

1. 프로젝트 환경설정

#1.인강/10.jpa활용편/활용편1/강의

- /프로젝트 생성
- /라이브러리 살펴보기
- /View 환경 설정
- /H2 데이터베이스 설치
- /JPA와 DB 설정, 동작확인

프로젝트 생성

- 스프링 부트 스타터(<https://start.spring.io/>)
- Project: **Gradle - Groovy** Project
- 사용 기능: web, thymeleaf, jpa, h2, lombok, **validation(추가)**
 - groupId: jpabook
 - artifactId: jpashop

주의! - 스프링 부트 3.x 버전 선택 필수

start.spring.io 사이트에서 스프링 부트 2.x에 대한 지원이 종료되어서 더는 선택할 수 없다.

이제는 스프링 부트 3.0 이상을 선택해야한다.

- 스프링 부트 버전은 3.x.x 버전 중에 최신 버전을 선택한다.
- 자바 버전은 17 또는 21을 선택한다.
- Validation (JSR-303 validation with Hibernate validator) 모듈을 꼭! 추가해야한다. (영상에 없음)

필독!

- 강의 영상은 스프링 부트 2.x를 기준으로 되어 있다.
- 대신에 강의 메뉴얼은 스프링 부트 최신 버전(3.x)으로 계속 업데이트 하고 있다.
- 이제는 스프링 부트 3.x를 사용해야 한다. 따라서 강의 영상을 보면서 제공되는 메뉴얼도 함께 보아야 한다.

필독! 주의!

잘 안되면 다음에 나오는 `build.gradle` 파일을 그대로 복사해서 사용하자. 강의 영상과 차이가 있다.

- 스프링 부트 버전이 2.1.x → 3.x.x로 업그레이드 되었다.
- validation 모듈이 추가되었다.
- 자바 버전이 1.8 → 17로 업그레이드 되었다.

build.gradle Gradle 전체 설정

```
plugins {  
    id 'java'  
    id 'org.springframework.boot' version '3.4.4'  
    id 'io.spring.dependency-management' version '1.1.7'  
}  
  
group = 'jpabook'  
version = '0.0.1-SNAPSHOT'  
  
java {  
    toolchain {  
        languageVersion = JavaLanguageVersion.of(17)  
    }  
}  
  
configurations {  
    compileOnly {  
        extendsFrom annotationProcessor  
    }  
}  
  
repositories {  
    mavenCentral()  
}  
  
dependencies {  
    implementation 'org.springframework.boot:spring-boot-starter-data-jpa'  
    implementation 'org.springframework.boot:spring-boot-starter-thymeleaf'  
    implementation 'org.springframework.boot:spring-boot-starter-validation'  
    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'  
    testRuntimeOnly 'org.junit.platform:junit-platform-launcher'  
}  
  
tasks.named('test') {  
    useJUnitPlatform()  
}
```

동작 확인

- 기본 테스트 케이스 실행
- 스프링 부트 메인 실행 후 에러페이지로 간단하게 동작 확인(<http://localhost:8080>)

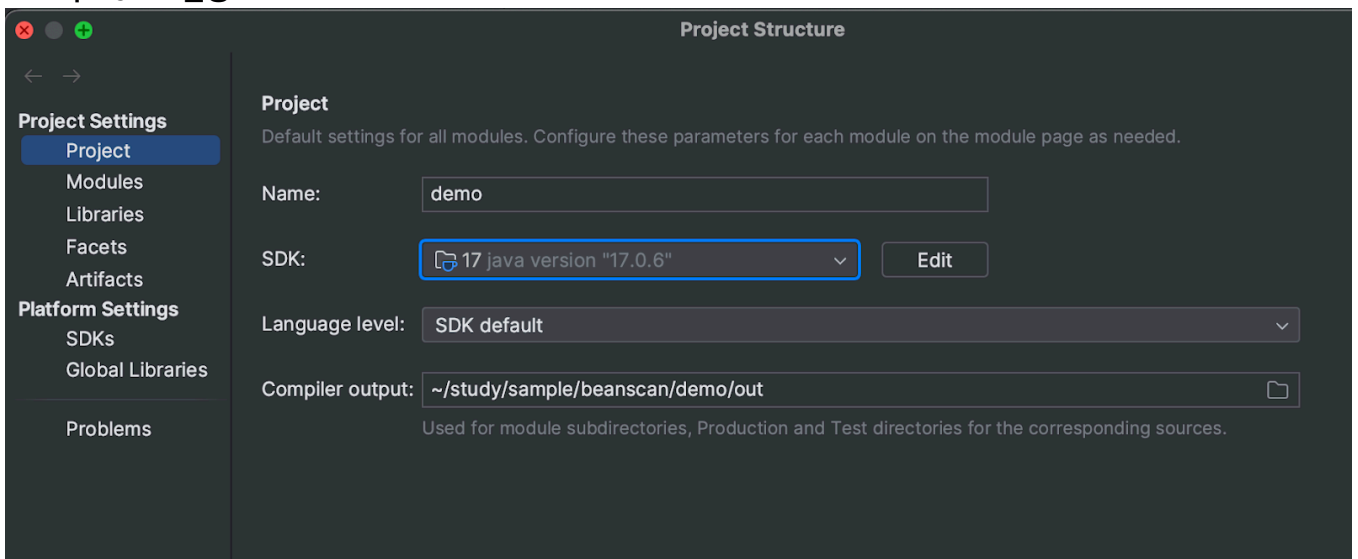
로복 적용

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

주의! 스프링 부트 3.2부터 JDK 17과 Gradle 옵션을 선택하자.

스프링 부트 3.2부터 JDK 17 이상과, 빌드시 IntelliJ IDEA가 아니라 Gradle을 선택해야 한다.

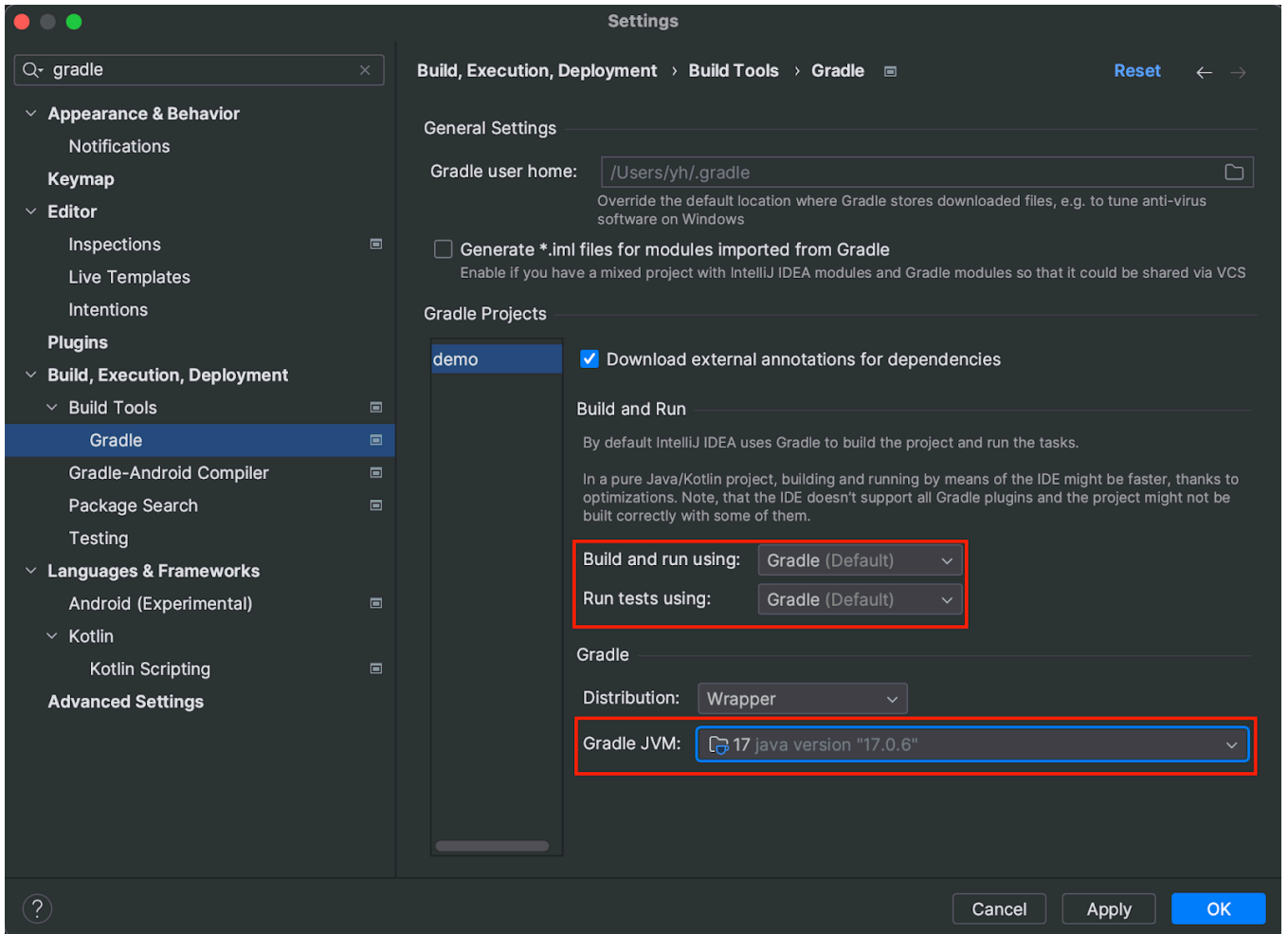
프로젝트 JDK 설정



다음으로 이동합니다.

- Windows: File → Project Structure(Ctrl+Alt+Shift+S)
- Mac: File → Project Structure (⌘;))

파란 박스의 JDK를 내가 새로 설치한 **자바 17로 지정**해줍니다.



다음으로 이동합니다.

- Windows: File → Settings(Ctrl+Alt+S)
- Mac: IntelliJ IDEA | Preferences(⌘,))
- Preferences → Build, Execution, Deployment → Build Tools → Gradle
- 빨간색 박스의 **Build and run using**를 **Gradle**로 선택합니다.
- 빨간색 박스의 **Build tests using**를 **Gradle**로 선택합니다.

빨간색 박스 Gradle JVM을 새로 설치한 자바 17또는 그 이상으로 지정합니다.

주의! 반드시 **Gradle**로 선택해야 한다! IntelliJ IDEA로 선택하면 이후에 문제가 될 수 있다.

라이브러리 살펴보기

gradle 의존관계 보기

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

스프링 부트 라이브러리 살펴보기

- spring-boot-starter-web
 - spring-boot-starter-tomcat: 톰캣 (웹서버)
 - spring-webmvc: 스프링 웹 MVC
- spring-boot-starter-thymeleaf: 타임리프 템플릿 엔진(View)
- 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
 - mockito: 목 라이브러리
 - assertj: 테스트 코드를 좀 더 편하게 작성하게 도와주는 라이브러리
 - spring-test: 스프링 통합 테스트 지원
- 핵심 라이브러리
 - 스프링 MVC
 - 스프링 ORM
 - JPA, 하이버네이트
 - 스프링 데이터 JPA
- 기타 라이브러리
 - H2 데이터베이스 클라이언트
 - 커넥션 풀: 부트 기본은 HikariCP
 - WEB(thymeleaf)
 - 로깅 SLF4J & LogBack
 - 테스트

참고: 스프링 데이터 JPA는 스프링과 JPA를 먼저 이해하고 사용해야 하는 응용기술이다.

View 환경 설정

thymeleaf 템플릿 엔진

- thymeleaf 공식 사이트: <https://www.thymeleaf.org/>
- 스프링 공식 튜토리얼: <https://spring.io/guides/gs/serving-web-content/>
- 스프링부트 메뉴얼: <https://docs.spring.io/spring-boot/reference/web/servlet.html#web.servlet.spring-mvc.template-engines>
- 스프링 부트 thymeleaf viewName 매핑
 - `resources:templates/` + {ViewName}+ `.html`

jpabook.jpashop.HelloController

```
package jpabook.jpashop;

import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.GetMapping;

@Controller
public class HelloController {

    @GetMapping("hello")
    public String hello(Model model) {
        model.addAttribute("data", "hello!!");
        return "hello";
    }
}
```

thymeleaf 템플릿엔진 동작 확인(hello.html)

`resources/templates/hello.html`

```
<!DOCTYPE HTML>
<html xmlns:th="http://www.thymeleaf.org">
<head>
  <title>Hello</title>
  <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
</head>
<body>
<p th:text="'안녕하세요. ' + ${data}" >안녕하세요. 손님</p>
</body>
</html>
```

index.html 하나 만들기

- static/index.html

static/index.html

```
<!DOCTYPE HTML>
<html xmlns:th="http://www.thymeleaf.org">
<head>
  <title>Hello</title>
  <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
</head>
<body>
Hello
<a href="/hello">hello</a>
</body>
</html>
```

참고: spring-boot-devtools 라이브러리를 추가하면, html 파일을 컴파일만 해주면 서버 재시작 없이 View 파일 변경이 가능하다.

인텔리J 컴파일 방법: 메뉴 build → Recompile

H2 데이터베이스 설치

개발이나 테스트 용도로 가볍고 편리한 DB, 웹 화면 제공

<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.
- 데이터베이스 파일 생성 방법
 - `jdbc:h2:~/jpashop` (최소 한번)
 - `~/jpashop.mv.db` 파일 생성 확인
 - 이후 부터는 `jdbc:h2:tcp://localhost/~/jpashop` 이렇게 접속

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

JPA와 DB 설정, 동작확인

main/resources/application.yml

```
spring:
  datasource:
    url: jdbc:h2:tcp://localhost/~/jpashop
    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 #스프링 부트 2.x, hibernate5
# org.hibernate.orm.jdbc.bind: trace #스프링 부트 3.x, hibernate6
```

[주의! - 영상과 다른점] ;MVCC=TRUE

데이터베이스 접속 url의 마지막 부분에 붙는 `;MVCC=TRUE` 는 다음과 같이 제거해주세요. 해당 옵션은 제거되었

습니다. 해당 옵션이 있으면 오류가 발생합니다.

- 영상 화면: `jdbc:h2:tcp://localhost/~/jpashop;MVCC=TRUE`
- 수정 내용: `jdbc:h2:tcp://localhost/~/jpashop`

- `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.

주의! `application.yml` 같은 `yml` 파일은 띄어쓰기(스페이스) 2칸으로 계층을 만든다. 따라서 띄어쓰기 2칸을 필수로 적어주어야 한다.

예를 들어서 아래의 `datasource` 는 `spring:` 하위에 있고 앞에 띄어쓰기 2칸이 있으므로

`spring.datasource` 가 된다. 다음 코드에 주석으로 띄어쓰기를 적어두었다.

yml 띄어쓰기 주의

```
spring: #띄어쓰기 없음
  datasource: #띄어쓰기 2칸
    url: jdbc:h2:tcp://localhost/~/jpashop #4칸
    username: sa
    password:
    driver-class-name: org.h2.Driver

  jpa: #띄어쓰기 2칸
    hibernate: #띄어쓰기 4칸
      ddl-auto: create #띄어쓰기 6칸
    properties: #띄어쓰기 4칸
      hibernate: #띄어쓰기 6칸
#      show_sql: true #띄어쓰기 8칸
      format_sql: true #띄어쓰기 8칸

logging.level: #띄어쓰기 없음
  org.hibernate.SQL: debug #띄어쓰기 2칸
#  org.hibernate.type: trace #띄어쓰기 2칸
```

실제 동작하는지 확인하기

회원 엔티티

```
package jpabook.jpashop;

import jakarta.persistence.Entity;
import jakarta.persistence.GeneratedValue;
import jakarta.persistence.Id;
import lombok.Getter;
import lombok.Setter;

@Entity
@Getter @Setter
public class Member {

    @Id @GeneratedValue
    private Long id;
    private String username;

}
```

[주의! 영상과 다른 점] - jakarta.persistence

패키지 import에서 `javax` 대신에 `jakarta`를 사용해야 한다. (스프링 부트 3.x 이상)

- 영상 화면: `import javax.persistence`
- 수정 내역: `import jakarta.persistence`

회원 리포지토리

```
package jpabook.jpashop;

import jakarta.persistence.EntityManager;
import jakarta.persistence.PersistenceContext;
import org.springframework.stereotype.Repository;

@Repository
public class MemberRepository {

    @PersistenceContext
```

```

EntityManager em;

public Long save(Member member) {
    em.persist(member);
    return member.getId();
}

public Member find(Long id) {
    return em.find(Member.class, id);
}
}

```

[주의! 영상과 다른 점] - jakarta.persistence

패키지 import에서 javax 대신에 jakarta를 사용해야 한다. (스프링 부트 3.x 이상)

- 영상 화면: import javax.persistence
- 수정 내역: import jakarta.persistence

앞으로 이 부분은 반복 설명하지 않겠다. javax 대신에 jakarta를 사용하자.

테스트

```

package jpabook.jpashop;

import jakarta.transaction.Transactional;
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.Rollback;

@SpringBootTest
public class MemberRepositoryTest {

    @Autowired
    MemberRepository memberRepository;

    @Test
    @Transactional
    @Rollback(false)
    public void testMember() {

```

```

    Member member = new Member();
    member.setUsername("memberA");
    Long savedId = memberRepository.save(member);

    Member findMember = memberRepository.find(savedId);

    Assertions.assertThat(findMember.getId()).isEqualTo(member.getId());

    Assertions.assertThat(findMember.getUsername()).isEqualTo(member.getUsername());
};

    Assertions.assertThat(findMember).isEqualTo(member); //JPA 엔티티 동일성
보장
    }
}

```

[주의! 영상과 다른 점] - JUnit 버전

강의 영상은 JUnit4를 사용하고, 스프링 부트 3.x 부터는 JUnit5를 사용한다. 따라서 사용하는 패키지가 다르다.

- 영상 화면: `org.junit.Test` 클래스 사용 (JUnit4)
- 수정 내역: `org.junit.jupiter.api.Test` 클래스 사용 (JUnit5)

앞으로 이 부분은 반복 설명하지 않겠다

[주의! 영상과 다른 점] - @RunWith 제거

스프링 부트 3.x 부터는 `@RunWith` 를 사용하지 않는다. 제거하면 된다.

- 영상 화면: `@RunWith(SpringRunner.class)` 사용 (JUnit4에서 필요)
- 수정 내역: `@RunWith(SpringRunner.class)` 사용하지 않음

앞으로 이 부분은 반복 설명하지 않겠다

- Entity, Repository 동작 확인
- jar 빌드해서 동작 확인

참고: 스프링 부트를 통해 복잡한 설정이 대부분 자동화 되었다. `persistence.xml` 도 없고,

`LocalContainerEntityManagerFactoryBean` 도 없다. 스프링 부트를 통한 추가 설정은 스프링 부트 메뉴얼을 참고하고, 스프링 부트를 사용하지 않고 순수 스프링과 JPA 설정 방법은 자바 ORM 표준 JPA 프로그래밍 책을 참고하자.

쿼리 파라미터 로그 남기기

- 로그에 다음을 추가하기: SQL 실행 파라미터를 로그로 남긴다.
- 주의! 스프링 부트 3.x를 사용한다면 영상 내용과 다르기 때문에 다음 내용을 참고하자.
 - 스프링 부트 2.x, hibernate5
 - ◆ `org.hibernate.type: trace`
 - 스프링 부트 3.x, hibernate6
 - ◆ `org.hibernate.orm.jdbc.bind: trace`
- 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:1.10.0'
```

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

[주의! 영상과 다른 점] - p6spy-spring-boot-starter 버전

스프링 부트 3.x 부터는 해당 라이브러리 버전을 최신 버전을 사용해야 한다.

- 영상 화면: 1.5.6
- 수정 내역: 1.10.0