포팅메뉴얼

개발 환경

- Backend
  - IDE : Intellij 2022.02.01 (Ultimate), MobaXterm
  - SpringBoot : 3.0.12
  - Java 17
  - MySQL
  - Redis
  - Docker
  - Jenkins

▼ Dockerfile

```
FROM openjdk:17-jdk-slim as builder

COPY gradlew .
COPY gradle gradle
COPY build.gradle .
COPY settings.gradle .
COPY src src

RUN chmod +x ./gradlew
RUN ./gradlew --stacktrace bootJar

FROM openjdk:17-jdk-slim
COPY --from=builder build/libs/*.jar app.jar
EXPOSE 8080

ARG SERVER_MODE
RUN echo "$SERVER_MODE"
ENV SERVER_MODE=$SERVER_MODE


ENTRYPOINT ["java", "-Dspring.profiles.active=${SERVER_MODE}","-Duser.timezone=Asia/Seoul", "-jar", "/app.jar"]
```

▼ application.yml

1. MESC

```
spring:
  datasource:
    driver-class-name: com.mysql.cj.jdbc.Driver
    url: jdbc:mysql://k9b201.p.ssafy.io:3306/mesc?useSSL=false&serverTimezone=Asia/Seoul&useUnicode=yes&characterEncoding=UTF-8
    username: ksol
    password: ksol1117

  jpa:
    hibernate:
      ddl-auto: update
    show-sql: false
    database: mysql
    properties:
      hibernate:
        format_sql: true
        dialect: org.hibernate.dialect.MySQL8Dialect
        discriminator:
          ignore_explicit_for_joined: true
    .open-in-view: false

  # redis 설정
  data:
    redis:
      host: localhost
      port: 6379

  # 이메일
  mail:
```

```
    host: smtp.gmail.com
    port: 587
    username: B201MESC
    password: szwj gsio ejxo zxqp
    properties:
      mail:
        smtp:
          auth: true
          timeout: 5000
          starttls:
            enable: true
  thymeleaf:
    cache: false

  servlet:
    multipart:
      enabled: true
      max-file-size: 50MB
      max-request-size: 50MB

# jwt secret key 설정
jwt:
  key: YourewaitingforatrainAtrainthatwilltakeyoufarawayYouknowwhereyouhope

server:
  port: 8080
  ec2-url: https://www.mesc.kr
```

2. MES

```
spring:
  # DB 설정
  datasource:
    driver-class-name: com.mysql.cj.jdbc.Driver
#    url: jdbc:mysql://127.0.0.1:3306/mes?useSSL=false&serverTimezone=Asia/Seoul&useUnicode=yes&characterEncoding=UTF-8&allowPu
    url: jdbc:mysql://k9b201a.p.ssafy.io:3306/mes?useSSL=false&serverTimezone=Asia/Seoul&useUnicode=yes&characterEncoding=UTF-8
    username: ksol
    password: ksol1117
#    url: jdbc:mysql://localhost:3306/mes?useSSL=false&serverTimezone=Asia/Seoul&useUnicode=yes&characterEncoding=UTF-8&allowPu
#    username: root
#    password: root

  jpa:
    hibernate:
      ddl-auto: update
    show-sql: true
    database: mysql
    properties:
      hibernate:
        format_sql: true
        dialect: org.hibernate.dialect.MySQL8Dialect
        discriminator:
          ignore_explicit_for_joined: true
    .open-in-view: false

  # redis 설정
  data:
    redis:
      host: localhost
      port: 6379

  # 이메일
  #  mail:
  #    host: smtp.gmail.com
  #    port: 587
  #    username: B201MESC
  #    password: szwj gsio ejxo zxqp
  #    properties:
  #      mail:
  #        smtp:
  #          auth: true
  #          timeout: 5000
  #          starttls:
  #            enable: true
  #  thymeleaf:
  #    cache: false

  #  firebase:
  #    serviceAccountPath: path/to/serviceAccountKey.json
  #  security:
  #    user:
```

```
  #      name: admin
  #      password: admin

#  servlet:
#    multipart:
#      enabled: true
#      max-file-size: 50MB
#      max-request-size: 50MB

# JPA log
logging:
  level:
    com.ksol.mes: debug
    org:
      hibernate:
        SQL: DEBUG
        type:
          descriptor:
            sql:
              BasicBinder: TRACE

# jwt secret key 설정
jwt:
  key: YourewaitingforatrainAtrainthatwilltakeyoufarawayYouknowwhereyouhope

server:
  port: 8081
  ec2-url: https://
```

▼ 빌드 스크립트

- **start-chatbot-be.sh**

```
sed -i 's/localhost:3306/mysql:3306/g' ./BE/mesc/src/main/resources/application.yml
sed -i 's/host: localhost/host: local-redis/g' ./BE/mesc/src/main/resources/application.yml
sed -i 's/http:\/\/localhost:8081\/mes/https:\/\/www.mescadmin.kr\/api\/mes/g' ./BE/mesc/src/main/java/com/ksol/mesc/global/cor
rm -rf ./BE/mesc/src/main/generated

docker-compose -f docker-compose-chatbot-be.yml pull //현재 프로젝트에있는 docker-compose

COMPOSE_DOCKER_CLI_BUILD=1 DOCKER_BUILDKIT=1 docker-compose -f docker-compose-chatbot-be.yml up --build -d

docker rmi -f $(docker images -f "dangling=true" -q) || true
```

- **docker-compose-chatbot-be.yml**

```
version: "3" #Compose 파일의 버전
services: # 서비스 정의를 시작합니다. 각 서비스는 별도의 컨테이너로 실행
  server:
    image: chatbot_be:latest # 이미지 이름
    container_name: chatbot_be # 컨테이너 이름
    build:
      context: ./BE/mesc # 컨테이너와 호스트 간의 포트 매핑을 설정합니다.

      args:
        SERVER_MODE: prod
    ports:
      - 8080:8080
    environment:
      - TZ=Asia/Seoul
    networks:
      - default

networks:
  default:
    external:
      name: chatbot_net
```

- **Jenkinsfile-chatbot-be**

```
pipeline {
    agent any
        stages {
            stage('Prepare') {
                steps {
                    sh 'echo "Clonning Repository"'
                    git branch: 'develop-BE',
```

```
                        url: 'https://lab.ssafy.com/s09-final/S09P31S105.git',
                        credentialsId: 'ldy'
                }
                post {
                    success {
                        sh 'echo "Successfully Cloned Repository"'
                    }
                    failure {
                        sh 'echo "Fail Cloned Repository"'
                    }
                }
            }
            stage('Docker stop'){
                steps {
                        sh 'echo "Docker Container Stop"'
                        sh '''
                        result=$( docker container ls -a --filter "name=chatbot_be*" -q )
                        if [ -n "$result" ]
                        then
                            docker stop $(docker container ls -a --filter "name=chatbot_be*" -q)
                        else
                            echo "No stop containers"
                        fi
                        '''
      sh 'docker-compose -f docker-compose-chatbot-be.yml down'
                }
                post {
                    failure {
                        sh 'echo "Docker Fail"'
                    }
                }
            }

stage('RM Docker') {
    steps {
        sh 'echo "Remove Docker"'

        // 정지된 도커 컨테이너 찾아서 컨테이너 ID로 삭제함
        sh '''
            result=$( docker container ls -a --filter "name=chatbot_be*" -q )
            if [ -n "$result" ]
            then
                docker rm $(docker container ls -a --filter "name=chatbot_be*" -q)
            else
                echo "No such containers"
            fi
        '''

        // homesketcher로 시작하는 이미지 찾아서 삭제함
        sh '''
            result=$( docker images -f "reference=chatbot_be*" -q )
            if [ -n "$result" ]
            then
                docker rmi -f $(docker images -f "reference=chatbot_be*" -q)
            else
                echo "No such container images"
            fi
        '''

        // 안쓰는이미지 -> <none> 태그 이미지 찾아서 삭제함
        sh '''
            result=$(docker images -f "dangling=true" -q)
            if [ -n "$result" ]
            then
                docker rmi -f $(docker images -f "dangling=true" -q)
            else
                echo "No such container images"
            fi
        '''
    }
    post {
        failure {
            sh 'echo "Remove Fail"'
        }
    }
}
            stage('Set Permissions') {
                        steps {
                            // 스크립트 파일에 실행 권한 추가
                            sh 'chmod +x start-chatbot-be.sh'
                        }
                    }
            stage('Execute start-prod.sh Script') {
                steps {
```

```
                    // start-mes-be.sh 스크립트 실행
                    sh 'sh start-chatbot-be.sh'
                }
            }
        }
    }
```

DB 접속 프로퍼티

- mysql

```
url: jdbc:mysql://k9b201.p.ssafy.io:3306/mesc?useSSL=false&serverTimezone=Asia/Seoul&useUnicode=yes&characterEncoding=UTF-8&allowPublic
    username: ksol
    password: ksol1117
```

NGINX 설정

- default.conf

```
server {
    listen  80;
    server_name mesc.kr www.mesc.kr;
    return 301 https://www.mesc.kr$request_uri;
}

server {
    listen 443 ssl;
    ssl on;
    server_name mesc.kr www.mesc.kr;

    ssl_certificate /etc/letsencrypt/live/mesc.kr/fullchain.pem;
    ssl_certificate_key /etc/letsencrypt/live/mesc.kr/privkey.pem;
    access_log   /var/log/nginx/nginx.vhost.access.log;
    error_log    /var/log/nginx/nginx.vhost.error.log;

    location / {
        proxy_pass http://localhost:3000/;
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
    }


    location /api/ {
        proxy_pass http://localhost:8080/;
        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-Original-URI $request_uri;
    }

    location /v3 {
        proxy_pass https://localhost:8080/v3;
    }
    location ~ ^/(swagger|webjars|configuration|swagger-resources|v2|csrf) {
                proxy_pass http://localhost:8080;
                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-Proto $scheme;
```