포팅 메뉴얼

사용 프로그램 버전

<markdown>
<markdown>

포팅 메뉴얼

1. 개발 환경

- Database : MySql 8
- JVM : 17
- Spring Boot : 3.0.1
- Jenkins : 2.423

2. 설정 파일

Spring Boot - application.yml

```
server:
  port : 8081
spring:
  datasource:
    url: jdbc:mysql://j9c102.p.ssafy.io:3306/mysql  # MySQL ?? ??? ??, ?????? ?? ??
    username: DB아이디  # MySQL ??? ??
    password: DB비밀번호  # MySQL ??
    driver-class-name: com.mysql.cj.jdbc.Driver  # MySQL ???? ??? ??
  jpa:
    hibernate:
      ddl-auto: update  # ?? DDL ?? ?? (update, create, create-drop, none ? ??)
    show-sql: true  # SQL ?? ?? ?? (?? ????? true? ???? ???? ??)
    database-platform: org.hibernate.dialect.MySQL8Dialect  # MySQL ??? ?? Dialect ??
logging:
  level:
    com.alphano.alphano: debug
jwt:
  secret:
c3ByaW5nLWJvb3Qtc2VjdXJpdHktand0LXR1dG9yaWFsLWppd29vbi1zcHJpbmctYm9vdC1zZWN1cml0eS1qd
3QtdHV0b3JpYWwK
```

위 예시는 Spring Boot의 `application.yml` 파일의 일부입니다. 여기에는 데이터베이스 연결 정보와 JPA 설정이 포함됩니다. `url`에는 데이터베이스의 URL을, `username`과 `password`에는

데이터베이스에 접속하기 위한 사용자 이름과 비밀번호를 입력해야 합니다. `dialect`는 사용하는 데이터베이스의 Dialect를 설정합니다. 저희는 MySQL 8을 사용하고 있습니다.

Spring Boot - build.gradle

```
plugins {
	id 'java'
	id 'org.springframework.boot' version '3.0.10'
	id 'io.spring.dependency-management' version '1.1.3'
}

group = 'com.alphano'
version = '0.0.1-SNAPSHOT'

java {
	sourceCompatibility = '17'
}


configurations {
	compileOnly {
		extendsFrom annotationProcessor
	}
}

repositories {
	mavenCentral()
}

dependencies {
	implementation 'org.springframework.boot:spring-boot-starter'
	implementation 'org.springframework.boot:spring-boot-starter-data-jpa'
	implementation 'org.springframework.boot:spring-boot-starter-jdbc'
	implementation 'org.springframework.boot:spring-boot-starter-data-redis'

	implementation 'org.springframework.boot:spring-boot-starter-security'
	implementation group: 'com.fasterxml.jackson.core', name: 'jackson-databind',
version: '2.12.3'

	implementation group: 'io.jsonwebtoken', name: 'jjwt-api', version: '0.11.5'
	runtimeOnly group: 'io.jsonwebtoken', name: 'jjwt-impl', version: '0.11.5'
	runtimeOnly group: 'io.jsonwebtoken', name: 'jjwt-jackson', version: '0.11.5'


	implementation 'org.springframework.boot:spring-boot-starter-web'
	implementation 'org.springframework.boot:spring-boot-starter-websocket'


	//swagger api
	implementation 'org.springdoc:springdoc-openapi-starter-webmvc-ui:2.0.2'

	implementation 'org.hibernate.validator:hibernate-validator:6.2.0.Final' //
Use the latest version
```

```
	//redis
	implementation 'org.springframework.boot:spring-boot-starter-data-redis'

	//querydsl
	implementation "com.querydsl:querydsl-core:5.0.0"
	implementation "com.querydsl:querydsl-collections"
	implementation("com.querydsl:querydsl-jpa:5.0.0:jakarta")
	annotationProcessor "com.querydsl:querydsl-apt:5.0.0:jakarta"
	annotationProcessor "jakarta.annotation:jakarta.annotation-api"
	annotationProcessor "jakarta.persistence:jakarta.persistence-api:3.1.0"

	//spring batch
// https://mvnrepository.com/artifact/org.springframework.boot/spring-boot-starter-batch
	implementation group: 'org.springframework.boot', name: 'spring-boot-starter-batch'



	compileOnly 'org.projectlombok:lombok'
	developmentOnly 'org.springframework.boot:spring-boot-devtools'
	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'
}

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

다음은 Docker 설치 과정입니다:

1. 우선, 시스템 패키지를 업데이트합니다:

```
sudo apt-get update
```

1. 필요한 패키지를 설치합니다:

```
sudo apt-get install ca-certificates curl gnupg
```

1. Docker 관련 디렉토리를 생성합니다:

```
sudo install -m 0755 -d /etc/apt/keyrings
```

1. Docker GPG 키를 다운로드하고 설치합니다:

```
curl -fsSL <https://download.docker.com/linux/ubuntu/gpg> | sudo gpg --dearmor -o /etc/apt/keyrings/docker.gpg
sudo chmod a+r /etc/apt/keyrings/docker.gpg
```

1. Docker 저장소를 설정합니다:

```
echo \
  "deb [arch="$(dpkg --print-architecture)" signed-by=/etc/apt/keyrings/docker.gpg]
<https://download.docker.com/linux/ubuntu> \
  $(. /etc/os-release && echo "$VERSION_CODENAME") stable" | \
  sudo tee /etc/apt/sources.list.d/docker.list > /dev/null
```

1. 다시 한 번 패키지를 업데이트합니다:

```
sudo apt-get update
```

1. Docker를 설치합니다:

```
sudo apt-get install docker-ce docker-ce-cli containerd.io docker-buildx-plugin
docker-compose-plugin
```

Docker 에 MySql 설치

```
// 도커 mysql 이미지 다운
$ sudo docker pull mysql

// 도커 이미지 확인
$ sudo docker images

// 3306 포트열기
$ sudo ufw allow 3306

// 컨테이너 mysql 적재
$ sudo docker run -d --name mysql -e MYSQL_ROOT_PASSWORD=[패스워드] -p 3306:3306
mysql

// 컨테이너 확인
$ sudo docker ps

// 컨테이너에 접속
$ sudo docker exec -it mysql bash

// 관리자로 접속
$ mysql -u root -p

$ 패스워드 입력

use mysql

CREATE USER '사용자명'@'%' IDENTIFIED BY '비밀번호';
```

```
GRANT ALL PRIVILEGES ON *.* TO '사용자명';

FLUSH PRIVILEGES;
```

Jenkins 설치

```
// jnekins image pull
$ sudo docker pull jenkins/jenkins:lts

// image 확인 (아래 사진)
$ sudo docker images

//8080 포트 열기
$ sudo ufw allow 8080

// docker in docker를 해결하기 위해 컨테이너를 삭제하고 다시 시도했음
$ sudo docker run --name jenkins -d -p 8080:8080 -p 50000:50000 -v
/home/jenkins:/var/jenkins_home -v /var/run/docker.sock:/var/run/docker.sock -e
TZ=Asia/Seoul -u root jenkins/jenkins:lts

// 도커 컨테이너 확인 (아래 사진)
$ sudo docker ps

// 초기 비밀번호 확인 (아래 사진)
$ sudo docker logs jenkins

// 도메인:8080으로 젠킨스 접근
```

Jenkins Pipeline Shell Script

```
pipeline {
    agent any       // 사용 가능한 에이전트에서 이 파이프라인 또는 해당 단계를 실행
    stages {

        stage('Build') {
            steps {
             // gralew이 있어야됨. git clone해서 project를 가져옴.
            sh 'ls -a'
            dir('../backend/alphano/alphano/') {
            // 해당 폴더에서 gradlew 파일에 실행 권한 부여
            sh 'pwd'
            sh 'ls -a'
            sh 'chmod +x gradlew'
            // gradlew를 사용하여 빌드 수행
            sh './gradlew --warning-mode=all --stacktrace clean build -x test'
        }
            }
            post {
```

```
            success {
                echo 'gradle build success'
            }

            failure {
                echo 'gradle build failed'
            }
        }
    }
    stage('Test') {
        steps {
            echo  '테스트 단계와 관련된 몇 가지 단계를 수행합니다.'
        }
    }
    stage('Prune Docker data') {
        steps {
            sh 'echo "Prune Docker data"'
            sh 'docker system prune -a --volumes -f'
        }

        post {
            success {
                sh 'echo "Prune Docker data Success"'
            }
            failure {
                sh 'echo "Prune Docker data Fail"'
            }
        }
    }

    stage('Docker Build'){
        steps{
         sh 'pwd'

        dir('../backend/alphano/alphano/') {
            sh 'pwd'
            sh 'echo " Image Bulid Start"'
            sh 'ls -a'
            sh 'docker stop backend || true'
            sh 'docker rm backend || true'
            sh 'docker image rm backend || true'
            sh 'docker build -t backend .'

            sh 'docker run -d --name backend -p 8081:8081 backend'
        }
        }
        post {
            success {
                sh 'echo "Bulid Docker Image Success"'
            }

            failure {
                sh 'echo "Bulid Docker Image Fail"'
            }
```

```
            }

        }

    }
}
```

Nginx 설치

NginX 설치
sudo apt-get -y install nginx

(SSL 설정) CertBot 다운
sudo snap install --classic certbot

sudo certbot --nginx -d k9c106.p.ssafy.io

proxy 설정

sudo vi /etc/nginx/sites-enabled/default (파일 들어가서)

코드 추가

```
location / {
        proxy_pass http://localhost:3000;
        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;
    }

    location /api {
                proxy_pass http://k9c106.p.ssafy.io:8081; # 자신의 springboot
app이사용하는 포트
                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;
                }

        location /swagger {
                proxy_pass http://k9c106.p.ssafy.io:8081/swagger; # 자신의 springboot
app이사용하는 포트
                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;
                }

server {
```

```
 if ($host = k9c106.p.ssafy.io) {
        return 301 http://$host$request_uri;
    } # managed by Certbot


       listen 80 ;
       listen [::]:80 ;
    server_name k9c106.p.ssafy.io;
return 404; # managed by Certbot

}
```

설정 적용 후 재시작

sudo service nginx restart

``