곰방 포팅 매뉴얼

Gitlab 소스 클론 이후 빌드 및 배포할 수 있도록 정리한 문서

1) 사용한 JVM, 웹서버, WAS, 제품 등의 종류와 설정값, 버전(IDE 포함) 기재

FE	
스택	버전
환경	
node	10.19.0
npm	6.14.4
프레임워크	
React	^18.2.0
라우터	
react-router-dom	^6.14.2
통신 라이브러리	
axios	^1.4.0
sockjs-client	^1.6.1
stompjs	^2.3.3

BE & Environment	
스택	버전
JDK	11.0.19
Spring Boot	2.4.5
Gradle	6.14.4
Ec2 ubuntu	20.0.4
Docker	24.0.5
Mysql	5.7
MongoDB	5.0.19
Redis	5.0.14
ElasticSearch	8.9.0
Kibana	8.9.0
Jenkins	2.401.3
Nginx	1.18.0
Git	2.25.1
Intellij	2023.1.3
VSCode	1.81.1

2) 빌드시 사용되는 환경변수

build.gradle

```
import org.apache.tools.ant.filters.ReplaceTokens

buildscript{
    ext {
        springBootVer = '2.4.5'
        querydslVer = '4.4.0'
        querydslPluginVer = '1.0.10'
        springDependencyMgmtVer = '1.0.11'
        springLoadedVer = '1.2.8'
        nodePluginVer = '1.3.1'
    }
    repositories {
        mavenCentral()
        jcenter()
    }
    dependencies {
        classpath "org.springframework.boot:spring-boot-gradle-
plugin:${springBootVer}"
        classpath "io.spring.gradle:dependency-management-
plugin:${springDependencyMgmtVer}.RELEASE"
        classpath
"org.springframework:springloaded:${springLoadedVer}.RELEASE"
        classpath "com.github.node-gradle:gradle-node-plugin:3.1.0"
    }
}

plugins {
    id 'java'
    id 'idea'
    id 'org.springframework.boot' version "${springBootVer}"
}

apply plugin: 'io.spring.dependency-management'
apply plugin: 'eclipse'
apply plugin: 'com.github.node-gradle.node'


repositories {
    mavenCentral()
    maven { url 'https://repo.spring.io/snapshot' }
    maven { url 'https://repo.spring.io/milestone' }
    maven { url "https://repo.spring.io/libs-release" }
    maven { url "https://repo.maven.apache.org/maven2" }
    maven { url
"https://build.shibboleth.net/nexus/content/repositories/releases" }
}

group 'com.ssafy'
version '1.0-SNAPSHOT'
sourceCompatibility = '11'


configurations {
    providedRuntime
```

```
}

def buildTime() {
    def date = new Date()
    def formattedDate = date.format('yyyyMMdd_HHmm')
    return formattedDate
}

project.ext.set("build.date", buildTime())

processResources {
    with copySpec {
        from "src/main/resources"
        include "**/application*.yml"
        include "**/application*.yaml"
        include "**/application*.properties"
        project.properties.findAll().each {
            prop ->
                if (prop.value != null) {
                    filter(ReplaceTokens, tokens: [ (prop.key):
String.valueOf(prop.value)])
                    filter(ReplaceTokens, tokens: [ ('project.' + prop.key):
String.valueOf(prop.value)])
                    filter(ReplaceTokens, tokens: [ ('project.ext.' +
prop.key): String.valueOf(prop.value)])
                }
        }
    }
}

dependencies {
    implementation("org.springframework.boot:spring-boot-starter-web")
    implementation("org.springframework.boot:spring-boot-starter-
websocket")
    implementation("org.springframework.boot:spring-boot-starter-
security")
    implementation("org.springframework.boot:spring-boot-starter-data-
jpa")
    implementation("org.springframework.boot:spring-boot-starter-
actuator")
    implementation("org.springframework.boot:spring-boot-starter-
validation")
    implementation("org.springframework.plugin:spring-plugin-
core:2.0.0.RELEASE")
    testImplementation("org.springframework.security:spring-security-
test")
    annotationProcessor("org.springframework.boot:spring-boot-starter-
data-jpa")
    runtimeOnly("mysql:mysql-connector-java")
    developmentOnly("org.springframework.boot:spring-boot-devtools")
    annotationProcessor("org.springframework.boot:spring-boot-
configuration-processor")

    implementation('commons-io:commons-io:2.6')
    implementation("org.apache.commons:commons-collections4:4.4")
    implementation("org.apache.commons:commons-lang3:3.9")

    implementation("com.querydsl:querydsl-jpa:${querydslVer}")
    implementation("com.querydsl:querydsl-apt:${querydslVer}")
```

```
    //STOMP 웹소캣 서버 사이드 테스트를 위한 의존성 추가
    implementation("org.springframework.boot:spring-boot-starter-
mustache")
    implementation("org.springframework.boot:spring-boot-starter-
websocket")

    //STOMP 관련 프론트 라이브러리
    implementation('org.webjars.bower:jquery:3.3.1')
    implementation('org.webjars:sockjs-client:1.1.2')
    implementation('org.webjars:stomp-websocket:2.3.3-1')
    implementation('org.webjars:webjars-locator:0.30')
    //WebRTC 클라이언트 의존성 추가
    implementation('org.webjars.bower:webrtc-adapter:7.4.0')
    //Kurento (미디어서버) 관련 의존성 추가
    implementation('org.kurento:kurento-client:6.16.0')
    implementation('org.kurento:kurento-utils-js:6.15.0')

    // validation
    implementation("org.springframework.boot:spring-boot-starter-
validation")

    // MongoDB
    // MongoDB 단독 사용이면 뒤에 reactive 뺄 것
    // 다른 DB랑 같이 쓸려면 reactive 적을 것
    implementation ('org.springframework.boot:spring-boot-starter-data-
mongodb')

    annotationProcessor("com.querydsl:querydsl-apt:${querydslVer}:jpa")

    implementation("com.squareup.retrofit2:retrofit:2.7.1")
    implementation("com.squareup.retrofit2:converter-jackson:2.7.1")
    implementation("com.squareup.okhttp3:logging-interceptor:3.9.0")

    implementation("com.google.guava:guava:29.0-jre")
    annotationProcessor("com.google.guava:guava:29.0-jre")

    testImplementation("com.jayway.jsonpath:json-path:2.4.0")

    implementation("com.auth0:java-jwt:3.10.3")

    implementation("io.springfox:springfox-swagger2:3.0.0")
    implementation("io.springfox:springfox-swagger-ui:3.0.0")
    implementation("io.springfox:springfox-data-rest:3.0.0")
    implementation("io.springfox:springfox-bean-validators:3.0.0")
    implementation("io.springfox:springfox-boot-starter:3.0.0")

    compile("javax.annotation:javax.annotation-api:1.2")

    implementation("org.projectlombok:lombok:1.18.20")
    annotationProcessor("org.projectlombok:lombok:1.18.20")

    testCompile('org.springframework.boot:spring-boot-starter-test')

    implementation('com.googlecode.json-simple:json-simple:1.1.1')
    implementation("org.springframework.boot:spring-boot-starter-
validation")
    implementation('org.springframework.boot:spring-boot-starter-
```

```
webflux')
    implementation 'org.springframework.boot:spring-boot-starter-data-
elasticsearch'

    // Redis
    implementation 'org.springframework.boot:spring-boot-starter-data-
redis'
    implementation 'it.ozimov:embedded-redis:0.7.2'

    // Cache
    implementation("org.springframework.boot:spring-boot-starter-cache")

    // S3
    implementation 'org.springframework.cloud:spring-cloud-starter-
aws:2.2.6.RELEASE'

}

test {
    useJUnitPlatform()
}
```

application.properties

```
#it will be set build date by gradle. if this value is @build.date@, front-
end is development mode
build.date=@build.date@
server.port={포트번호}
server.address={ip 주소}
server.servlet.contextPath={contextPath 값}
# Charset of HTTP requests and responses. Added to the "Content-Type"
header if not set explicitly.
server.servlet.encoding.charset=UTF-8
# Enable http encoding support.
server.servlet.encoding.enabled=true
# Force the encoding to the configured charset on HTTP requests and
responses.
server.servlet.encoding.force=true

# for SPA
spring.web.resources.static-locations=classpath:/dist/
spa.default-file=/dist/index.html
spring.mvc.throw-exception-if-no-handler-found=true
spring.web.resources.add-mappings=false

# Swagger
springfox.documentation.swagger.use-model-v3=false

#database
spring.jpa.hibernate.naming.implicit-
strategy=org.springframework.boot.orm.jpa.hibernate.SpringImplicitNamingS
trategy
spring.jpa.hibernate.naming.physical-
strategy=org.springframework.boot.orm.jpa.hibernate.SpringPhysicalNamingS
trategy
spring.jpa.hibernate.ddl-auto=none
spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.MySQL57Dial
```

```
ect
spring.data.web.pageable.one-indexed-parameters=true
spring.datasource.url=jdbc:mysql://{ip 주소}:{포트번호}/{스키마이름}?useUnicode=true&characterEncoding=utf8&serverTimezone=Asia/Seoul&zeroDateTimeBehavior=convertToNull&rewriteBatchedStatements=true
spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver
spring.datasource.hikari.username={mysql 계정 이름}
spring.datasource.hikari.password={mysql 계정 비밀번호}

# jwt
jwt.secret={jwt secretKey}
# unit is ms. 15 * 24 * 60 * 60 * 1000 = 15days
jwt.expiration=1296000000

#logging
logging.file.name={로깅 파일 이름}
logging.level.root=INFO
logging.level.com.samsung.security=DEBUG
logging.level.org.springframework.web=DEBUG
logging.level.org.apache.tiles=INFO
logging.level.org.sringframework.boot=DEBUG
logging.level.org.sringframework.security=DEBUG

spring.devtools.livereload.enabled=true

#gzip compression
server.compression.enabled=true
server.compression.mime-types=application/json,application/xml,text/html,text/xml,text/plain,application/javascript,text/css

#for health check
management.servlet.context-path=/manage
management.health.db.enabled=true
management.health.default.enabled=true
management.health.diskspace.enabled=true

# KakaoLoginAPI
kakao.client.id={KakaoLoginAPI id}
kakao.client.secret={KakaoLoginAPI secretKey}
kakao.redirect.url={redirect url}

# MongoDB
spring.data.mongodb.uri=mongodb://{ip 주소}:{포트번호}
spring.data.mongodb.database={스키마이름}

# Redis
spring.redis.host={ip 주소}
spring.redis.port={포트번호}

# S3
cloud.aws.credentials.accessKey={S3 accessKey}
cloud.aws.credentials.secretKey={S3 sscretKey}
cloud.aws.s3.bucket={S3 버킷 이름}
cloud.aws.region.static={S3 지역}
cloud.aws.stack.auto-=false
```

프로젝트에서 사용하는 외부 서비스 정보를 정리한 문서

카카오 소셜 로그인

1. Kakao developers에서 새 어플리케이션을 생성한다.
2. '카카오 로그인'을 활성화 하고, redirection url을 입력한다.
3. 로그인 시 정보를 제공받을 범위를 설정하기 위해, 동의 할목 탭에서 원하는 항목들을 체크한다.
4. application.properties에 다음과 같이 정보를 입력한다.

```
# KakaoLoginAPI
kakao.client.id={KakaoLoginAPI id}
kakao.client.secret={KakaoLoginAPI secretKey}
kakao.redirect.url={KakaoLoginAPI URL}
```

카카오 지도

1. Kakao developers에서 새 어플리케이션을 생성한다.
2. 해당 어플리케이션에 도메인 호스트명을 등록한다.
3. 발급 받은 key로 CDN형태의 Kakaomap Library에 접근한다.
4. WGS84, zoomlevel등 지도 생성에 필요한 parameter를 입력하고 원하는 형태로 호출한다.

AWS S3

1. AWS에서 S3 버킷을 생성한다.
2. 권한 탭에서 버킷 정책을 생성해 제 3자의 객체 읽기, 쓰기 권한을 부여한다.
3. application.properties에 다음과 같이 정보를 입력한다.

```
# S3
cloud.aws.credentials.accessKey={S3 accessKey}
cloud.aws.credentials.secretKey={S3 secretKey}
cloud.aws.s3.bucket={S3 버킷 이름}
cloud.aws.region.static={S3 지역}
cloud.aws.stack.auto-=false
```