나와 색을 맞추다

깔맞춤

포팅 매뉴얼

삼성SW청년아카데미 대전캠퍼스 7기

공통 프로젝트 B208

2022. 07. 05. ~ 2022. 08. 19.

송다경 강민성 김민영 김찬일 오정환 이한기

개발 환경

형상 관리

- Gitlab

이슈 관리

- Jira

Communication

- Mattermost
- Webex
- Notion

OS

- Windows 10

UI/UX

- Figma

IDE

- IntelliJ (2022.1.3)
- Visual Studio Code (1.69.X)

DataBase

- MySQL (8.0.29)

Server

- AWS EC2
  - Ubuntu 20.04 LTS
  - Docker 20.10.17

기타 편의 툴

- Postman 9.28.2
- Source Tree 3.4.9
- Termius 7.46.2

Front-End

- HTML5, CSS3, JavaScript(ES6)
- Vue 2.6.14
  - Vue/cli 5.0.8
  - Vuex 3.6.2
  - Vue Router 3.5.1

Back-End

- Java Open-JDK zulu 8.33.0.1
- SpringBoot Gradle 2.7.2
  - Spring Data JPA
  - Lombok
  - Swagger 2.9.2
  - jjwt 0.11.2

WebRTC

- OpenVidu 2.22.0

AWS S3

- aws-sdk 2.1192.0

Open Source API

- color-name-list 9.19.0

- nearest-color 0.4.4
- face-api.js 0.22.2

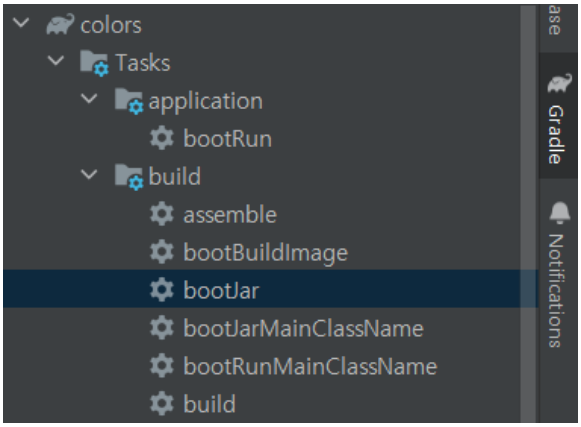
Node Package

```
"dependencies": {
    "@caohenghu/vue-colorpicker": "^1.2.4",
    "@tensorflow/tfjs-core": "^3.19.0",
    "@tensorflow/tfjs-node": "^3.19.0",
    "aos": "^2.3.4",
    "aws-sdk": "^2.1192.0",
    "axios": "^0.27.2",
    "color-name-list": "^9.19.0",
    "core-js": "^3.8.3",
    "dotenv": "^16.0.1",
    "express": "^4.17.1",
    "face-api.js": "^0.22.2",
    "fs": "^0.0.1-security",
    "html2canvas": "^1.4.1",
    "jquery": "^3.6.0",
    "materialize-css": "^1.0.0-rc.2",
    "nearest-color": "^0.4.4",
    "node-sass": "^7.0.1",
    "openvidu-browser": "^2.22.0",
    "request": "^2.88.2",
    "sass-loader": "^13.0.2",
    "sweetalert": "^2.1.2",
    "tslib": "^1.6.1",
    "util": "^0.12.4",
    "vue": "^2.6.14",
    "vue-router": "^3.5.1",
    "vue-aos": "^2.3.0",
    "vuex": "^3.6.2",
    "vuex-persistedstate": "^4.1.0"
  },
  "devDependencies": {
    "@babel/core": "^7.12.16",
    "@babel/eslint-parser": "^7.12.16",
    "@vue/cli-plugin-babel": "~5.0.0",
    "@vue/cli-plugin-eslint": "~5.0.0",
    "@vue/cli-plugin-router": "~5.0.0",
    "@vue/cli-plugin-vuex": "~5.0.0",
    "@vue/cli-service": "~5.0.0",
    "eslint": "^7.32.0",
    "eslint-config-prettier": "^8.3.0",
    "eslint-plugin-prettier": "^4.0.0",
    "eslint-plugin-vue": "^8.0.3",
    "prettier": "^2.4.1",
    "vue-template-compiler": "^2.6.14"
  }
```

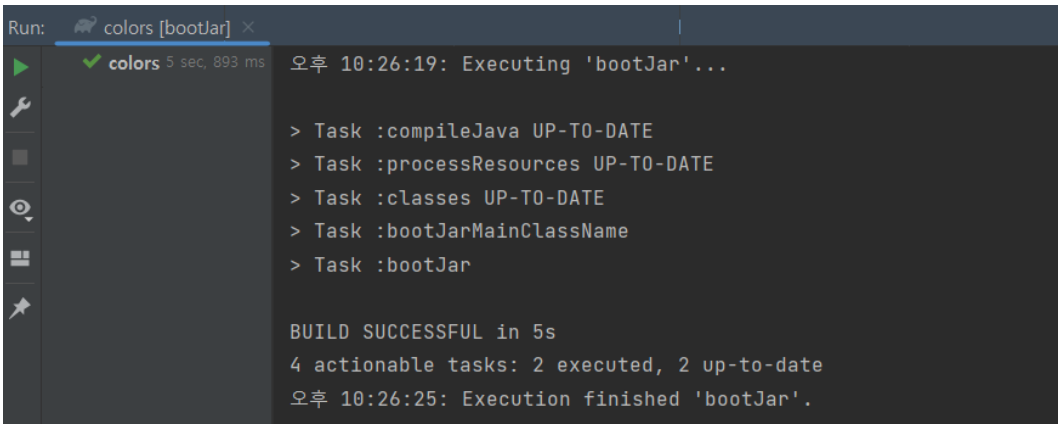
Back-end Build

GUI 빌드 방법

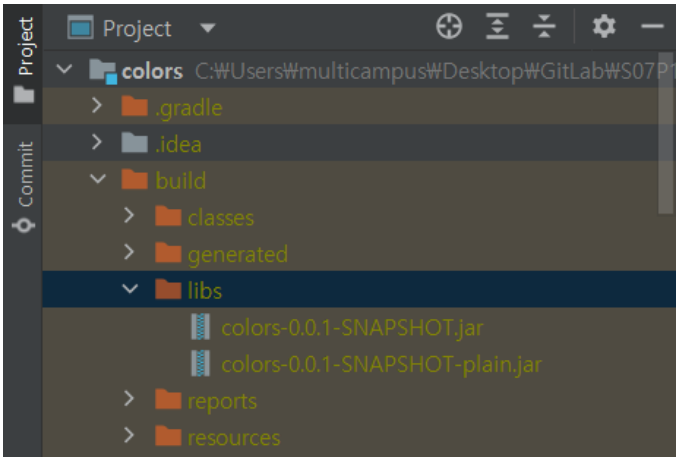
1. IntelliJ로 프로젝트를 열고, 우측의 Gradle 탭에서 Tasks > build > bootJar를 더블 클릭하여 실행한다.



2. 빌드 성공 시 다음과 같은 문구가 나타난다.



3. build > libs 디렉토리에 아래와 같이 빌드 된 jar 파일이 생성되어 있다.



CLI 빌드 방법

1. 다음과 같이 프로젝트의 위치로 이동하여 `.\gradlew.bat build` 명령어를 수행한다.

```
Terminal:  Local
PS C:\Users\multicampus\Desktop\GitLab\S07P12B208\BE\colors> .\gradlew.bat build

BUILD SUCCESSFUL in 3s
7 actionable tasks: 1 executed, 6 up-to-date
```

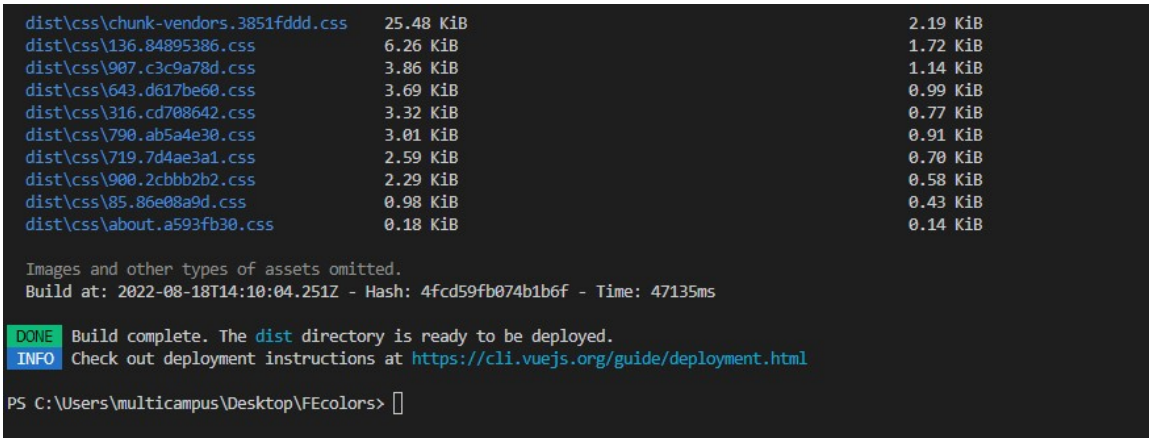
2. 빌드에 성공하면 build > libs로 이동하였을 때, 빌드 된 jar 파일을 확인할 수 있다.

```
Terminal:  Local
PS C:\Users\multicampus\Desktop\GitLab\S07P12B208\BE\colors> cd .\build\libs\
PS C:\Users\multicampus\Desktop\GitLab\S07P12B208\BE\colors\build\libs> dir


    디렉터리: C:\Users\multicampus\Desktop\GitLab\S07P12B208\BE\colors\build\libs


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
-a----        2022-08-18  오후 10:40          88245 colors-0.0.1-SNAPSHOT-plain.jar
-a----        2022-08-18  오후 10:44       51812744 colors-0.0.1-SNAPSHOT.jar
```

Front-end Build

Vue Project 빌드

1. vue project에서 `npm run build` 명령어로 프로젝트를 빌드한다.
2. 빌드에 성공하면 결과물이 dist 폴더 안에 생성되며, 터미널에는 아래와 같은 문구가 출력된다.



배포 명령어 정리

도커 설치

```
// curl 툴 없을 시 curl 툴 먼저 설치
sudo apt-get install curl
curl -fsSL https://get.docker.com. | sudo sh
```

도커 이미지 관련 명령어

```
이미지 검색
docker search 이미지

이미지 다운로드
docker pull 이미지
docker pull 이미지

이미지 확인
docker images

이미지 아이디만 검색
docker image ls -q

이미지 삭제
docker rmi 이미지
```

도커 컨테이너 관련 명령어

```
컨테이너 생성
docker create 이미지

실행중인 컨테이너
docker ps

실행중이지 않은것도 포함
docker ps -a

컨테이너 삭제
docker rm 컨테이너아이디

컨테이너 실행
docker start 컨테이너 이름

만드는 동시에 실행
docker run

컨테이너 종료
docker stop myubuntu

일시 중지
docker pause
```

Dockerfile 작성 명령어

- 명령 + 인수로 구성 (명령은 통상적으로 대문자로 표시)
- FROM : 베이스 이미지 설정
- LABEL : key+ value 로 간단한 정보 저장 (이미지에 영향 X)

- CMD : 컨테이너가 시작할 때 실행하는 명령어
- RUN : 이미지 작성시 실행하는 명령어
- ENTRYPOINT : docker 컨테이너가 시작할 때 실행하는 명령을 지정하는 명령
- EXPOSE : 컨테이너 외부에 오픈할 포트 설정
- ENV : 컨테이너 내부에서 사용할 환경 변수 지정
- WORKDIR : 컨테이너에서의 작업 디렉토리 설정
- COPY : 파일 또는 디렉토리를 container 에 복사

doker-compose

```
#Docker Compose 파일 버전 지정

version:

# 하나 또는 여러개의 컨테이너 설정
service:

# 컨테이너에서 사용하는 volumes
volumes:

# 컨테이너간 네트워크 분리를 위한 설정
networks:

실행 명령

# 백그라운드에서 실행
docker-compose up -d

# 이미지 재빌드가 필요하면 --build옵션을 추가
docker-compose up --build -d

service 작성
services:
  db:                       # 컨테이너의 이름 설정
    image:                  # 사용하는 이미지
    restart:                # 컨테이너 재시작
    volumes:                # volume 설정
    environment:            # Dockerfile의 ENV 옵션과 동일
```

배포 관련 파일

- docker-compose.yml

```
version: "3"

services:
  webserver:
    image: nginx:latest
    container_name: proxy
    restart: always
    ports:
      - "80:80"
      - "443:443"
    volumes:
      - ./dist:/usr/share/nginx/html
      - ./nginx/nginx.conf:/etc/nginx/nginx.conf
      - ./certbot-etc:/etc/letsencrypt
  nginx:
    image: nginx:latest
    container_name: front
    restart: always
    volumes:
      - ./dist:/usr/share/nginx/html
      - ./default.conf:/etc/nginx/conf.d/default.conf

  certbot:
    depends_on:
      - webserver
    image: certbot/certbot
    container_name: certbot
    volumes:
      - ./certbot-etc:/etc/letsencrypt
      - ./dist:/usr/share/nginx/html
    command: certonly --webroot --webroot-path=/usr/share/nginx/html --email test@test.com --agree-tos --no-eff-email --keep-until-exp

  db:
    image: mysql:5.7
    restart: always
    container_name: db
    volumes:
      - ./mysqldata:/var/lib/mysql
    environment:
      - MYSQL_ROOT_PASSWORD=root
      - MYSQL_DATABASE=colors

  back:
    container_name: back
    build:
      context: ./back
      dockerfile: Dockerfile
    links:
      - "db:mysqldb"
```

- default.conf

```
server {
    listen       80;
    listen  [::]:80;
    server_name  localhost;

    #access_log  /var/log/nginx/host.access.log  main;

    location / {
        root   /usr/share/nginx/html;
        index  index.html index.htm;
  try_files $uri $uri/ /index.html $uri.html = 404;
    }

    #error_page  404              /404.html;

    # redirect server error pages to the static page /50x.html
    #
    error_page   500 502 503 504  /50x.html;
    location = /50x.html {
```

```
        root   /usr/share/nginx/html;
    }

    # proxy the PHP scripts to Apache listening on 127.0.0.1:80
    #
    #location ~ \.php$ {
    #    proxy_pass   http://127.0.0.1;
    #}

    # pass the PHP scripts to FastCGI server listening on 127.0.0.1:9000
    #
    #location ~ \.php$ {
    #    root           html;
    #    fastcgi_pass   127.0.0.1:9000;
    #    fastcgi_index  index.php;
    #    fastcgi_param  SCRIPT_FILENAME  /scripts$fastcgi_script_name;
    #    include        fastcgi_params;
    #}

    # deny access to .htaccess files, if Apache's document root
    # concurs with nginx's one
    #
    #location ~ /\.ht {
    #    deny  all;
    #}
}
```

- Dockerfile

```
FROM openjdk:8-jdk-alpine
ARG JAR_FILE=./*.jar
COPY ${JAR_FILE} app.jar
ENTRYPOINT ["java", "-jar", "/app.jar"]
EXPOSE 8080
```

Nginx

```
user nginx;
worker_processes  auto;

error_log  /var/log/nginx/error.log warn;
pid        /var/run/nginx.pid;

events {
    worker_connections 1024;
}

http {
    include       /etc/nginx/mime.types;
    default_type  application/octet-stream;
    log_format  main  '$remote_addr - $remote_user [$time_local] "$request" '
                      '$status $body_bytes_sent "$http_referer" "$request_uri" "$uri"'
                      '"$http_user_agent" "$http_x_forwarded_for"';
    access_log  /var/log/nginx/access.log  main;
    sendfile on;
    keepalive_timeout 65;

    upstream docker-web {
        server nginx:80;
    }

    upstream docker-back {
        server back:8080;
    }

    server {
        listen 80;
        server_name i7b208.p.ssafy.io;

        location ~ /.well-known/acme-challenge {
                allow all;
                root /usr/share/nginx/html;
                try_files $uri = 404;
        }

        location / {
                return 301 https://$host$request_uri;
        }
    }

    server {
        listen 443 ssl;
```

```
        server_name i7b208.p.ssafy.io;

        ssl_certificate /etc/letsencrypt/live/i7b208.p.ssafy.io/fullchain.pem;
        ssl_certificate_key /etc/letsencrypt/live/i7b208.p.ssafy.io/privkey.pem;
        include /etc/letsencrypt/options-ssl-nginx.conf;
        ssl_dhparam /etc/letsencrypt/ssl-dhparams.pem;

       location / {
            proxy_pass         http://docker-web; # docker-web 컨테이너로 포워딩
            proxy_redirect     off;
            proxy_set_header   Host $host;
            proxy_set_header   X-Real-IP $remote_addr;
            proxy_set_header   X-Forwarded-For $proxy_add_x_forwarded_for;
            proxy_set_header   X-Forwarded-Host $server_name;
            proxy_set_header   X-Forwarded-Proto $scheme;
        }

        location /api/ {
            proxy_pass         http://docker-back;    # docker-back 컨테이너로 포워딩
            proxy_redirect     off;
            proxy_set_header   Host $host;
            proxy_set_header   X-Real-IP $remote_addr;
            proxy_set_header   X-Forwarded-For $proxy_add_x_forwarded_for;
            proxy_set_header   X-Forwarded-Host $server_name;
            proxy_set_header   X-Forwarded-Proto $scheme;
        }
    }
}
```

AWS S3 Bucket

Bucket

- Name : ssafy7color
- region : “ap-northeast-2”
- IdentityPoolId : “”

Bucket Policy

```
{
  "Version": "2008-10-17",
  "Id": "Policy1335892530063",
  "Statement": [
    {
      "Sid": "Stmt1335892150622",
      "Effect": "Allow",
      "Principal": {
        "Service": "billingreports.amazonaws.com"
      },
      "Action": [
        "s3:GetBucketAcl",
        "s3:GetBucketPolicy"
      ],
      "Resource": "arn:aws:s3:::ssafy7color",
      "Condition": {
        "StringEquals": {
          "aws:SourceAccount": "873095655836",
          "aws:SourceArn": "arn:aws:cur:us-east-1:dummy:ArnCode/*"
        }
      }
    },
    {
      "Sid": "Stmt1335892526596",
      "Effect": "Allow",
      "Principal": {
        "Service": "billingreports.amazonaws.com"
      },
      "Action": "s3:PutObject",
      "Resource": "arn:aws:s3:::ssafy7color/",
      "Condition": {
        "StringEquals": {
          "aws:SourceAccount": "873095655836",
          "aws:SourceArn": "arn:aws:cur:us-east-1:dummy:ArnCode/*"
        }
```

```
      }
    }
  ]
}
```

ACL(액세스 제어 목록)

CORS(Cross-origin 리소스 공유)

```
[
    {
        "AllowedHeaders": [
            "*"
```

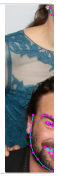
```
        ],
        "AllowedMethods": [
            "HEAD",
            "GET",
            "PUT",
            "POST",
            "DELETE"
        ],
        "AllowedOrigins": [
            "*"
        ],
        "ExposeHeaders": [
            "ETag"
        ]
    }
]
```

face-api

GitHub - justadudewhohacks/face-api.js: JavaScript API for face detection and face recognition in the browser and nodejs with tensorflow.js

JavaScript face recognition API for the browser and nodejs implemented on top of tensorflow.js core (tensorflow/tfjs-core) Clone the repository: git clone https://github.com/justadudewhohacks/face-api.js.git cd face-api.js/examples/examples-browser npm i npm start Browse to http://localhost:3000/. cd face-api.js/examples/examples-nodejs npm i Now run one of the examples using ts-node: Or simply compile and run them with node: tsc faceDetection.ts node

https://github.com/justadudewhohacks/face-api.js



https://www.notion.so/face-api-43138615e7274185a94893c97fe2553f#57ef64e3547c417a89cca3dbd1bc76bd

개요

- Javascript 기반의 tensorflow-core를 활용하여 안면인식을 수행하는 외부 라이브러리
- 프로젝트에 직접 tensorflow.js를 추가할 필요가 있음
- 엔진을 가져 옴으로 인해서 원하는 모델을 학습시키거나 주어진 모델을 추가로 학습하는 것이 가능
- 기본적으로 일정부분 학습된 모델을 제공 함
- 이번 프로젝트에서는 제공된 **Tiny Face Detector** 모델을 사용하여 안면 인식을 수행함
- 해당 모델은 비교적 작고 가벼운 모델로써 웹상에서 사용하기 좋음

설치방법

- 위 라이브러리는 npm 명령어를 지원하는 라이브러리로 쉽게 설치가 가능함

```
###git clone https://github.com/justadudewhohacks/face-api.js.git
cd face-api.js/examples/examples-browser
npm i
npm start
```

- 비교적 간단하게 설치가 가능하지만 의존성 출동이 일어날 수 있다.
- 경험에 의하면 npm cache를 삭제한후 —legacy-peer-deps 옵션을 주어 설치하면 해결 가능하다

```
npm cache clean
npm install --legacy-peer-deps
```

- face-api를 사용하기 위해서는 미리 학습된 모델의 출력 파일을 가지고 있어야 하며 해당 파일을 라이브러리와 연결하여 주는 과정이 필수적이다.

사용 방법

- 아래와 같이 모델의 상대적인 경로를 지정하여 주어 학습된 모델을 불러 올 수 있다.

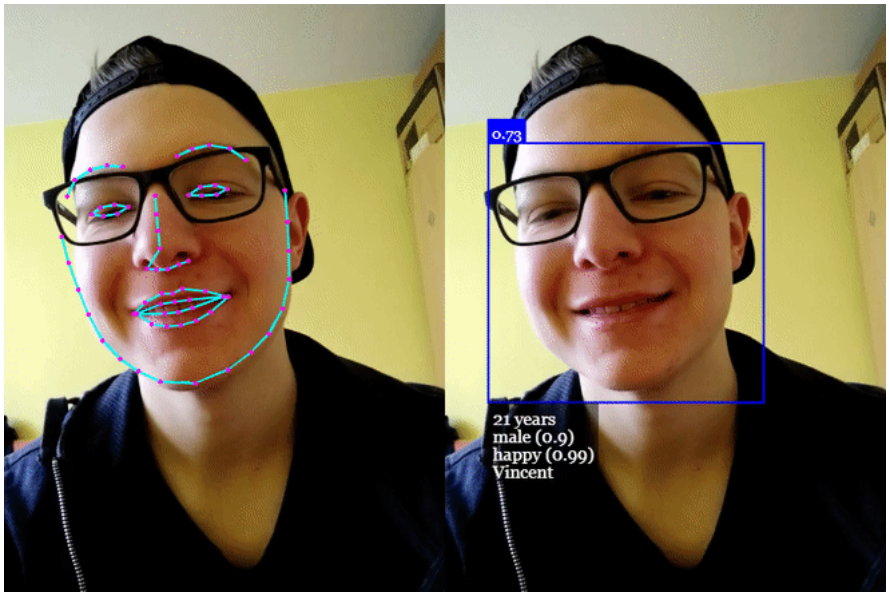
```
const net = new faceapi.SsdMobilenetv1()
await net.loadFromUri('/models')
```

- 프로젝트에서 단일 대상에게 화면을 제공하는 것을 목표로 하므로 아래와 같이 안면을 인식함

```
const detection = await faceapi.detectSingleFace(input)
```

※ 이외에도 당양한 메소드와 모델을 제공하므로 필요에 맞추어 사용 가능하다.

예시

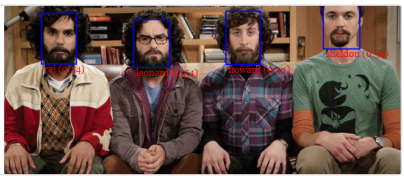


- 얼굴 인식 및 윤곽선 인식
- 이외에도 다양한 사진 및 웹 캠을 활용한 예제가 존재함

- 다양한 모델과 다양한 메소드를 경험할 수 있는 사이트

face-api.js

https://justadudewhohacks.github.io/face-api.js/face_and_landmark_detection



https://www.notion.so/face-api-43138615e7274185a94893c97fe2553f#e9b5c81c6cc7491aa0e86d510a446c5c