

서울 2반 3팀 A203 공통 프로젝트 (SSAMART CART PJT)  
포팅 매뉴얼

팀장 : 김태환

팀원 : 박중현, 노현진, 박광우, 정원영, 조성호

목차

1. 사용 도구
2. 개발 도구
3. 개발 환경
4. 환경변수 형태
5. 서버 설정
6. 빌드 및 실행
7. **사용 도구**

* 이슈 관리 : Jira
* 형상 관리 : GitLab
* 커뮤니케이션 : Notion, MatterMost
* 디자인 : Figma
* UCC : Windows XBOX Game bar

1. **개발 도구**

* Visual Studio Code : 1.85.1
* Intellij : 2024.3.2 (Ultimate Edition)
* Datagrip : 2024.3.2
* MySQL Workbench : 8.0.36

1. **개발 환경**

* FrontEnd
  + Vue.js : 3.3.11
  + Node.js : 20.11.0
  + Axios : 1.6.7
  + Pinia : 2.1.7
  + Pinia-plugin-persistedstate : 3.2.1
* BackEnd
  + JVM : 17
  + 플러그인 버전
    - Spring Boot: 3.2.1
    - Spring Dependency Management: 1.1.4
  + 의존성 버전
    - Spring Boot Starter Data JPA: 버전이 명시되지 않았으므로 Spring Boot 플러그인의 버전인 3.2.1에 맞게 의존성 버전이 관리될 것입니다.
    - Spring Boot Starter Security: 동일하게 Spring Boot 플러그인의 버전에 따라 관리됩니다.
    - ModelMapper: 3.1.1
    - JSON Simple: 1.1.1
    - JJWT API: 0.11.5
    - JJWT Impl: 0.11.5
    - JJWT Jackson: 0.11.5
    - Swagger Annotations: 2.2.20
    - Springdoc OpenAPI Starter Web MVC UI: 2.0.2
    - Spring Cloud Starter AWS: 2.2.6.RELEASE
    - JAXB API: 2.3.1
    - MariaDB Java Client: 3.0.7

1. **환경변수 형태**
2. **데이터베이스 설정**

* DB : MySQL Community Server : 8.0.36
  + Driver Class Name: com.mysql.cj.jdbc.Driver
  + DataSourceURL:

jdbc:mysql://stg-yswa-kr-practice-db-master.mariadb.database.azure.com:3306/S10P12A203?serverTimezone=UTC&useUnicode=true&characterEncoding=utf8

* + Username: S10P12A203@stg-yswa-kr-practice-db-master
  + password: a20312
  + **JPA & Hibernate 설정:**
    - spring.jpa.show-sql=true
    - spring.jpa.hibernate.ddl-auto=update
    - spring.jpa.properties.hibernate.format\_sql=true
    - spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.MySQLDialect

**2. JWT 설정**

jwt.secret.key= 7Iqk7YyM66W07YOA7L2U65Sp7YG065+9U3ByaW5n6rCV7J2Y7Yqc7YSw7LWc7JuQ67mI7J6F64uI64ukLg==

jwt.expiration\_time=864000000

**3. AWS S3 설정**

cloud.aws.credentials.accessKey=AKIA6PPFBKVH6ABYXZJN

cloud.aws.credentials.secretKey=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

cloud.aws.s3.bucketName=ssamart-productimg-ec2-a20312

cloud.aws.region.static=ap-northeast-2

cloud.aws.stack.auto-=false

**4. 기타 설정**

gpu.ip=70.12.130.101

gpu.port=41234

1. **서버 설정**
2. **Nginx 설정**

server {

listen 80 default\_server;

listen [::]:80 default\_server;

root /var/www/html;

# Add index.php to the list if you are using PHP

index index.html index.htm index.nginx-debian.html;

server\_name \_;

location / {

# First attempt to serve request as file, then

# as directory, then fall back to displaying a 404.

try\_files $uri $uri/ =404;

}

}

server {

root /var/www/html;

# Add index.php to the list if you are using PHP

index index.html index.htm index.nginx-debian.html;

server\_name i10a203.p.ssafy.io; # managed by Certbot

location /api/ {

rewrite ^/api/(.\*)$ /$1 break;

proxy\_pass http://localhost:8080;

proxy\_set\_header X-Real-IP $remote\_addr;

proxy\_set\_header X-Forwarded-For $proxy\_add\_x\_forwarded\_for;

proxy\_set\_header Host $http\_host;

}

location / {

proxy\_pass http://localhost:3000; # Node.js 앱으로 요청을 전달

add\_header Cache-Control no-cache;

proxy\_http\_version 1.1;

proxy\_set\_header Upgrade $http\_upgrade;

proxy\_set\_header Connection 'upgrade';

proxy\_set\_header Host $host;

proxy\_cache\_bypass $http\_upgrade;

}

listen [::]:443 ssl ipv6only=on; # managed by Certbot

listen 443 ssl; # managed by Certbot

ssl\_certificate /etc/letsencrypt/live/i10a203.p.ssafy.io/fullchain.pem; # managed by Certbot

ssl\_certificate\_key /etc/letsencrypt/live/i10a203.p.ssafy.io/privkey.pem; # managed by Certbot

include /etc/letsencrypt/options-ssl-nginx.conf; # managed by Certbot

ssl\_dhparam /etc/letsencrypt/ssl-dhparams.pem; # managed by Certbot

}

server {

if ($host = i10a203.p.ssafy.io) {

return 301 https://$host$request\_uri;

} # managed by Certbot

listen 80 ;

listen [::]:80 ;

server\_name i10a203.p.ssafy.io;

return 404; # managed by Certbot

}

1. **빌드 및 실행**
2. **Docker 설정**

**FrontEnd Docker**

1. Dockerfile 설정

FROM node:lts-alpine AS builder

# install simple http server for serving static content

RUN npm install -g http-server

# make the 'app' folder the current working directory

# WORKDIR ./

WORKDIR /app

# copy both 'package.json' and 'package-lock.json' (if available)

COPY package\*.json ./

# install project dependencies

RUN npm install

# copy project files and folders to the current working directory(i.e. 'app'folder)

COPY .

# build app for production with minification

RUN npm run build

# CMD [ "http-server", "dist", "-p", "8901" ]

FROM nginx:alpine

COPY nginx.conf /etc/nginx/conf.d/default.conf

COPY --from=builder /app/dist /usr/share/nginx/html

EXPOSE 3000

CMD ["nginx", "-g", "daemon off;"]

1. Nginx.conf 설정

server {

    listen 3000;

    location / {

        root /usr/share/nginx/html;

        index index.html;

        try\_files $uri $uri/ /index.html;

    }

}

**BackEnd Docker 설정**

1. Dockerfile 설정

# Dockerfile  
  
# jdk17 Image Start  
FROM openjdk:17  
  
# 인자 설정 - JAR\_File  
ARG JAR\_FILE=build/libs/\*.jar  
  
# jar 파일 복제  
COPY ${JAR\_FILE} app.jar  
  
# 인자 설정 부분과 jar 파일 복제 부분 합쳐서 진행해도 무방  
#COPY build/libs/\*.jar app.jar  
  
# 실행 명령어  
ENTRYPOINT ["java", "-jar", "app.jar"]

1. BackEnd 설정-Intellij 실행 후 Open -> SpringBoot 폴더로 이동 -> build.gradle 파일 열기  
   - Gradle 파일 설치 완료될 때까지 잠시 기다린 후 Project Structure 들어가서 JDK와 SDK 17로 모두 수정
   * 상단의 SmartcartApplication 설정에서 똑같이 JDK, SDK 17로 설정
   * src/resources/application.properties 파일 들어가서 DB 설정 확인
   * 상단의 SmartcartApplication Run 클릭해서 Springboot 실행
2. FrontEnd 설정

* Visual Studio Code 에서 파일이 담긴 폴더 -> Open Project
* src/util/http-common.js 파일 들어가서 baseURL 이 일치하는지 확인
* node\_modules 폴더가 존재하지 않으면 powershell로 npm install 로 실행 파일 설치
* npm run dev로 프로젝트 실행