포팅매뉴얼

버전 명세서

허브 버전 명세서

프로젝트 정보

- **Group:** `org.binddog`
- **Version:** `0.0.1-SNAPSHOT`
- **Java Version:** `17` (Toolchain 사용)

플러그인

1. `java`
2. `org.springframework.boot` - **Version:** `3.3.5`
3. `io.spring.dependency-management` - **Version:** `1.1.6`

Dependencies

Spring Boot

- **spring-boot-starter-web**: Spring Boot 기반의 웹 애플리케이션 개발 지원
- **spring-boot-starter-data-jpa**: JPA 사용을 위한 의존성
- **spring-boot-starter-data-mongodb**: MongoDB 사용을 위한 의존성
- **spring-boot-starter-data-redis**: Redis 사용을 위한 의존성
- **spring-boot-starter-validation**: 데이터 유효성 검증 지원
- **spring-boot-starter-security**: Spring Security 기반 인증 및 권한 관리
- **spring-boot-starter-test**: Spring Boot 테스트 의존성
- **spring-boot-devtools**: 개발 편의성을 위한 도구 (Development Only)

QueryDSL

- **querydsl-jpa**: **Version:** `5.0.0:jakarta`
- **querydsl-apt**: **Version:** `5.0.0:jakarta` (Annotation Processor)
- **jakarta.annotation-api**
- **jakarta.persistence-api**

MapStruct

- **mapstruct**: **Version:** `1.4.2.Final`

- **mapstruct-processor**: **Version:** `1.4.2.Final` (Annotation Processor)

JWT (JSON Web Token)

- **jjwt**: **Version:** `0.12.6`
- **jjwt-api**: **Version:** `0.12.6`
- **jjwt-jackson**: **Version:** `0.12.6` (Runtime Only)

Swagger/OpenAPI

- **springdoc-openapi-starter-webmvc-ui**: **Version:** `2.0.4`

MinIO

- **minio**: **Version:** `8.5.4`
- **javax.annotation-api**: **Version:** `1.3.2`

Lombok

- **lombok** (Compile Only & Annotation Processor)

Database

- **mysql-connector-j**: MySQL 데이터베이스 연동 (Runtime Only)

Test

- **spring-security-test**: Spring Security 테스트 지원
- **junit-platform-launcher**: JUnit 플랫폼 런처 (Runtime Only)

라이브러리 버전 명세서

프로젝트 정보

- **Group:** `org.binddog`
- **Artifact:** `binddog`
- **Version:** `0.0.1`
- **Java Version:** `17` (Toolchain 사용)

플러그인

1. `java-library`
2. `maven-publish`
3. `org.springframework.boot` - **Version:** `3.3.5` (apply false)
4. `io.spring.dependency-management` - **Version:** `1.1.6` (apply false)
5. `com.vanniktech.maven.publish` - **Version:** `0.28.0`
6. `signing`

Dependencies

Spring Boot

- **spring-boot-starter-web**: **Version:** `3.3.5`
- **spring-boot-starter-test**: **Version:** `3.3.5`

Swagger/OpenAPI

- **springdoc-openapi-starter-webmvc-ui**: **Version:** `2.6.0`

Test

- **junit-platform-launcher**

Maven Publishing

- **Publishing Host:** Maven Central (via Sonatype Central Portal)
- **Coordinates:**
  - Group: `org.binddog`
  - Artifact: `binddog`
  - Version: `0.0.1`

POM 정보

- **Name:** `org.binddog`
- **Description:** `Binddog library`
- **Project URL:** `<https://binddog.org>`

라이선스

- **Name:** MIT License
- **URL:** `<https://lab.ssafy.com/s11-final/S11P31A401/-/blob/master/LICENSE>`

개발자 정보

- **ID:** `wnso-kim`
- **Name:** `wnso`
- **Email:** `wnso.kim@gmail.com`

SCM 정보

- **Connection:** `scm:git:lab.ssafy.com/s11-final/S11P31A401.git`
- **Developer Connection:** `scm:git:ssh://lab.ssafy.com/s11-final/S11P31A401.git`
- **SCM URL:** `<https://lab.ssafy.com/s11-final/S11P31A401/-/tree/master>`

환경변수

허브

```
spring:
  application:
    name: binddog-hub
  datasource:
    url: ${DB_URL} # DB 접근 url
    username: ${DB_USER} # DB 유저
    password: ${DB_PASSWORD} # DB 비밀번호
    driver-class-name: com.mysql.cj.jdbc.Driver
  jpa:
    properties:
      hibernate:
        dialect: org.hibernate.dialect.MySQLDialect
    hibernate:
      ddl-auto: update
    show-sql: false
  data:
    mongodb:
```

```
      host: ${MONGODB_HOST} # 몽고디비 실행 url
      username: ${MONGODB_USER} # 몽고디비 유저
      password: ${MONGODB_PASSWORD} # 몽고디비 비밀번호
      database: ${MONGODB_DB} # 몽고디비 DB
      authentication-database: ${MONGODB_AUTH_DB} # 몽고디비 접
      port: ${MONGODB_PORT} # 몽고디비 포트
    redis:
      host: ${REDIS_HOST} # 레디스 실행 url
      port: ${REDIS_PORT} # 레디스 포트 번호
      password: ${REDIS_PASSWORD} #  레디스 접속 비밀번호
  servlet:
    multipart:
      max-file-size: 100MB
      max-request-size: 100MB

server:
  port: ${HUB_PORT} # 애플리케이션 실행 port 번호
  servlet:
    context-path: /api
  tomcat:
    max-http-form-post-size: 100MB

springdoc:
  api-docs:
    path: /v3/api-docs
  swagger-ui:
    path: /swagger-ui.html

jwt:
  secret-key: ${JWT_SECRET_KEY} # jwt 시크릿 키

minio:
  endpoint: ${MINIO_ENDPOINT}       # MinIO 서버 엔드포인트
  access-key: ${MINIO_ACCESS_KEY}   # MinIO 액세스 키
  secret-key: ${MINIO_SECRET_KEY}   # MinIO 비밀 키
  bucket-name: ${MINIO_BUCKET_NAME} # 사용할 버킷 이름
  presigned-url : ${MINIO_PRESIGNED_URL} # 이미지 조회 시 생성된
```

빌드 방법

docker-compose.yml

```
services:
  hub:
    container_name: hub
    build:
      context: ./backend/binddog-hub
    ports:
      - "8080:8080"
    depends_on:
      hub-db:
        condition: service_started
    networks:
      - bindDog-net
    environment:
      - DB_URL=jdbc:mysql://hub-db:3306/binder?serverTimezone=
      - DB_USER=<masked_user>
      - DB_PASSWORD=<masked_password>
      - HUB_PORT=8080
      - MONGODB_HOST=hub-mongo-db
      - MONGODB_USER=<masked_user>
      - MONGODB_PASSWORD=<masked_password>
      - MONGODB_DB=project
      - MONGODB_AUTH_DB=admin
      - MONGODB_PORT=27017
      - REDIS_HOST=hub-redis
      - REDIS_PORT=6379
      - REDIS_PASSWORD=<masked_password>
      - MINIO_ENDPOINT=http://minio:9000
      - MINIO_ACCESS_KEY=<masked_key>
      - MINIO_SECRET_KEY=<masked_key>
      - MINIO_BUCKET_NAME=binder
      - MINIO_PRESIGNED_URL=http://k11a401.p.ssafy.io:9000
      - JWT_SECRET_KEY=<masked_jwt_secret>
    healthcheck:
      test: ["CMD-SHELL", "curl -f http://localhost:8080/actu
```

```
      interval: 10s
      timeout: 3s
      retries: 3

  hub-db:
    image: mysql:8.0.40
    container_name: hub-db
    environment:
      MYSQL_ROOT_PASSWORD=<masked_password>
      MYSQL_DATABASE=binder
      MYSQL_USER=<masked_user>
      MYSQL_PASSWORD=<masked_password>
      TZ: Asia/Seoul
    ports:
      - "3306:3306"
    volumes:
      - hub_mysql_data:/var/lib/mysql
    networks:
      - bindDog-net
    restart: unless-stopped

  hub-mongo-db:
    image: mongo:latest
    container_name: hub-mongo-db
    environment:
      MONGO_INITDB_ROOT_USERNAME=<masked_user>
      MONGO_INITDB_ROOT_PASSWORD=<masked_password>
    ports:
      - "27017:27017"
    networks:
      - bindDog-net
    volumes:
      - mongo_data:/data/db

  hub-redis:
    image: redis:latest
    container_name: hub-redis
    command: redis-server --requirepass <masked_password>
```

```
    ports:
      - "6379:6379"
    networks:
      - bindDog-net
    volumes:
      - redis_data:/data

  hub-mongo-express:
    image: mongo-express:latest
    container_name: hub-mongo-express
    restart: always
    environment:
      ME_CONFIG_MONGODB_SERVER: hub-mongo-db
      ME_CONFIG_MONGODB_ADMINUSERNAME: <masked_user>
      ME_CONFIG_MONGODB_ADMINPASSWORD: <masked_password>
      ME_CONFIG_MONGODB_PORT: 27017
      ME_CONFIG_BASICAUTH_USERNAME: <masked_user>
      ME_CONFIG_BASICAUTH_PASSWORD: <masked_password>
    ports:
      - "8081:8081"
    networks:
      - bindDog-net

  minio:
    image: quay.io/minio/minio
    container_name: minio
    ports:
      - "9000:9000"
      - "8107:8107"
    environment:
      MINIO_ROOT_USER: <masked_user>
      MINIO_ROOT_PASSWORD: <masked_password>
    command: server /data --console-address ":8107" --address
    volumes:
      - minio_data:/data
    networks:
      - bindDog-net
```

```
networks:
  bindDog-net:

volumes:
  hub_mysql_data:
  mongo_data:
  redis_data:
  minio_data:
```

허브 배포 및 실행 방법

1. **Docker 및 Docker Compose 설치**

   Docker와 Docker Compose가 설치되어 있는지 확인합니다. 설치되어 있지 않다면 아래 명령어로 설치할 수 있습니다:

   ```
   # Docker 설치
   sudo apt update
   sudo apt install docker.io

   # Docker Compose 설치
   sudo apt install docker-compose
   ```

2. **프로젝트 클론**
   olivepay 프로젝트를 클론하고 back/hub/master 브랜치로 체크아웃 합니다.

   ```
   git clone https://lab.ssafy.com/s11-final/S11P31A401
   git checkout back/hub/master
   ```

3. **Docker Compose yml을 프로젝트의 최상단에 복사합니다.**

4. **Docker Compose 실행**
   모든 컨테이너를 동시에 실행하기 위해 Docker Compose 명령어를 사용합니다:

   ```
   docker-compose up --build
   ```

MavenCentral 배포

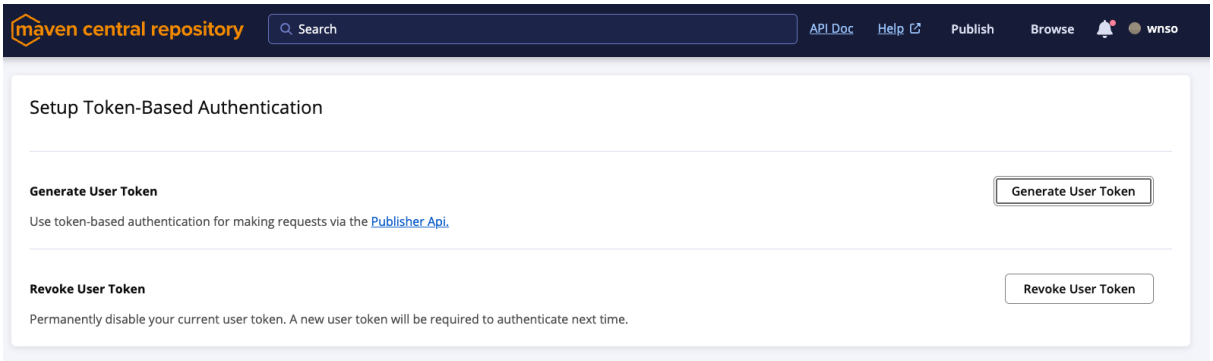
1. **build.gradle**

Maven 배포를 위해 SonatypeHost, signing, mavenPublishing을 작성합니다.

```
import com.vanniktech.maven.publish.SonatypeHost

plugins {
    id 'java-library'
    id 'maven-publish'
    id 'org.springframework.boot' version '3.3.5' apply false
    id 'io.spring.dependency-management' version '1.1.6' appl
    id 'com.vanniktech.maven.publish' version '0.28.0'
    id 'signing'
}

group = 'org.binddog'
version = '0.0.1'

java {
    toolchain {
        languageVersion = JavaLanguageVersion.of(17)
    }
}

repositories {
    mavenCentral()
}

signing {
    useGpgCmd()
    sign publishing.publications
}

dependencies {
    implementation 'org.springframework.boot:spring-boot-star
    testImplementation 'org.springframework.boot:spring-boot-
    testRuntimeOnly 'org.junit.platform:junit-platform-launch
    api 'org.springdoc:springdoc-openapi-starter-webmvc-ui:2.
}
```

```
tasks.named('test') {
    useJUnitPlatform()
}


mavenPublishing {
    publishToMavenCentral(SonatypeHost.CENTRAL_PORTAL)

    coordinates("org.binddog", "binddog", "0.0.1")

    pom {
        name = 'org.binddog'
        description = 'Binddog library'
        url = '<https://binddog.org>'

        licenses {
            license {
                name = 'MIT License'
                url = '<https://lab.ssafy.com/s11-final/S11P3
            }
        }

        developers {
            developer {
                id = 'wnso-kim'
                name = 'wnso'
                email = 'wnso.kim@gmail.com'
            }
        }

        scm {
            connection = 'scm:git:lab.ssafy.com/s11-final/S11
            developerConnection = 'scm:git:ssh://lab.ssafy.co
            url = '<https://lab.ssafy.com/s11-final/S11P31A40
        }
    }
}
```

2. MavenCentral 회원가입 및 token 생성



3. gradle.properties 작성

MavenCentral 에서 발급한 Token을 등록합니다.

```
mavenCentralUsername=
mavenCentralPassword=
```

4. GPG키 생성

배포할 운영체제에서 GPG키 생성 및 MavenCentral에 등록합니다.

Maven Central Repository에서 사용하는 공개 키 서버는 3가지가 있습니다.

```
.
keyserver.ubuntu.com
keys.openpgp.org
pgp.mit.edu

$ gpg --keyserver keyserver.ubuntu.com --send-keys 63196272
gpg: sending key A5E971A363196272 to hkp://keyserver.ubuntu.c
```

5. org.springframework.boot.autoconfigure.AutoConfiguration.imports 추가

```
org.binddog.core.configuration.BindDogConfiguration
```

6. 프론트 빌드 파일

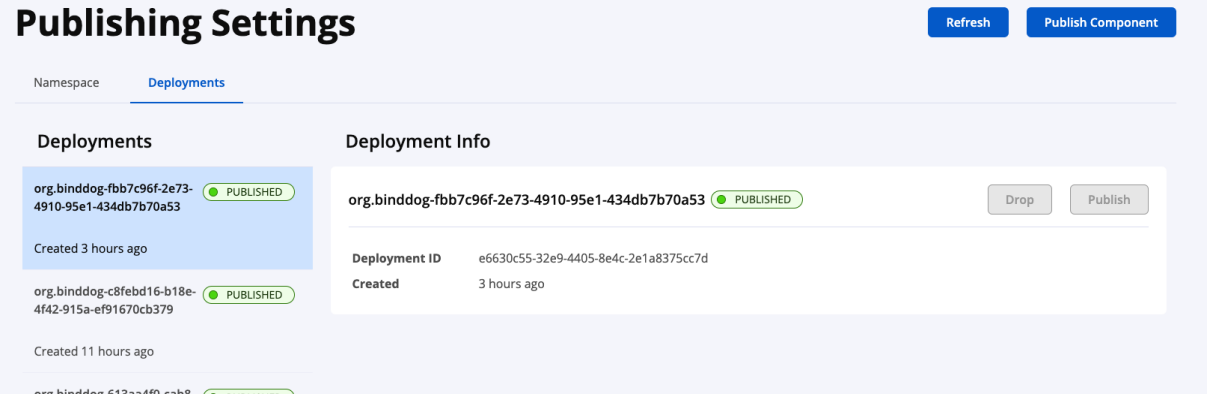
index.html은 `static/binddog/ui` 에 위치 합니다.

그 외의 파일은 `static`에 배치합니다.

7. Maven 배포

```
./gradlew publishAllPublicationsToMavenCentralRepository
```

8. Publish



Docker(React+Nginx) 배포

1. Docker file

```
# 베이스 이미지로 Node.js 20-alpine을 사용
FROM node:20-alpine as builder

# 앱 디렉토리를 생성하고 작업 디렉토리로 설정
WORKDIR /app

# package.json과 package-lock.json을 복사 (가능한 경우)
COPY ./package*.json ./
```

```
# 의존성 설치
RUN npm install

# 앱 소스 복사
COPY . .

# 빌드
RUN npm run build

# nginx 설정
FROM nginx
COPY ./nginx/default.conf /etc/nginx/conf.d/default.conf
COPY --from=builder /app/build /usr/share/nginx/html
```

2. **Jenkins**

```
pipeline {
    agent any

    environment {
        GIT_REPO_URL = 'https://lab.ssafy.com/s11-final/S11P3
        GIT_BRANCH = 'front/hub/master'
        IMAGE_NAME = 'frontend-app-image:latest' // Docker 이
        DOCKERFILE_PATH = 'frontend/binddog-hub' // Dockerfil
        ENV_FILE_PATH = 'frontend/binddog-hub/.env' // env 파
    }

    stages {
        stage('Checkout') {
            steps {
                // 브랜치 체크아웃
                git branch: env.GIT_BRANCH, url: env.GIT_REPO
            }
        }
        stage('Add Env') {
            steps {
```

```
                script {
                    withCredentials([file(credentialsId: 'fro
                        sh '''
                        chmod -R 755 .
                        cp $ENV_FILE ${ENV_FILE_PATH}
                        '''
                    }
                }
            }
        }


        stage('Build Docker Image') {
            steps {
                script {
                    // Dockerfile을 사용하여 Docker 이미지 빌드
                    docker.build("${IMAGE_NAME}", "${DOCKERFI
                }
            }
        }

        stage('Run Docker Container') {
            steps {
                script {
                    // 기존 컨테이너가 있다면 중지 및 삭제한 후 새 컨
                    sh """
                    docker stop hub-front || true
                    docker rm hub-front || true
                    docker run -d --name hub-front -p 3000:80
                    """
                }
            }
        }

        stage('Clean up unused images') {
            steps {
                script {
                    // 사용되지 않는 이미지를 자동으로 삭제
```

```
                    sh 'docker image prune -a -f'
                }
            }
        }
    }


    post {

        always {
            cleanWs()  // 빌드 완료 후 작업 공간을 정리
        }
        success {
            echo 'Deployment succeeded!'
        }
        failure {
            echo 'Deployment failed!'
        }
    }
}
```

Tag/Release 자동화

.gitlab-ci.yml

```
image: node:17.1-alpine3.14

stages:
  - tag
  - release

create_tag:
  stage: tag
  script:
    - echo "Installing git..."
    - apk add --no-cache git
```

```
    - echo "Fetching existing tags..."
    - git fetch --tags

    - echo "Getting the latest tag..."
    - LAST_TAG=$(git describe --tags $(git rev-list --tags --

    - echo "Last tag is ${LAST_TAG}"

    - |
      if [ -z "${LAST_TAG}" ]; then
        TAG_NAME="v0.0.1";  # 최초 태그가 없을 경우 기본값 설정
        echo "${TAG_NAME}"
      else
        # 기존 태그에서 버전 숫자 추출
        VERSION=${LAST_TAG:1}  # 'v' 제거
        echo "VERSION: ${VERSION}"
        # IFS를 설정하고 read로 배열에 분리
        OLD_IFS=$IFS
        IFS='.'
        set -- ${VERSION}  # 공백으로 분리된 인수로 설정
        MAJOR=$1
        MINOR=$2
        PATCH=$3
        IFS=$OLD_IFS

        echo "Major version: ${MAJOR}"
        echo "Minor version: ${MINOR}"
        echo "Patch version: ${PATCH}"

        # 수정 버전 증가
        PATCH=$((PATCH + 1))
        TAG_NAME="v${MAJOR}.${MINOR}.${PATCH}"  # 새로운 태그 생
        echo "New tag name:${TAG_NAME}"
      fi


    - git tag $TAG_NAME
    - git push https://wnso.kim:${CI_JOB_TOKEN}@lab.ssafy.com
```

```
    # 새로운 태그와 마지막 태그를 파일에 저장
    - echo "TAG_NAME=${TAG_NAME}" > tag_name.env
    - echo "LAST_TAG=${LAST_TAG}" >> tag_name.env

  # master 브랜치에 Merge될 때만 실행
  rules:
   - if: '$CI_COMMIT_BRANCH == "master" && $CI_PIPELINE_SOURC
  artifacts:
    paths:
      - tag_name.env  # 생성된 태그 파일을 아티팩트로 저장
  tags:
    - binddog


release_job:
  stage: release
  needs:
    - create_tag  # create_tag 작업이 완료된 후에 실행
  script:
    - echo "Installing git..."
    - apk add --no-cache git

    - echo "Installing curl..."
    - apk add --no-cache curl  # curl 설치

    - echo "Loading tag name from file..."
    - source tag_name.env  # 파일에서 변수 불러오기

    # 모든 커밀 불러오기
    # - COMMITS=$(git log --merges $LAST_TAG..HEAD --pretty=%
    # - echo "ALL COMMITS\n$COMMITS"

    # Commit 메시지 추출 및 description 자동 생성
    - echo "Running release job for tag $TAG_NAME"
    - echo "Generating release description..."
    - |
      # 이전 태그 이후 Commit 내역을 불러오거나 빈 문자열 반환
```

```
        COMMITS=$(git log --merges $LAST_TAG..HEAD --pretty=%B

        DESCRIPTION="### 📝 릴리즈 노트 ${TAG_NAME}\n"
        # 현재 날짜를 YYYY-MM-DD 형식으로 가져오기
        DEPLOY_DATE=$(date +'%Y-%m-%d')
        DESCRIPTION="$DESCRIPTION\nRelease date: ${DEPLOY_DATE}
        DESCRIPTION="$DESCRIPTION > 🚨 현재 Binddog은 베타 버전입니

        FEAT_COMMITS=""
        FIX_COMMITS=""
        REFACTOR_COMMITS=""

        # 커밋 메시지 분석 및 분류
        echo "Commits list ${COMMITS}"
        OLD_IFS=$IFS
        IFS=$'\n'  # 줄바꿈으로 분리
        for COMMIT in $COMMITS; do
          case "$COMMIT" in
            *"[기능 개발]"*)
              FEAT_COMMITS="${FEAT_COMMITS}- $COMMIT\n"
              ;;
            *"[기능 수정]"*)
              REFACTOR_COMMITS="${REFACTOR_COMMITS}- $COMMIT\n"
              ;;
            *"[버그 수정]"*)
              FIX_COMMITS="${FIX_COMMITS}- $COMMIT\n"
              ;;
          esac
        done
        IFS=$OLD_IFS  # 기본 IFS로 복원

        # 카테고리별로 DESCRIPTION에 추가
        if [ -n "$FEAT_COMMITS" ]; then
          DESCRIPTION="$DESCRIPTION---\n### 🆕 새로운 기능\n$FEAT_
        fi
        if [ -n "$REFACTOR_COMMITS" ]; then
          DESCRIPTION="$DESCRIPTION---\n### ♻️ 리팩토링\n$REFACTO
        fi
```

```
      if [ -n "$FIX_COMMITS" ]; then
        DESCRIPTION="$DESCRIPTION---\n### 🐛 버그 수정\n$FIX_CO
      fi

      echo "FEAT COMMITS---\n$FEAT_COMMITS"
      echo "REFACTOR COMMITS---\n$REFACTOR_COMMITS"
      echo "FIX COMMITS---\n$FIX_COMMITS"
      echo "DESCRIPTION---\n$DESCRIPTION"



  - |
      curl --header 'Content-Type: application/json' \
       --header "PRIVATE-TOKEN: ${CI_JOB_TOKEN}" \
       --data '{
         "name": "Binddog '"${TAG_NAME:1}"'",
         "tag_name": "'"$TAG_NAME"'",
         "description": "'"${DESCRIPTION}"'",
         "assets": {
           "links": [
             {
               "name": "🚀 Binddog Hub Site",
               "url": "https://binddog.org",
               "link_type": "other"
             }
           ]
         }
       }' \
       --request POST "https://lab.ssafy.com/api/v4/projects/
rules:
  - if: '$CI_COMMIT_BRANCH == "master" && $CI_PIPELINE_SOUR

tags:
  - binddog
```