

E106 finBall포팅 매뉴얼

프로젝트 기간 : 2023.08.21 ~ 2023.10.06(7주)

팀장 : 정영빈

팀원 : 김정희, 서정희, 신현탁, 정현우, 하성호

목차

[1. 사용한 JVM, 웹서버, WAS 제품 등의 종류와 설정 값, 버전 기재 2](#_Toc147455660)

[2. 빌드 시 사용되는 환경 변수 등의 내용 3](#_Toc147455661)

[3. 배포 시 특이사항 9](#_Toc147455662)

[4. DB 접속 정보 12](#_Toc147455663)

# 사용한 JVM, 웹서버, WAS 제품 등의 종류와 설정 값, 버전 기재

1. Java

1.8

1. 웹서버

NginX

1. WAS

Tomcat

1. Gradle

6.8.3

1. Spring Boot

2.7.15

1. Node

18.17.0

1. React

18.2.0

1. Intellij

2019.3.5, 2023.1.5

1. Visual Studio Code

1.81.1

# 2. 빌드 시 사용되는 환경 변수 등의 내용

### Backend

- application.yml

|  |
| --- |
| * *# 공통 부분 -> 아마 프로파일과 무관* spring:  datasource:  driver-class-name: com.mysql.cj.jdbc.Driver  url: jdbc:mysql://j9e106.p.ssafy.io:3306/finball?characterEncoding=UTF-8&serverTimezone=UTC  username: 사용자아이디  password: 비번  profiles:  active: local  jpa:  properties:  hibernate:  globally\_quoted\_identifiers: 'true'  format\_sql: 'true'  show-sql: 'true'  hibernate:  ddl-auto: none  auth:  secret:  key: asdfasdf  redis:  host: j9e106.p.ssafy.io  port: 6379 *# MongoDB* data:  mongodb:  uri: mongodb://j9e106.p.ssafy.io:27017/finball   *# MatterMost* notification:  mattermost:  enabled: true  webhook-url: https://meeting.ssafy.com/hooks/wn83kosiefggdpp71mfhc1bbho jwt:  secret: VlwEyVBsYt9V7zq57TejMnVUyzblYcfPQye08f7MGVA9XkHa sms:  serviceId: 서비스ID  accessKey: 접근키  secretKey: 시크릿키  phoneNumber: 01000000000 --- *#set1 프로파일* spring:  config:  activate:  on-profile: set1 server:  port: 8081  --- *#set2 프로파일* spring:  config:  activate:  on-profile: set1 server:  port: 8082 |

- build.gralde

|  |
| --- |
| * buildscript {  ext {  queryDslVersion = "5.0.0"  } }  plugins {  id 'java'  id 'org.springframework.boot' version '2.7.15'  id 'io.spring.dependency-management' version '1.0.15.RELEASE' }  group = 'com.ssafy' version = '0.0.1-SNAPSHOT'  java {  sourceCompatibility = JavaVersion.VERSION\_1\_8  targetCompatibility = JavaVersion.VERSION\_1\_8 }  configurations {  compileOnly {  extendsFrom annotationProcessor  } }  repositories {  mavenCentral() }  dependencies {  implementation 'org.springframework.boot:spring-boot-starter-data-jpa'  implementation("org.springframework.boot:spring-boot-starter-validation")  implementation 'org.springframework.boot:spring-boot-starter-web'  testImplementation 'org.springframework.boot:spring-boot-starter-test'  implementation 'org.springframework.boot:spring-boot-starter-security'  testImplementation 'org.projectlombok:lombok:1.18.28'  runtimeOnly 'com.mysql:mysql-connector-j'  compileOnly 'org.projectlombok:lombok'   // Jwt  implementation("io.jsonwebtoken:jjwt-api:0.11.2")  implementation("commons-codec:commons-codec:1.5")  implementation("io.jsonwebtoken:jjwt-impl:0.11.2")  implementation("io.jsonwebtoken:jjwt-jackson:0.11.2")  implementation("com.auth0:java-jwt:3.13.0")   // builder  annotationProcessor('org.projectlombok:lombok')  testAnnotationProcessor('org.projectlombok:lombok')   // Mongo DB  implementation 'org.springframework.boot:spring-boot-starter-data-mongodb'   // queryDSL 설정  implementation "com.querydsl:querydsl-jpa"  implementation "com.querydsl:querydsl-core"  implementation "com.querydsl:querydsl-collections"  annotationProcessor "com.querydsl:querydsl-apt:${dependencyManagement.importedProperties['querydsl.version']}:jpa" // querydsl JPAAnnotationProcessor 사용 지정  annotationProcessor "jakarta.annotation:jakarta.annotation-api" // java.lang.NoClassDefFoundError (javax.annotation.Generated) 대응 코드  annotationProcessor "jakarta.persistence:jakarta.persistence-api" // java.lang.NoClassDefFoundError (javax.annotation.Entity) 대응 코드   //헬스 체크 의존 추가  implementation 'org.springframework.boot:spring-boot-starter-actuator'   // redis  implementation 'org.springframework.boot:spring-boot-starter-data-redis'   // json 파싱  implementation group: 'com.googlecode.json-simple', name: 'json-simple', version: '1.1.1'   // PathVariable Validation  implementation 'org.hibernate.validator:hibernate-validator'   // java8 LocalDate  implementation 'com.fasterxml.jackson.datatype:jackson-datatype-jsr310'   // sms  implementation group: 'org.apache.httpcomponents', name: 'httpclient', version: '4.5.13'   // Gson  implementation 'com.google.code.gson:gson:2.9.0'  }  // Querydsl 설정부 def generated = 'src/main/generated'  // querydsl QClass 파일 생성 위치를 지정 tasks.withType(JavaCompile) {  options.getGeneratedSourceOutputDirectory().set(file(generated)) }  // java source set 에 querydsl QClass 위치 추가 sourceSets {  main.java.srcDirs += [ generated ] }  // gradle clean 시에 QClass 디렉토리 삭제 clean {  delete file(generated) }  tasks.named('test') {  useJUnitPlatform() } |

### mydata

- application.properties

|  |
| --- |
| server.port=8081 spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver spring.datasource.url=jdbc:mysql://j9e106.p.ssafy.io:3306/mydata?characterEncoding=UTF-8&serverTimezone=UTC spring.datasource.username=아이디 spring.datasource.password=비번 spring.jpa.show-sql=true spring.jpa.hibernate.ddl-auto=none spring.jpa.properties.hibernate.format\_sql=true spring.jpa.properties.hibernate.globally\_quoted\_identifiers=true spring.auth.secret.key=asdfasdf jwt.secret=VlwEyVBsYt9V7zq57TejMnVUyzblYcfPQye08f7MGVA9XkHa  # redis spring.redis.host=j9e106.p.ssafy.io spring.redis.port=6379 |

- build.gradle

|  |
| --- |
| buildscript **{** ext **{** queryDslVersion = "5.0.0"  **} }** plugins **{** id 'java'  id 'org.springframework.boot' version '2.7.15'  id 'io.spring.dependency-management' version '1.0.15.RELEASE' **}** group = 'com.finball' version = '0.0.1-SNAPSHOT'  java **{** sourceCompatibility = JavaVersion.*VERSION\_1\_8* targetCompatibility = JavaVersion.*VERSION\_1\_8* **}** configurations **{** compileOnly **{** extendsFrom annotationProcessor  **} }** repositories **{** mavenCentral() **}** dependencies **{** implementation 'org.springframework.boot:spring-boot-starter-data-jpa'  implementation 'org.springframework.boot:spring-boot-starter-security'  implementation 'org.springframework.boot:spring-boot-starter-web'  compileOnly 'org.projectlombok:lombok'  runtimeOnly 'com.mysql:mysql-connector-j'  annotationProcessor 'org.projectlombok:lombok'  testImplementation 'org.springframework.boot:spring-boot-starter-test'  testImplementation 'org.springframework.security:spring-security-test'   // https://mvnrepository.com/artifact/com.auth0/java-jwt  implementation group: 'com.auth0', name: 'java-jwt', version: '3.10.3'   // json parsing  implementation group: 'com.googlecode.json-simple', name: 'json-simple', version: '1.1.1'   // java8 LocalDate  implementation 'com.fasterxml.jackson.datatype:jackson-datatype-jsr310'   // queryDSL 설정  implementation "com.querydsl:querydsl-jpa"  implementation "com.querydsl:querydsl-core"  implementation "com.querydsl:querydsl-collections"  annotationProcessor "com.querydsl:querydsl-apt:$**{**dependencyManagement.importedProperties['querydsl.version']**}**:jpa" // querydsl JPAAnnotationProcessor 사용 지정  annotationProcessor "jakarta.annotation:jakarta.annotation-api" // java.lang.NoClassDefFoundError (javax.annotation.Generated) 대응 코드  annotationProcessor "jakarta.persistence:jakarta.persistence-api" // java.lang.NoClassDefFoundError (javax.annotation.Entity) 대응 코드 **}** // Querydsl 설정부 def generated = 'src/main/generated'  // querydsl QClass 파일 생성 위치를 지정 tasks.withType(JavaCompile) **{** options.getGeneratedSourceOutputDirectory().set(file(generated)) **}** // java source set 에 querydsl QClass 위치 추가 sourceSets **{** main.java.srcDirs += [ generated ] **}** // gradle clean 시에 QClass 디렉토리 삭제 clean **{** delete file(generated) **}** tasks.named('test') **{** useJUnitPlatform() **}** |

# 3. 배포 시 특이사항

### NginX설정

|  |
| --- |
| 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;  #proxy\_pass $service\_url;  proxy\_pass http://localhost:5173;  }  location /api {  #proxy\_pass $service\_url;  #proxy\_pass http://i9e206.p.ssafy.io:5000;  #proxy\_pass http://backend;  proxy\_pass $service\_url;  }  include /etc/nginx/conf.d/service-url.inc;  location /chat\_bot {  proxy\_pass http://localhost:8000;  }  #location /profile {  #proxy\_pass $service\_url;  #proxy\_pass http://i9e206.p.ssafy.io:5000;  #proxy\_pass http://backend;  # proxy\_pass http://j9e106.p.ssafy.io:8082;  #}  listen 443 ssl;  ssl\_certificate /etc/letsencrypt/live/j9e106.p.ssafy.io/fullchain.pem;  ssl\_certificate\_key /etc/letsencrypt/live/j9e106.p.ssafy.io/privkey.pem;  # pass PHP scripts to FastCGI server  #location ~ \.php$ {  # include snippets/fastcgi-php.conf;  #  # # With php-fpm (or other unix sockets):  # fastcgi\_pass unix:/var/run/php/php7.4-fpm.sock;  # # With php-cgi (or other tcp sockets):  # fastcgi\_pass 127.0.0.1:9000;  #}  # deny access to .htaccess files, if Apache's document root  # concurs with nginx's one  #  #location ~ /\.ht {  # deny all;  #}  }  server {  if ($host = j9e106.p.ssafy.io) {  return 301 https://$host$request\_uri;  }  listen 80;  server\_name j9e106.p.ssafy.io;  return 404;  } |

sudo service nginx start

# 4. DB 접속 정보

----MySql---

Host : j9e106.p.ssafy.io

Port : 3306

User : root

Password : finball

Database : finball, mydata