미세먼지 웹사이트

융합소프트웨어학부 60171670 홍유진

2019.05.07.

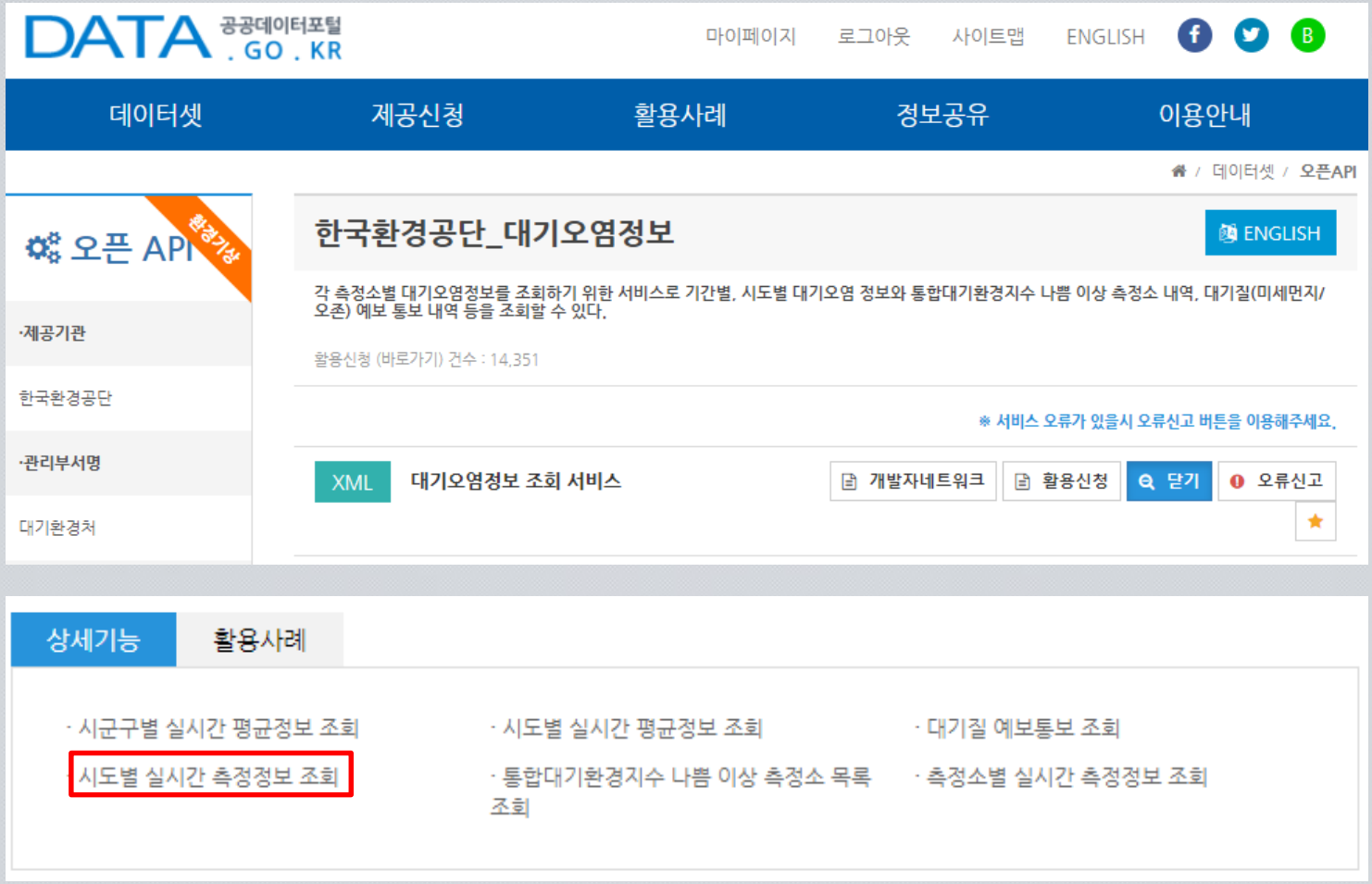
Contents

01

사용 데이터 및 API

1. 미세 먼지 측정치
2. 측정소 정보
3. 지도

미세 먼지 측정치 공공데이터 포털



공공 데이터 포털 (www.data.go.kr)에서 제공하는 대기 오염 정보 조회 API를 사용
시도별 실시간 측정 정보 조회 기능을 사용

미세 먼지 측정치 공공데이터 포털

사용 오퍼레이션 명 : getCtprvnRltmMesureDnsty

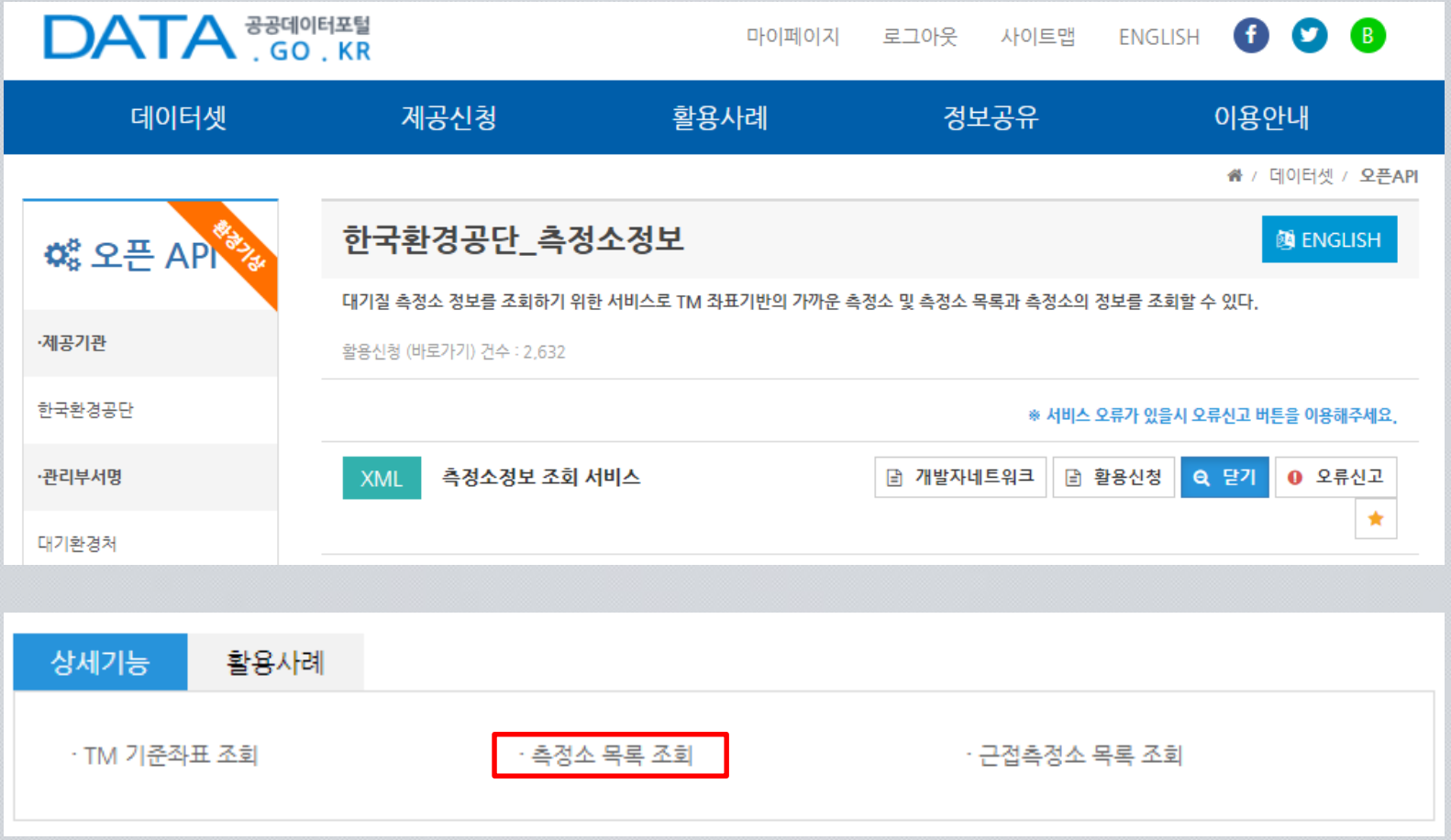
오퍼레이션 기능 : 시도별 측정소목록에 대한 일반 항목 (SO2, pm10, pm2.5…),CAI 최종 실시간 측정값과 지수 정보 조회 기능 제공

Request Parameters : sidoName (시도 이름 (서울, 부산, 대구, 인천, 광주, 대전, 울산, 경기, 강원, 충북, 충남, 전북, 전남, 경북, 경남, 제주, 세종))

미세 먼지 측정치 공공데이터 포털

항목명(영문)	항목명(국문)	항목설명
resultCode	결과코드	결과코드
resultMsg	결과메세지	결과메세지
numOfRows	한 페이지 결과 수	한 페이지 결과 수
pageNo	페이지 번호	페이지 번호
totalCount	전체 결과 수	전체 결과 수
items	목록	목록
stationName	측정소 명	측정소 이름
mangName	측정망 정보	측정망 정보 (국가배경, 교외대기, 도시대기, 도로변대기)
dataTime	측정일시	오염도 측정 연-월-일 시간 : 분
so2Value	아황산가스 농도	아황산가스 농도(단위 : ppm)
coValue	일산화탄소 농도	일산화탄소 농도(단위 : ppm)
o3Value	오존 농도	오존 농도(단위 : ppm)
no2Value	이산화질소 농도	이산화질소 농도(단위 : ppm)
pm10Value	미세먼지(PM_{10}) 농도	미세먼지(PM_{10}) 농도 (단위 : $\mu g/m^3$)
pm10Value24	미세먼지(PM_{10}) 24시간예측이동농도	미세먼지(PM_{10})24시간예측이동농도(단위 : $\mu g/m^3$)
pm25Value	미세먼지($PM_{2.5}$) 농도	미세먼지($PM_{2.5}$) 농도(단위 : $\mu g/m^3$)
pm25Value24	미세먼지($PM_{2.5}$) 24시간예측이동농도	미세먼지($PM_{2.5}$) 24시간예측이동농도(단위 : $\mu g/m^3$)
khai	통합대기환경수치	통합대기환경수치
khaiGrade	통합대기환경지수	통합대기환경지수
so2Grade	아황산가스 지수	아황산가스 지수
coGrade	일산화탄소 지수	일산화탄소 지수
o3Grade	오존 지수	오존 지수
no2Grade	이산화질소 지수	이산화질소 지수
pm10Grade	미세먼지(PM_{10}) 24시간 등급	미세먼지(PM_{10}) 24시간 등급자료
pm25Grade	미세먼지($PM_{2.5}$) 24시간 등급	미세먼지($PM_{2.5}$) 24시간 등급자료
pm10Grade1H	미세먼지(PM_{10}) 1시간 등급	미세먼지(PM_{10}) 1시간 등급자료
pm25Grade1H	미세먼지($PM_{2.5}$) 1시간 등급	미세먼지($PM_{2.5}$) 1시간 등급자료

측정소 목록 조회 공공데이터 포털



측정소 목록 조회 공공데이터 포털

사용 오퍼레이션 명 : getMsrstnList

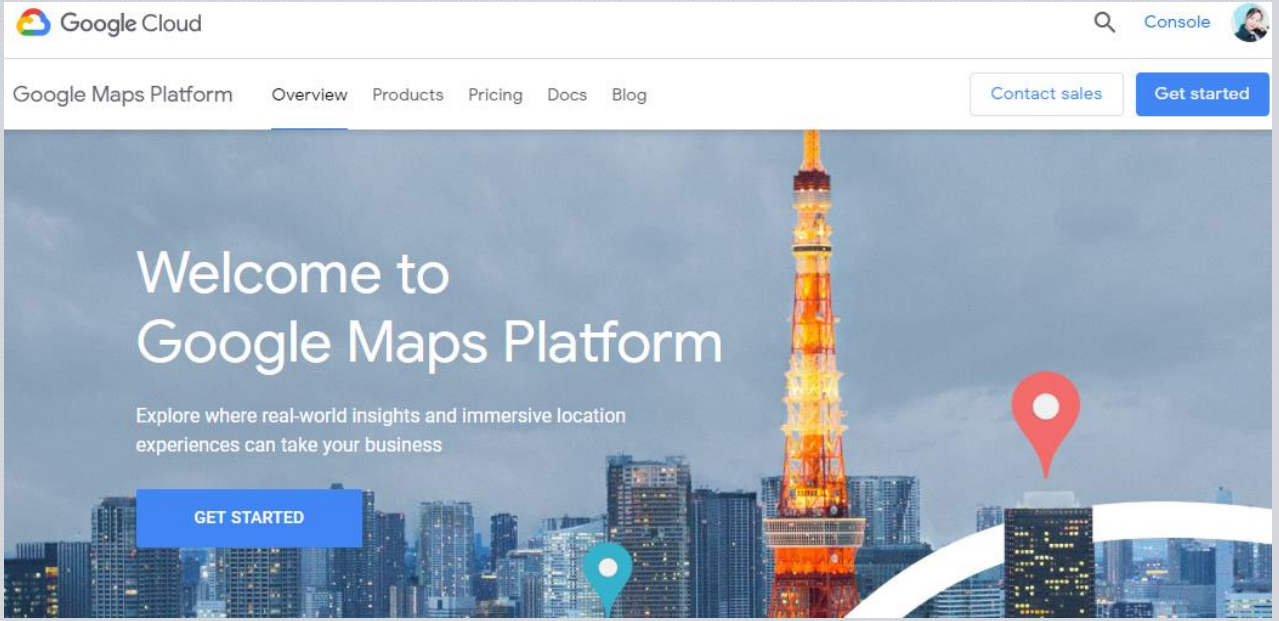
오퍼레이션 기능 : 측정소 주소/명칭으로 측정소 목록/단 1건의 측정소 상세 정보 제공

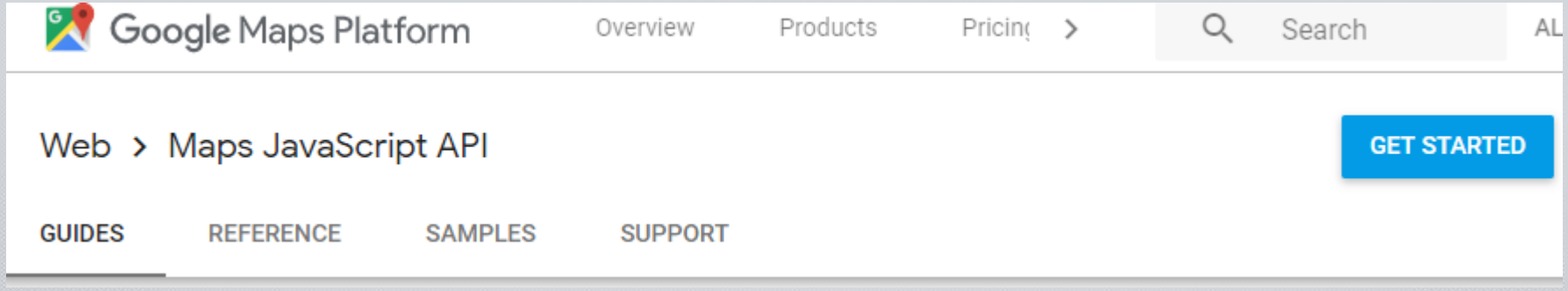
Request Parameters : stationName (측정소 이름)

측정소 목록 조회 공공데이터 포털

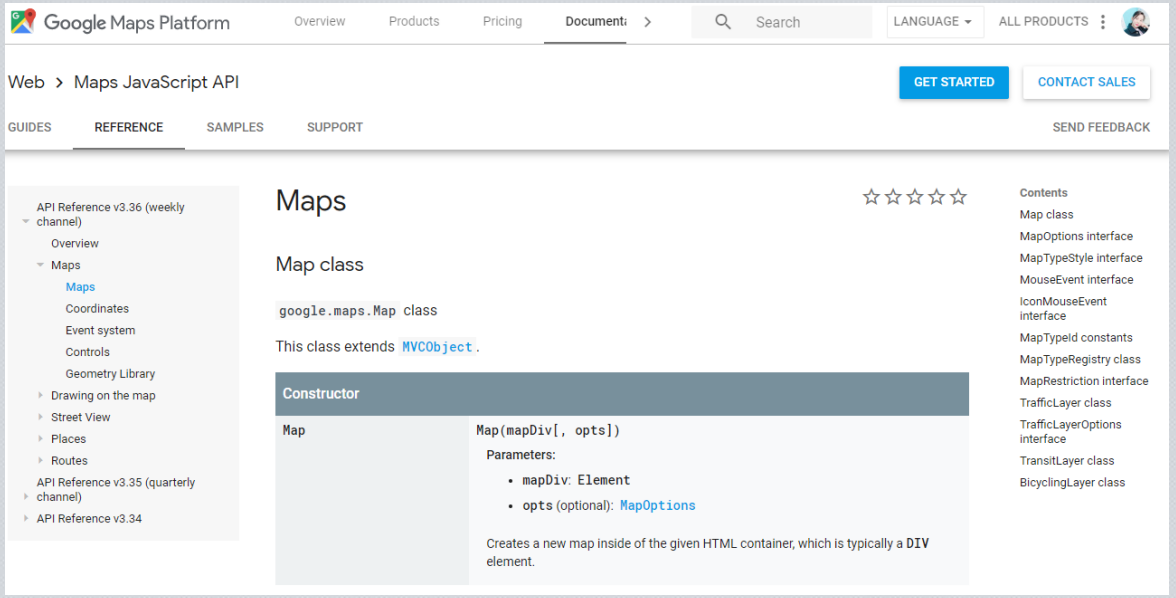
항목명(영문)		항목명(국문)	항목설명
resultCode		결과코드	결과코드
resultMsg		결과메세지	결과메세지
numOfRows		한 페이지 결과 수	한 페이지 결과 수
pageNo		페이지 번호	페이지 번호
totalCount		전체 결과 수	전체 결과 수
items		목록	목록
	stationName	측정소 명	측정소 이름
	addr	측정소 주소	측정소가 위치한 주소
	year	설치년도	측정소 설치년도
	oper	관리기관명	측정소 관리기관 이름
	photo	측정소 이미지	측정소 이미지
	vrml	측정소 전경	측정소의 주변 전경
	map	측정소 지도이미지	측정소가 설치된 장소 지도 이미지
	mangName	측정망	측정망
	item	측정항목	측정소 측정항목 (SO_2, CO, O_3, NO_2, PM10)
	dmX	위도(Latitude)	WGS84기반 X좌표
	dmY	경도(Longitude)	WGS84기반 Y좌표

지도 Google Maps





지도 Google Maps



MapOptions interface

`google.maps.MapOptions` interface

MapOptions object used to define the properties that can be set on a Map.

Properties	
`backgroundColor`	Type: `string` Color used for the background of the Map div. This color will be visible when tiles have not yet loaded as the user pans. This option can only be set when the map is initialized.
`center`	Type: `LatLng|LatLngLiteral` The initial Map center. Required.

02

코드

1. 검색
2. 측정소 정보
3. 지도

코드

Reducer

```
import axios from 'axios';

const API_KEY = 'zH931mm1oAC1kVXsp2LBfgb55%2BZFsQNKQQpxMTo2XqeVsxMeA3bjw3aHIqZ5Hl0RhQZOCfB08mLPMzdTSsjjPw%3D%3D';
const ROOT_URL = 'http://openapi.airkorea.or.kr/openapi/services/rest/ArpltnInforInqireSvc/getCtprvnRltmMesureDnsty';
const proxyurl = "https://cors-anywhere.herokuapp.com/";



export const FETCH_AIR = 'FETCH_AIR';
export const FETCH_INDEX = 'FETCH_INDEX';
export const FETCH_ADDRESS = 'FETCH_ADDRESS';

export function fetchAir(city) {
  const url = `${proxyurl}${ROOT_URL}?sidoName=${city}&ServiceKey=${API_KEY}&ver=1.3&numOfRows=100&_returnType=json`;
  const request = axios.get(url, {
    headers:{'Access-Control-Allow-Origin': '*',
    'Content-Type': 'application/json',
    'Access-Control-Allow-Headers': 'Origin, X-Requested-With, Content-Type, Accept, Authorization'},
  });
  return {
    type: FETCH_AIR,
    payload: request
  };
}

export function fetchIndex(index){
  return{
    type: FETCH_INDEX,
    index
  }
}
```

CORS Policy 때문에 localhost로 받아지지 않기 때문에, proxy를 header에 설정

코드

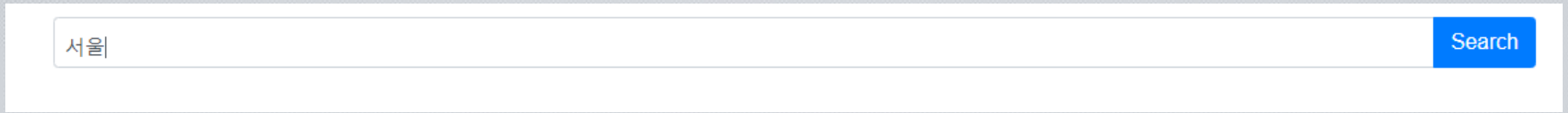
Reducer

```
export function fetchAddress(station) {
  const url = `${proxyurl}http://openapi.airkorea.or.kr/openapi/services/rest/MsrstnInfoInqireSvc/getMsrstnList?stationName=${station}&ServiceKey=${API_KEY}&_returnType=json`;
  const request = axios.get(url, {
    headers:{'Access-Control-Allow-Origin': '*',
    'Content-Type': 'application/json',
    'Access-Control-Allow-Headers': 'Origin, X-Requested-With, Content-Type, Accept, Authorization'},
  });
  return {
    type: FETCH_ADDRESS,
    payload: request
  };
}
```

CORS Policy 때문에 localhost로 받아지지 않기 때문에, proxy를 header에 설정

코드

검색




```
import { FETCH_AIR } from '../actions';

function filterMangName(sites){
  return sites.mangName === '도시대기';
}

export default function(state = {
  loading: false, error: '', data: []
}, action) {
  console.log(action);
  switch (action.type) {
  case `${FETCH_AIR}_PENDING`:
    return {
      loading: true,
      error: '',
      data: [...state.data]
    };
  case `${FETCH_AIR}_FULFILLED`:
  console.log(action.payload);
    var city = {
      name: action.payload.data.parm.sidoName,
      sites: action.payload.data.list
      }
    var s = city.sites.filter(filterMangName);
    s.sort(function(s1, s2){
      if (s1.stationName > s2.stationName) { return 1; }
      if (s1.stationName < s2.stationName) { return -1; }
      return 0;
    })
    return {
      loading: false,
      error: '',
      data: [s]
    };
```

sites: payload에 담긴 측정소 정보 list만 추출
s: sites 중 mangName이 도시대기인 것만 추출

그 후, s를 오름차순으로 정렬

filter와 sort를 거친 결과를 data에 담음

코드

검색

```
  renderStation(s){
    var x = s;
    return(
      <ul>
        {x.map(this.renderItem.bind(this))}
      </ul>
    );
  }

  render(){
    console.log("STATION_LIST GOT THIS.PROPS.AIR: ", this.props.air)
    if (this.props.air.length === 0){
      return(<ul></ul>)
    }
    else {
      return(
          <ul>
            {this.props.air.map(this.renderStation.bind(this))}
          </ul>
      )
    }
  }
}

function mapStateToProps(state) {
  return {
    air: state.air.data
  };
}

function mapDispatchToProps(dispatch) {
  return bindActionCreators({ fetchIndex, fetchAddress }, dispatch);
}

export default connect(mapStateToProps, mapDispatchToProps)(StationList);
```

```
function Button(props) {
  return (
    <button value={props.index} type='button' className='stations'
    onClick={props.onClick}>
      <span>{props.name}</span>
    </button>
  )
}

class StationList extends Component {
  constructor(props) {
    super(props);
  }

  renderItem(item, index) {
    return <Button key={index} index={index} name={item.stationName} onClick={()=> {
      this.handleClick(index);
    }}/>
  }

  handleClick(index){
    this.props.fetchIndex(index);
    var i = index;
    var s = this.props.air[0];
    var final = parseInt(i, 10);
    var st = s[final].stationName;
    this.props.fetchAddress(st);
  }
```

각 Button에 Map()함수를 통하여 index값 부여
Button이 클릭될 때마다 handleClick 함수 실행

fetchIndex 함수에 index를 인자로 주어 실행
fetchAddress 함수에 측정소 이름을 인자로 주어 실행

코드

동네 선택


City
Search
강남구
강동구
강북구
강서구
관악구
광진구
구로구
금천구
노원구
도봉구
동대문구
동작구
마포구
서대문구
서초구
성동구
성북구
송파구
양천구
영등포구
용산구
은평구
종로구
중구
중랑구

코드

동네 선택

```
import { FETCH_INDEX } from '../actions';

export default function(state = -1, action) {
    switch (action.type) {
        case FETCH_INDEX:
        console.log("INDEX_REDUCER GOT INDEX:", action.index)
            return action.index;

        default:
            return state
        }
}
```

받은 index에 해당하는 측정소명, 대기통합지수, pm10 측정치, pm2.5 특정치를 render()

```
render() {
  if (parseInt(this.props.index, 10) === -1){
    return(
      <div className='air-list' >
      { this.handleError() }
    </div>
    );

  }
  else {
    console.log("AIR_INFO PROPS.AIR: ", this.props.air);
    var i = this.props.index;
    var s = this.props.air[0];
    var final = parseInt(i, 10);
     return (
      <div className='air-list' >
        { this.handleError() }
        <table className='table table-hover'>
          <thead>
            <tr>
              <th>Station</th>
              <th>KhaiGrade</th>
              <th>pm10</th>
              <th>pm2.5</th>
            </tr>
          </thead>
          <tbody>
            <tr>
              <td>{s[final].stationName}</td>
              <td>{s[final].khaiGrade}</td>
              <td>{s[final].pm10Value}</td>
              <td>{s[final].pm25Value}</td>
            </tr>
          </tbody>
        </table>
      </div>
    );
  }
 }
}
```

코드

동네 선택

```
import { FETCH_ADDRESS } from '../actions';

export default function(state = { addr: [] }, action) {
  console.log(action);
  switch (action.type) {
  case `${FETCH_ADDRESS}_PENDING`:
    return {
      addr: [...state.addr]
    };
  case `${FETCH_ADDRESS}_FULFILLED`:
  console.log("FETCH ADDRESS GOT: ", action.payload);
    var a = {
      xc: action.payload.data.list[0].dmX,
      yc: action.payload.data.list[0].dmY
      }
    return {
      addr: [a]
    };
  case `${FETCH_ADDRESS}_REJECTED`:
    return {
      addr: [...state.addr],
    };
  default:
    return state;
  }
}
```

```
render() {
  if (this.props.address.addr.length === 0){
    console.log("ADDR LENGTH 0")
    return(
      <div>
      </div>
    )
  }
  else {
    console.log("ADDRESS GOT: ", this.props.address);
    var lat = parseFloat(this.props.address.addr[0].xc, 10);
    var lng = parseFloat(this.props.address.addr[0].yc, 10);

    return (
        <GoogleMap lat={lat} lng={lng}/>
      );
    }
  }
}
```

a: payload에 담긴 좌표값 dmX와 dmY만 추출

address에 담긴 xc값과 yc값을 GoogleMap의 인자로 주어 새로운 map 생성 후 render()

코드 측정치 및 지도

City Search

강남구 강동구 강북구 강서구 관악구 광진구 구로구 금천구

노원구 도봉구 동대문구 동작구 마포구 서대문구 서초구

성동구 성북구 송파구 양천구 영등포구 용산구 은평구

종로구 중구 중랑구

Station	KhaiGrade	pm10	pm2.5
강남구	2	40	13