



Spring  
Boot

12

Lombok 및 JPA로 DataBase 사용하기

# 롬복(Lombok) 사용하기

# 롬복 사용하기

3

## ▣ 룸복(Lombok)

- ▣ 자바 클래스를 만들 때 항상 만들게 되는 **Getter, Setter** 메서드 등을 어노테이션을 이용해서 자동으로 만들어주는 유틸리티 라이브러리

## ▣ 룸복을 사용하기 위해

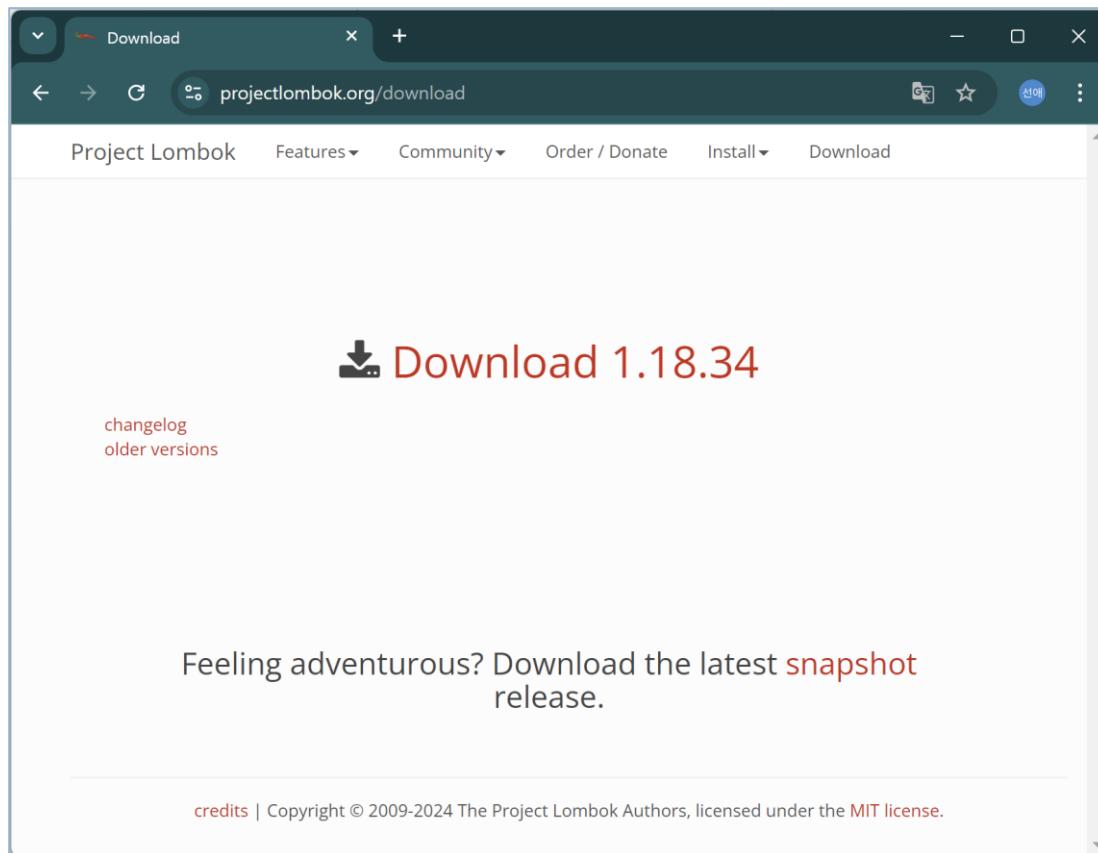
- ▣ 룸복 플러그인 설치
- ▣ 프로젝트 생성 시 디펜던시 추가

# 롬복 사용하기

4

## ▣ 룸복 다운로드

▣ <https://projectlombok.org/download>

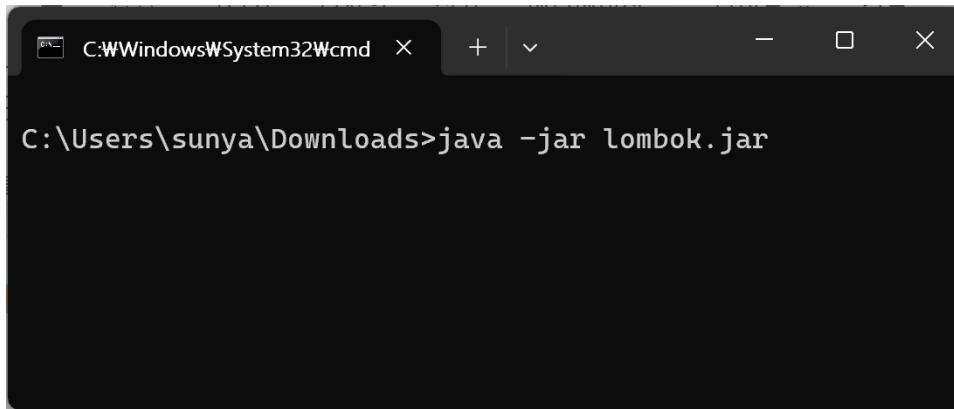


# 롬복 사용하기

5

## ▣ 룸복 설치

- ▣ 명령 프롬프트 창 > **java -jar lombok.jar**



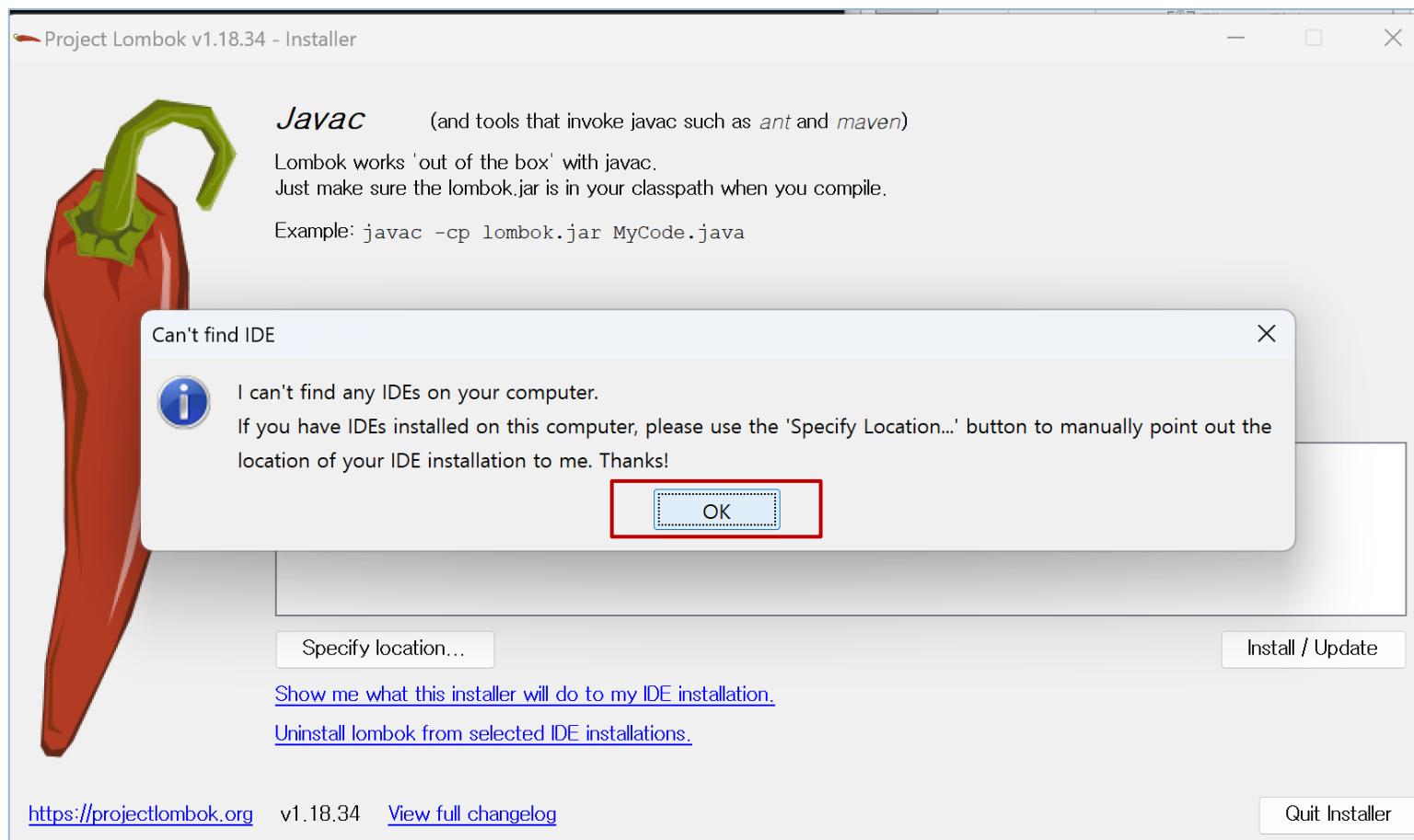
A screenshot of a Windows Command Prompt window titled 'cmd'. The window shows the path 'C:\Windows\System32\cmd'. In the main area, the command 'C:\Users\sunya\Downloads>java -jar lombok.jar' is typed. The window has standard window controls (minimize, maximize, close) at the top.

# 롬복 사용하기

6

## ▣ 룸복 설치

- ▣ 명령 프롬프트 창 > `java -jar lombok.jar`

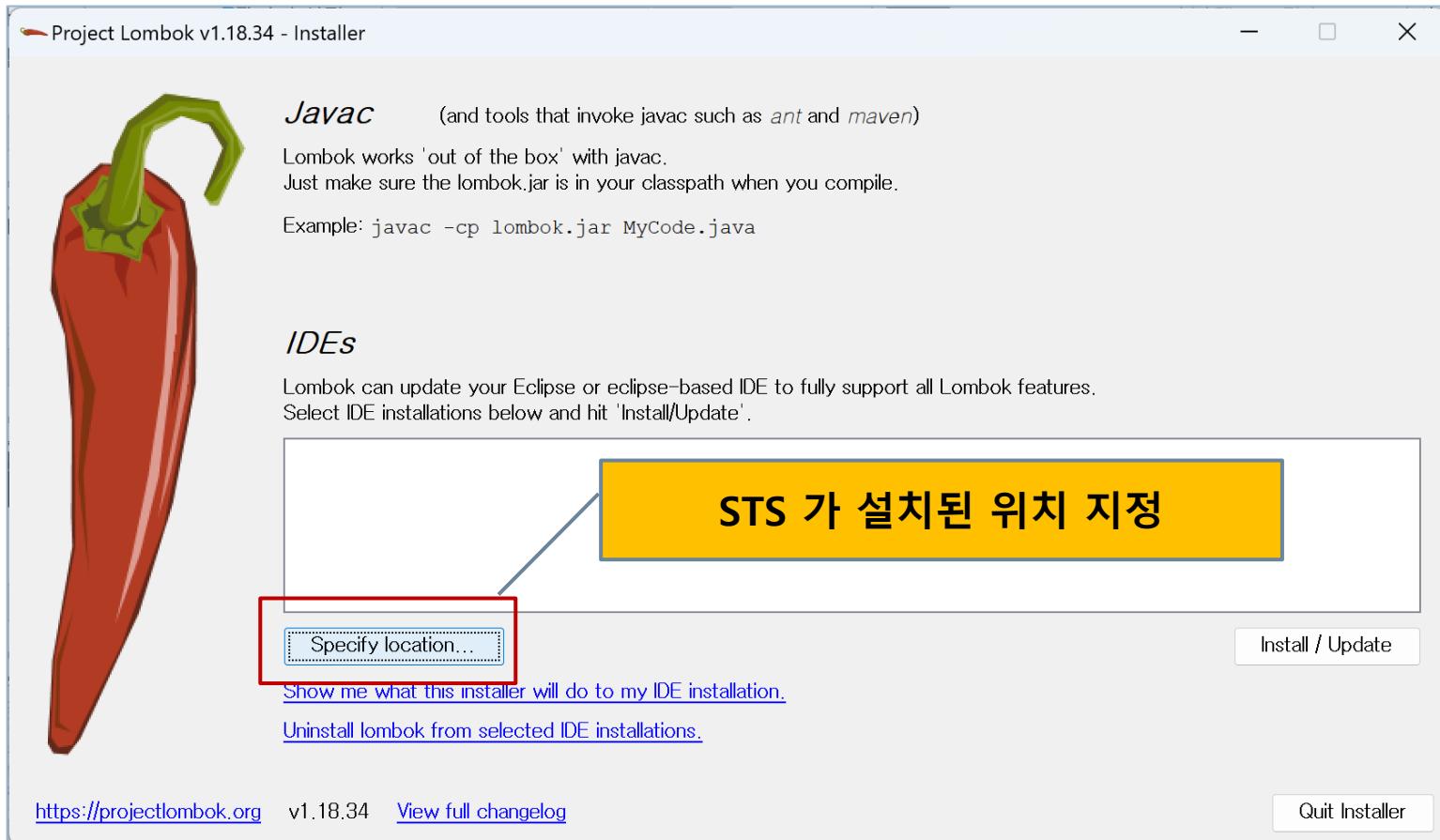


# 롬복 사용하기

7

## ▣ 룸복 설치

- ▣ 명령 프롬프트 창 > `java -jar lombok.jar`

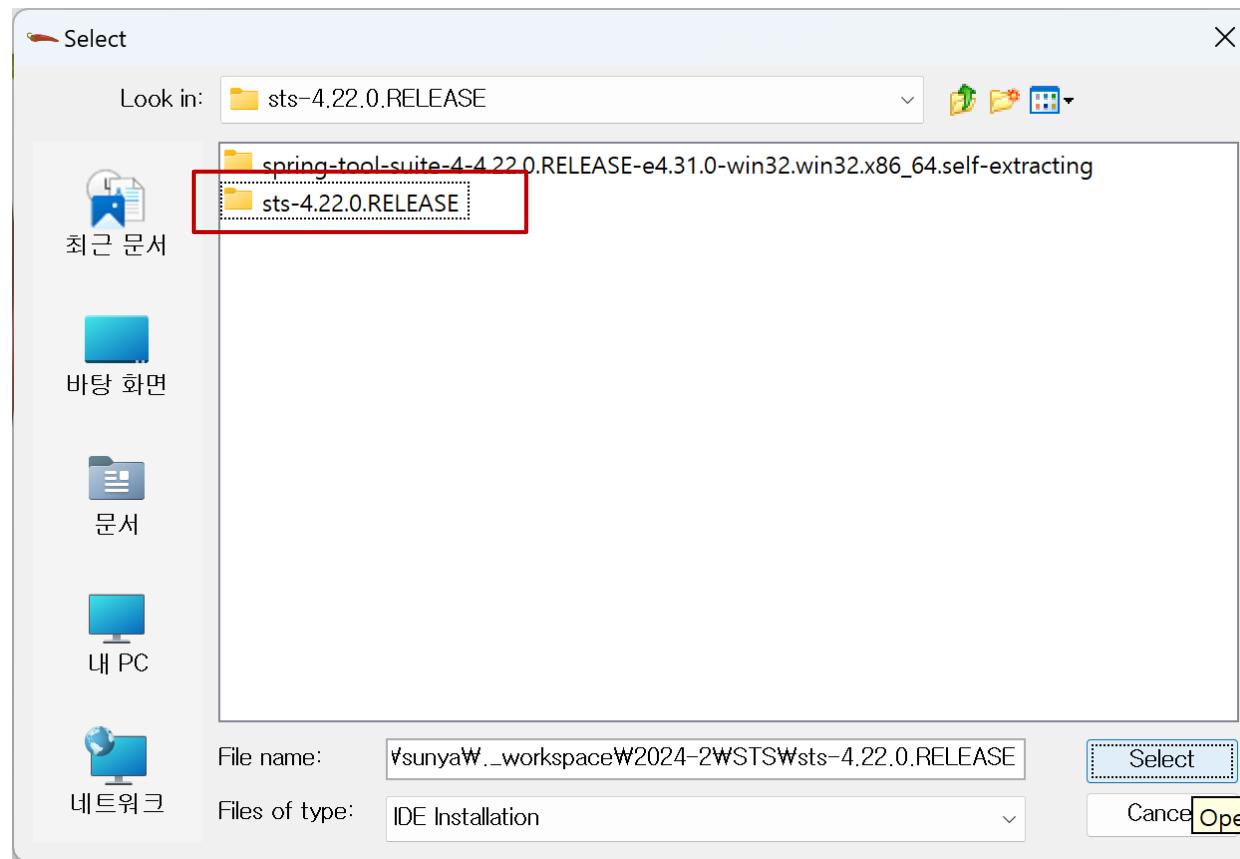


# 롬복 사용하기

8

## ▣ 룸복 설치

- ▣ 명령 프롬프트 창 > java -jar lombok.jar

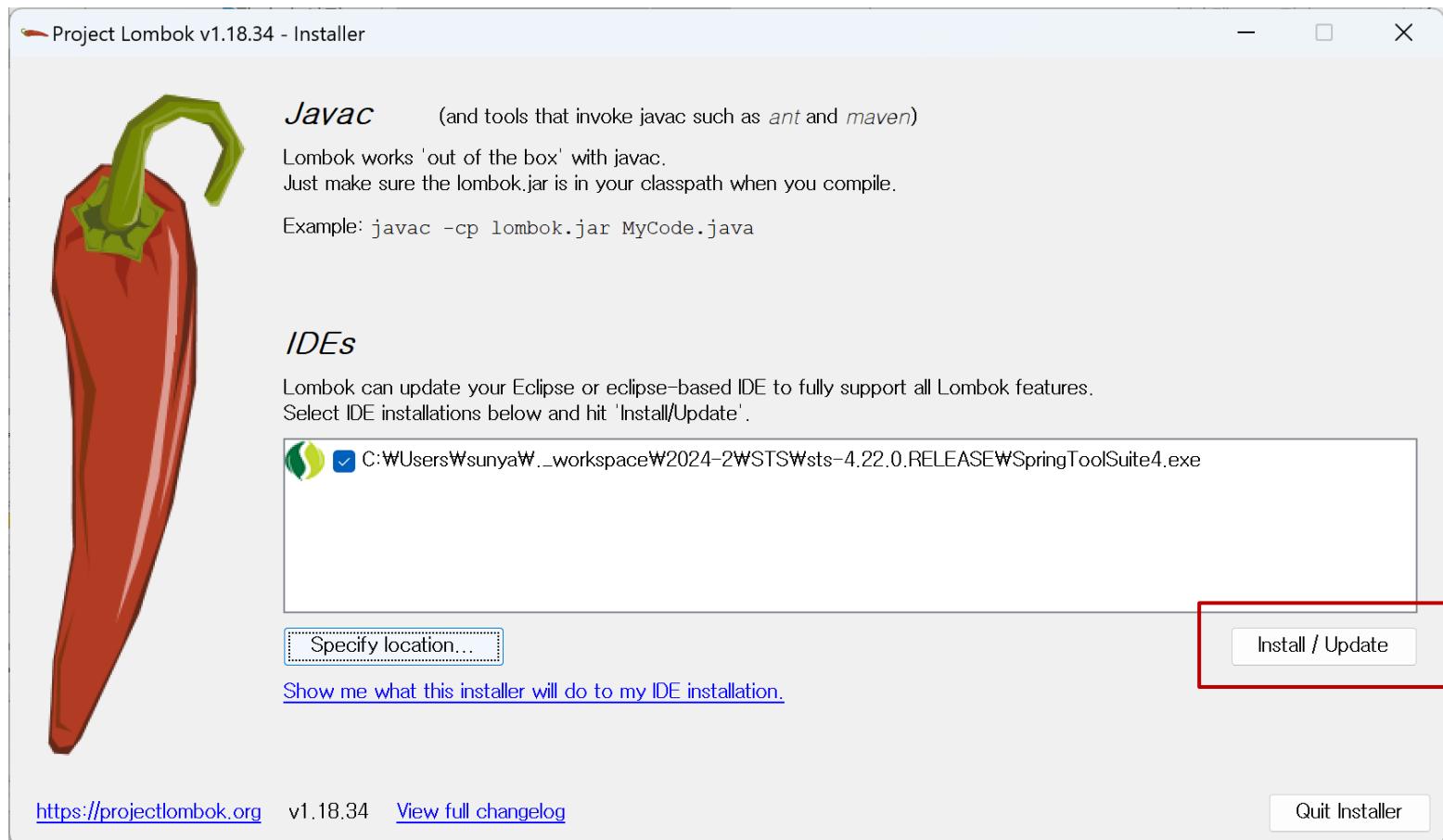


# 롬복 사용하기

9

## ▣ 룸복 설치

- ▣ 명령 프롬프트 창 > java -jar lombok.jar

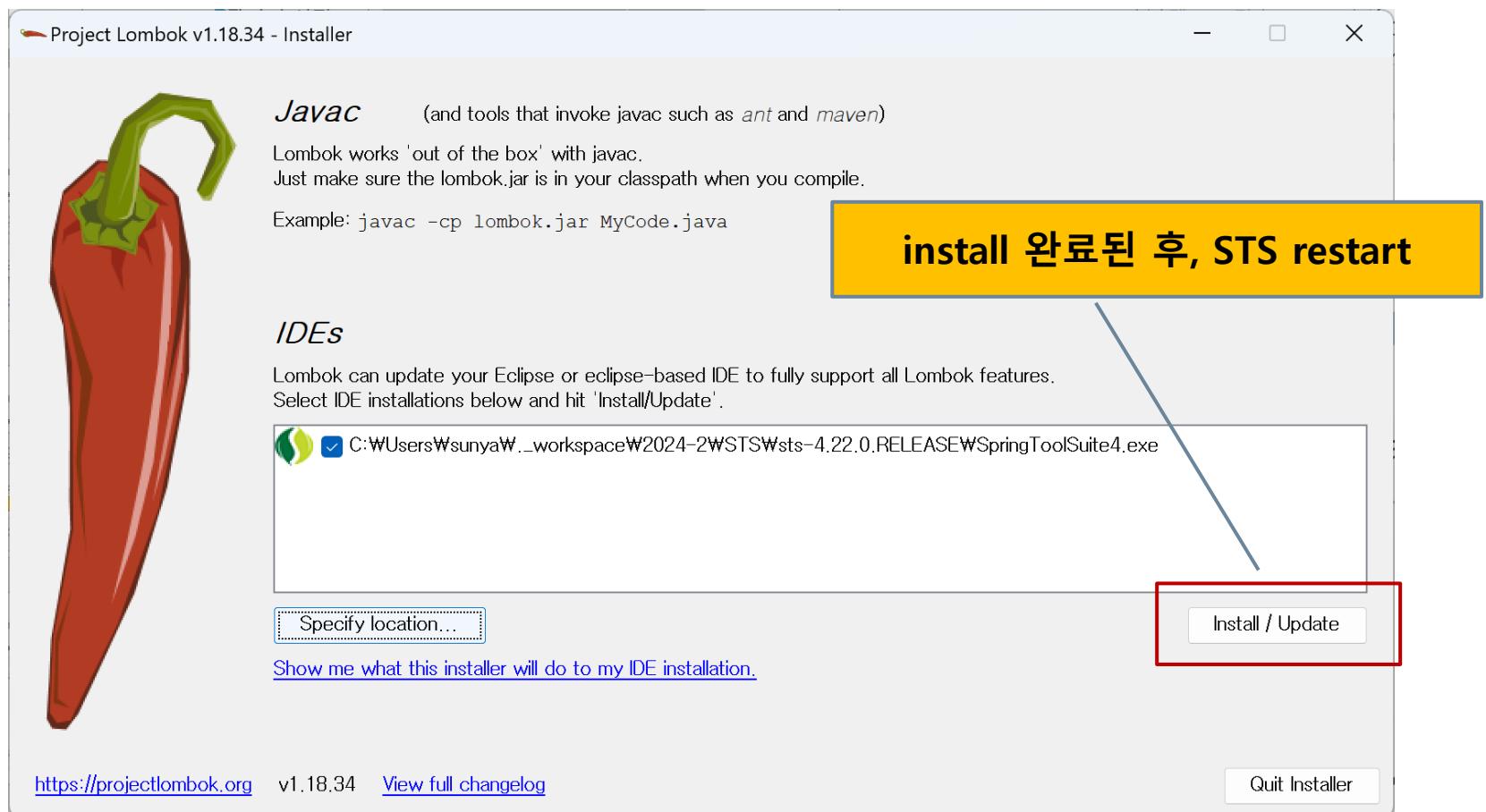


# 롬복 사용하기

10

## ▣ 룸복 설치

- ▣ 명령 프롬프트 창 > java -jar lombok.jar



# 롬복 사용하기

11

- 룸복 사용 프로젝트 생성(week12\_01Lombok)
  - 디펜던시 설정
    - SpringWeb
    - **Lombok** (implementation 'org.projectlombok:lombok' )
  - build.gradle
    - JSP 설정 / 버전 설정
    - Refresh Gradle Project
  - application.properties
    - 포트 설정
    - JSP 파일 경로 설정
  - File > Properties
    - Project Facets > Dynamic Web Module 버전 > 5.0으로 수정
  - JSP 사용을 위한 폴더 구성
    - src/main/webapp/WEB-INF/views

# 롬복 사용하기

12

- 커맨드(Command) 객체 만들기
  - Member 클래스

The screenshot shows an IDE interface with two main panes. On the left is the code editor for `Member.java`, and on the right is the `Outline` view.

**Member.java Content:**

```
1 package com.study.springboot;
2
3 import lombok.Data;
4
5 @Data
6 public class Member {
7     private String id;
8     private String name;
9 }
```

**Outline View:**

- com.study.springboot
- └ Member
  - getId() : String
  - getName() : String
  - setId(String) : void
  - setName(String) : void
  - equals(Object) : boolean
  - hashCode() : int
  - toString() : String
  - Member()
  - id : String
  - name : String

# 롬복 사용하기

13

## □ 리퀘스트 맵핑

### □ MyController

```
MyContoller.java ×
1 package com.study.springboot;
2
3 import org.springframework.stereotype.Controller;
4 import org.springframework.web.bind.annotation.GetMapping;
5 import org.springframework.web.bind.annotation.RequestMapping;
6 import org.springframework.web.bind.annotation.ResponseBody;
7
8 @Controller
9 public class MyContoller {
10    @RequestMapping("/")
11    public @ResponseBody String root() {
12        return "Lombok 사용하기";
13    }
14
15    @GetMapping("/test3")
16    public String test3(Member member) {
17        // 파라미터와 일치하는 빈을 만들어서 사용할 수 있다.
18        // View 페이지에서 model을 사용하지 않고 member를 사용한다.
19        return "test3";
20    }
21 }
```

# 롬복 사용하기

14

## □ 리퀘스트 맵핑

### □ test3.jsp

```
test3.jsp ×
1 <%@ page language="java" contentType="text/html; charset=UTF-8"
2     pageEncoding="UTF-8"%>
3 <!DOCTYPE html>
4<html>
5<head>
6 <meta charset="UTF-8">
7 <title>Insert title here</title>
8 </head>
9<body>
10<%
11     out.println("<h1>#03 : Hello World </h1>");
12 %>
13 <br>
14<h3>
15 당신의 아이디는 ${member.id }입니다. <br>
16 당신의 이름은 ${member.name }입니다.
17 </h3>
18 </body>
19 </html>
```

# JPA로 데이터베이스 사용하기

# JPA로 데이터베이스 사용하기

16

## □ ORM과 JPA 이해하기

### □ **ORM(Object Relational Mapping)**

- Java 문법으로도 데이터베이스를 다룰 수 있는 도구
- SQL을 직접 작성하지 않아도 데이터베이스의 데이터 처리가 가능

### □ **JPA(Java Persistence API)**

- 스프링 부트는 JPA를 사용하여 데이터베이스를 관리
- **스프링 부트는 JPA를 ORM의 기술 표준으로 사용**

# JPA로 데이터베이스 사용하기

17

## □ ORM과 JPA 이해하기

### □ ORM(Object Relational Mapping)

- Java 문법으로도 데이터베이스를 다룰 수 있는 도구
- SQL을 직접 작성하지 않아도 데이터베이스의 데이터 처리가 가능

### □ ORM의 장점

- MySQL, Oracle DB, MS SQL 과 같은 **DBMS의 종류에 관계없이 일관된 자바 코드 사용이 가능**
- **프로그램 유지, 보수, 관리가 편리**
- **코드 내부에서 안정적인 SQL 쿼리문을 생성**
- 개발자가 달라져도 통일된 쿼리문 작성이 가능
- 오류 발생률도 감소

# JPA로 데이터베이스 사용하기

18

## □ ORM과 JPA 이해하기

### □ JPA(Java Persistence API)

- 스프링 부트는 JPA를 사용하여 데이터베이스를 관리
- 스프링 부트는 JPA를 ORM의 기술 표준으로 사용

### □ JPA 특징

- JPA는 인터페이스 모음으로 이 인터페이스를 구현한 실체 클래스가 필요
- 대표적인 JPA 실체 클래스 => 하이버네이트(Hibernate)
- Hibernate
  - JPA의 인터페이스를 구현한 실체 클래스
  - 자바의 ORM 프레임워크로 스프링 부트에서 데이터베이스 관리를 쉽게 하는 도구

# JPA로 데이터베이스 사용하기

19

## □ ORM과 JPA 이해하기

### □ ORM(Object Relational Mapping)

- Java 문법으로도 데이터베이스를 다룰 수 있는 도구
- SQL을 직접 작성하지 않아도 데이터베이스의 데이터 처리가 가능

### □ JPA(Java Persistence API)

**데이터베이스 관리 = JPA + Hibernate**

- 스프링 부트는 JPA를 사용하여 데이터베이스를 관리

- 스프링 부트는 JPA를 ORM의 기술 표준으로 사용

- JPA는 인터페이스 모음으로 이 인터페이스를 구현한 실제 클래스가 필요

- 대표적인 JPA 실제 클래스 => 하이버네이트(Hibernate)

- Hibernate

- JPA의 인터페이스를 구현한 실제 클래스

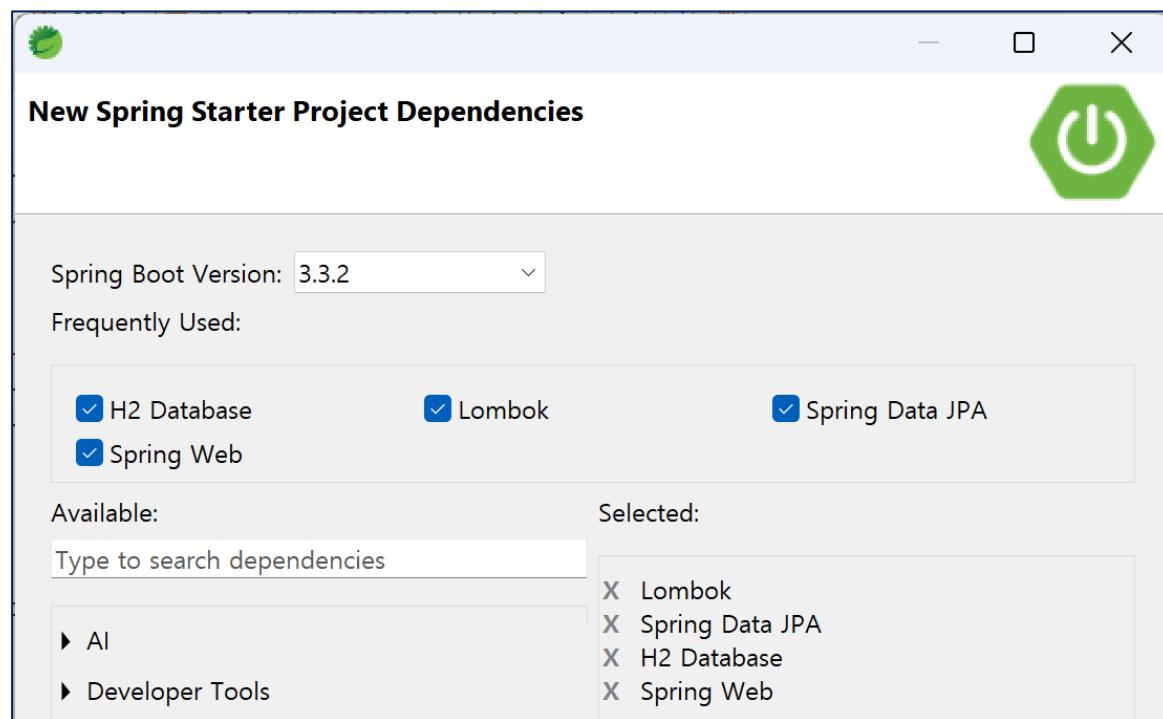
- 자바의 ORM 프레임워크로 스프링 부트에서 데이터베이스 관리를 쉽게하는 도구

# JPA로 데이터베이스 사용하기

20

## □ H2 데이터베이스 사용

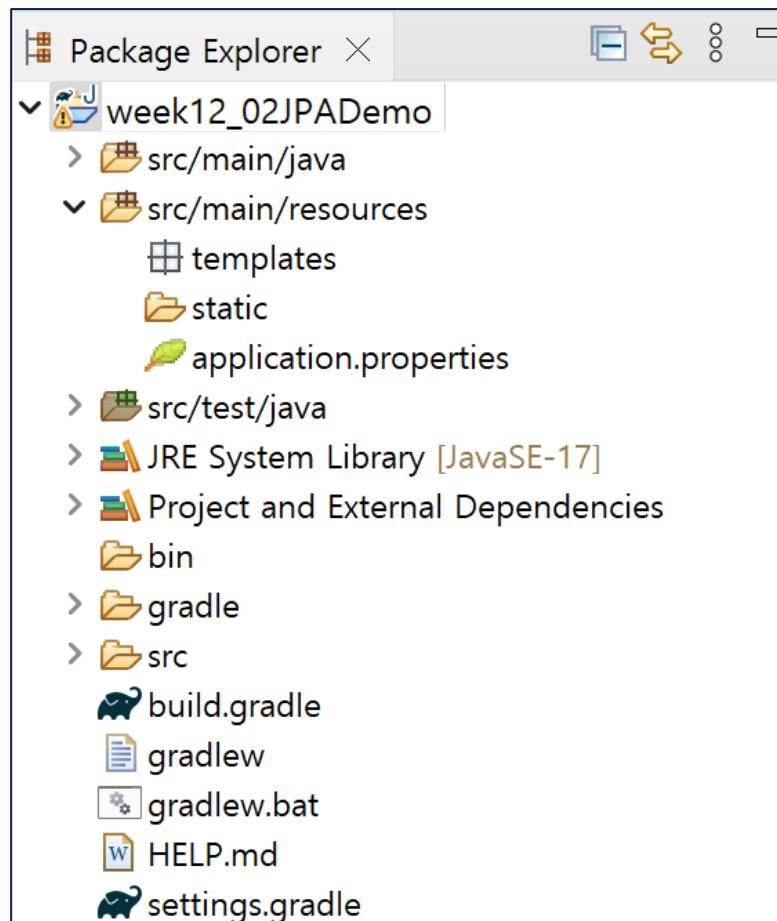
- 주로 개발 환경에서 사용하는 자바 기반의 경량 DBMS
- 스프링 부트 프로젝트 생성(**week12\_02JPADemo**)
- 디펜던시 설정
  - **Spring Web**
  - **Lombok**
  - **Spring Data JPA**
  - **H2 Database**



# JPA로 데이터베이스 사용하기

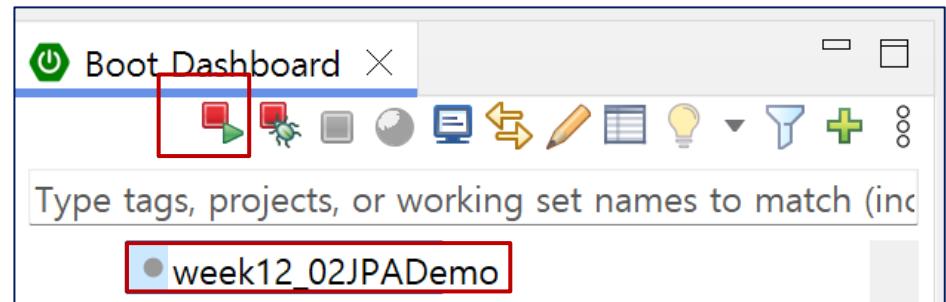
21

- H2 데이터베이스 사용
  - ▣ application.properties



The screenshot shows the Eclipse IDE's editor window displaying the 'application.properties' file. The content is as follows:

```
1 spring.application.name=week12_02JPADemo
2 server.port=8081
3 spring.h2.console.enabled=true
```



# JPA로 데이터베이스 사용하기

22

- H2 데이터베이스 사용
  - ▣ 프로젝트 실행 후 콘솔 창

```
main] c.s.springboot.Ex12JpaDemoApplication      : Starting Ex12JpaDemoApplication using Java 17.0.10 with PID 13324 (C:\Users\sunya\._workspace\2024-2\S
main] c.s.springboot.Ex12JpaDemoApplication      : No active profile set, falling back to 1 default profile: "default"
main] .s.d.r.c.RepositoryConfigurationDelegate   : Bootstrapping Spring Data JPA repositories in DEFAULT mode.
main] .s.d.r.c.RepositoryConfigurationDelegate   : Finished Spring Data repository scanning in 16 ms. Found 0 JPA repository interfaces.
main] o.s.b.w.embedded.tomcat.TomcatWebServer    : Tomcat :
main] o.apache.catalina.core.StandardService     : Starting
main] o.apache.catalina.core.StandardEngine       : Starting
main] o.a.c.c.C.[Tomcat].[localhost].[]          : Initializing Spring embedded WebApplicationContext
main] w.s.c.ServletWebServerApplicationContext     : Root WebApplicationContext: initialization completed in 1306 ms
main] com.zaxxer.hikari.HikariDataSource         : HikariPool-1 - Starting...
main] com.zaxxer.hikari.pool.HikariPool           : HikariPool-1 - Added connection conn0: url=jdbc:h2:mem:1efa9784-88af-4876-95ae-270fffc9e591b user=SA
main] com.zaxxer.hikari.HikariDataSource         : HikariPool-1 - Start completed.
main] o.s.b.a.h2.H2ConsoleAutoConfiguration       : H2 console available at '/h2-console'. Database available at 'jdbc:h2:mem:1efa9784-88af-4876-95ae-270f
main] o.hibernate.jpa.internal.util.LogHelper     : HHH000204: Processing PersistenceUnitInfo [name: default]
main] org.hibernate.Version                      : HHH000412: Hibernate ORM core version 6.5.2.Final
main] o.h.c.internal.RegionFactoryInitiator      : HHH000026: Second-level cache disabled
main] o.s.o.j.p.SpringPersistenceUnitInfo        : No LoadTimeWeaver setup: ignoring JPA class transformer
main] o.h.e.t.j.p.i.JtaPlatformInitiator        : HHH000489: No JTA platform available (set 'hibernate.transaction.jta.platform' to enable JTA platform
main] j.LocalContainerEntityManagerFactoryBean     : Initialized JPA EntityManagerFactory for persistence unit 'default'
main] JpaBaseConfiguration$JpaWebConfiguration     : spring.jpa.open-in-view is enabled by default. Therefore, database queries may be performed during vie
main] o.s.b.w.embedded.tomcat.TomcatWebServer    : Tomcat started on port 8081 (http) with context path '/'
main] c.s.springboot.Ex12JpaDemoApplication      : Started Ex12JpaDemoApplication in 3.659 seconds (process running for 8.982)
```

H2 DB 연결 url => 복사

# JPA로 데이터베이스 사용하기

23

## □ H2 데이터베이스 연결

- 브라우저 창 > **http://localhost:8081/h2-console**
- 복사 한 url 주소 => **JDBC URL** 붙여넣기

The screenshot shows the H2 Console login interface. A red box highlights the 'JDBC URL' field, which contains the value 'jdbc:h2:mem:f7d79527-4384-4a00-9de9-d3b081049466'. Another red box highlights the '연결' (Connect) button at the bottom left.

한국어 설정 도구 도움말

로그인

저장한 설정: Generic H2 (Embedded)

설정 이름: Generic H2 (Embedded) 저장 삭제

드라이버 클래스: org.h2.Driver

JDBC URL: jdbc:h2:mem:f7d79527-4384-4a00-9de9-d3b081049466

사용자명: sa

비밀번호:

연결 연결 시험

# JPA로 데이터베이스 사용하기

24

## □ H2 데이터베이스

The screenshot shows the H2 Console interface running in a browser at `localhost:8081/h2-console/login.do?jsessionid=719c41e7aef04b88a115f75e5b248a9d`. The left sidebar displays the database structure:

- jdbc:h2:~/local
- INFORMATION\_SCHEMA
- 사용자
  - SA
  - Admin

The right panel contains a SQL editor with tabs: 실행 (Run), Run Selected, 자동 완성 (Auto complete), 지우기 (Delete), and SQL 문 (SQL Query). Below the editor is a section titled "중요 명령" (Important Commands) with the following table:

	이 도움말 페이지 보기
	명령 이력 보기
	Ctrl+엔터 현재의 SQL 문 실행
	Shift+엔터 Executes the SQL statement defined by the text selection
	Ctrl+Space 자동 완성
	데이터베이스 연결 끊기

Below this is a section titled "샘플 SQL 스크립트" (Sample SQL Script) with the following table:

테이블이 존재하는 경우 삭제하기	DROP TABLE IF EXISTS TEST;
새 테이블 만들기	CREATE TABLE TEST(ID INT PRIMARY KEY, NAME VARCHAR(255));
컬럼은 ID와 NAME	INSERT INTO TEST VALUES(1, 'Hello');
행 추가	INSERT INTO TEST VALUES(2, 'World');
행 추가	SELECT * FROM TEST ORDER BY ID;
테이블 질의	UPDATE TEST SET NAME='Hi' WHERE ID=1;
행 데이터 변경	

# JPA로 데이터베이스 사용하기

25

## □ H2 데이터베이스

### □ url 고정

The screenshot shows the 'application.properties' file and the 'H2 Console' configuration interface side-by-side.

**application.properties:**

```
1 spring.application.name=week12_02JPADemo
2 server.port=8081
3 spring.h2.console.enabled=true
4 spring.datasource.url= jdbc:h2:mem:f7d79527-4384-4a00-9de9-d3b081049466
```

**H2 Console Configuration:**

- Language: 한국어
- Buttons: 설정, 도구, 도움말
- 로그인** section:
  - 저장한 설정: Generic H2 (Embedded)
  - 설정 이름: Generic H2 (Embedded) [저장, 삭제]
- 드라이버 클래스:** org.h2.Driver
- JDBC URL:** jdbc:h2:mem:f7d79527-4384-4a00-9de9-d3b081049466 (highlighted with a red border)
- 사용자명:** sa
- 비밀번호:** (empty field)
- Buttons: 연결, 연결 시험

A blue arrow points from the highlighted JDBC URL in the configuration interface to the corresponding line in the application.properties file.

# JPA로 데이터베이스 사용하기

26

- Entity로 테이블 매핑하기
  - Member 클래스

The screenshot shows the Eclipse IDE interface. On the left, the Package Explorer view displays the project structure:

- week12\_02JPADemo
- src/main/java
  - com.study.springboot
    - Ex12JpaDemoApplication.java
    - ServletInitializer.java
  - com.study.springboot.entity
    - Member.java
- src/main/resources
  - templates
  - static
  - application.properties
- src/test/java

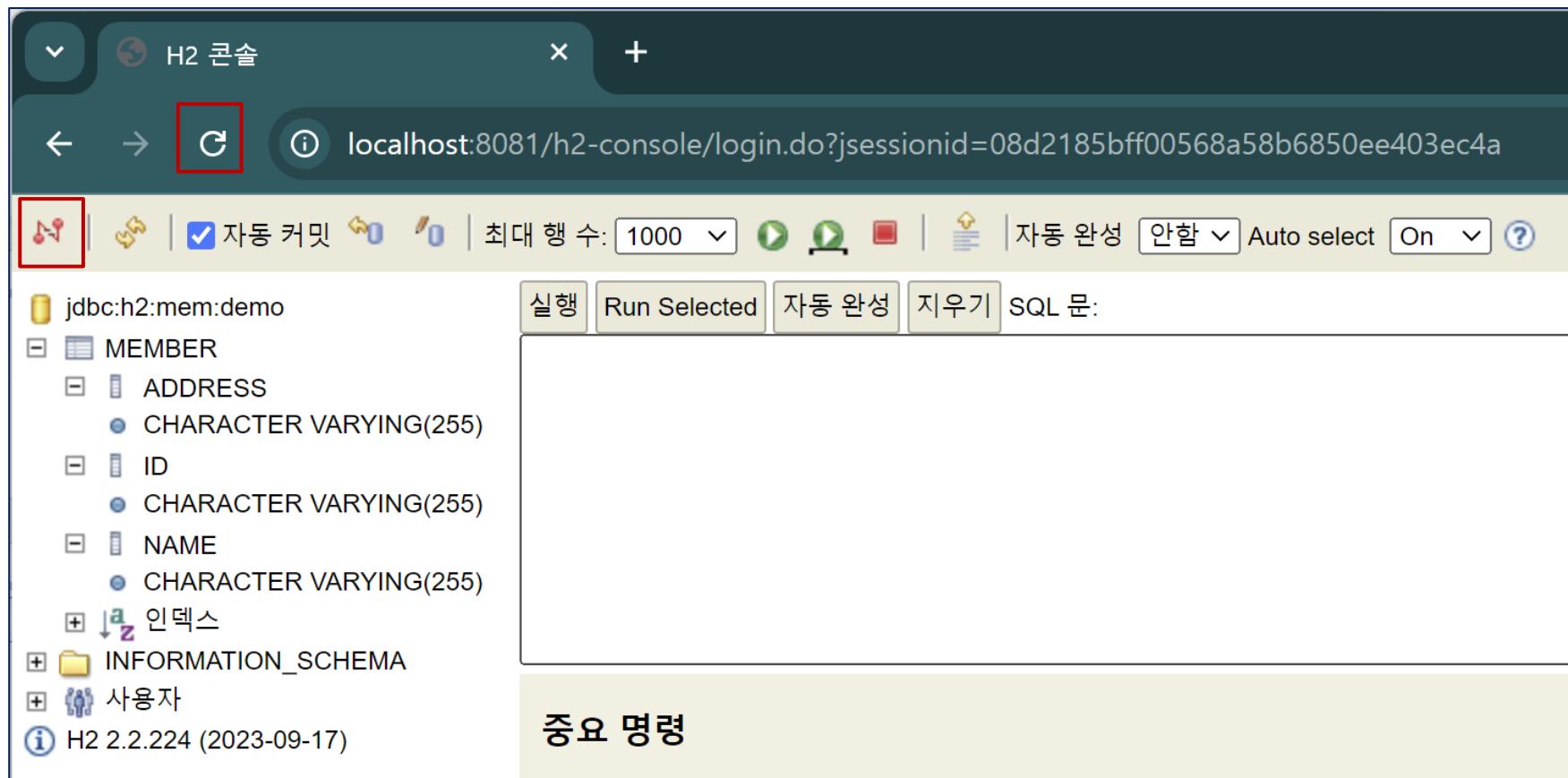
The Member.java file is selected in the Package Explorer and is displayed in the main editor area. The code is as follows:

```
1 package com.study.springboot.entity;
2
3 import jakarta.persistence.Entity;
4 import jakarta.persistence.Id;
5
6 @Entity
7 public class Member {
8     @Id
9     private String id;
10    private String name;
11    private String address;
12 }
```

# JPA로 데이터베이스 사용하기

27

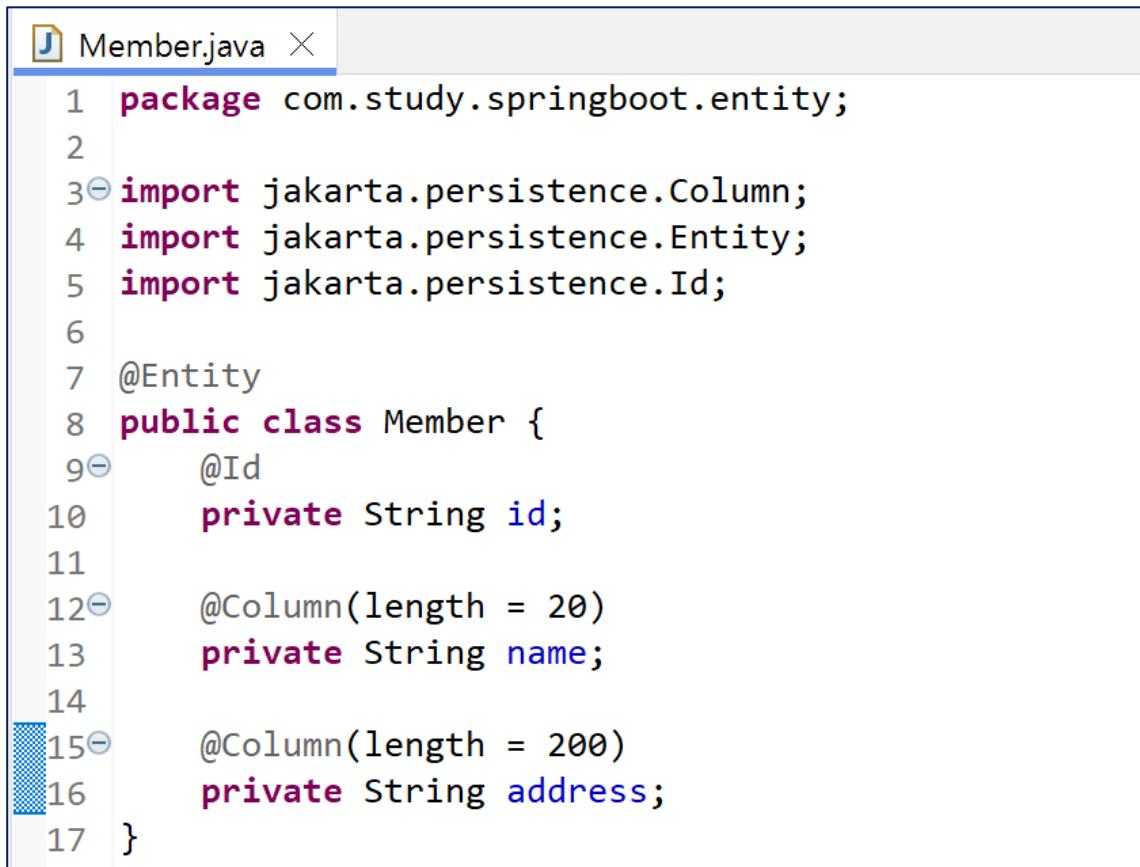
- Entity로 테이블 매핑하기
  - ▣ 프로젝트 재실행 후 > H2 콘솔 확인



# JPA로 데이터베이스 사용하기

28

- Entity로 테이블 매핑하기
  - ▣ 프로젝트 재실행 후 > H2 콘솔 확인

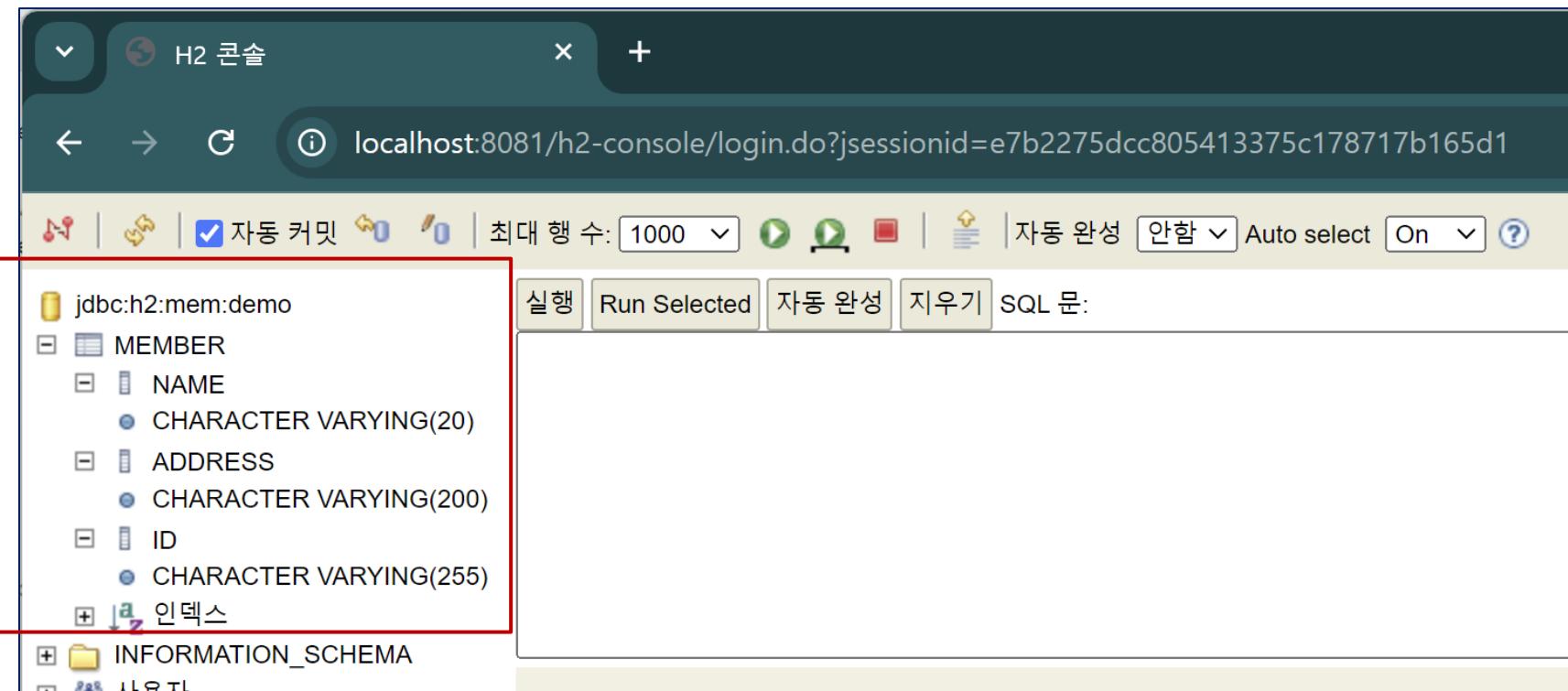


```
Member.java
1 package com.study.springboot.entity;
2
3 import jakarta.persistence.Column;
4 import jakarta.persistence.Entity;
5 import jakarta.persistence.Id;
6
7 @Entity
8 public class Member {
9     @Id
10    private String id;
11
12    @Column(length = 20)
13    private String name;
14
15    @Column(length = 200)
16    private String address;
17 }
```

# JPA로 데이터베이스 사용하기

29

- Entity로 테이블 매핑하기
  - ▣ 프로젝트 재실행 후 > H2 콘솔 확인



# Repository로 데이터베이스 사용하기

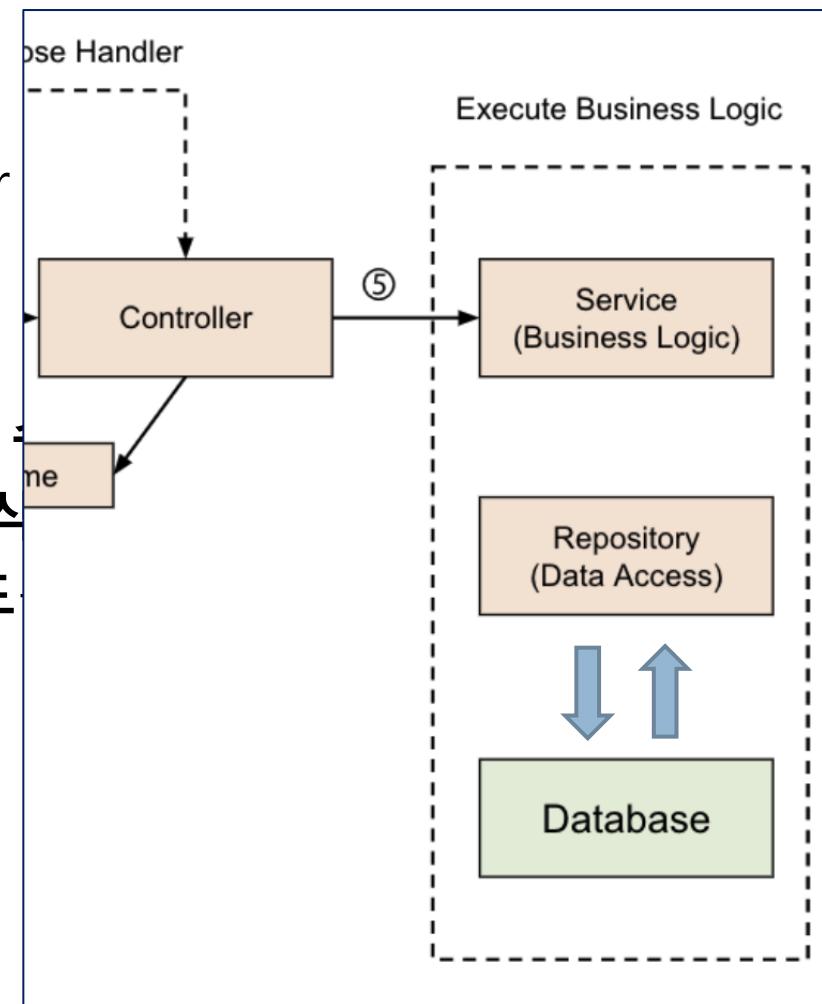
30

- 테이블을 구성하기 위해
  - Entity를 사용
  - 예) Question, Answer, Member 등
- **JPA Repository**
  - 테이블의 데이터를 저장, 조회, 수정, 삭제 등을 지원
  - JpaRepository 인터페이스 형식을 이용
  - 데이터를 관리하는 여러 메서드를 제공
    - findAll(), save() 등

# Repository로 데이터베이스 사용하기

31

- 테이블을 구성하기 위해
  - Entity를 사용
  - 예) Question, Answer, Member
- JPA Repository
  - 테이블의 데이터를 저장, 조회, 업데이트
  - JpaRepository 인터페이스 형식
  - 데이터를 관리하는 여러 메서드
    - findAll(), save() 등

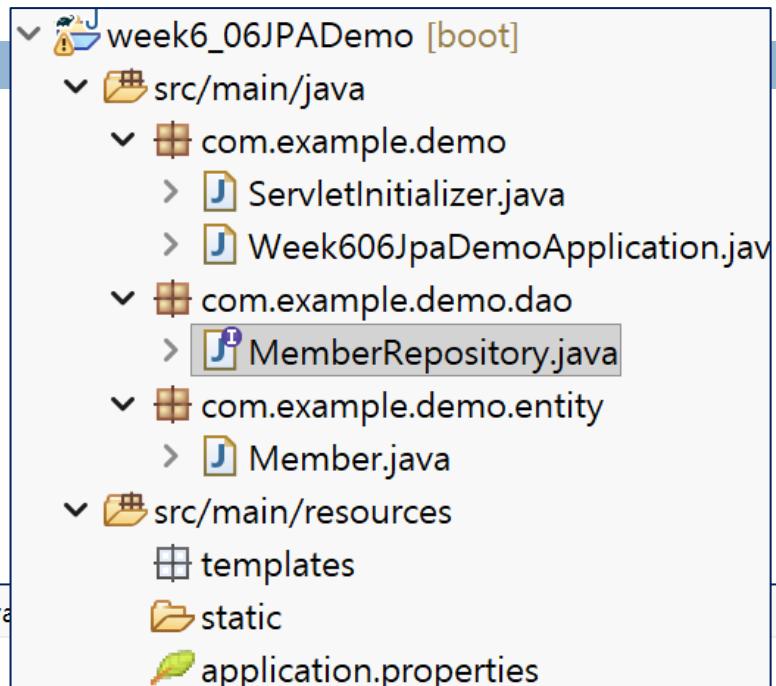


# Repository로 데이터베이스 사용하기

32

- Repository 생성하기
  - ▣ MemberRepository 인터페이스

```
MemberRepository.java × MemberController.java Member.java
1 package com.example.demo.dao;
2
3 import org.springframework.data.jpa.repository.JpaRepository;
4
5 import com.example.demo.entity.Member;
6
7 public interface MemberRepository extends JpaRepository<Member, String> {
8
9 }
10
```



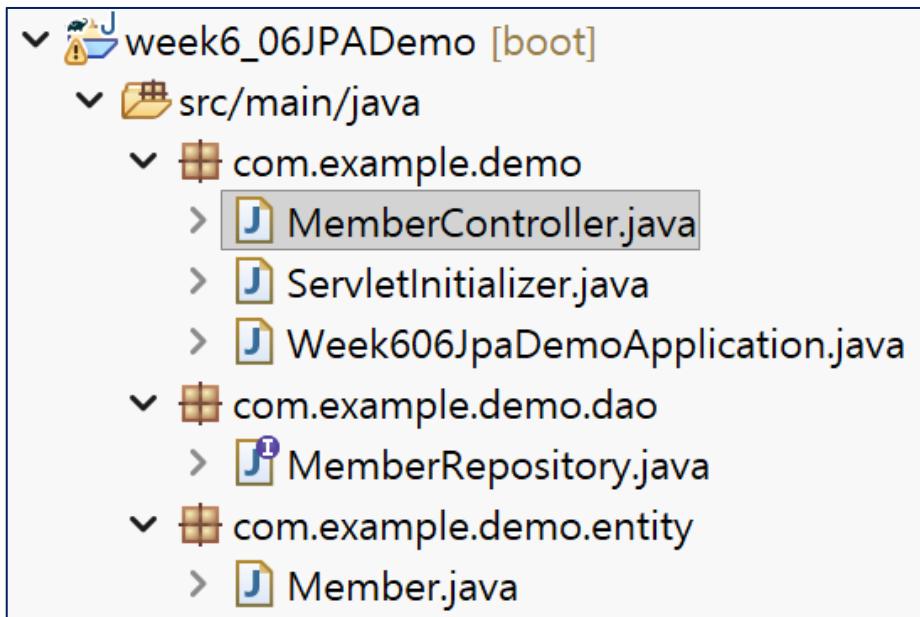
해당 Entity의  
키 타입

JPA로 핸들링 할  
Entity 명(테이블)

# Repository로 데이터베이스 사용하기

33

- Repository를 이용하여 데이터 생성하기
  - ▣ 클라이언트 요청정보("insert") 처리 클래스
  - ▣ **MemberController** 클래스



# Repository로 데이터베이스 사용하기

34

- Repository를 이용하여 데이터 생성하기
  - ▣ 클라이언트 요청정보("insert") 처리 클래스
  - ▣ **MemberController** 클래스

```
@RestController
public class MemberController {
    private MemberRepository memberRepository;

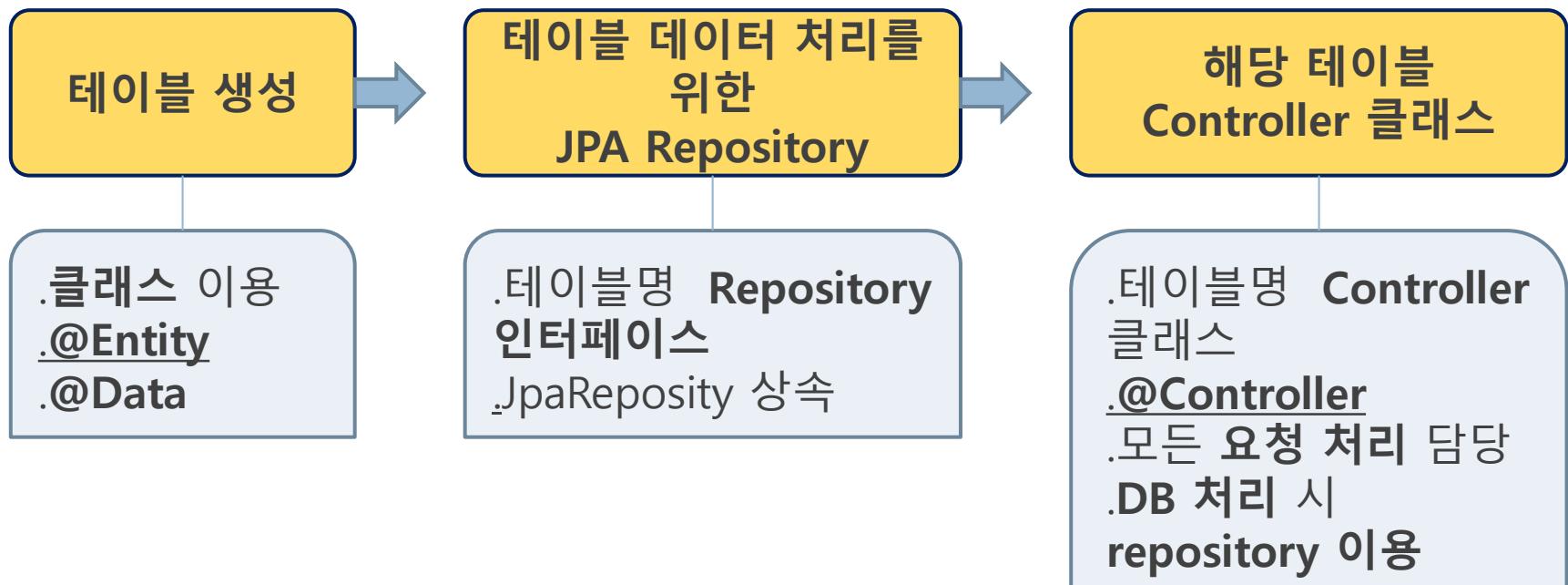
    public MemberController(MemberRepository repository) {
        memberRepository = repository;
    }

    @GetMapping("/insert")
    public void insert() {
        Member member = new Member();
        member.setId("admin");
        member.setName("홍길동");
        member.setAddress("경기도 화성시");
        memberRepository.save(member);
    }
}
```

# JPA로 데이터베이스 사용하기

35

- 테이블 생성 후 JPA로 DB 처리를 위한 설정 순서



# JPA로 데이터베이스 사용하기

36

## □ Entity로 테이블 매핑하기

### ▣ Question 클래스

데이터를 저장할 때 해당 속성 값을  
자동으로 1씩 증가하여 저장

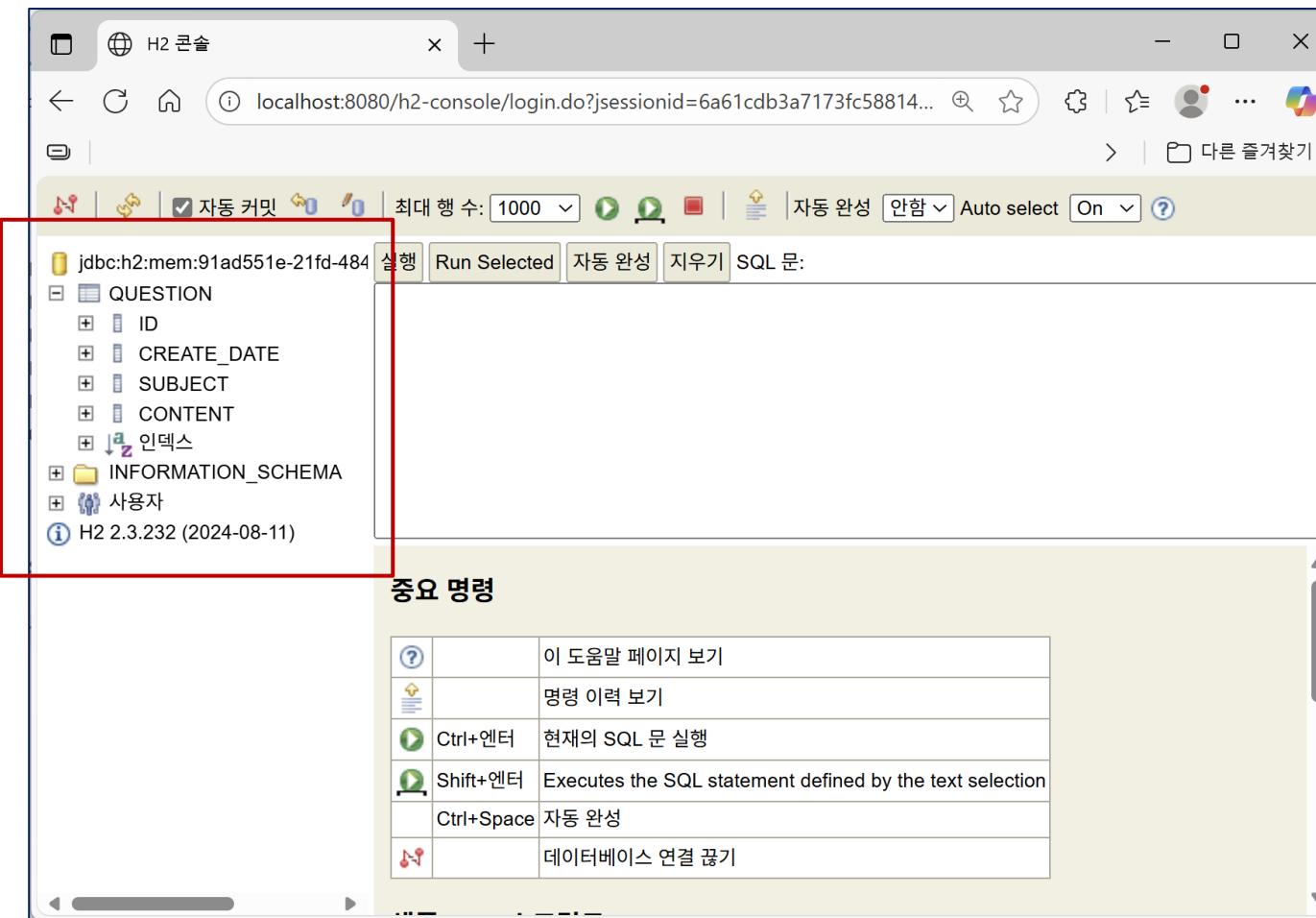
```
Question.java x
1 package com.example.demo.entity;
2
3 import java.time.LocalDateTime;
4
5 import jakarta.persistence.Column;
6 import jakarta.persistence.Entity;
7 import jakarta.persistence.GeneratedValue;
8 import jakarta.persistence.GenerationType;
import jakarta.persistence.Id;
import lombok.Data;

12 @Data
13 @Entity
14 public class Question {
15     @Id
16     @GeneratedValue(strategy = GenerationType.IDENTITY)
17     private int id;
18
19     @Column(length = 200)
20     private String subject;
21
22     @Column(columnDefinition = "TEXT")
23     private String content;
24
25     private LocalDateTime createDate;
26 }
```

# JPA로 데이터베이스 사용하기

37

- Entity로 테이블 매핑하기
  - ▣ 프로젝트 재실행 후 > H2 콘솔 확인



# Repository로 데이터베이스 사용하기

38

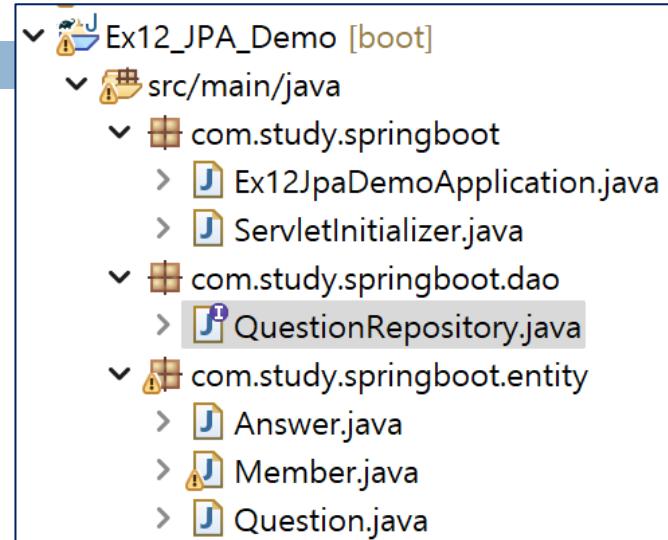
- 테이블을 구성하기 위해
  - Entity를 사용
  - 예) Question, Answer, Member 등
- **JPA Repository**
  - 테이블의 데이터를 저장, 조회, 수정, 삭제 등을 지원
  - JpaRepository 인터페이스 형식을 이용
  - 데이터를 관리하는 여러 메서드를 제공
    - findAll(), save() 등

# Repository로 데이터베이스 사용하기

39

- Repository 생성하기
  - ▣ **QuestionRepository** 인터페이스

Repository는 테이블 당 하나씩

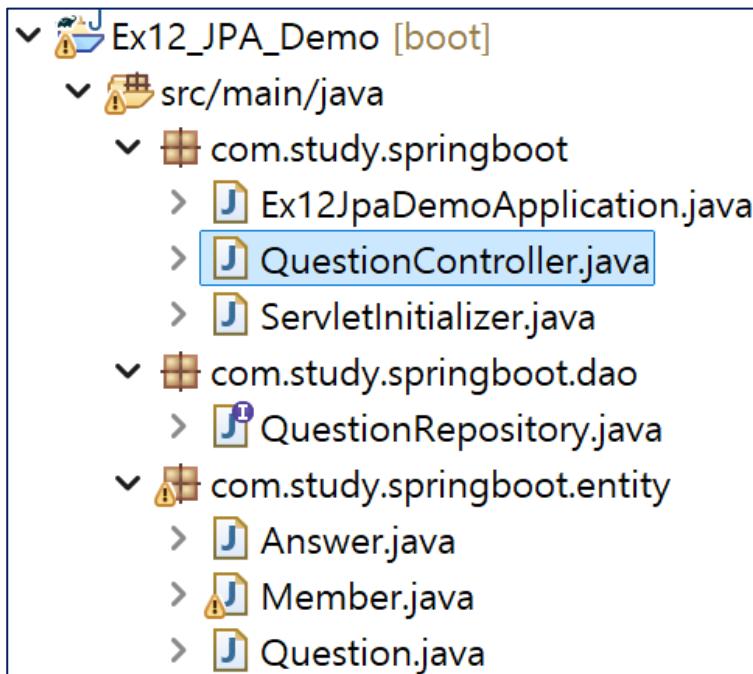


```
QuestionRepository.java ×  
1 package com.study.springboot.dao;  
2  
3 import org.springframework.data.jpa.repository.JpaRepository;  
4  
5 import com.study.springboot.entity.Question;  
6  
7 public interface QuestionRepository extends JpaRepository<Question, Integer> {  
8  
9 }  
10
```

# Repository로 데이터베이스 사용하기

40

- Repository를 이용하여 데이터 생성하기
  - ▣ 클라이언트 요청정보("insert") 처리 클래스
  - ▣ **QuestionController** 클래스



# Repository로 데이터베이스 사용하기

41

```
J QuestionController.java ×
1 package com.study.springboot;
2
3 import java.time.LocalDateTime;
4
5 import org.springframework.web.bind.annotation.GetMapping;
6 import org.springframework.web.bind.annotation.RestController;
7
8 import com.study.springboot.dao.QuestionRepository;
9 import com.study.springboot.entity.Question;
10
11 @RestController
12 public class QuestionController {
13     private QuestionRepository repository;
14
15     public QuestionController(QuestionRepository repository) {
16         this.repository = repository;
17     }
18
19     @GetMapping("/insert")
20     public void insert() {
21         Question question = new Question();
22         question.setSubject("sbb가 무엇인가요?");
23         question.setContent("sbb에 대해 알고 싶습니다");
24         question.setCreateDate(LocalDateTime.now());
25         repository.save(question);
26     }
27 }
```

# Repository로 데이터베이스 사용하기

42

- Repository를 이용하여 데이터 생성하기
  - ▣ 프로젝트 재실행 > "localhost:8080/insert"
  - ▣ H2 콘솔 창 > 새로고침 > SQL 문장으로 결과 확인

The screenshot shows the H2 Console interface. At the top, there is a toolbar with buttons for '실행' (Run), 'Run Selected', '자동 완성' (Autocomplete), '지우기' (Delete), and a text input field labeled 'SQL 문:' (SQL Query). Below the toolbar, a code editor window contains the SQL query: 'select \* from question'. In the bottom half of the interface, the results of the query are displayed. A message says 'select \* from question;' followed by a table with one row. The table has columns: ID, CREATE\_DATE, SUBJECT, and CONTENT. The data in the table is:

ID	CREATE_DATE	SUBJECT	CONTENT
1	2024-08-11 12:53:31.894076	sbb가 무엇인가요?	sbb에 대해 알고 싶습니다

Below the table, it says '(1 row, 1 ms)'. At the bottom left of the results area, there is a button labeled '편집' (Edit).

# Repository로 데이터베이스 사용하기

43

```
J QuestionController.java ×
12
13 @RestController
14 public class QuestionController {
15     private QuestionRepository repository;
16
17     public QuestionController(QuestionRepository repository) {
18         this.repository = repository;
19     }
20
21     @GetMapping("/insert")
22     public void insert() {
23         Question question1 = new Question();
24         question1.setSubject("sbb가 무엇인가요?");
25         question1.setContent("sbb에 대해 알고 싶습니다");
26         question1.setCreateDate(LocalDateTime.now());
27         repository.save(question1);
28
29         Question question2 = new Question();
30         question2.setSubject("스프링 부트 모델 질문입니다.");
31         question2.setContent("id는 자동으로 생성되나요?");
32         question2.setCreateDate(LocalDateTime.now());
33         repository.save(question2);
34     }
35
36     @GetMapping("/display")
37     @ResponseBody
38     public List<Question> display() {
39         List<Question> q = repository.findAll();
40         return q;
41     }
42 }
```

# Repository로 데이터베이스 사용하기

44

- Repository를 이용하여 데이터 생성하기
  - ▣ 프로젝트 재실행 > localhost:8081/insert > localhost:8081/display
  - ▣ H2 콘솔 창 > 새로고침 > SQL 문장으로 결과 확인

The screenshot shows the H2 Database Console interface. At the top, there are several buttons: '실행' (Run), 'Run Selected', '자동 완성' (Autocomplete), '지우기' (Delete), and 'SQL 문:' (SQL Query). Below these buttons is a text input field containing the SQL query: 'select \* from question'. The main area of the console is empty, indicating no results have been displayed yet. At the bottom, there is a summary message: 'select \* from question;' followed by a table with two rows of data.

ID	CREATE_DATE	SUBJECT	CONTENT
1	2024-08-11 13:51:56.017478	sbb가 무엇인가요?	sbb에 대해 알고 싶습니다
2	2024-08-11 13:51:56.107715	스프링 부트 모델 질문입니다.	id는 자동으로 생성되나요?

(2 행, 4 ms)

# JSP 기반 웹 프로젝트 구성

45

- JSP 설정
  - build.gradle에 디펜던시 추가
  - Gradle > Refresh Gradle Project
  - 프로젝트 > properties > Project Facets
    - Dynamic Web Module > 5.0
  - resources 폴더
    - application.properties 수정
      - spring.mvc.view.prefix=/WEB-INF/views/
      - spring.mvc.view.suffix=.jsp
    - JSP 폴더 생성
      - src > main > webapp > WEB-INF > views
    - jsp 파일 생성
      - views 폴더에 생성

# JSP 기반 웹 프로젝트 구성 => JSP 설정

46

QuestionController.java ×

```
5 @Controller
6 public class QuestionController {
7     private QuestionRepository repository;
8
9     public QuestionController(QuestionRepository repository) {
10         this.repository = repository;
11     }
12
13     @GetMapping("/")
14     public String index() {
15         return "index";
16     }
17
18     @GetMapping("/insertForm")
19     public String insertForm() {
20         return "insertForm";
21     }
22 }
```

# JSP 기반 웹 프로젝트 구성

47

QuestionController.java ×

```
3 @GetMapping("/insert")
4 public String insert(Question question) {
5     question.setCreateDate(LocalDateTime.now());
6     repository.save(question);
7     return "redirect:list";
8 }
9
10 @GetMapping("/list")
11 public String list(Model model) {
12     List<Question> questionList = repository.findAll();
13     model.addAttribute("questionList", questionList);
14     return "question_list";
15 }
```

# JSP 기반 웹 프로젝트 구성

48

## □ index.jsp

```
index.jsp ×
1 <%@ page language="java" contentType="text/html; charset=UTF-8"
2     pageEncoding="UTF-8"%>
3 <!DOCTYPE html>
4 ⊖<html>
5 ⊖<head>
6 <meta charset="UTF-8">
7 <title>Insert title here</title>
8 <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/css/bootstrap.min.css" rel="stylesheet"
9 integrity="sha384-QWTKZyjpPEjISv5WaRU9OFeRpok6YctnYmDr5pNLyT2bRjXh0JMhjY6hW+ALEwIH" crossorigin="anonymous">
0 </head>
1 ⊖<body>
2 ⊖<div class="container" style="padding: 20px;">
3     <h3> ♠ 게시판 프로그램 ♠</h3>
4     <br>
5     <a href="/insertForm" class="btn btn-primary">등록</a>
6     <a href="/List" class="btn btn-primary">조회</a>
7 </div>
8
9
0 ⊖<script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/js/bootstrap.bundle.min.js"
1 integrity="sha384-Yvpcryf0tY3lHB60NNkmXc5s9fDVZLESaAA55NDz0xhy9GkcIdsLK1eN7N6jIeHz" crossorigin="anonymous"></script>
2 </body>
3 </html>
```

# JSP 기반 웹 프로젝트 구성

49

## □ insertForm.jsp

```
1 insertForm.jsp ×
L0 </head>
L1 ⊖ <body>
L2
L3 ⊖ <div class="container" style="padding: 20px;">
L4     <h3> ♠ 게시판 등록 ♠</h3>
L5     <br>
L6 ⊖   <form action="/insert">
L7     제 목 : <input type="text" name="subject"> <br>
L8     내용 : <br>
L9         <textarea rows="5" cols="30" name="content"></textarea><br><br>
L10    <input type="submit" value="등록">
L11  </form>
L12 </div>
L13
```

# JSP 기반 웹 프로젝트 구성

50

## □ question\_list.jsp

```
question_list.jsp ×
3  <%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
4  <!DOCTYPE html><html><head><meta charset="UTF-8">
5  <title>Insert title here</title>
6  <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/css/bootstrap.min.css" rel="stylesheet"
7  integrity="sha384-QWTKZyjpPEjISv5WaRU9OFeRpok6YctnYmDr5pNLyT2bRjXh0JMhjY6hW+ALEwIH" crossorigin=
8  </head><body>
9  <div class="container" >
  <h3> ♣ 게시판 목록 ♣</h3>
10 <br>
11 <table border="1" cellpadding="5">
12   <tr>
13     <th>ID</th>
14     <th>Subject</th>
15     <th>Content</th>
16     <th>CreateDate</th>
17   </tr>
18   <c:forEach var="q" items="${questionList }">
19     <tr>
20       <td>${q.id } </td>
21       <td>${q.subject } </td>
22       <td>${q.content } </td>
23       <td>${q.createDate } </td>
24     </tr>
25   </c:forEach>
26 </table>
27
28 <br><br>
29 <a href="/insertForm" class="btn btn-primary">입력화면</a>
30 <a href="/" class="btn btn-primary">초기화면</a>
31
32 </div>
```

# 과제물

51

- **기말 프로젝트**
  - ▣ 스프링부트 기반 웹 시스템 개발 프로젝트
  - ▣ **프로젝트 주제** 정하기
    - 선정 동기
    - 벤치마킹 사이트
    - 컨셉트 등 그 외 부가사항 포함 가능
  - ▣ **화면 구성도**
    - 페이지 디자인 및 내용 구성
  - ▣ **DB 테이블**
    - 테이블 구성도

# 과제물

52

- 제출 형식
  - 제출 파일 : 학번\_이름.pdf
  - ppt로 작업하고 pdf 형식으로 저장 후 제출