
# 12

# Lombok 및 JPA로 DataBase 사용하기

2

# 롬복(Lombok) 사용하기

# 롬복 사용하기



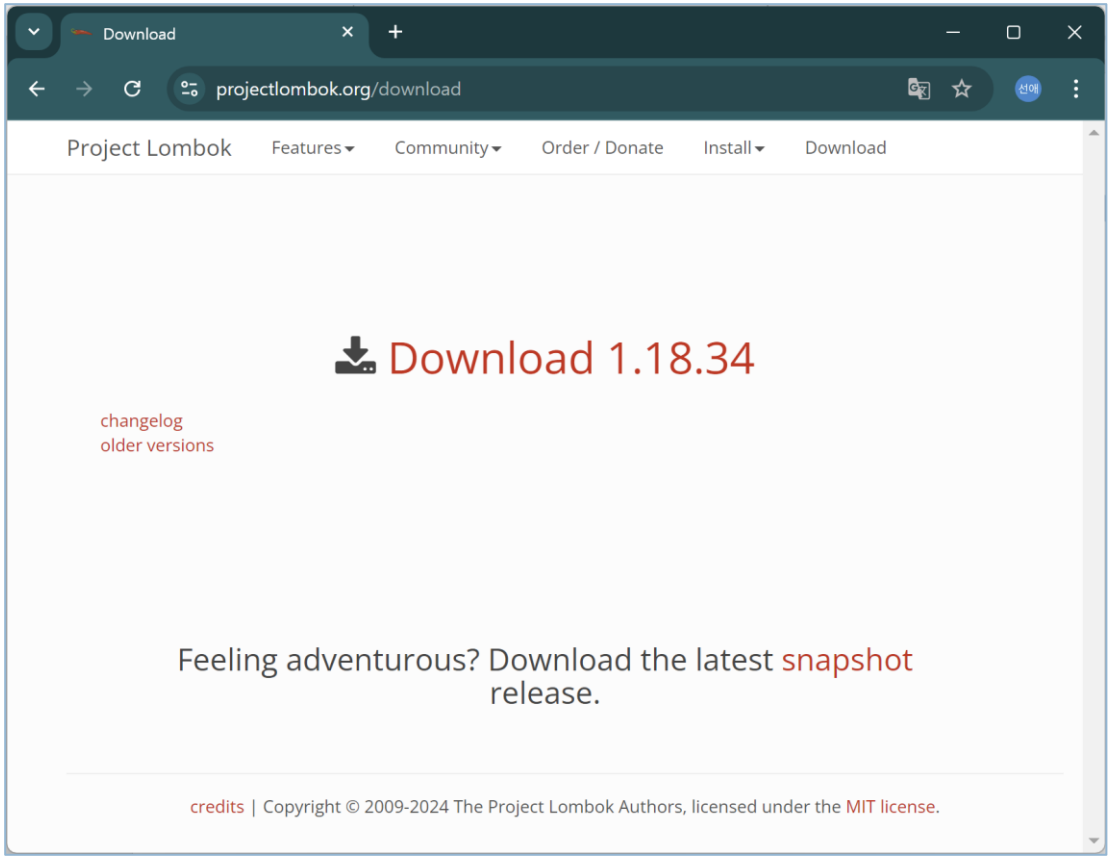
- **롬복(Lombok)**
  - 자바 클래스를 만들 때 항상 만들게 되는 **Getter, Setter 메서드** 등을 어노테이션을 이용해서 **자동으로 만들어주는 유틸리티 라이브러리**

- 롬복을 사용하기 위해
  - **롬복 플러그인 설치**
  - 프로젝트 생성 시 **디펜던시 추가**

# 롬복 사용하기



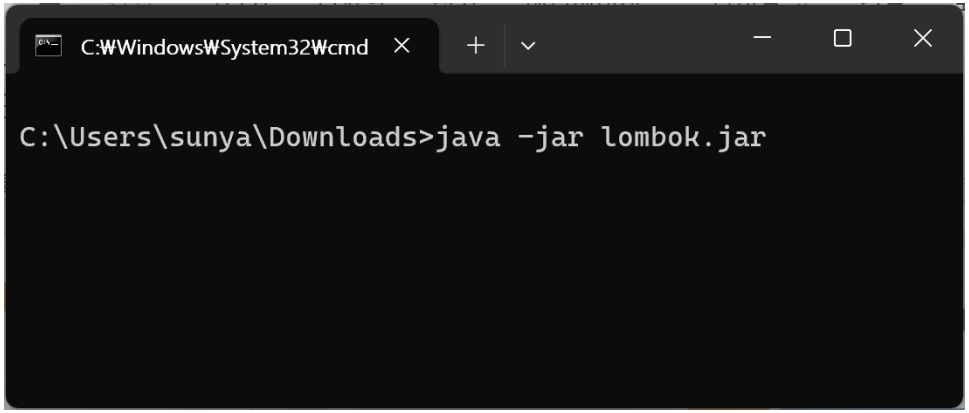
- **롬복 다운로드**
  - https://projectlombok.org/download



# 롬복 사용하기



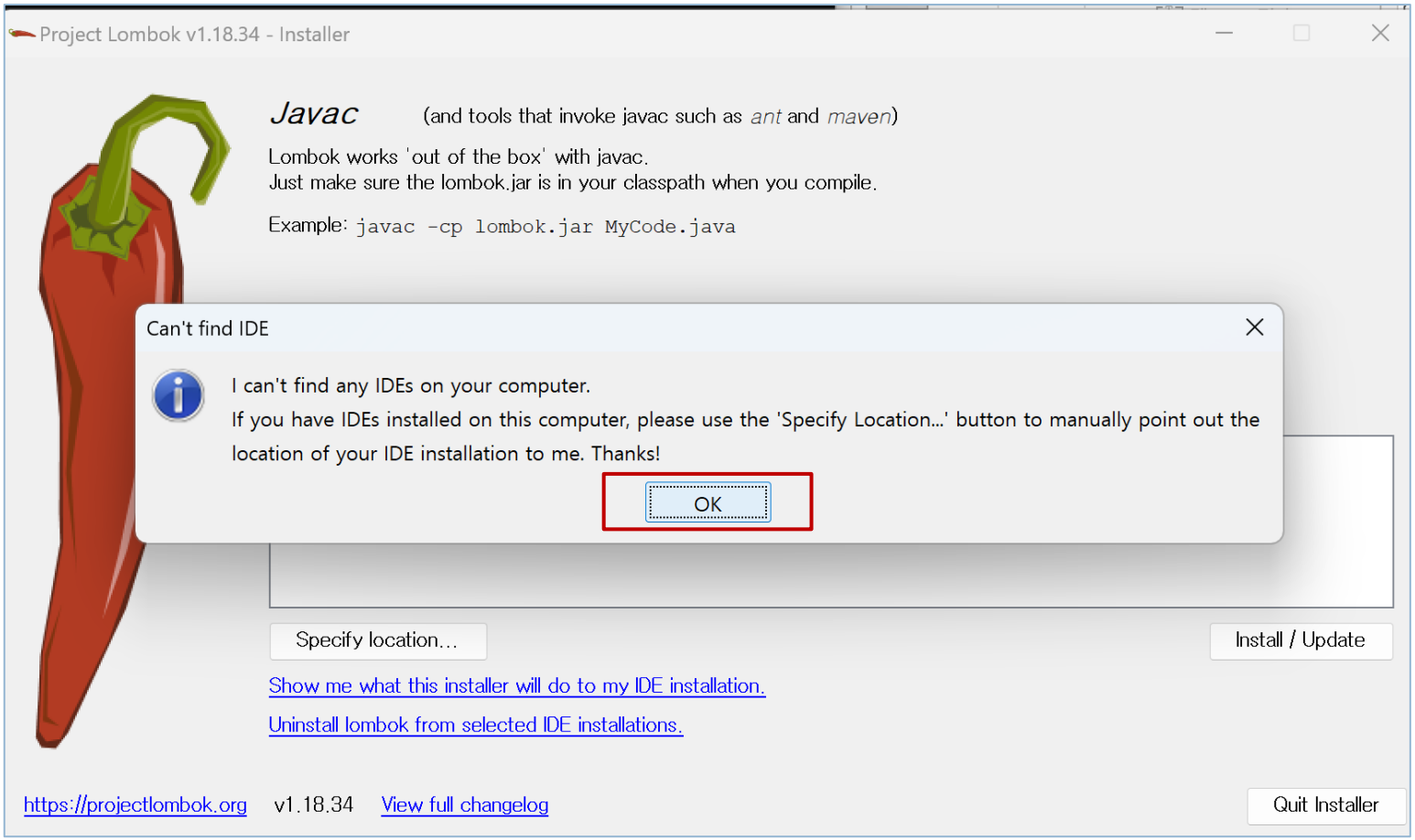
- **롬복 설치**
  - 명령 프롬프트 창 > **java –jar lombok.jar**



# 롬복 사용하기



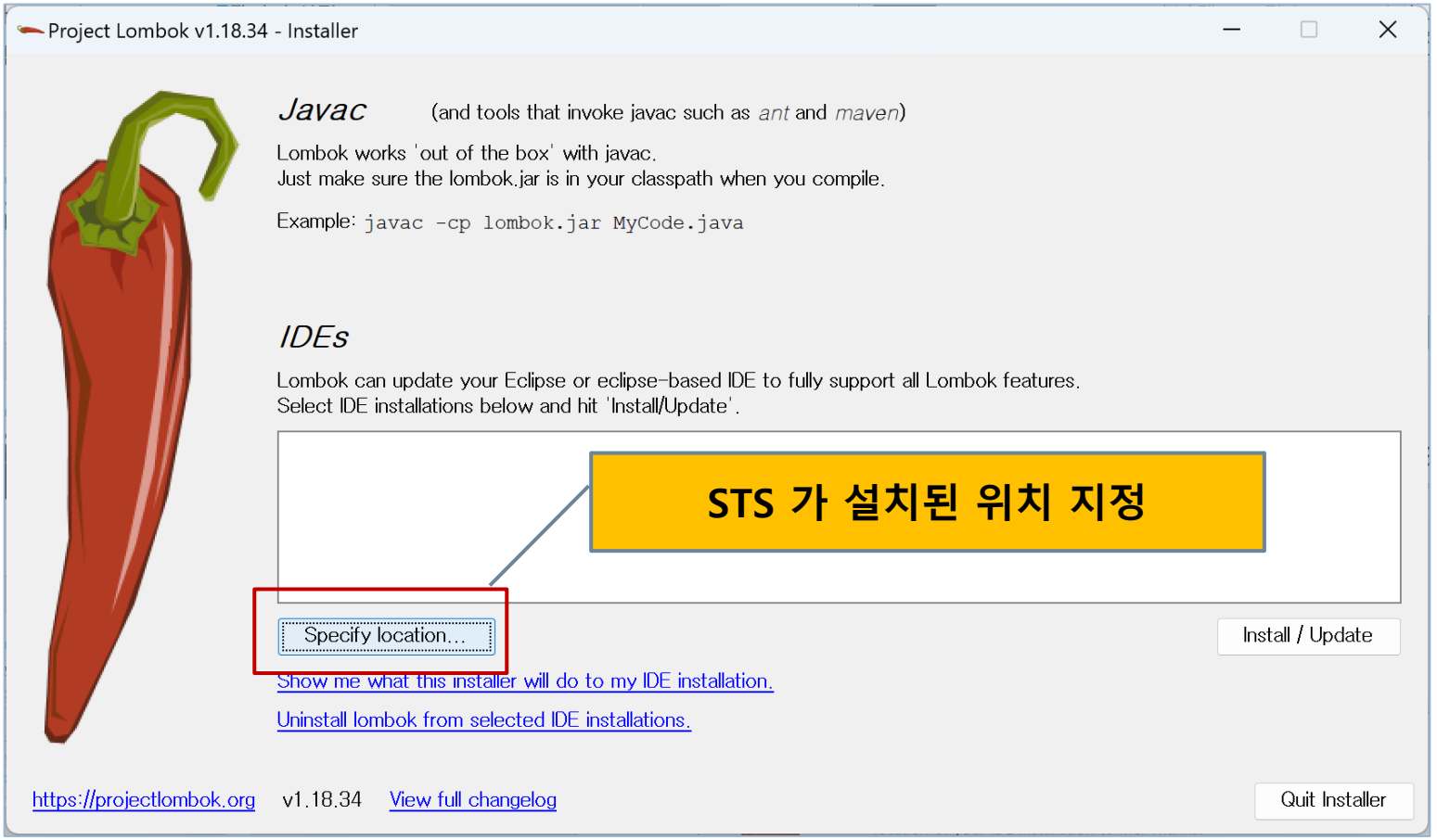
- 롬복 설치
  - 명령 프롬프트 창 > java –jar lombok.jar



# 롬복 사용하기



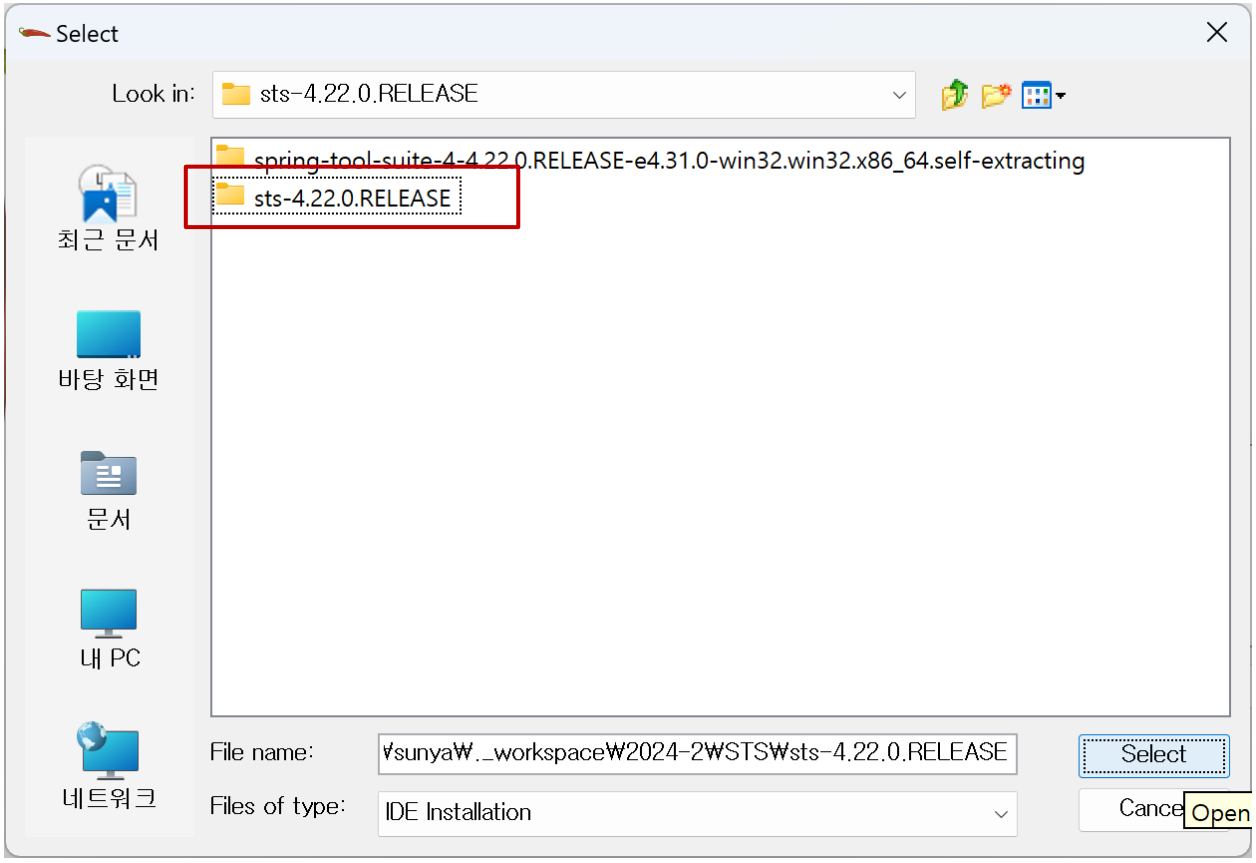
- 롬복 설치
  - 명령 프롬프트 창 > java –jar lombok.jar



# 롬복 사용하기



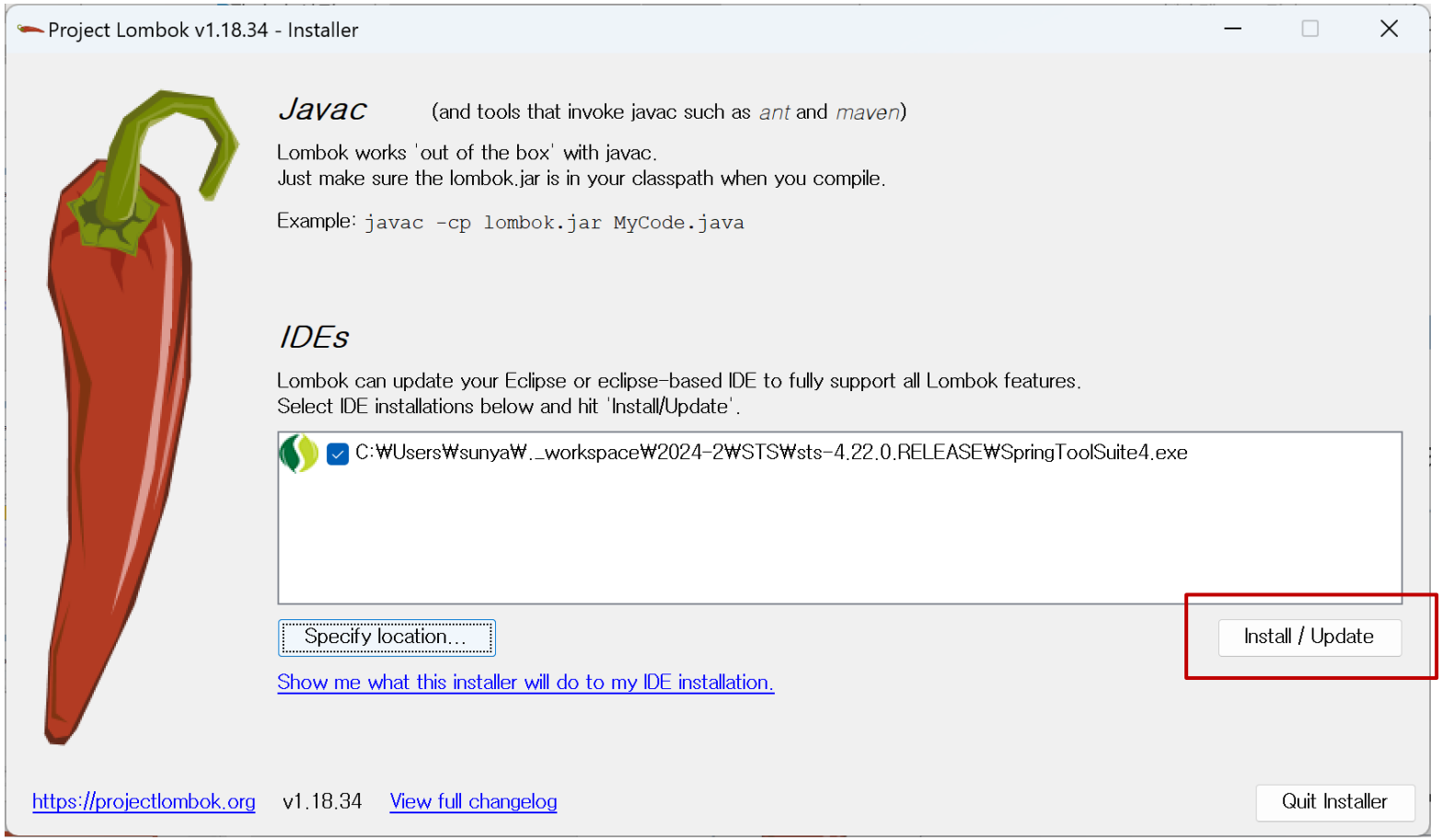
- 롬복 설치
  - 명령 프롬프트 창 > java –jar lombok.jar



# 롬복 사용하기



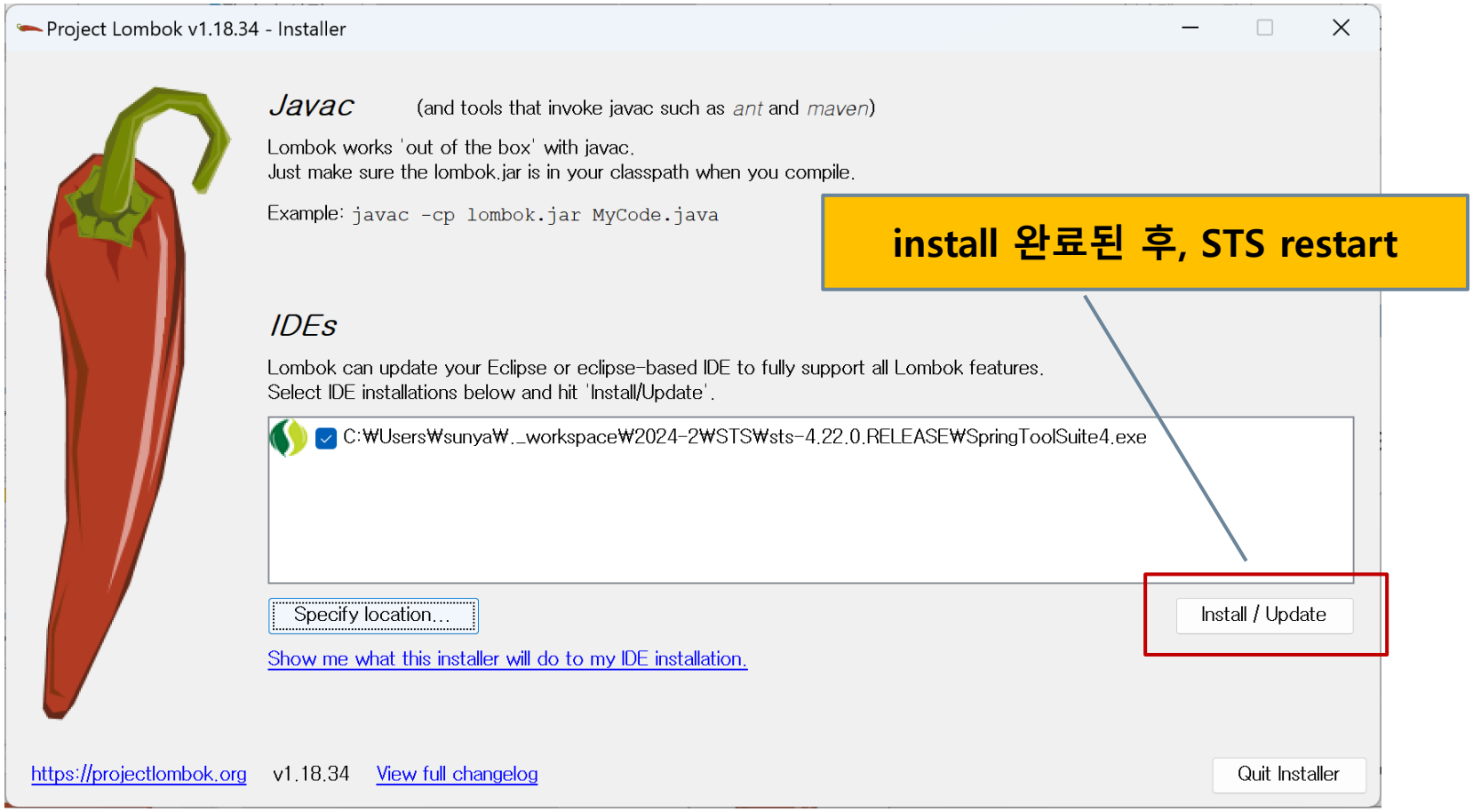
- 롬복 설치
  - 명령 프롬프트 창 > java –jar lombok.jar



# 롬복 사용하기



- 롬복 설치
  - 명령 프롬프트 창 > java –jar lombok.jar



# 롬복 사용하기

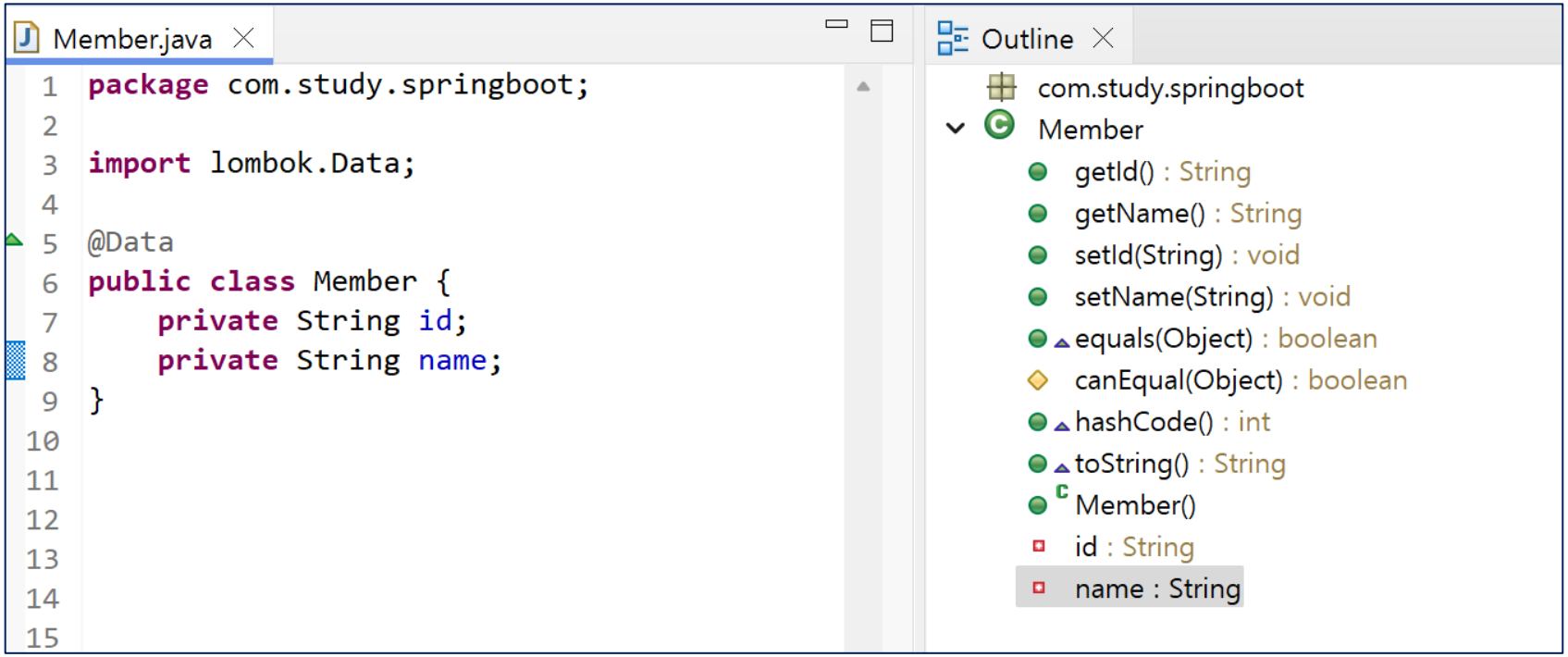


- 롬복 사용 프로젝트 생성(**week12_01Lombok**)
  - 디펜던시 설정
    - SpringWeb
    - **Lombok (**`implementation 'org.projectlombok:lombok')`
  - build.gradle
    - JSP 설정 / 버전 수정
    - Refresh Gradle Project
  - application.properties
    - 포트 설정
    - JSP 파일 경로 설정
  - File > Properties
    - Project Facets > Dynamic Web Module 버전 > 5.0으로 수정
  - JSP 사용을 위한 폴더 구성
    - src/main/webapp/WEB-INF/views

# 롬복 사용하기



- 커맨드(Command) 객체 만들기
  - Member 클래스



# 롬복 사용하기



- 리퀘스트 매핑
  - MyController

MyContoller.java

```java
package com.study.springboot;

import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.ResponseBody;

@Controller
public class MyContoller {
    @RequestMapping("/")
    public @ResponseBody String root() {
        return "Lombok 사용하기";
    }

    @GetMapping("/test3")
    public String test3(Member member) {
        // 파라미터와 일치하는 빈을 만들어서 사용할 수 있다.
        // View 페이지에서 model을 사용하지 않고 member를 사용한다.
        return "test3";
    }
}
```

# 롬복 사용하기



- 리퀘스트 매핑
  - test3.jsp

test3.jsp

```jsp
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Insert title here</title>
</head>
<body>
<%
    out.println("<h1>#03 : Hello World </h1>");
%>
<br>
<h3>
당신의 아이디는 ${member.id }입니다. <br>
당신의 이름은 ${member.name }입니다.
</h3>
</body>
</html>
```

15

# JPA로 데이터베이스 사용하기

# JPA로 데이터베이스 사용하기



- 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 부트**는 **JPA**를 **ORM의 기술 표준**으로 사용

# JPA로 데이터베이스 사용하기



- 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 내부**에서 안정적인 **SQL 쿼리문을 생성**
    - 개발자가 달라져도 통일된 쿼리문 작성이 가능
    - 오류 발생률도 감소

# JPA로 데이터베이스 사용하기



- ORM과 JPA 이해하기
  - **JPA(Java Persistence API)**
    - 스프링 부트는 JPA를 사용하여 데이터베이스를 관리
    - 스프링 부트는 JPA를 ORM의 기술 표준으로 사용
  - **JPA 특징**
    - JPA는 **인터페이스 모음**으로 이 **인터페이스를 구현한 실체 클래스가 필요**
    - **대표적인 JPA 실체 클래스 => 하이버네이트(Hibernate)**
    - **Hibernate**
      - JPA의 인터페이스를 구현한 **실체 클래스**
      - 자바의 ORM 프레임워크로 **스프링 부트**에서 **데이터베이스 관리를 쉽게 하는 도구**

# JPA로 데이터베이스 사용하기



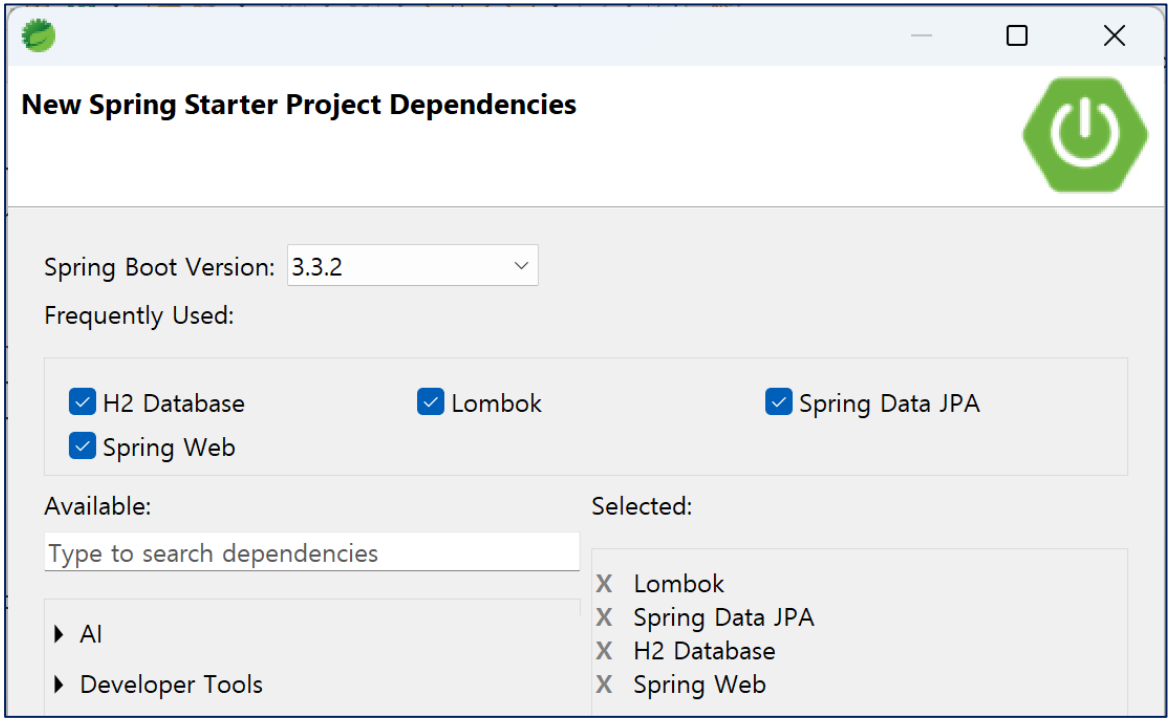
- ORM과 JPA 이해하기
  - ORM(Object Relational Mapping)
    - Java 문법으로도 데이터베이스를 다룰 수 있는 도구
    - SQL을 직접 작성하지 않아도 데이터베이스의 데이터 처리가 가능
  - JPA(Java Persistence API)
    - 스프링 부트는 JPA를 사용하여 데이터베이스를 관리
    - 스프링 부트는 JPA를 ORM의 기술 표준으로 사용
    - JPA는 인터페이스 모음으로 이 인터페이스를 구현한 실체 클래스가 필요
    - 대표적인 JPA 실체 클래스 => 하이버네이트(Hibernate)
    - Hibernate
      - JPA의 인터페이스를 구현한 실체 클래스
      - 자바의 ORM 프레임워크로 스프링 부트에서 데이터베이스 관리를 쉽게하는 도구

**데이터베이스 관리 = JPA + Hibernate**

# JPA로 데이터베이스 사용하기



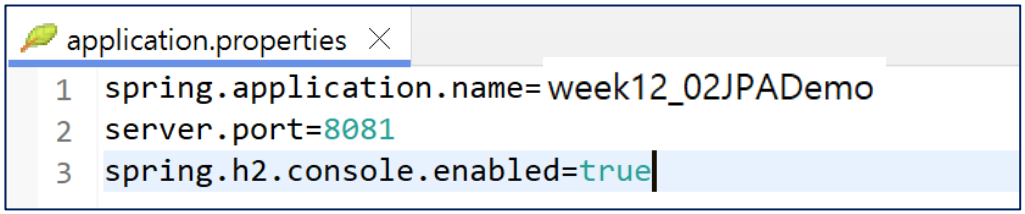
- H2 데이터베이스 사용
  - 주로 개발 환경에서 사용하는 자바 기반의 경량 DBMS
  - 스프링 부트 프로젝트 생성(**week12_02JPADemo**)
  - 디펜던시 설정
    - **Spring Web**
    - **Lombok**
    - **Spring Data JPA**
    - **H2 Database**

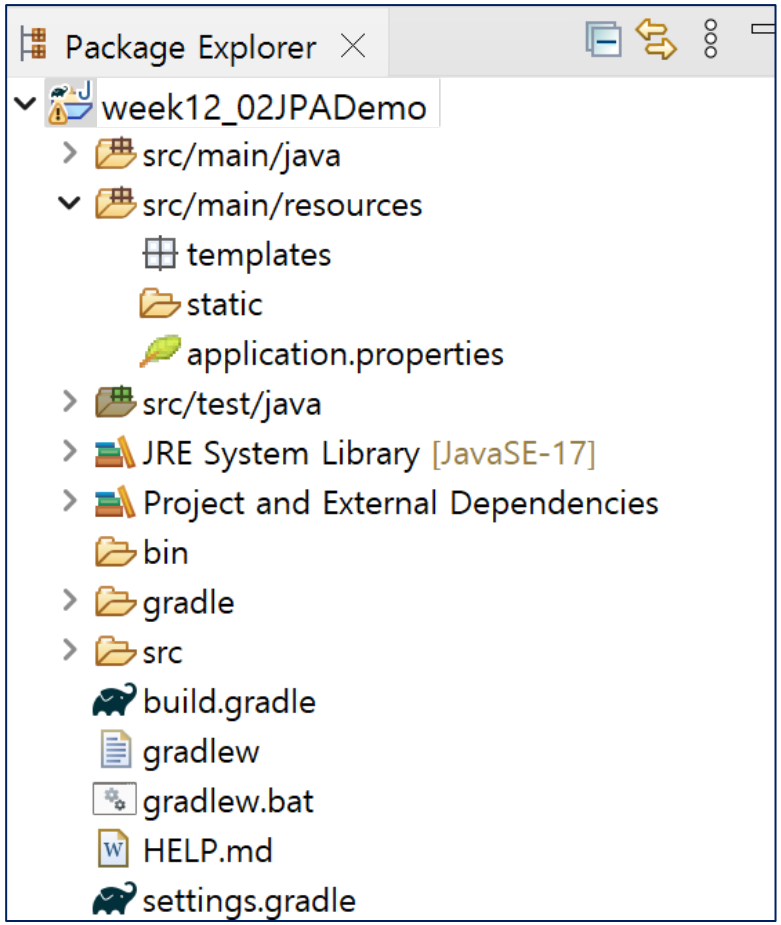


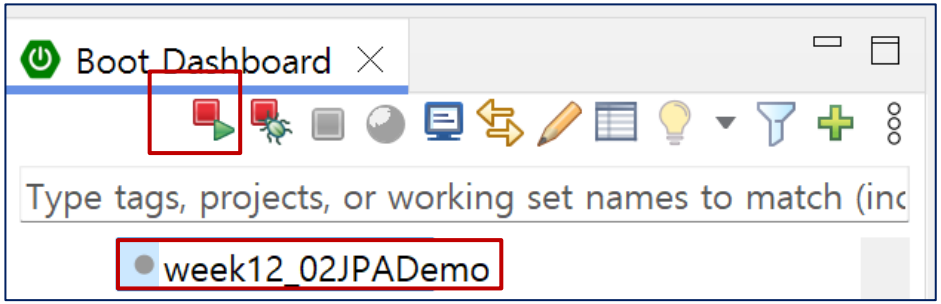
# JPA로 데이터베이스 사용하기



- H2 데이터베이스 사용
  - **application.properties**



# JPA로 데이터베이스 사용하기



- H2 데이터베이스 사용
  - 프로젝트 실행 후 콘솔 창

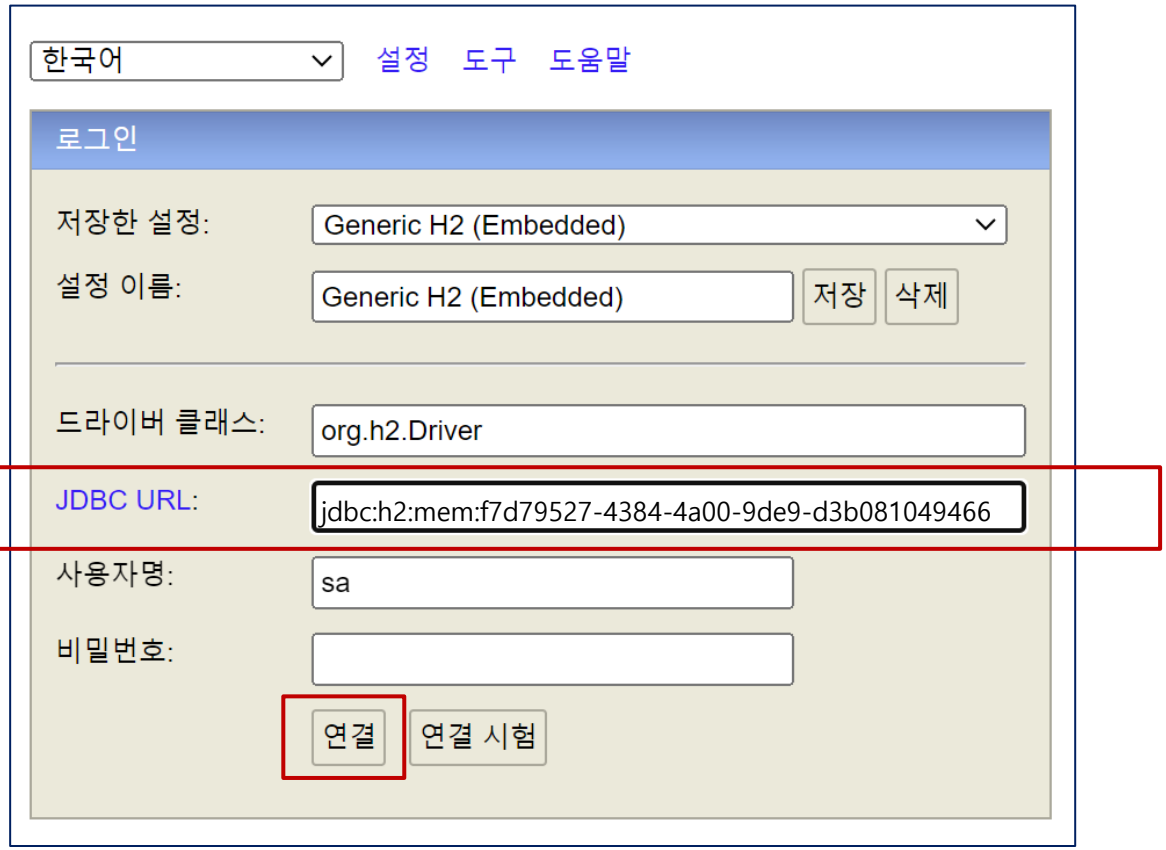
```
main] c.s.springboot.Ex12JpaDemoApplication    : Starting Ex12JpaDemoApplication using Java 17.0.10 with PID 13324 (C:\Users\sunya\._workspace\2024-2\S
main] c.s.springboot.Ex12JpaDemoApplication    : No active profile set, falling back to 1 default profile: "default"
main] .s.d.r.c.RepositoryConfigurationDelegate : 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 i
main] o.apache.catalina.core.StandardService   : Startin
main] o.apache.catalina.core.StandardEngine    : Startin
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-270ffc9e591b 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



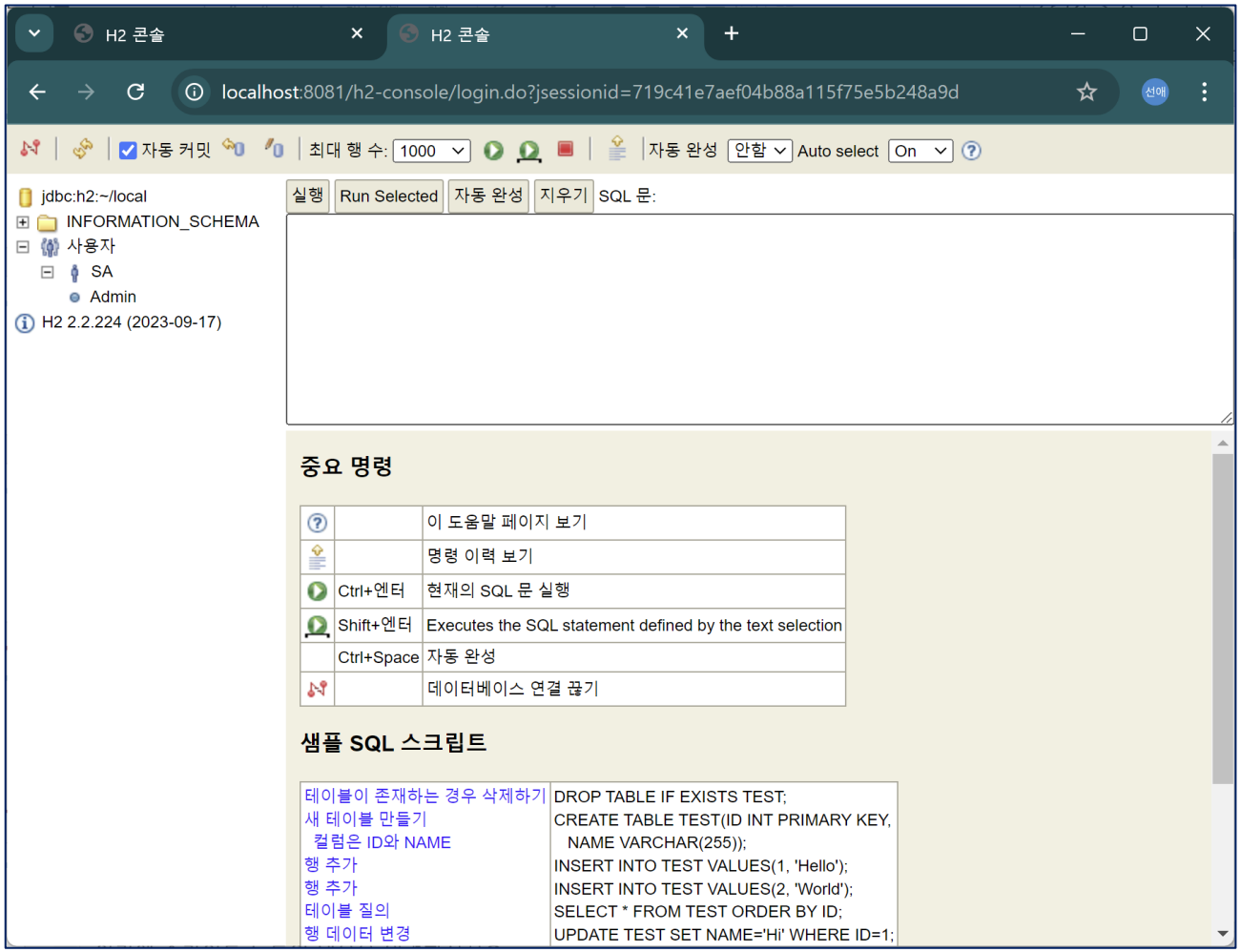
- H2 데이터베이스 연결
  - 브라우저 창 > **http://localhost:8081/h2-console**
  - 복사 한 url 주소 => **JDBC URL** 붙여넣기



# JPA로 데이터베이스 사용하기



- H2 데이터베이스



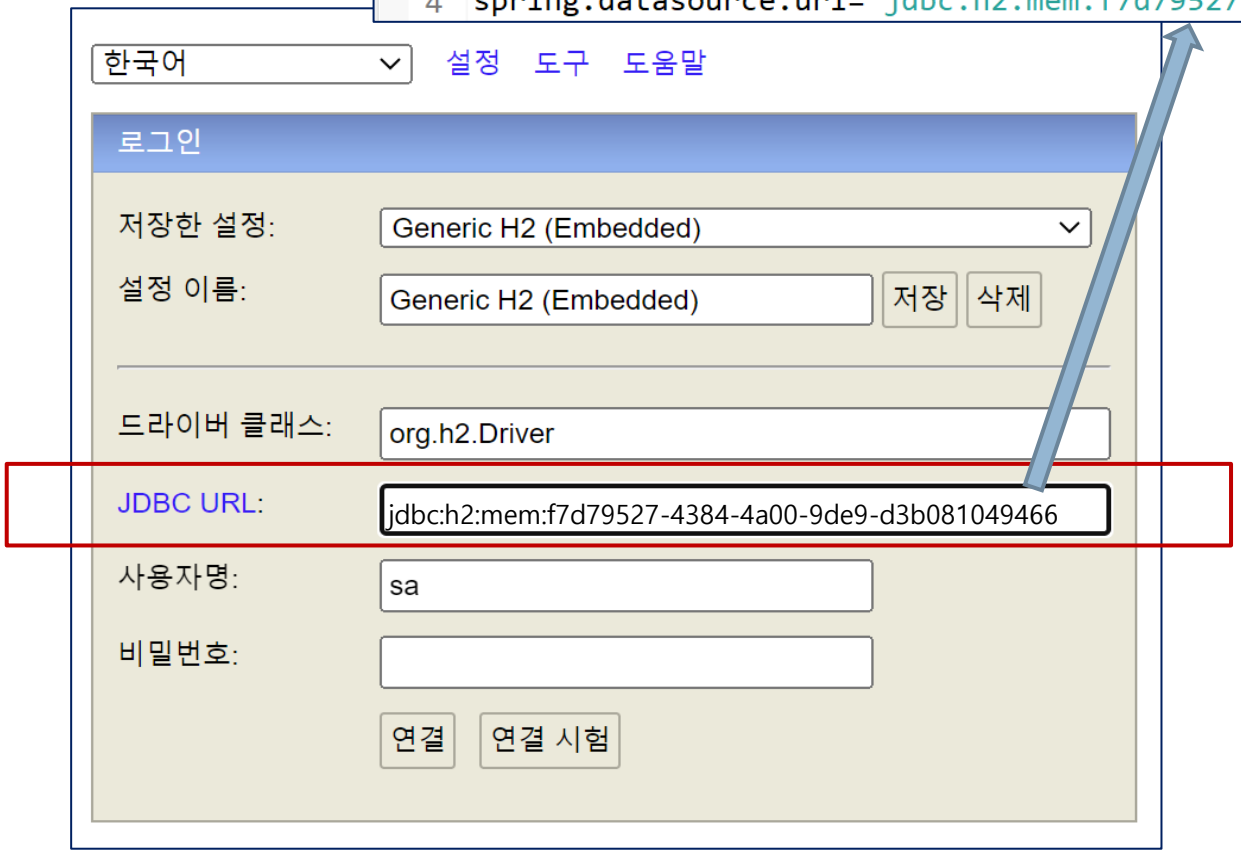
# JPA로 데이터베이스 사용하기



- H2 데이터베이스
  - **url 고정**

application.properties

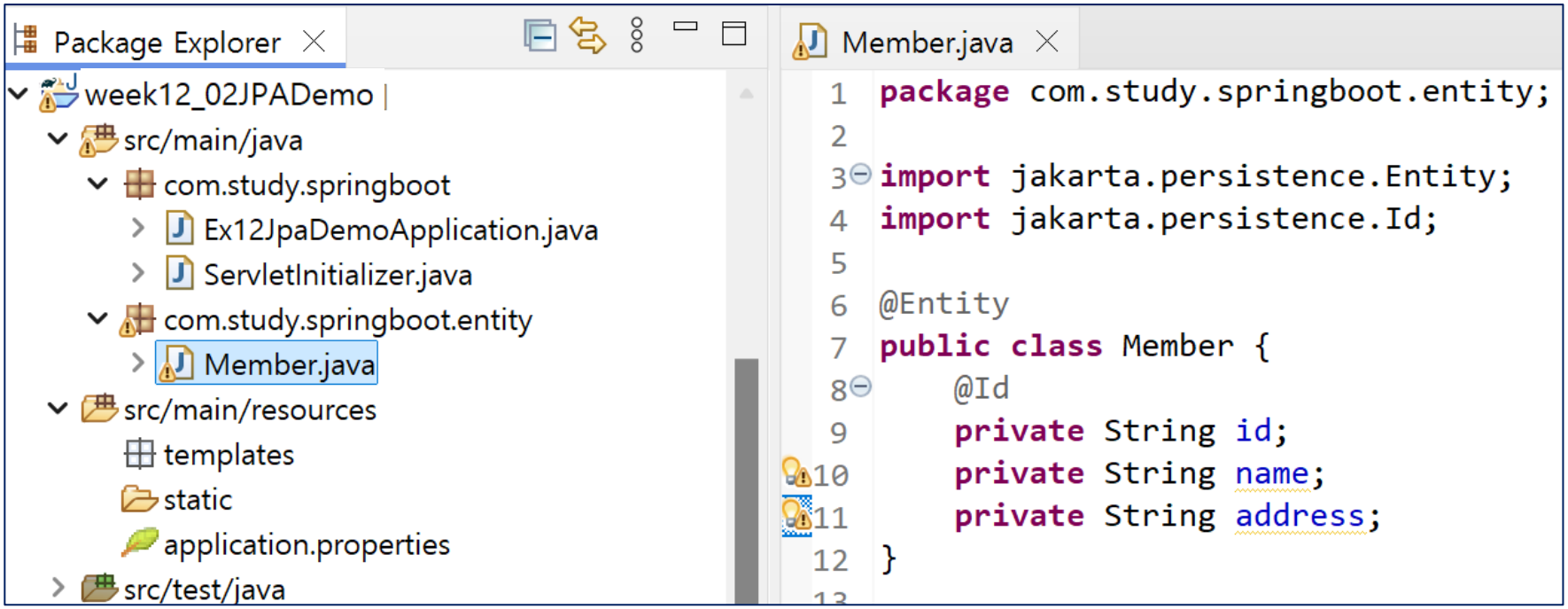
```
1 spring.application.name=week12_02JPADemo
2 server.port=8081
3 spring.h2.console.enabled=true
4 spring.datasource.url= jdbc:h2:mem:f7d79527-4384-4a00-9de9-d3b081049466
```



# JPA로 데이터베이스 사용하기



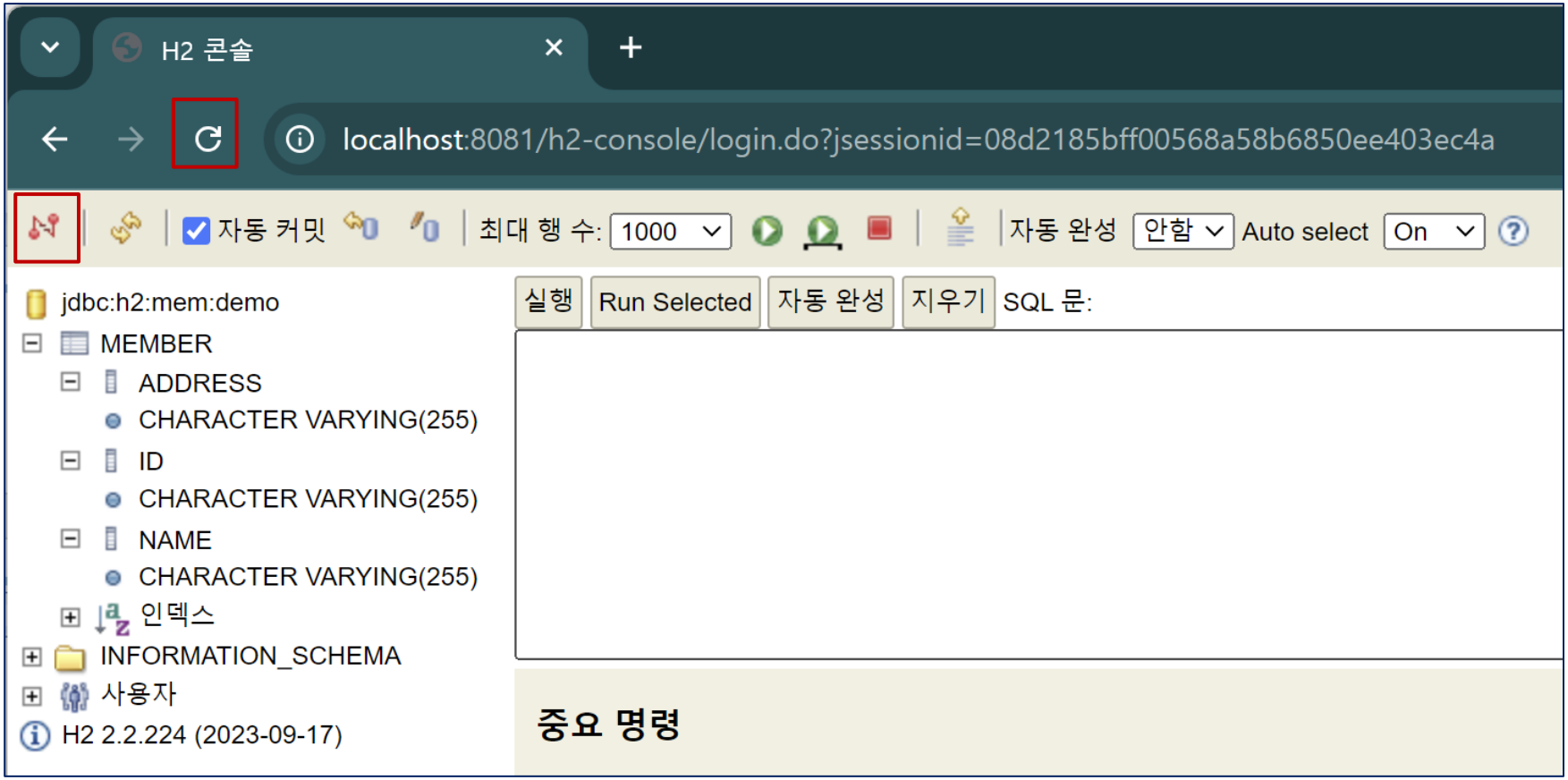
- Entity로 테이블 매핑하기
  - Member 클래스



# JPA로 데이터베이스 사용하기



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



# JPA로 데이터베이스 사용하기



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

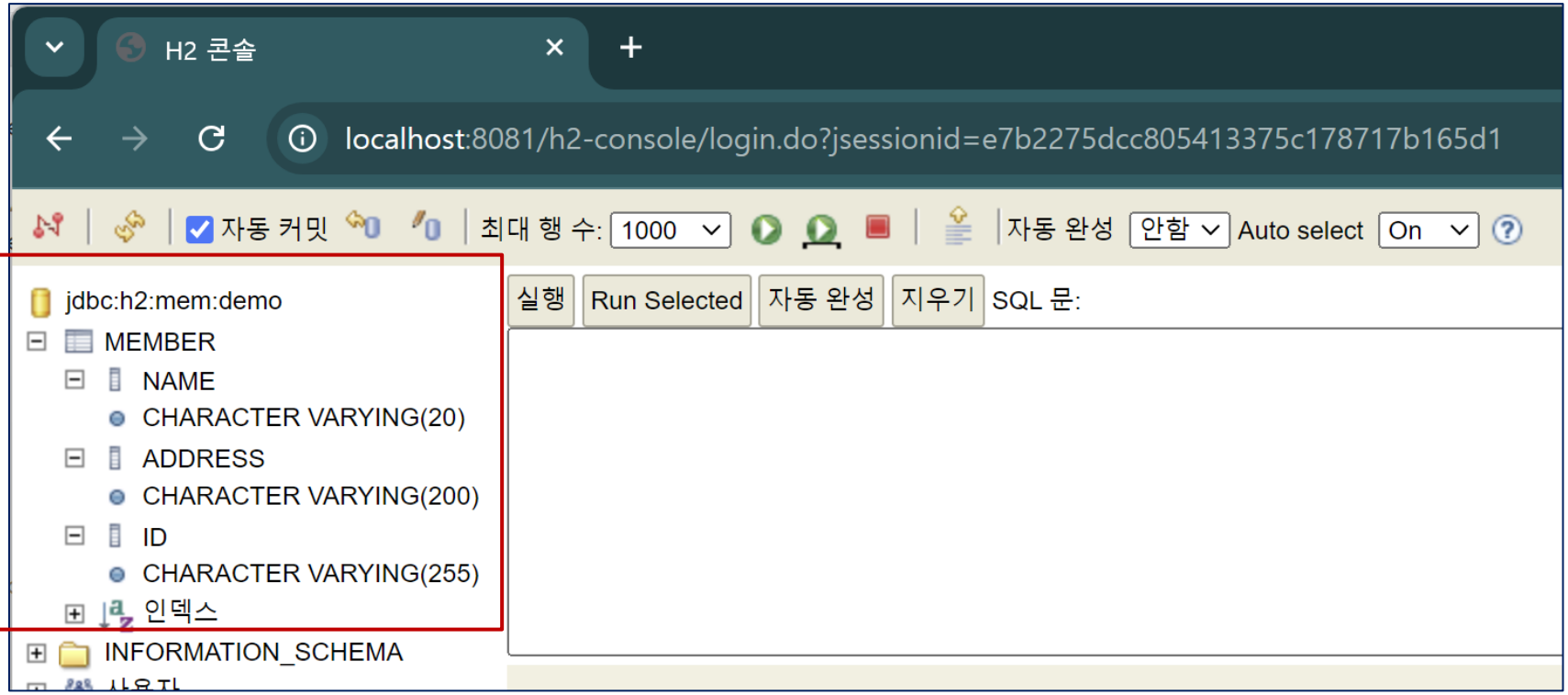
Member.java

```java
package com.study.springboot.entity;

import jakarta.persistence.Column;
import jakarta.persistence.Entity;
import jakarta.persistence.Id;

@Entity
public class Member {
    @Id
    private String id;

    @Column(length = 20)
    private String name;

    @Column(length = 200)
    private String address;
}
```

# JPA로 데이터베이스 사용하기



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



# Repository로 데이터베이스 사용하기



- 테이블을 구성하기 위해
  - Entity를 사용
  - 예) Question, Answer, Member 등

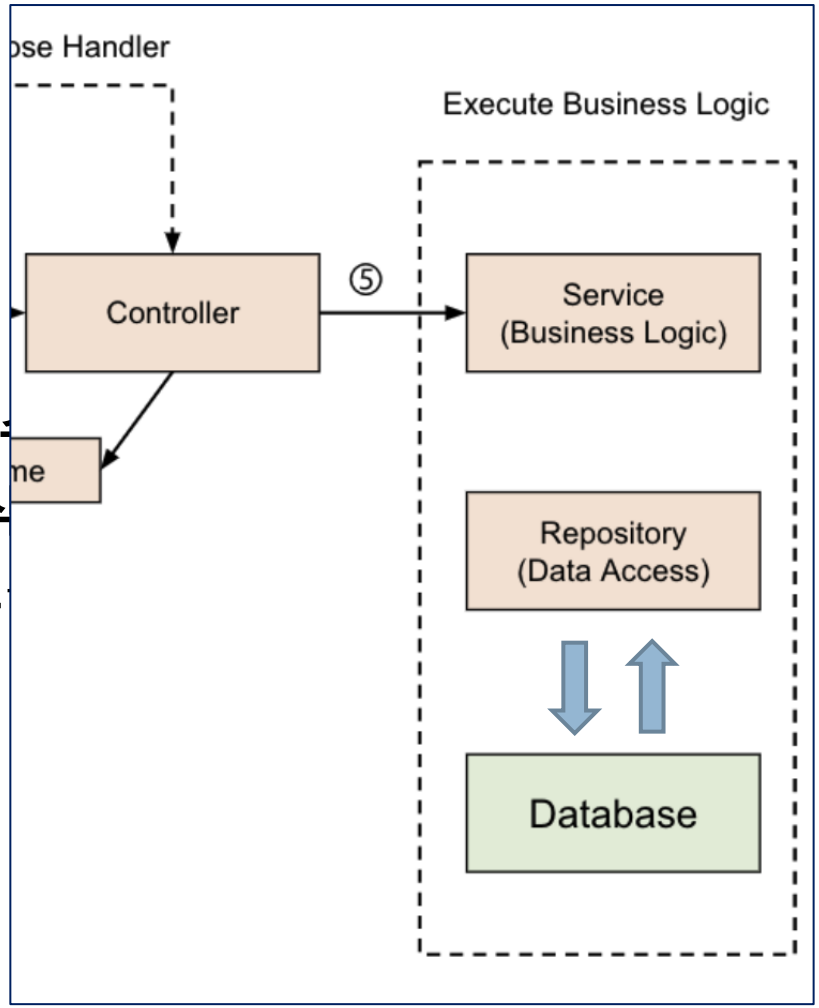
- **JPA Repository**
  - **테이블의 데이터**를 **저장, 조회, 수정, 삭제** 등을 지원
  - **JpaRepository 인터페이스 형식**을 이용
  - **데이터를 관리**하는 여러 **메서드**를 제공
    - findAll(), save() 등

# Repository로 데이터베이스 사용하기



- 테이블을 구성하기 위해
  - Entity를 사용
  - 예) Question, Answer, Member

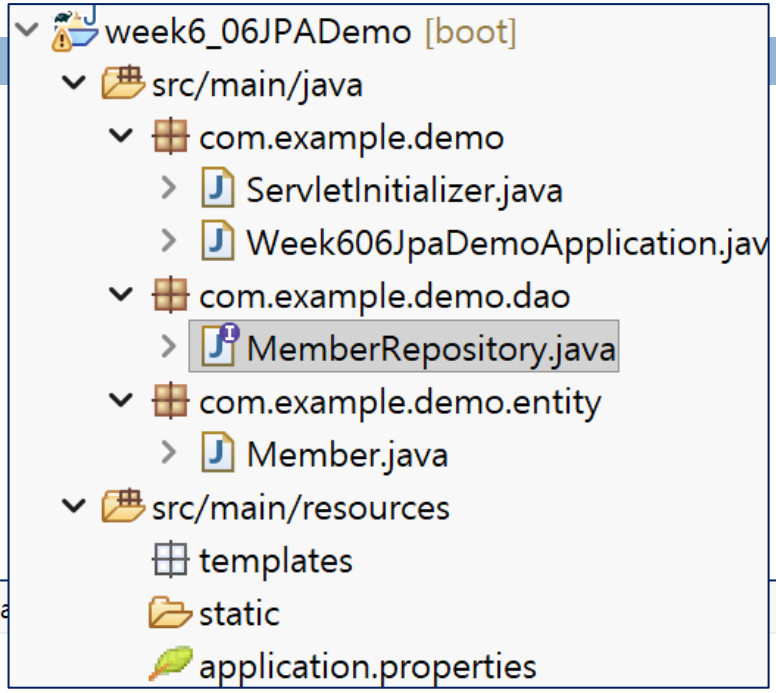
- **JPA Repository**
  - **테이블의 데이터를 저장, 조회,**
  - **JpaRepository 인터페이스 형식**
  - **데이터를 관리**하는 여러 **메서드**
    - findAll(), save() 등



# Repository로 데이터베이스 사용하기



- 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
```

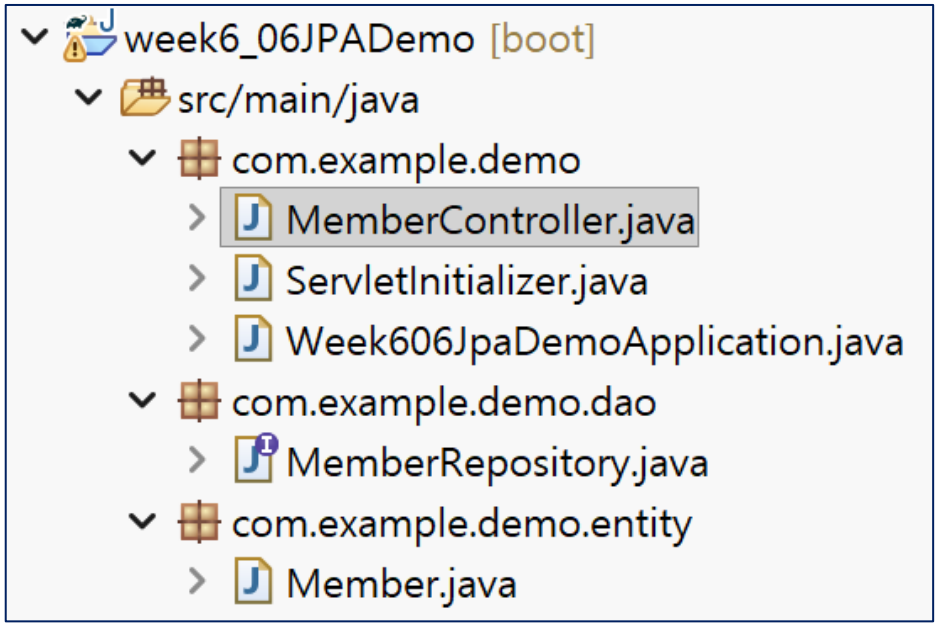
해당 Entity의 키 타입

JPA로 핸들링 할 Entity 명(테이블)

# Repository로 데이터베이스 사용하기



- Repository를 이용하여 데이터 생성하기
  - 클라이언트 요청정보(“insert”) 처리 클래스
  - **MemberController** 클래스



# Repository로 데이터베이스 사용하기



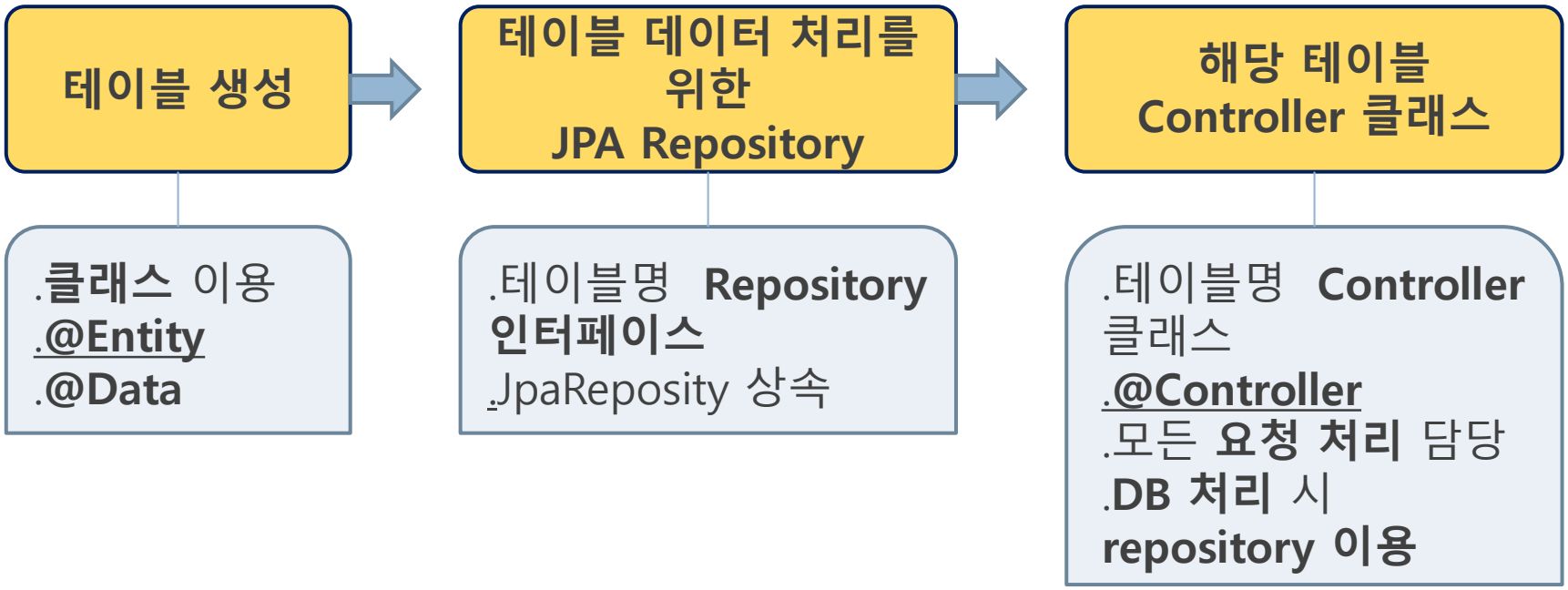
- Repository를 이용하여 데이터 생성하기
  - 클라이언트 요청정보(“insert”) 처리 클래스
  - **MemberController** 클래스

```
@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



- 테이블 생성 후 JPA로 DB 처리를 위한 설정 순서



# JPA로 데이터베이스 사용하기



- Entity로 테이블 매핑하기
  - **Question** 클래스

**데이터를 저장할 때 해당 속성 값을 자동으로 1씩 증가하여 저장**

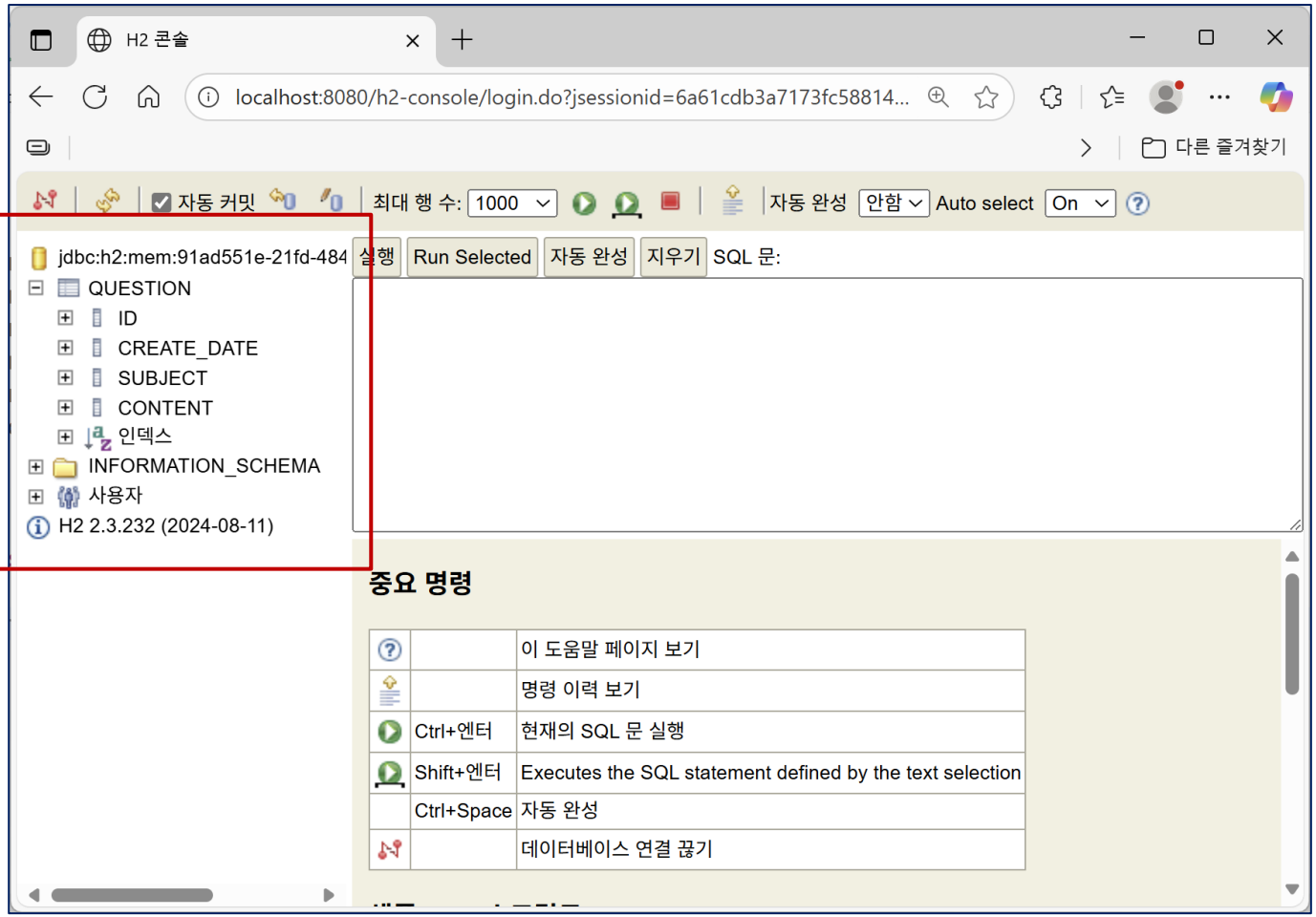
Question.java

```java
package com.example.demo.entity;

import java.time.LocalDateTime;

import jakarta.persistence.Column;
import jakarta.persistence.Entity;
import jakarta.persistence.GeneratedValue;
import jakarta.persistence.GenerationType;
import jakarta.persistence.Id;
import lombok.Data;

@Data
@Entity
public class Question {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private int id;

    @Column(length = 200)
    private String subject;

    @Column(columnDefinition = "TEXT")
    private String content;

    private LocalDateTime createDate;
}
```

# JPA로 데이터베이스 사용하기



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



# Repository로 데이터베이스 사용하기



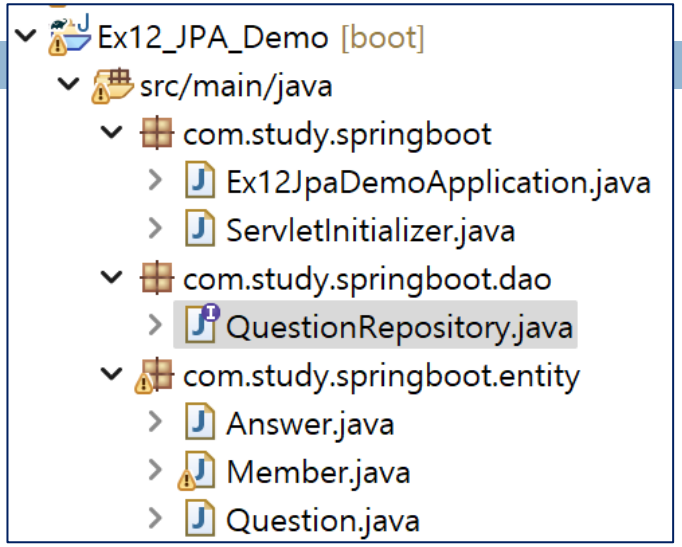
- 테이블을 구성하기 위해
  - Entity를 사용
  - 예) Question, Answer, Member 등

- **JPA Repository**
  - **테이블의 데이터**를 **저장, 조회, 수정, 삭제** 등을 지원
  - **JpaRepository 인터페이스 형식**을 이용
  - **데이터를 관리**하는 여러 **메서드**를 제공
    - findAll(), save() 등

# Repository로 데이터베이스 사용하기



- Repository 생성하기
  - **QuestionRepository** 인터페이스




**Repository는 테이블 당 하나씩**

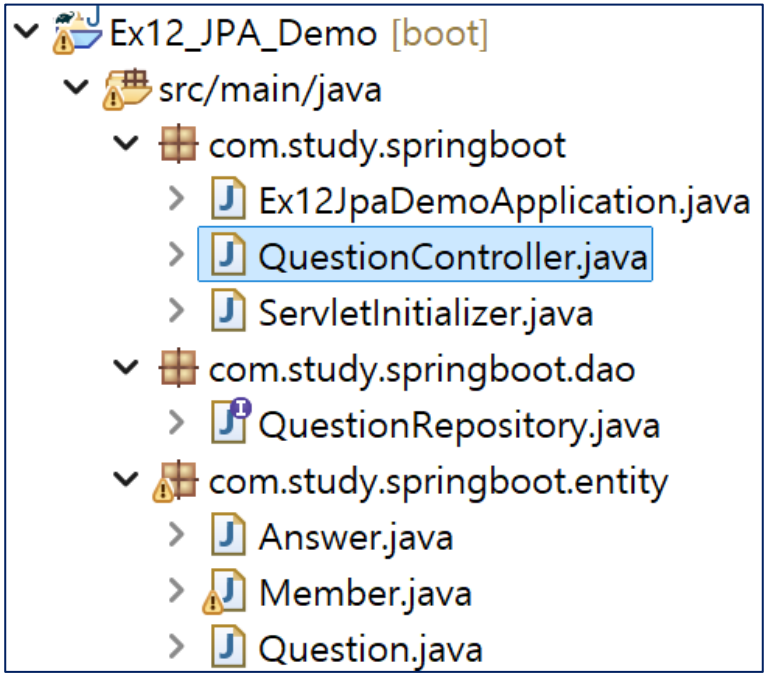
QuestionRepository.java

```
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



- Repository를 이용하여 데이터 생성하기
  - 클라이언트 요청정보(“insert”) 처리 클래스
  - **QuestionController** 클래스



# Repository로 데이터베이스 사용하기



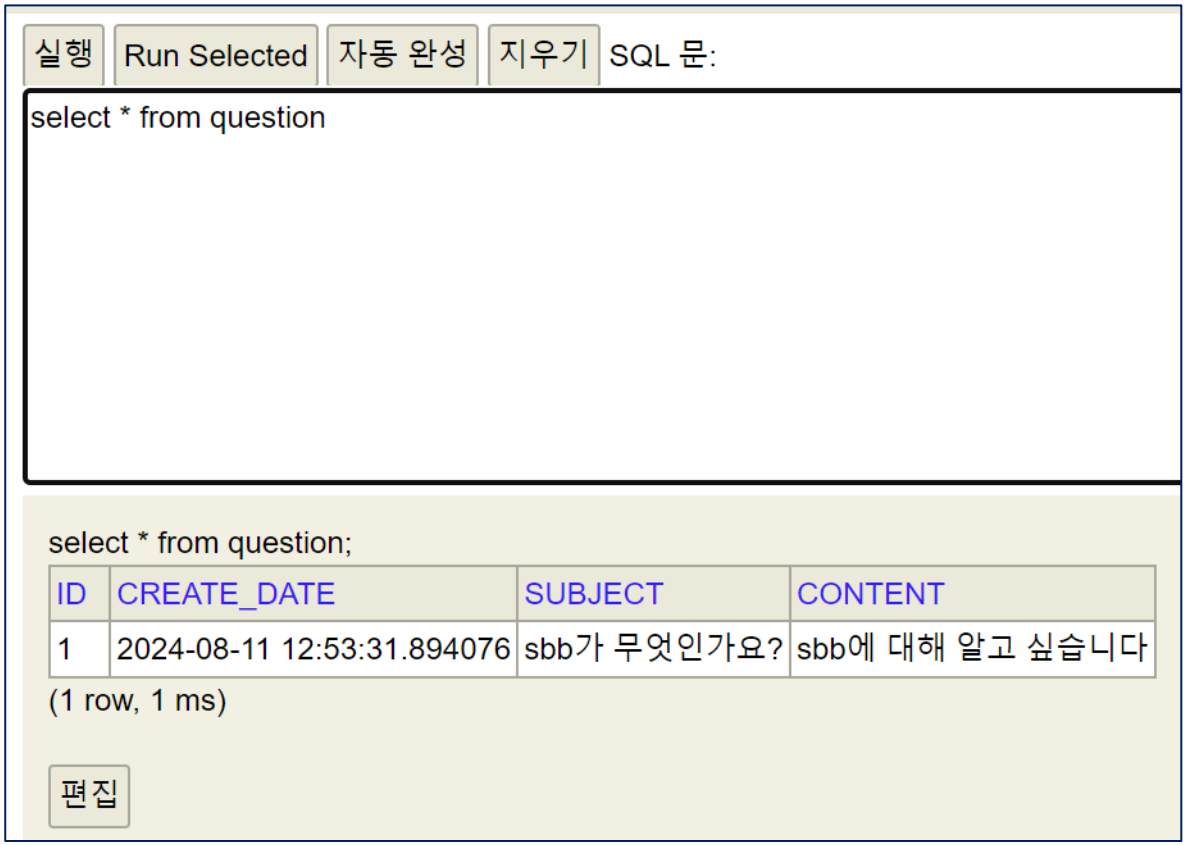
QuestionController.java

```java
package com.study.springboot;

import java.time.LocalDateTime;

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

import com.study.springboot.dao.QuestionRepository;
import com.study.springboot.entity.Question;

@RestController
public class QuestionController {
    private QuestionRepository repository;

    public QuestionController(QuestionRepository repository) {
        this.repository = repository;
    }

    @GetMapping("/insert")
    public void insert() {
        Question question = new Question();
        question.setSubject("sbb가 무엇인가요?");
        question.setContent("sbb에 대해 알고 싶습니다");
        question.setCreateDate(LocalDateTime.now());
        repository.save(question);
    }
}
```

# Repository로 데이터베이스 사용하기



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



# Repository로 데이터베이스 사용하기



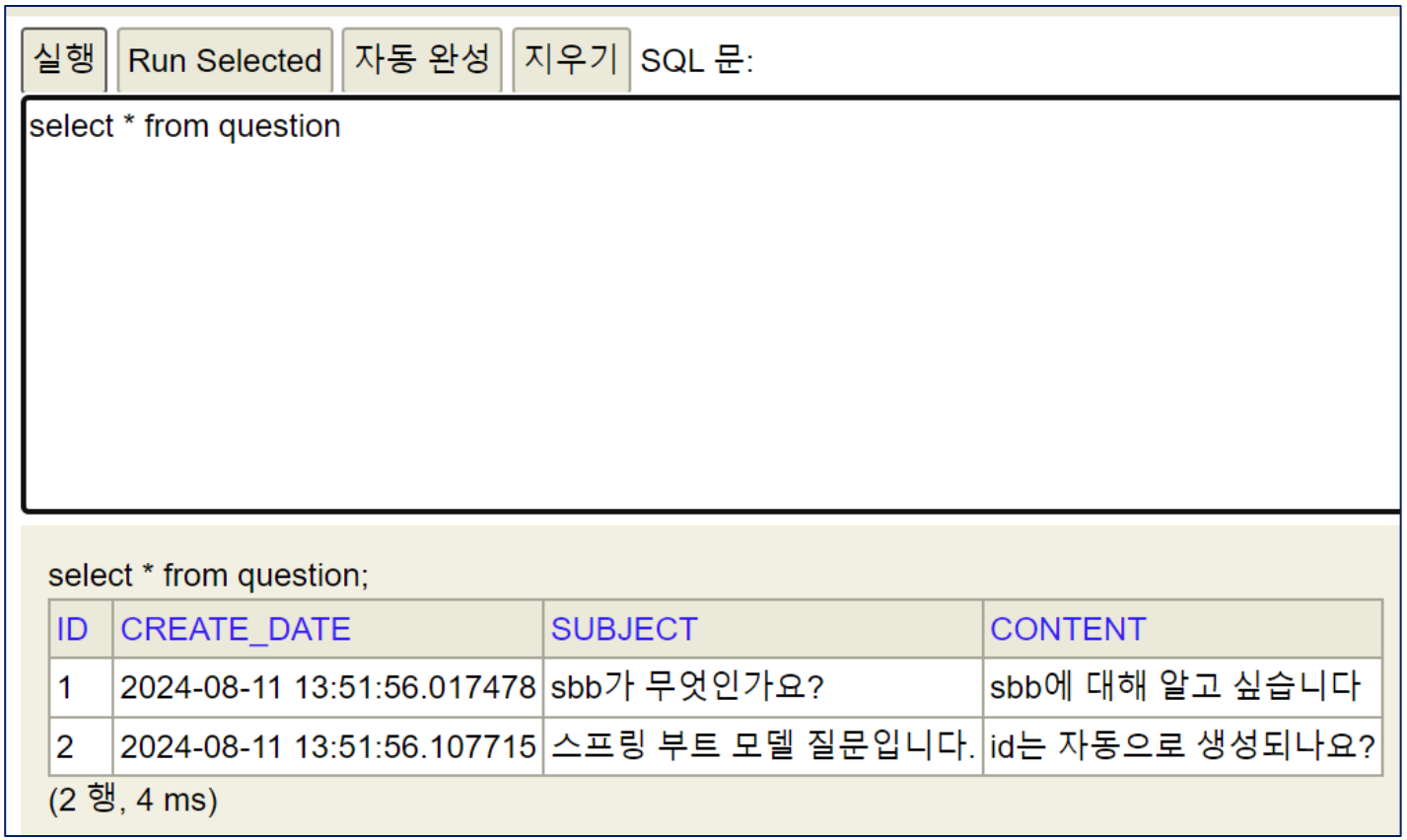
QuestionController.java

```
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



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



# JSP 기반 웹 프로젝트 구성



- 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.surfix=.jsp**
    - **JSP 폴더 생성**
      - src > main > **webapp > WEB-INF > views**
    - jsp 파일 생성
      - **views 폴더에 생성**

# JSP 기반 웹 프로젝트 구성 => JSP 설정



QuestionController.java

```
@Controller
public class QuestionController {
    private QuestionRepository repository;

    public QuestionController(QuestionRepository repository) {
        this.repository = repository;
    }

    @GetMapping("/")
    public String index() {
        return "index";
    }

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

# JSP 기반 웹 프로젝트 구성



QuestionController.java

```
    @GetMapping("/insert")
    public String insert(Question question) {
        question.setCreateDate(LocalDateTime.now());
        repository.save(question);
        return "redirect:list";
    }

    @GetMapping("/list")
    public String list(Model model) {
        List<Question> questionList = repository.findAll();
        model.addAttribute("questionList", questionList);
        return "question_list";
    }
```

# JSP 기반 웹 프로젝트 구성



- index.jsp

index.jsp

```jsp
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Insert title here</title>
<link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/css/bootstrap.min.css" rel="stylesheet"
integrity="sha384-QWTKZyjpPEjISv5WaRU9OFeRpok6YctnYmDr5pNlyT2bRjXh0JMhjY6hW+ALEwIH" crossorigin="anonymous">
</head>
<body>
<div class="container" style="padding: 20px;">
    <h3> ♠ 게시판 프로그램 ♠</h3>
    <br>
    <a href="/insertForm" class="btn btn-primary">등록</a>
    <a href="/list" class="btn btn-primary">조회</a>
</div>


<script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/js/bootstrap.bundle.min.js"
integrity="sha384-YvpcrYf0tY3lHB60NNkmXc5s9fDVZLESaAA55NDzOxhy9GkcIdslK1eN7N6jIeHz" crossorigin="anonymous"></script>
</body>
</html>
```

# JSP 기반 웹 프로젝트 구성



- insertForm.jsp

insertForm.jsp

```
</head>
<body>

<div class="container" style="padding: 20px;">
    <h3> ♠ 게시판 등록 ♠</h3>
    <br>
    <form action="/insert">
        제 목 : <input type="text" name="subject"> <br>
        내 용 : <br>
                <textarea rows="5" cols="30" name="content"></textarea><br><br>
        <input type="submit" value="등록">
    </form>
</div>

```

# JSP 기반 웹 프로젝트 구성



- question_list.jsp

question_list.jsp

```jsp
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
<!DOCTYPE html><html><head><meta charset="UTF-8">
<title>Insert title here</title>
<link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/css/bootstrap.min.css" rel="styles
integrity="sha384-QWTKZyjpPEjISv5WaRU9OFeRpok6YctnYmDr5pNlyT2bRjXh0JMhjY6hW+ALEwIH" crossorigin=
</head><body>
<div class="container" >
    <h3> ♠ 게시판 목록 ♠</h3>
    <br>
    <table border="1" cellpadding="5">
        <tr>
            <th>ID</th>
            <th>Subject</th>
            <th>Content</th>
            <th>CreateDate</th>
        </tr>
        <c:forEach var="q" items="${questionList }">
            <tr>
                <td>${q.id } </td>
                <td>${q.subject } </td>
                <td>${q.content } </td>
                <td>${q.createDate } </td>
            </tr>
        </c:forEach>
    </table>

    <br><br>
    <a href="/insertForm" class="btn btn-primary">입력화면</a>
    <a href="/" class="btn btn-primary">초기화면</a>
</div>
```

# 과제물



- **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



- 제출 형식
  - **제출 파일 : 학번_이름.pdf**
  - **ppt로 작업하고 pdf 형식으로 저장 후 제출**