1. 프로젝트 환경설정

#1.인강/JPA활용편/querydsl/강의

- /프로젝트 생성
- /Querydsl 설정과 검증
- /라이브러리 살펴보기
- /H2 데이터베이스 설치
- /스프링 부트 설정 - JPA, DB

프로젝트 생성

- 스프링 부트 스타터(https://start.spring.io/)
- Project: **Gradle - Groovy** Project
- 사용 기능: Spring Web, jpa, h2, lombok
  - SpringBootVersion: **3.x.x**
  - groupId: study
  - artifactId: querydsl

주의! - 스프링 부트 3.x 버전 선택 필수

start.spring.io 사이트에서 스프링 부트 2.x에 대한 지원이 종료되어서 더는 선택할 수 없습니다.
이제는 스프링 부트 3.0 이상을 선택해주세요.

스프링 부트 3.0을 선택하게 되면 다음 부분을 꼭 확인해주세요.

- **1. Java 17 이상**을 사용해야 합니다.
- **2. javax 패키지 이름을 jakarta로 변경**해야 합니다.
  - 오라클과 자바 라이센스 문제로 모든 `javax` 패키지를 `jakarta`로 변경하기로 했습니다.
- **3. H2 데이터베이스를 2.1.214 버전 이상 사용해주세요.**

패키지 이름 변경 예)

- **JPA 애노테이션**
  - javax.persistence.Entity ➜ jakarta.persistence.Entity
- **스프링에서 자주 사용하는 @PostConstruct 애노테이션**
  - javax.annotation.PostConstruct ➜ jakarta.annotation.PostConstruct
- **스프링에서 자주 사용하는 검증 애노테이션**

- javax.validation ➜ jakarta.validation

스프링 부트 3.0 관련 설명 방법은 다음 링크를 확인해주세요: https://bit.ly/springboot3

Gradle 전체 설정 - 스프링 부트 3.x

```
plugins {
    id 'java'
    id 'org.springframework.boot' version '3.2.0'
    id 'io.spring.dependency-management' version '1.1.4'
}

group = 'study'
version = '0.0.1-SNAPSHOT'
sourceCompatibility = '17'

configurations {
    compileOnly {
        extendsFrom annotationProcessor
    }
}

repositories {
    mavenCentral()
}

dependencies {
    implementation 'org.springframework.boot:spring-boot-starter-data-jpa'
    implementation 'org.springframework.boot:spring-boot-starter-web'
    implementation 'com.github.gavlyukovskiy:p6spy-spring-boot-starter:1.9.0'
    compileOnly 'org.projectlombok:lombok'
    runtimeOnly 'com.h2database:h2'
    annotationProcessor 'org.projectlombok:lombok'
    testImplementation 'org.springframework.boot:spring-boot-starter-test'

    //test 롬복 사용
    testCompileOnly 'org.projectlombok:lombok'
    testAnnotationProcessor 'org.projectlombok:lombok'
}

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

주의!

Querydsl 스프링 부트 3.0 관련 설정 방법은 다음 링크를 확인해주세요: https://bit.ly/springboot3

Gradle 전체 설정 - 스프링 부트 2.x

```
plugins {
    id 'org.springframework.boot' version '2.2.2.RELEASE'
    id 'io.spring.dependency-management' version '1.0.8.RELEASE'
    id 'java'
}

group = 'study'
version = '0.0.1-SNAPSHOT'
sourceCompatibility = '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-web'
    compileOnly 'org.projectlombok:lombok'
    runtimeOnly 'com.h2database:h2'
    annotationProcessor 'org.projectlombok:lombok'
    testImplementation('org.springframework.boot:spring-boot-starter-test') {
        exclude group: 'org.junit.vintage', module: 'junit-vintage-engine'
    }
}

test {
    useJUnitPlatform()
}
```

- 동작 확인
  - 기본 테스트 케이스 실행
  - 스프링 부트 메인 실행 후 에러페이지로 간단하게 동작 확인(http://localhost:8080)
  - 테스트 컨트롤러를 만들어서 spring web 동작 확인(http://localhost:8080/hello)

테스트 컨트롤러

```
package study.querydsl.controller;

import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;

@RestController
public class HelloController {

    @GetMapping("/hello")
    public String hello() {
        return "hello!";
    }
}
```

IntelliJ Gradle 대신에 자바로 바로 실행하기

최근 IntelliJ 버전은 Gradle로 실행을 하는 것이 기본 설정이다. 이렇게 하면 실행속도가 느리다. 다음과 같이 변경하면 자바로 바로 실행하므로 좀 더 빨라진다.

1. Preferences ➜ Build, Execution, Deployment ➜ Build Tools ➜ Gradle
2. Build and run using: Gradle ➜ IntelliJ IDEA
3. Run tests using: Gradle ➜ IntelliJ IDEA

롬복 적용

1. Preferences ➜ plugin ➜ lombok 검색 실행 (재시작)
2. Preferences ➜ Annotation Processors 검색 ➜ Enable annotation processing 체크 (재시작)
3. 임의의 테스트 클래스를 만들고 @Getter, @Setter 확인

Querydsl 설정과 검증

`build.gradle`에 주석을 참고해서 querydsl 설정 추가

Gradle 전체 설정 - 스프링 부트 3.x

```
plugins {
    id 'java'
    id 'org.springframework.boot' version '3.2.0'
    id 'io.spring.dependency-management' version '1.1.4'
}

group = 'study'
version = '0.0.1-SNAPSHOT'
sourceCompatibility = '17'

configurations {
    compileOnly {
        extendsFrom annotationProcessor
    }
}

repositories {
    mavenCentral()
}

dependencies {
    implementation 'org.springframework.boot:spring-boot-starter-data-jpa'
    implementation 'org.springframework.boot:spring-boot-starter-web'
    implementation 'com.github.gavlyukovskiy:p6spy-spring-boot-starter:1.9.0'
    compileOnly 'org.projectlombok:lombok'
    runtimeOnly 'com.h2database:h2'
    annotationProcessor 'org.projectlombok:lombok'
    testImplementation 'org.springframework.boot:spring-boot-starter-test'

    //test 롬복 사용
    testCompileOnly 'org.projectlombok:lombok'
    testAnnotationProcessor 'org.projectlombok:lombok'

    //Querydsl 추가
    implementation 'com.querydsl:querydsl-jpa:5.0.0:jakarta'
    annotationProcessor "com.querydsl:querydsl-apt:$
{dependencyManagement.importedProperties['querydsl.version']}:jakarta"
    annotationProcessor "jakarta.annotation:jakarta.annotation-api"
```

```
    annotationProcessor "jakarta.persistence:jakarta.persistence-api"
}

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

clean {
    delete file('src/main/generated')
}
```

주의!

Querydsl 스프링 부트 3.0 관련 설정 방법은 다음 링크를 확인해주세요: https://bit.ly/springboot3

Gradle 전체 설정 - 스프링 부트 2.x

```
plugins {
    id 'org.springframework.boot' version '2.2.2.RELEASE'
    id 'io.spring.dependency-management' version '1.0.8.RELEASE'
    //querydsl 추가
    id "com.ewerk.gradle.plugins.querydsl" version "1.0.10"
    id 'java'
}

group = 'study'
version = '0.0.1-SNAPSHOT'
sourceCompatibility = '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-web'

    //querydsl 추가
```

```
    implementation 'com.querydsl:querydsl-jpa'

    compileOnly 'org.projectlombok:lombok'
    runtimeOnly 'com.h2database:h2'
    annotationProcessor 'org.projectlombok:lombok'
    testImplementation('org.springframework.boot:spring-boot-starter-test') {
        exclude group: 'org.junit.vintage', module: 'junit-vintage-engine'
    }
}

test {
    useJUnitPlatform()
}

//querydsl 추가 시작
def querydslDir = "$buildDir/generated/querydsl"

querydsl {
    jpa = true
    querydslSourcesDir = querydslDir
}
sourceSets {
    main.java.srcDir querydslDir
}
configurations {
    querydsl.extendsFrom compileClasspath
}
compileQuerydsl {
    options.annotationProcessorPath = configurations.querydsl
}
//querydsl 추가 끝
```

Querydsl 환경설정 검증

검증용 엔티티 생성

```
package study.querydsl.entity;

import lombok.Getter;
import lombok.Setter;

import javax.persistence.Entity;
```

```
import javax.persistence.GeneratedValue;
import javax.persistence.Id;

@Entity
@Getter @Setter
public class Hello {

    @Id @GeneratedValue
    private Long id;
}
```

검증용 Q 타입 생성

Gradle IntelliJ 사용법

- Gradle ➜ Tasks ➜ build ➜ clean
- Gradle ➜ Tasks ➜ other ➜ compileQuerydsl

Gradle 콘솔 사용법

- ./gradlew clean compileQuerydsl

Q 타입 생성 확인

- build ➜ generated ➜ querydsl
  - study.querydsl.entity.QHello.java 파일이 생성되어 있어야 함

> 참고: Q타입은 컴파일 시점에 자동 생성되므로 버전관리(GIT)에 포함하지 않는 것이 좋다. 앞서 설정에서 생성 위치를 gradle build 폴더 아래 생성되도록 했기 때문에 이 부분도 자연스럽게 해결된다. (대부분 gradle build 폴더를 git에 포함하지 않는다.)

테스트 케이스로 실행 검증

```
package study.querydsl;

import com.querydsl.jpa.impl.JPAQueryFactory;
import org.assertj.core.api.Assertions;
import org.junit.jupiter.api.Test;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.boot.test.context.SpringBootTest;
import org.springframework.test.annotation.Commit;
import org.springframework.transaction.annotation.Transactional;
import study.querydsl.entity.Hello;
```

```java
import study.querydsl.entity.QHello;

import javax.persistence.EntityManager;
import java.util.List;

@SpringBootTest
@Transactional
class QuerydslApplicationTests {

    @Autowired
    EntityManager em;

    @Test
    void contextLoads() {

        Hello hello = new Hello();
        em.persist(hello);

        JPAQueryFactory query = new JPAQueryFactory(em);
        QHello qHello = QHello.hello; //Querydsl Q타입 동작 확인

        Hello result = query
                .selectFrom(qHello)
                .fetchOne();

        Assertions.assertThat(result).isEqualTo(hello);
        //lombok 동작 확인 (hello.getId())
        Assertions.assertThat(result.getId()).isEqualTo(hello.getId());
    }
}
```

- Querydsl Q타입이 정상 동작하는가?
- lombok이 정상 동작 하는가?

> 참고: 스프링 부트에 아무런 설정도 하지 않으면 h2 DB를 메모리 모드로 JVM안에서 실행한다.

라이브러리 살펴보기

gradle 의존관계 보기

```
./gradlew dependencies --configuration compileClasspath
```

Querydsl 라이브러리 살펴보기

- querydsl-apt: Querydsl 관련 코드 생성 기능 제공
- querydsl-jpa: querydsl 라이브러리

스프링 부트 라이브러리 살펴보기

- spring-boot-starter-web
  - spring-boot-starter-tomcat: 톰캣 (웹서버)
  - spring-webmvc: 스프링 웹 MVC
- spring-boot-starter-data-jpa
  - spring-boot-starter-aop
  - spring-boot-starter-jdbc
    - HikariCP 커넥션 풀 (부트 2.0 기본)
  - hibernate + JPA: 하이버네이트 + JPA
  - spring-data-jpa: 스프링 데이터 JPA
- spring-boot-starter(공통): 스프링 부트 + 스프링 코어 + 로깅
  - spring-boot
    - spring-core
  - spring-boot-starter-logging
    - logback, slf4j

테스트 라이브러리

- spring-boot-starter-test
  - junit: 테스트 프레임워크, 스프링 부트 2.2부터 junit5(`jupiter`) 사용
    - 과거 버전은 `vintage`
  - mockito: 목 라이브러리
  - assertj: 테스트 코드를 좀 더 편하게 작성하게 도와주는 라이브러리
    - https://joel-costigliola.github.io/assertj/index.html
  - spring-test: 스프링 통합 테스트 지원
- 핵심 라이브러리
  - 스프링 MVC
  - JPA, 하이버네이트
  - 스프링 데이터 JPA
  - Querydsl

- 기타 라이브러리
  - H2 데이터베이스 클라이언트
  - 커넥션 풀: 부트 기본은 HikariCP
  - 로깅 SLF4J & LogBack
  - 테스트

H2 데이터베이스 설치

개발이나 테스트 용도로 가볍고 편리한 DB, 웹 화면 제공

> **주의! 스프링 부트 3.0 이상을 사용한다면 H2 데이터베이스를 2.1.214 버전 이상으로 사용해주세요.**

- https://www.h2database.com
- 다운로드 및 설치
  - 스프링 부트 2.x를 사용하면 **1.4.200 버전**을 다운로드 받으면 된다.
  - 스프링 부트 3.x를 사용하면 **2.1.214 버전 이상** 사용해야 한다.
  - h2 데이터베이스 버전은 스프링 부트 버전에 맞춘다.
- 권한 주기: `chmod 755 h2.sh`
- 데이터베이스 파일 생성 방법
  - `jdbc:h2:~/querydsl` (최소 한번)
  - `~/querydsl.mv.db` 파일 생성 확인
  - 이후 부터는 `jdbc:h2:tcp://localhost/~/querydsl` 이렇게 접속

> **주의**: H2 데이터베이스의 MVCC 옵션은 H2 1.4.198 버전부터 제거되었습니다. **최신 버전에서는 MVCC 옵션을 사용하면 오류가 발생**합니다. 영상에서 나오는 MVCC 옵션은 제거해주세요.

스프링 부트 설정 - JPA, DB

`application.yml`

```
spring:
```

```
  datasource:
    url: jdbc:h2:tcp://localhost/~/querydsl
    username: sa
    password:
    driver-class-name: org.h2.Driver

  jpa:
    hibernate:
      ddl-auto: create
    properties:
      hibernate:
#        show_sql: true
        format_sql: true

logging.level:
  org.hibernate.SQL: debug
#  org.hibernate.type: trace
```

- spring.jpa.hibernate.ddl-auto: create
  - 이 옵션은 애플리케이션 실행 시점에 테이블을 drop 하고, 다시 생성한다.

> 참고: 모든 로그 출력은 가급적 로거를 통해 남겨야 한다.
> `show_sql` : 옵션은 `System.out` 에 하이버네이트 실행 SQL을 남긴다.
> `org.hibernate.SQL` : 옵션은 logger를 통해 하이버네이트 실행 SQL을 남긴다.

쿼리 파라미터 로그 남기기

- 로그에 다음을 추가하기 `org.hibernate.type` : SQL 실행 파라미터를 로그로 남긴다.
- 외부 라이브러리 사용
  - https://github.com/gavlyukovskiy/spring-boot-data-source-decorator

스프링 부트를 사용하면 이 라이브러리만 추가하면 된다.

```
implementation 'com.github.gavlyukovskiy:p6spy-spring-boot-starter:1.5.8'
```

> 참고: 쿼리 파라미터를 로그로 남기는 외부 라이브러리는 시스템 자원을 사용하므로, 개발 단계에서는 편하게 사용해도 된다. 하지만 운영시스템에 적용하려면 꼭 성능테스트를 하고 사용하는 것이 좋다.

쿼리 파라미터 로그 남기기 - 스프링 부트 3.0

스프링 부트 3.0 이상을 사용하면 라이브러리 버전을 1.9.0 이상을 사용해야 한다.

```
implementation 'com.github.gavlyukovskiy:p6spy-spring-boot-starter:1.9.0'
```