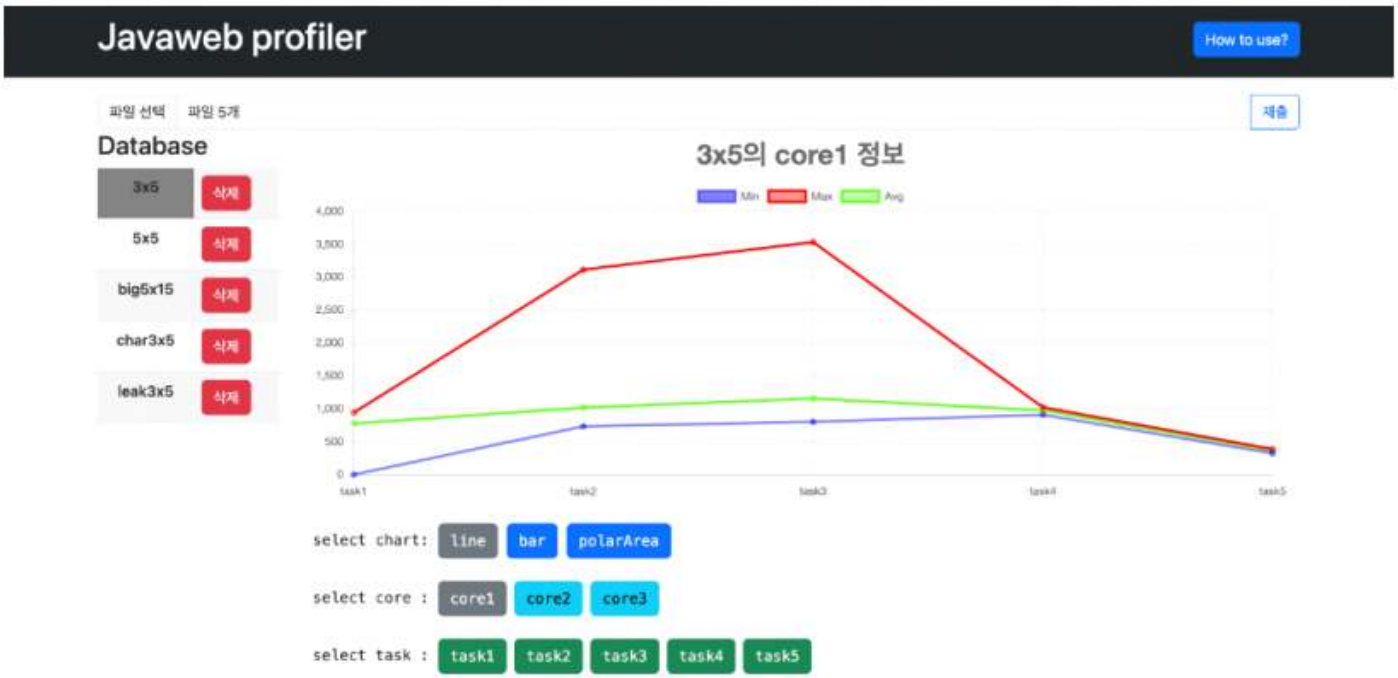
터미널에 있는 주소에 접속한다.

1-5. 파일 선택 (웹)



Profiling 할 파일들을 선택한다.

1-6. 파일 선택 (웹)



좌측엔 파일들의 리스트가 있고, 차트 타입과 core, task를 선택해 자신이 원하는 화면을 볼 수 있다. 또한 좌측 리스트에서 파일들을 삭제할 수 있다.

2. 소스 코드 분석

2-1. config.json

```
{
    "development": {
      "username": "root",
      "password": "*****",
      "database": "javaweb",
      "host": "127.0.0.1",
      "dialect": "mysql",
      "logging": false
    }
}
```

데이터베이스에 연결하기 위한
정보들을 제공한다.

2-2. index.js(1)

```
const Sequelize = require('sequelize');

const env = process.env.NODE_ENV || 'development';
const config = require('../config/config')[env];
const db = {};

const sequelize = new Sequelize(config.database,
    config.username, config.password, config);


async function createTable(tableName){
    const Model = sequelize.define(
        tableName,
        {
            core : {
                type: Sequelize.STRING(20),
                allowNull : false,
            },
            task : {
                type: Sequelize.STRING(20),
                allowNull : false,
            },
            usaged : {
                type:Sequelize.INTEGER.UNSIGNED,
                allowNull:false,
            },
        },
        {
            sequelize,
            timestamps: false,
            underscored: false,
            modelName: 'Profile',
            tableName: tableName,
            paranoid: false,
            charset: 'utf8',
            collate: 'utf8_general_ci',
        }
    );

    await Model.sync();

    return Model;
}
```

동적으로 테이블 이름을 지정하여
데이터베이스에 테이블을 생성한다.

2-2. index.js(2)

```
async function dropTable(tableName) {
    try {
        await sequelize.query(`DROP TABLE IF EXISTS \`${tableName}\``);
        console.log(`테이블 '${tableName}'이(가) 삭제되었습니다.`);
    } catch (error) {
      console.error(`테이블 삭제 중 오류가 발생했습니다: ${error}`);
    }
}

async function createDynamicTable(profile){
    // console.log(profile);
    const tableName = profile[0][0];
    const DynamicModel = await createTable(tableName);

    let core_row = -1;
    for(let row = 1; row<profile.length; row++){
        if(core_row == -1){
            core_row = row;
            continue;
        }
        if(profile[row].length==1){
            core_row = -1;
            continue;
        }
        for(let column = 1; column < profile[row].length; column++){
            try{

            await DynamicModel.create({
                task: profile[core_row][column-1],
                core : profile[row][0],
                usaged : profile[row][column],
            });
            }catch(e){
                console.log(`Error: ${tableName} 파일 데이터 오류 발생`);
            }
        }
    }
}
```

2차원 배열 데이터를 바탕으로 테이블의 구조를 동적으로 생성/삭제하고 , 각 행과 열에 따라 데이터를 삽입하는 과정을 자동화한다.

2-2. index.js(3)

```
async function getTableList() {
    const query = 'SHOW TABLES';

    const [results, metadata] = await sequelize.query(query);

    const tableList = results.map((result) => result.Tables_in_javaweb);

    return tableList;
}

db.sequelize = sequelize;

module.exports = {db, createDynamicTable, sequelize, getTableList, dropTable};
```

데이터베이스 내 모든 테이블의 이름을 조회하고, 이를 모듈의 일부로 내보낸다.

2-3. profile.js

```
const Sequelize = require('sequelize');

class Profile extends Sequelize.Model{
    static initiate(sequelize,tableName){
        Profile.init({
            core : {
                type: Sequelize.STRING(20),
                allowNull : false,
            },
            task : {
                type: Sequelize.STRING(20),
                allowNull : false,
            },
            usaged : {
                type:Sequelize.INTEGER.UNSIGNED,
                allowNull:false,
            },
        },
        {
            sequelize,
            timestamps: false,
            underscored: false,
            modelName: 'Profile',
            tableName: tableName,
            paranoid: false,
            charset: 'utf8',
            collate: 'utf8_general_ci',
        });
    }
    static associations(db){
    }
};
module.exports = Profile;
```

Sequelize를 사용하여 동적으로 테이블 이름을 가진 모델을 정의하고 초기화한다.

2-4. nodes_modules

npm을 통해 받은 외부 라이브러리들이 있다.

2-5. sequelize.js(1)

```
let fileName="";
let selete="";

const profileList = document.querySelectorAll('#profile_list tr td:first-child');
profileList.forEach((el) => {
    el.addEventListener('click', function () {
        fileName = el.textContent;
        profileList.forEach((otherEl) => {
            otherEl.style.setProperty("background-color", "white");
        });
        this.style.setProperty("background-color", "#888888");
        select = undefined;
        if (chart) {
            chart.destroy();
        }
        getdata();
    });
});
```

이벤트 리스너를 추가해 클릭 이벤트를 처리한다..

2-5. sequelize.js(2)

```
20  document.getElementById('profile_form').addEventListener('submit', async (e) => {
21      e.preventDefault();
22
23      const files = document.querySelector('#input_profile').files;
24      let profiles = [];
25      let is_error = false;
26      if(!files){
27          return alert('파일을 등록하세요');
28      }
29      const filePromises = Array.from(files).map((file) => {
30          if (file.name.split(".").pop().toLowerCase() === 'txt') {
31              return new Promise((resolve, reject) => {
32                  readTextFile(file, (data) => {
33                      profiles.push(data);
34                      resolve();
35                  });
36              });
37          } else {
38              alert(".txt파일만 입력해주세요");
39              is_error = true;
40          }
41      });
42
43      await Promise.all(filePromises);
44
45      if(!is_error){
46          const response = await fetch('/profiles',{
47              method: 'POST',
48              headers: {
49                  'Content-Type' : 'application/json'
50              },
51              body: JSON.stringify(profiles)
52          });
53
54          if (response.ok) {
55              response.json().then(data => {
56                  getList();
57                  alert(data.message);
58              });
59          } else {
60              console.error('파일 전송 중 오류가 발생하였습니다.');
61          }
62      }
63  });
```

사용자가 텍스트 파일을 선택하여 제출하면, 이 파일들의 내용을 서버로 전송한다.

2-5. sequelize.js(3)

```
async function getList(){
    const res = await axios.get('profiles');
    const profiles = res.data;

    const tbody = document.querySelector('#profile_list tbody');
    tbody.innerHTML = '';
    profiles.map(function(profile){
        const row = document.createElement('tr');
        const td = document.createElement('td');
        td.textContent = profile;
        td.className= 'text-center fw-semibold';
        td.addEventListener('click',function(){
            fileName = profile;
            const profileList = document.querySelectorAll('#profile_list tr td:first-child');
            profileList.forEach((otherEl) => {
                otherEl.style.setProperty("background-color", "white");
            });
            this.style.setProperty("background-color", "#888888");
            if (chart) {
                chart.destroy();
            }
            getdata();
        });
        if(profile==fileName) td.style.setProperty("background-color", "#888888");
        row.appendChild(td);
        const td2 = document.createElement('td');
        const btndrop = document.createElement('button');
        btndrop.textContent="삭제";
        btndrop.className="btn btn-danger";
        btndrop.addEventListener('click',function(){ deleteTable(`${profile}`); });
        td2.appendChild(btndrop);
        row.appendChild(td2);

        tbody.appendChild(row);
    });
}
```

서버로부터 정보를 가져와 테이블에 동적으로 표시하며, 클릭 및 삭제 기능을 제공한다.

2-5. sequelize.js(4)

```
async function deleteTable(name){
    await axios.delete(`profiles/drop/${name}`);
    if(fileName==name && chart) {
        chart.destroy();
        const task_div = document.querySelector('#core');
        task_div.innerHTML="";
        const core_div = document.querySelector('#task');
        core_div.innerHTML = '';
        fileName = "";
    };
    setTimeout(getList,50);
}
```

정보를 삭제하고, 필요 시 UI를 갱신한다.

2-5. sequelize.js(5)

```
115  async function getdata(){
116
117      const res = await axios.get(`profiles/data/${fileName}`);
118      const cores = res.data.cores;
119      const tasks = res.data.tasks;
120
121      const task_div = document.querySelector('#core');
122      task_div.innerHTML = 'select core : ';
123      tasks.map(function(task){
124          let button = document.createElement('button');
125          button.className = 'btn btn-info me-2';
126          button.textContent = task.core;
127          button.addEventListener('click', function(){
128              updateChart('task', task.core);
129              const coreDiv = document.getElementById('core');
130              const coreBtns = coreDiv.getElementsByClassName('btn');
131              for (let i = 0; i < coreBtns.length; i++) {
132                  coreBtns[i].className = "btn btn-info me-2";
133              }
134              const taskDiv = document.getElementById('task');
135              const taskBtns = taskDiv.getElementsByClassName('btn');
136              for (let i = 0; i < taskBtns.length; i++) {
137                  taskBtns[i].className = "btn btn-success me-2";
138              }
139              this.className = "btn btn-secondary me-2";
140          });
141          task_div.appendChild(button);
142      });
```

```
144      const core_div = document.querySelector('#task');
145      core_div.innerHTML = 'select task : ';
146      cores.map(function(core){
147          let button = document.createElement('button');
148          button.className = 'btn btn-success me-2';
149          button.textContent = core.task;
150          button.addEventListener('click', function(){
151              updateChart('core', core.task);
152              const coreDiv = document.getElementById('core');
153              const coreBtns = coreDiv.getElementsByClassName('btn');
154              for (let i = 0; i < coreBtns.length; i++) {
155                  coreBtns[i].className = "btn btn-info me-2";
156              }
157              const taskDiv = document.getElementById('task');
158              const taskBtns = taskDiv.getElementsByClassName('btn');
159              for (let i = 0; i < taskBtns.length; i++) {
160                  taskBtns[i].className = "btn btn-success me-2";
161              }
162              this.className = "btn btn-secondary me-2";
163          });
164          core_div.appendChild(button);
165      });
166
167  }
```

서버에서 데이터를 가져와서 tasks와 cores에 대한 버튼을 생성하고, 각 버튼에 클릭 이벤트를 추가하여 UI를 동적으로 생성한다.

2-5. sequelize.js(6)

```
169   function readTextFile(file, save) {
170       const reader = new FileReader();
171
172       reader.onload = async function(event) {
173           const contents = event.target.result;
174           let line_parse = contents.split("\n");
175           const parse = [[file.name]];
176           for(let i=0; i<line_parse.length; i++){
177               parse.push(line_parse[i].trim().split(/\t| |,|\//));
178           }
179           save(parse);
180       };
181
182       reader.onerror = function(event){
183           console.error("잘못된 파일");
184       }
185
186       reader.readAsText(file, 'UTF-8');
187
188   }
```

주어진 파일을 읽고, 그 내용을 파싱하여 주어진 save 콜백 함수로 전달한다.

2-5. sequelize.js(7)

```
190   let chart;
191   let chart_type = 'line';
192   let labels = [];
193   let minData = [];
194   let maxData = [];
195   let avgData = [];
196
197   const btnline = document.getElementById('line');
198   const btnbar = document.getElementById('bar');
199   const btnpolarArea = document.getElementById('polarArea');
200
201   btnline.addEventListener('click', function () {
202       chart_type = 'line';
203       btnline.className="btn btn-secondary"; btnbar.className="btn btn-primary"; btnpolarArea.className="btn btn-primary";
204       if(fileName.length!=0) updateChart(null,null);
205   })
206   btnbar.addEventListener('click', function () {
207       chart_type = 'bar';
208       btnline.className="btn btn-primary"; btnbar.className="btn btn-secondary"; btnpolarArea.className="btn btn-primary";
209       if(fileName.length!=0) updateChart(null,null);
210   });
211   btnpolarArea.addEventListener('click', function () {
212       chart_type = 'polarArea';
213       btnline.className="btn btn-primary"; btnbar.className="btn btn-primary"; btnpolarArea.className="btn btn-secondary";
214       if(fileName.length!=0) updateChart(null,null);
215    });
```

웹 페이지에서 차트 유형을 변경할 수 있는 버튼들과 그에 대한 이벤트 리스너를 설정한다.

2-5. sequelize.js(8)

```
async function updateChart(type, choose_name){

    const profiler = document.getElementById('profiler').getContext('2d');
    if (chart) {
        chart.destroy();
    }

    if(type == 'core'){
        select = choose_name;
        const res = await axios.get(`profiles/taskdata/${fileName}/${select}`);
        const datas = res.data;

        labels = [];
        minData = [];
        maxData = [];
        avgData = [];
        datas.forEach((data) => {
            labels.push(data.core);
            minData.push(data.min_usaged);
            maxData.push(data.max_usaged);
            avgData.push(data.avg_usaged);
        });

    }else if(type == 'task'){
        select = choose_name;
        const res = await axios.get(`profiles/coredata/${fileName}/${select}`);
        const datas = res.data;

        labels = [];
        minData = [];
        maxData = [];
        avgData = [];
        datas.forEach((data) => {
            labels.push(data.task);
            minData.push(data.min_usaged);
            maxData.push(data.max_usaged);
            avgData.push(data.avg_usaged);
        });

    }
```

2-5. sequelize.js(9)

```
257     if(fileName==undefined || select==undefined) return;
258
259     chart = new Chart(profiler, {
260         type: `${chart_type}`,
261         data: {
262           labels: labels,
263           datasets: [{
264             label: 'Min',
265             data: minData,
266             borderColor: 'rgba(0, 0, 255, 0.5)',
267             backgroundColor: 'rgba(0, 0, 255, 0.5)',
268           }, {
269             label: 'Max',
270             data: maxData,
271             borderColor: 'rgba(255, 0, 0, 1)',
272             backgroundColor: 'rgba(255, 0, 0, 0.5)',
273           }, {
274             label: 'Avg',
275             data: avgData,
276             borderColor: 'rgba(100, 255, 30, 1)',
277             backgroundColor: 'rgba(100, 255, 30, 0.5)',
278           }]
279         },
280         options: {
281           maintainAspectRatio: false,
282           scales: {
283             y: {
284               beginAtZero: true
285             }
286           },
287           plugins: {
288             title: {
289               display: true,
290               text: `${fileName}의 ${select} 정보`,
291               font: {
292                 size: 30
293               }
294             }
295           }
296         },
297     });
298
299 }
```

차트의 유형에 따라 다른 데이터를 가져와서 차트를 업데이트하는 기능을 제공한다.

2-6. index.js

```
const express = require('express');
const router = express.Router();

const { getTableList } = require('../models/index');
router.get('/', async (req,res)=>{

    getTableList()

        .then((tableList) => {
            res.render('index', {tableList});
        })

        .catch((error) => {
        console.error('테이블 리스트 조회 중 오류가 발생하였습니다:', error);
        });
});
module.exports = router;
```

Express를 사용하여 웹에서 데이터베이스의 테이블 목록을 조회하고, 조회 결과를 웹 페이지에 렌더링하는 기능을 구현한다.

2-7. index.html

사용자로부터 데이터 파일을 입력 받아, 데이터베이스에 저장하고, 저장된 데이터를 바탕으로 차트를 보여준다.

2-8. profiles.js(1)

```
const express = require('express');
const router = express.Router();
const { createDynamicTable, getTableList, sequelize, dropTable } = require('../models/index');
const profile_model = require('../models/profile');

router.post('/', async (req, res) => {
    const profiles = req.body;
    let count = 0;
    try {
        const tableList = await getTableList();

    for (let file_num = 0; file_num < profiles.length; file_num++) {
        profiles[file_num][0][0] = profiles[file_num][0][0].toLowerCase().slice(0,-4);

        if (tableList.includes(profiles[file_num][0][0])) {
          console.log("이미 존재하는 파일입니다");
          continue;
        }

        await createDynamicTable(profiles[file_num]);
        count++;
    }
        if(count===0){
            res.json({ status: 'success', message: `저장 가능한 파일이 존재하지 않습니다.` });
        }else if(count==profiles.length){
            res.json({ status: 'success', message: `${count}개의 프로파일이 정상적으로 저장되었습니다.` });
        }else{
            res.json({ status: 'success', message: `중복된 이름의 파일을 제외한 ${count}개의 프로파일이 저장되었습니다.` });
        }

    } catch (error) {
        console.error('오류가 발생하였습니다:', error);
        res.json({ status: 'error', message: '오류가 발생하였습니다.' });
    }
});

router.get('/', async (req,res)=>{
    const tableList = await getTableList();
    res.json(tableList);
});
```

데이터베이스와의 상호작용을 통해 동적으로 테이블을 생성하고, 테이블 목록을 관리한다.

2-8. profiles.js(2)

```
router.get('/', async (req,res)=>{
    const tableList = await getTableList();
    res.json(tableList);
});

router.get('/data/:tableName', async (req,res)=>{
    try{
        const {tableName} = req.params;

        const tableList = await getTableList();

        if(!tableList.includes(tableName)){
            return res.status(404).json({error:'존재하지 않는 파일입니다.'});
        }

        profile_model.initiate(sequelize, tableName);

        const datas = await profile_model.findAll();

        const tasks = await profile_model.findAll({
            attributes: [sequelize.fn('DISTINCT', sequelize.col('core')), 'core'],
        });

        const cores = await profile_model.findAll({
            attributes: [sequelize.fn('DISTINCT', sequelize.col('task')), 'task'],
        });

        res.json({datas: datas, cores : cores, tasks : tasks});
    }catch(error){
        console.error('데이터 조회 오류', error);
    }
});

router.delete('/drop/:tableName', async(req,res)=>{
    try{
        const {tableName} = req.params;
        dropTable(tableName);
        res.json({state:'success'});
    }catch(error){
        res.json({state:'error'});
    }
});
```

특정 데이터베이스 테이블의 데이터를 조회하거나 삭제한다.

2-8. profiles.js(3)

```
router.get('/coredata/:tableName/:core', async(req,res)=>{

    const { tableName, core } = req.params;

    profile_model.initiate(sequelize, tableName);

    const data = await profile_model.findAll({
        attributes: [
          'task',
          [sequelize.fn('max', sequelize.col('usaged')), 'max_usaged'],
          [sequelize.fn('min', sequelize.col('usaged')), 'min_usaged'],
          [sequelize.fn('avg', sequelize.col('usaged')), 'avg_usaged']
        ],
        where: {
          core: core
        },
        group: ['task']
      });

    res.json(data);
});

router.get('/taskdata/:tableName/:task', async(req,res)=>{
    const { tableName, task } = req.params;

    profile_model.initiate(sequelize, tableName);

    const data = await profile_model.findAll({
        attributes: [
          'core',
          [sequelize.fn('max', sequelize.col('usaged')), 'max_usaged'],
          [sequelize.fn('min', sequelize.col('usaged')), 'min_usaged'],
          [sequelize.fn('avg', sequelize.col('usaged')), 'avg_usaged']
        ],
        where: {
          task: task,
        },
        group: ['core']
      });

    res.json(data);
});

module.exports = router;
```

core 또는 task 값에 따라 데이터를 조회하고, 그룹화된 데이터에서 최대값, 최소값, 평균값을 계산하여 클라이언트에게 반환한다.

2-9. app.js

```
const express = require('express');
const morgan = require('morgan');
const path = require('path');
const nunjucks = require('nunjucks');
const indexRouter = require('./routes');
const profilesRouter = require('./routes/profiles');
const {sequelize} = require('./models');
const app = express();
app.set('port', process.env.PORT || 3000);
app.set('view engine', 'html');

nunjucks.configure('views',{
    express: app,
    watch: true,
});

sequelize.sync({force : false})
    .then(()=>{
        console.log('데이터베이스 연결 성공');
    })
    .catch((err)=>{
        console.error(err);
    })

app.use(express.static(path.join(__dirname,'public')));
app.use(express.json());
app.use(express.urlencoded({extended: false}));

app.use('/',indexRouter);
app.use('/profiles',profilesRouter);

app.use((req,res,next)=>{
    const error = new Error(`${req.url}은 잘못된 주소입니다.`);
    error.status = 404;
    next(error);
});

app.use((err,req,res,next)=>{
    res.locals.message = err.message;
    res.status(err.status || 500);
    res.render('error');
});

app.listen(app.get('port'), () =>{
    console.log("http://localhost:"+app.get('port')+" server open");
});
```

기본적인 웹 서버 기능 외에 데이터베이스 연결, 템플릿 엔진 사용, 정적 파일 제공, 에러 처리 등 웹 애플리케이션 개발에 필요한 다양한 기능을 포함한다.