

1장. 개발 환경 구축

- 1. 웹 애플리케이션 개발을 위해서 사용하는 스프링부트의 특징 확인
- 2. 개발 환경 구축을 위한 자바, 인텔리제이, MySQL을 설치

1.1 스프링 부트의 특징

1. 내장 서버를 이용해 별도의 설정 없이 독립 실행이 가능한 스프링 애플리케이션
 2. 톰캣, 제티 또는 언더토우와 같은 웹 애플리케이션서버(WAS) 자체 내장
 3. 빌드 구성을 단순화하기 위한 'Spring Boot Starter' 의존성 제공
 4. XML 설정 없이 단순 자바 수준의 설정 방식 제공
 5. JAR를 이용해 자바 옵션만으로 배포 가능
 6. 애플리케이션의 모니터링과 관리를 위한 스프링 액추에이터 제공
-

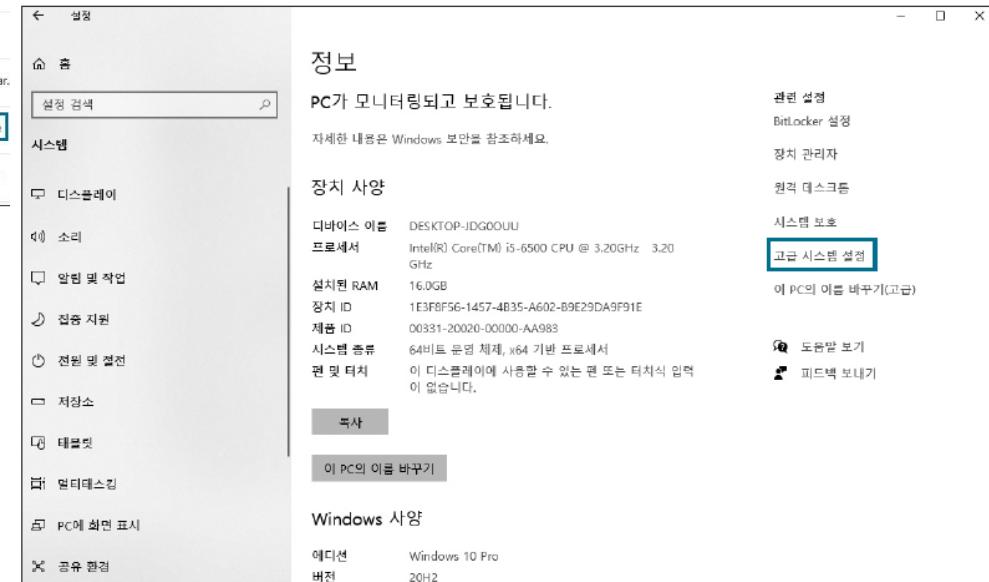
1.2 JDK 설치

Java SE Development Kit 11.0.10

This software is licensed under the Oracle Technology Network License Agreement for Oracle Java SE

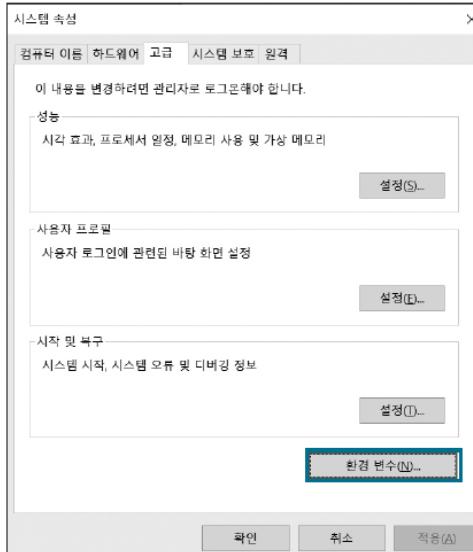
Product / File Description	File Size	Download
Linux ARM 64 Debian Package	145.64 MB	*jdk-11.0.10_linux-aarch64_bin.deb
Linux ARM 64 RPM Package	152.22 MB	*jdk-11.0.10_linux-aarch64_bin.rpm
Linux ARM 64 Compressed Archive	169.37 MB	*jdk-11.0.10_linux-aarch64_bin.tar.gz
Linux x64Debian Package	149.39 MB	*jdk-11.0.10_linux-x64_bin.deb
Linux x64 RPM Package	156.12 MB	*jdk-11.0.10_linux-x64_bin.rpm
Linux x64 Compressed Archive	173.31 MB	*jdk-11.0.10_linux-x64_bin.tar.gz
macOS Installer	167.51 MB	*jdk-11.0.10_osx-x64_bin.dmg
macOS Compressed Archive	167.84 MB	*jdk-11.0.10_osx-x64_bin.tar.gz
Solaris SPARC Compressed Archive	184.82 MB	*jdk-11.0.10_solaris-sparcv9_bin.tar.gz
Windows x64 Installer	152.32 MB	*jdk-11.0.10_windows-x64_bin.exe
Windows x64 Compressed Archive	171.67 MB	*jdk-11.0.10_windows-x64_bin.zip

[그림 1-1] JDK 11 다운로드

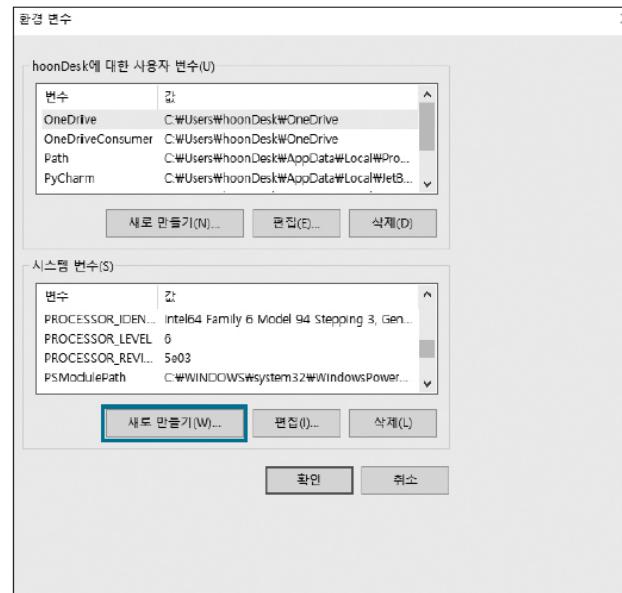


[그림 1-2] 자바 환경 변수 설정 1단계

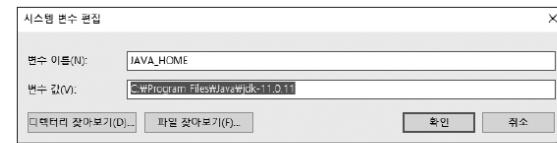
JDK 설치



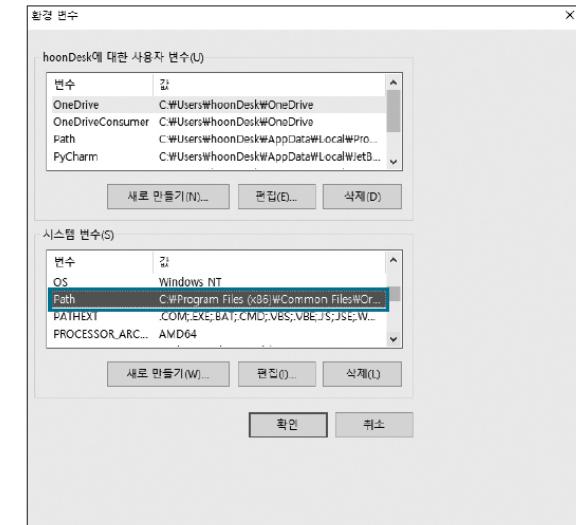
[그림 1-3] 자바 환경 변수 설정 2단계



[그림 1-4] 자바 환경 변수 설정 3단계



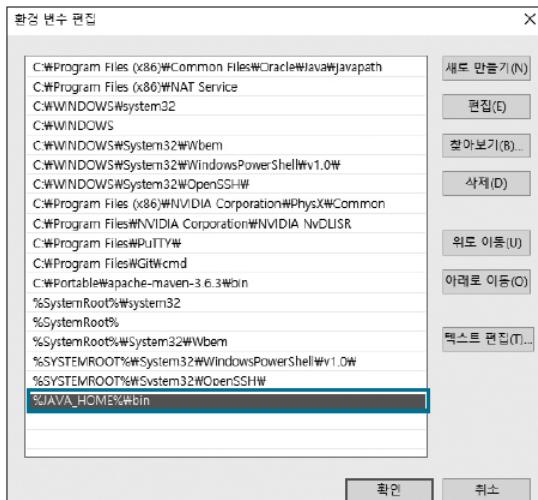
[그림 1-5] 자바 환경 변수 설정 4단계



[그림 1-6] 자바 환경 변수 설정 5단계

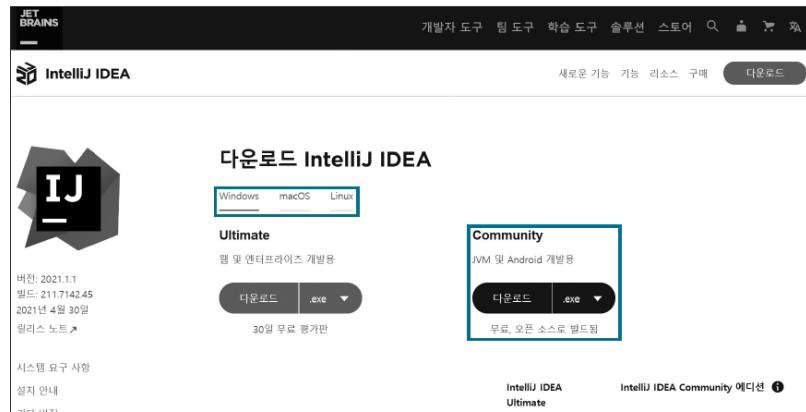
커맨드 창에서 최종확인

```
java -version
```



[그림 1-7] 자바 환경 변수 설정 6단계

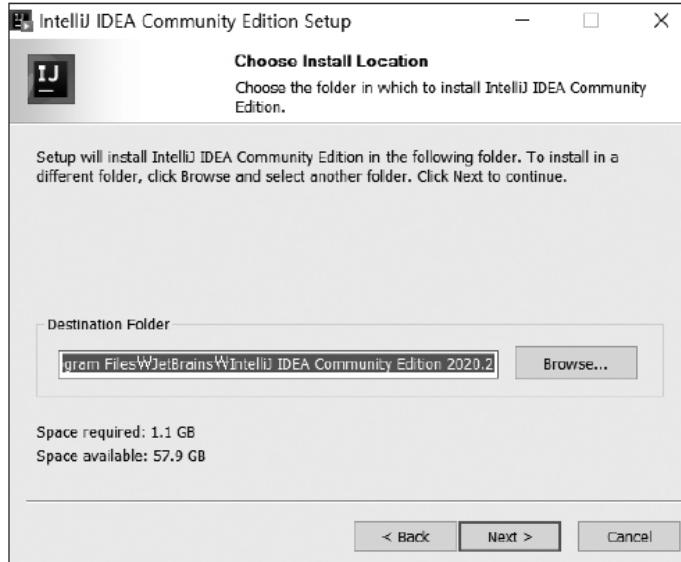
1.3 인텔리제이 설치



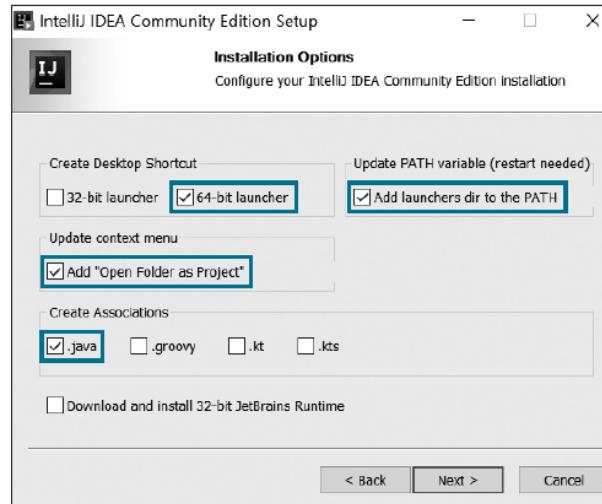
[그림 1-9] IntelliJ Community 버전 다운로드



[그림 1-10] IntelliJ Community 버전 설치 1단계

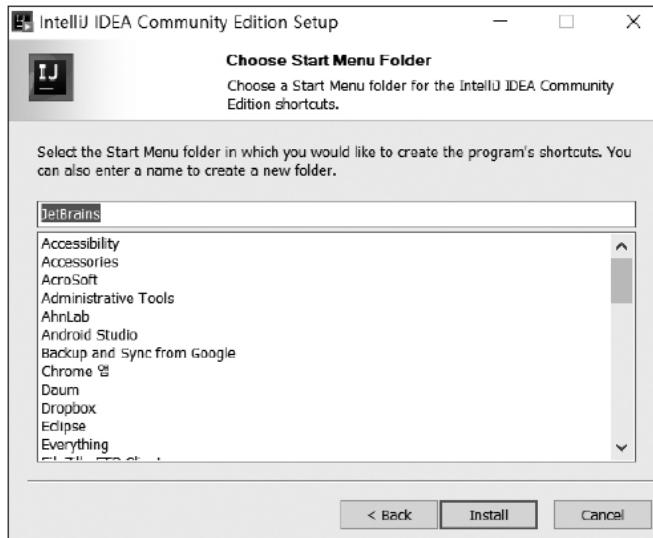


[그림 1-11] IntelliJ Community 버전 설치 2단계

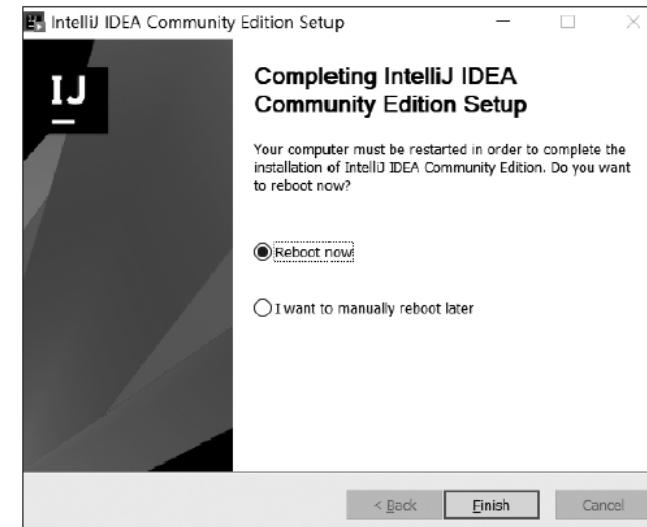


[그림 1-12] IntelliJ Community 버전 설치 3단계

인텔리제이 설치



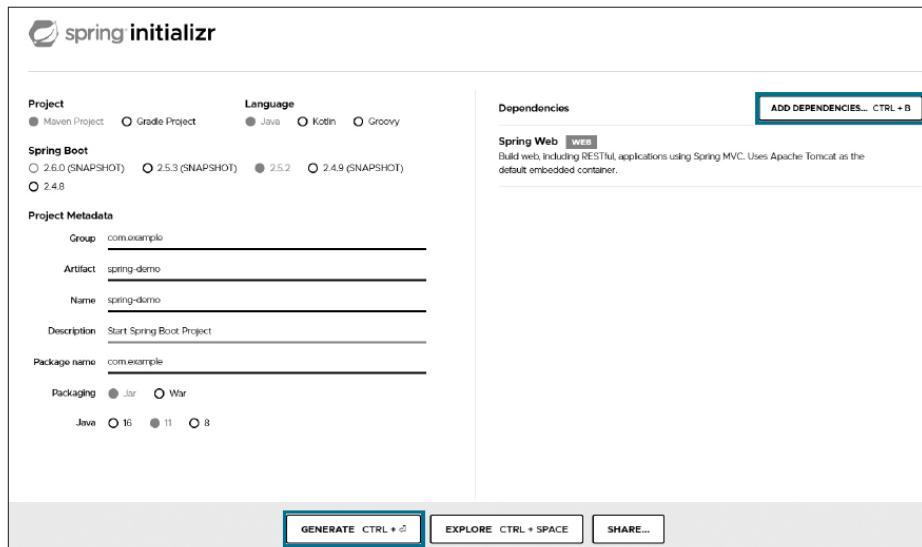
[그림 1-13] IntelliJ Community 버전 설치 4단계



[그림 1-14] IntelliJ Community 버전 설치 5단계

1.4 애플리케이션 실행하기

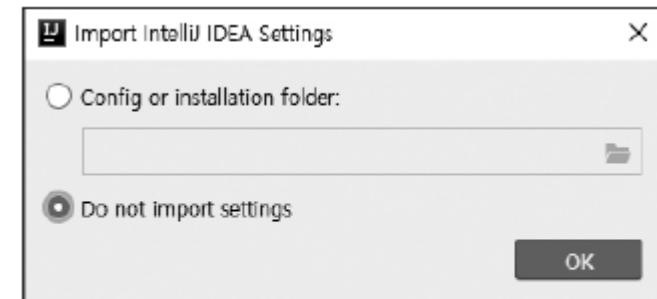
1.4.1 Spring Boot Project 생성하기



[그림 1-15] 스프링부트 프로젝트 생성 1단계

프로젝트 설정

- 빌드 툴 - 메이븐
- 언어 - Java 11
(기존에 사용하던 Java 1.8 이상의 버전이 있을 경우 해당 버전 선택 가능)
- 스프링 부트 버전: 2.5.2
- 패키징: Jar
- 의존성: Spring Web



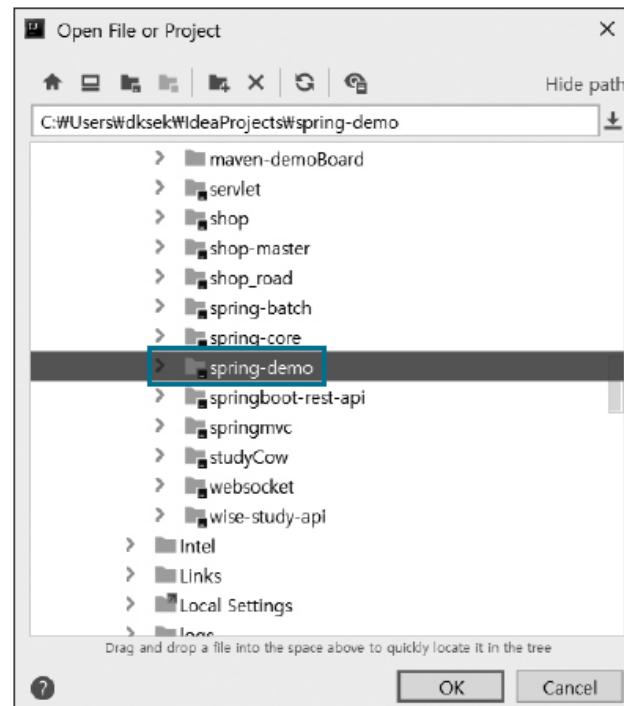
[그림 1-16] 스프링부트 프로젝트 생성 2단계

애플리케이션 실행하기

1.4.1 Spring Boot Project 생성하기



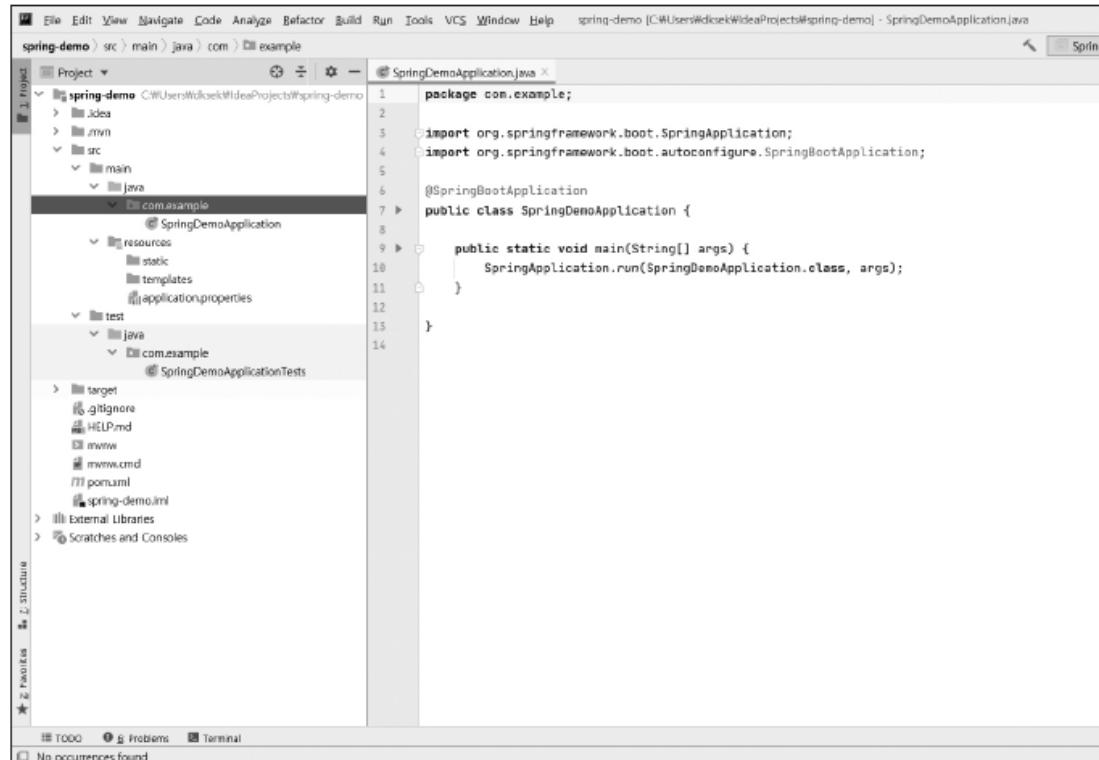
[그림 1-17] 스프링부트 프로젝트 생성 3단계



[그림 1-18] 스프링부트 프로젝트 생성 4단계

애플리케이션 실행하기

1.4.1 Spring Boot Project 생성하기

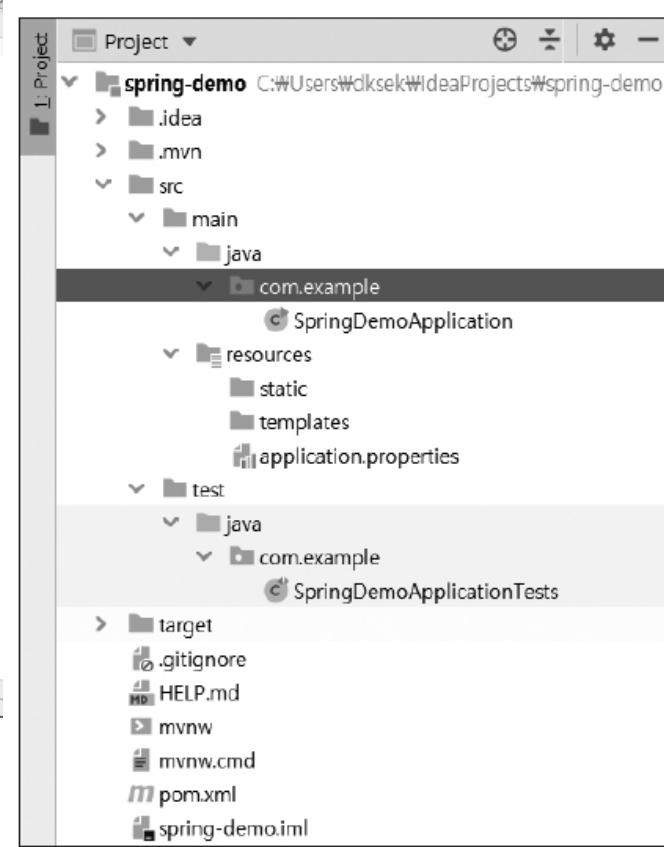


The screenshot shows the IntelliJ IDEA interface with the following details:

- File Path:** spring-demo > src > main > java > com > example
- Code Editor:** SpringDemoApplication.java

```
1 package com.example;
2
3 import org.springframework.boot.SpringApplication;
4 import org.springframework.boot.autoconfigure.SpringBootApplication;
5
6 @SpringBootApplication
7 public class SpringDemoApplication {
8
9     public static void main(String[] args) {
10         SpringApplication.run(SpringDemoApplication.class, args);
11     }
12 }
```

[그림 1-19] 스프링부트 프로젝트 생성 완료



[그림 1-20] 프로젝트 패키지 구조

애플리케이션 실행하기

1.4.2 빌드 도구

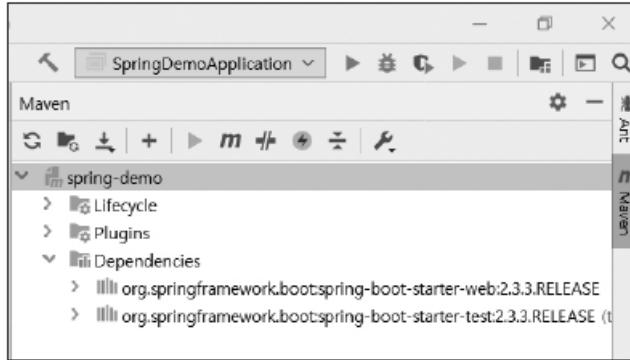
- ① 스프링부트 최상위 모듈로서 스프링부트에 필요한 의존성(dependency)를 자동으로 추가
- ② 웹 애플리케이션에 필요한 라이브러리
- ③ Spring Test Framework 라이브러리

pom.xml

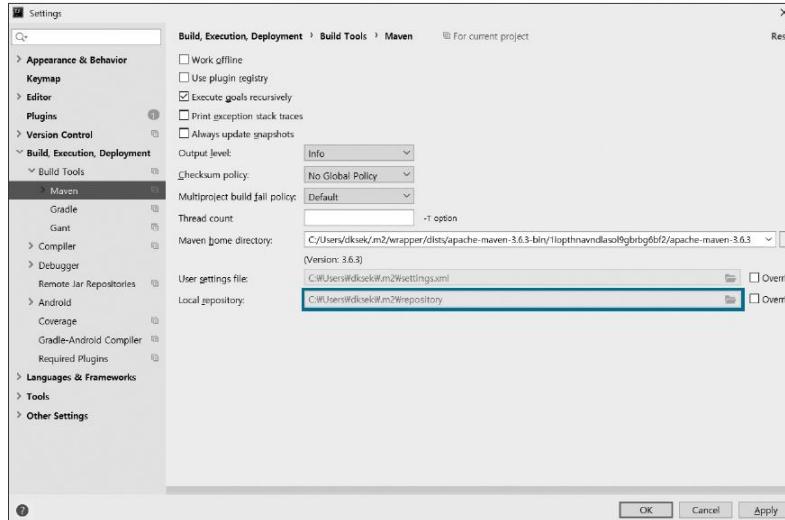
```
01 <?xml version="1.0" encoding="UTF-8"?>
02 <project xmlns="http://maven.apache.org/POM/4.0.0"
03   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
04   xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
05     https://maven.apache.org/xsd/maven-4.0.0.xsd">
06   <modelVersion>4.0.0</modelVersion>
07   <parent> ..... ①
08     <groupId>org.springframework.boot</groupId>
09     <artifactId>spring-boot-starter-parent</artifactId>
10     <version>2.5.2</version> ..... 22
11     <relativePath/> <!-- lookup parent from repo: 23
12       <dependency> ..... 24
13         <artifactId>spring-boot-starter-web</artifactId>
14       </dependency> ..... 25
15     </parent> ..... 26
16     <groupId>com.example</groupId>
17     <artifactId>spring-demo</artifactId>
18     <version>0.0.1-SNAPSHOT</version>
19     <name>spring-demo</name>
20     <description>Start Spring Boot Project</descript: 29
21     <properties>
22       <java.version>11</java.version>
23     </properties>
24     <dependencies>
25       <dependency> ..... 26
26         <groupId>org.springframework.boot</groupId>
27         <artifactId>spring-boot-starter-test</artifactId>
28         <scope>test</scope>
29       </dependency> ..... 27
30     </dependencies> ..... 28
31   </parent> ..... 29
32   <groupId>org.springframework.boot</groupId>
33   <artifactId>spring-boot-starter-test</artifactId>
34   <version>2.5.2</version>
35   <dependencies>
36     <dependency> ..... 30
37       <groupId>org.springframework.boot</groupId>
38       <artifactId>spring-boot-starter-test</artifactId>
39       <version>2.5.2</version>
40     </dependency> ..... 31
41   </dependencies> ..... 32
42 
```

애플리케이션 실행하기

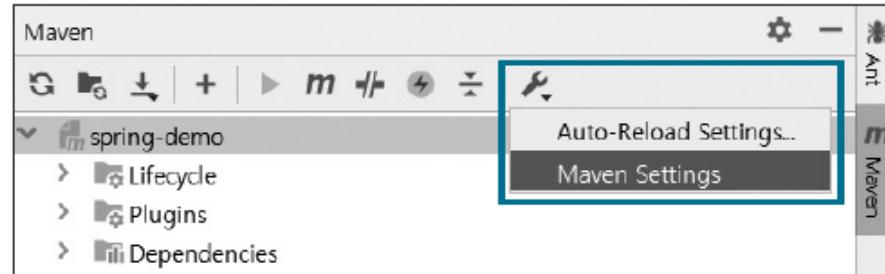
1.4.2 빌드 도구



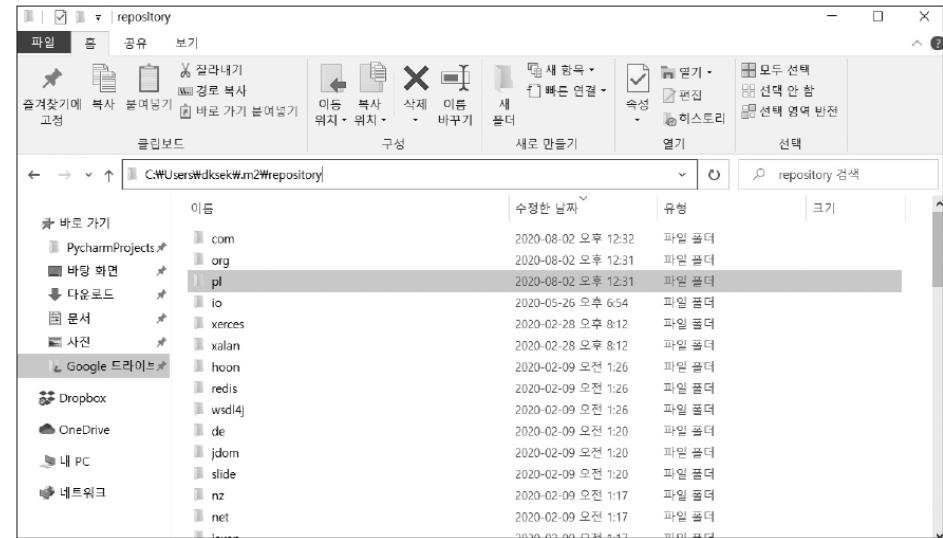
[그림 1-21] maven dependency 확인



[그림 1-23] local maven repository 위치 확인



[그림 1-22] maven settings



[그림 1-24] local maven repository

애플리케이션 실행하기

1.4.3 설정 파일(application.properties)

application-{profile}.properties



[함께 해봐요 1-1] application.properties 설정하기

```
01 server.port = 80 ..... ①  
02 application.name = spring-demo ..... ②
```

애플리케이션 실행하기



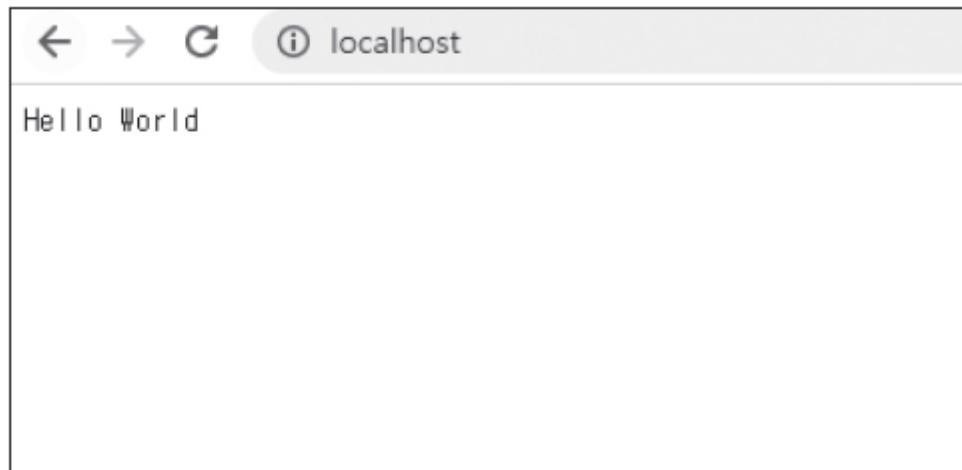
[함께 해봐요 1-2] Hello World 출력하기

com.example.SpringDemoApplication.java

```
01 package com.example;
02
03 import org.springframework.boot.SpringApplication;
04 import org.springframework.boot.autoconfigure.SpringBootApplication;
05 import org.springframework.web.bind.annotation.GetMapping;
06 import org.springframework.web.bind.annotation.RestController;
07
08 @RestController
09 @SpringBootApplication
10 public class SpringDemoApplication {
11
12     public static void main(String[] args) {
13         SpringApplication.run(SpringDemoApplication.class, args);
14     }
15
16     @GetMapping(value = "/")
17     public String HelloWorld(){
18         return "Hello World";
19     }
20
21 }
```

애플리케이션 실행하기

- 웹 브라우저 url 창에 `http://localhost`를 입력하면 “Hello World”라는 문자열 출력



[그림 1-28] Hello World 예제 실행 결과

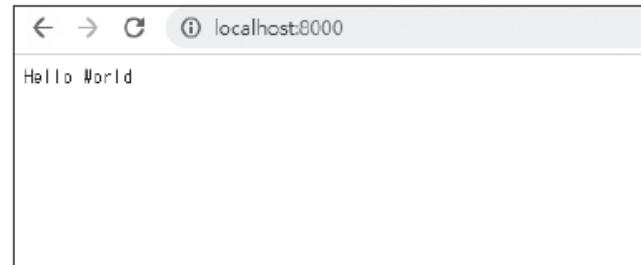
애플리케이션 실행하기

- application.properties 포트 번호 수정



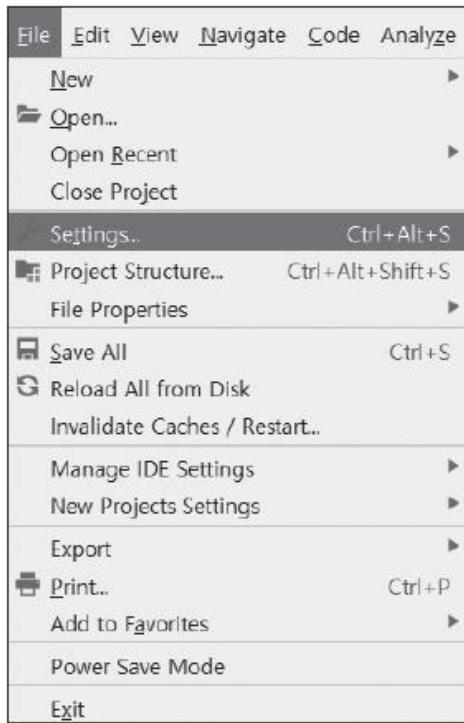
[함께 해봐요 1-3] 애플리케이션 포트 변경하기

```
server.port = 8000
```

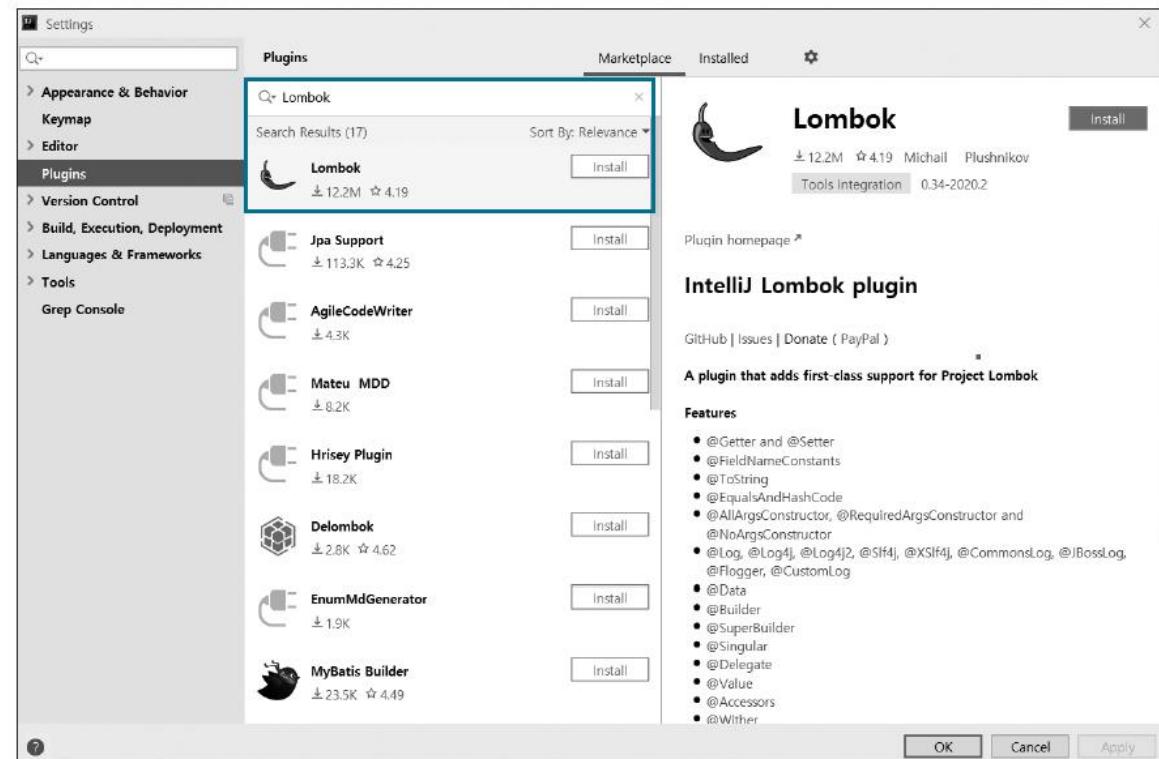


[그림 1-29] 애플리케이션 포트 번호 변경 결과

1.5 Lombok 라이브러리

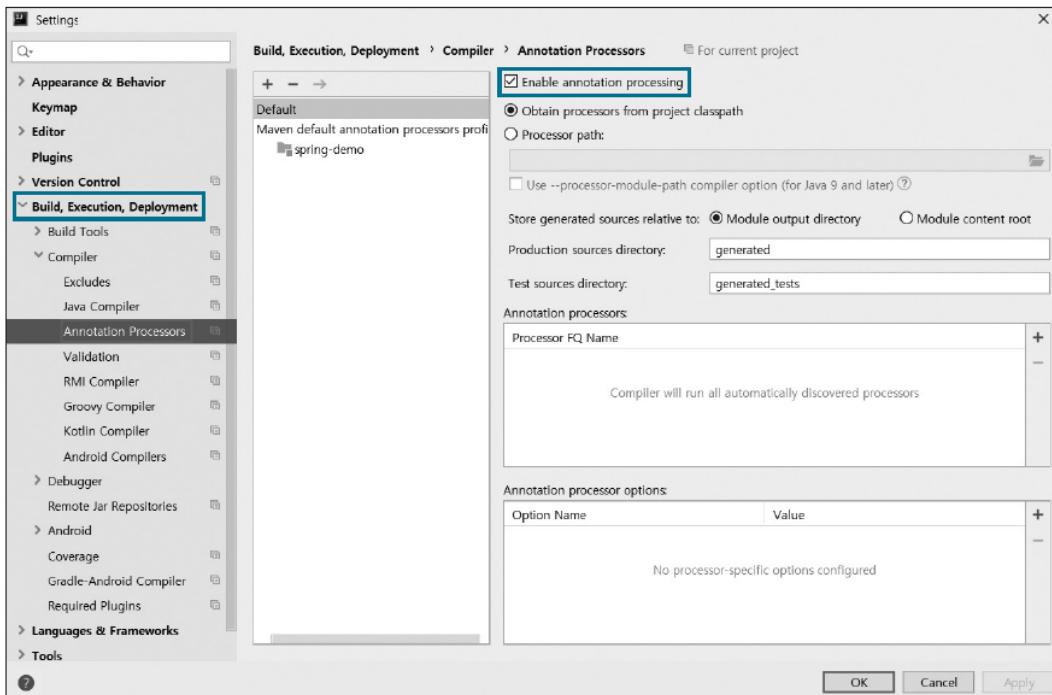


[그림 1-30] Lombok 라이브러리 설치 1단계



[그림 1-31] Lombok 라이브러리 설치 2단계

Lombok 라이브러리



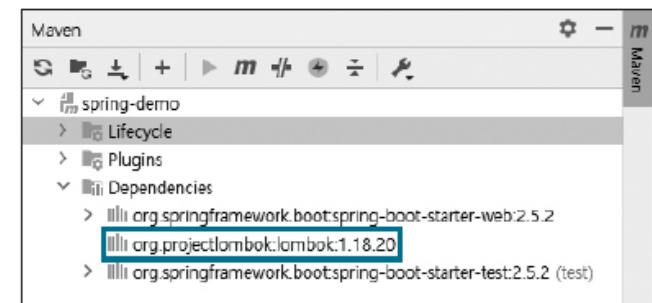
[그림 1-32] Lombok 라이브러리 설치 3단계

```
<dependencies>
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-web</artifactId>
    </dependency>

    <dependency>
        <groupId>org.projectlombok</groupId>
        <artifactId>lombok</artifactId>
    </dependency>

    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-test</artifactId>
        <scope>test</scope>
    </dependency>
</dependencies>
```

[그림 1-33] Lombok 라이브러리 설치 4단계



[그림 1-34] Lombok 라이브러리 설치 완료

Lombok 라이브러리

자주 사용하는 어노테이션

어노테이션	설명
@Getter/Setter	코드를 컴파일할 때 속성들에 대한 Getter/Setter 메소드 생성
@ToString	toString() 메소드 생성
@ToString(exclude={"변수명"})	원하지 않는 속성을 제외한 toString() 메소드 생성
@NonNull	해당 변수가 null 체크. NullPointerException 예외 발생
@EqualsAndHashCode	equals()와 hashCode() 메소드 생성
@Builder	빌더 패턴을 이용한 객체 생성
@NoArgsConstructor	파라미터가 없는 기본 생성자 생성
@AllArgsConstructor	모든 속성에 대한 생성자 생성
@RequiredArgsConstructor	초기화되지 않은 Final, @NonNull 어노테이션이 붙은 필드에 대한 생성자 생성
@Log	log 변수 자동 생성
@Value	불변(immutable) 클래스 생성
@Data	@ToString, @EqualsAndHashCode, @Getter, @Setter, @RequiredArgsConstructor를 합친 어노테이션

Lombok 라이브러리



[함께 해봐요 1-4] Lombok 라이브러리 적용하기

com.example.UserDto.java

```
01 package com.example;
02
03 import lombok.Getter;
04 import lombok.Setter;
05 import lombok.ToString;
06
07 @Getter
08 @Setter
09 @ToString
10 public class UserDto {
11
12     private String name;
13     private Integer age;
14
15 }
```

Lombok 라이브러리

com.example.TestController.java

```
01 package com.example;
02
03 import org.springframework.web.bind.annotation.GetMapping;
04 import org.springframework.web.bind.annotation.RestController;
05
06 @RestController
07 public class TestController {
08
09     @GetMapping(value = "/test")
10     public UserDto test(){
11
12         UserDto userDto = new UserDto();
13         userDto.setAge(20);
14         userDto.setName("hoon");
15
16         return userDto;
17     }
18 }
```

Lombok 라이브러리

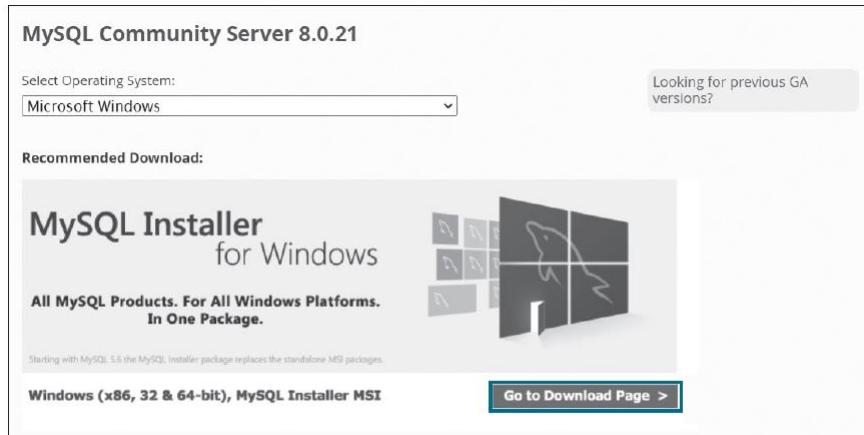


The screenshot shows a browser window with the address bar set to `localhost:8000/test`. The page content displays the following JSON response:

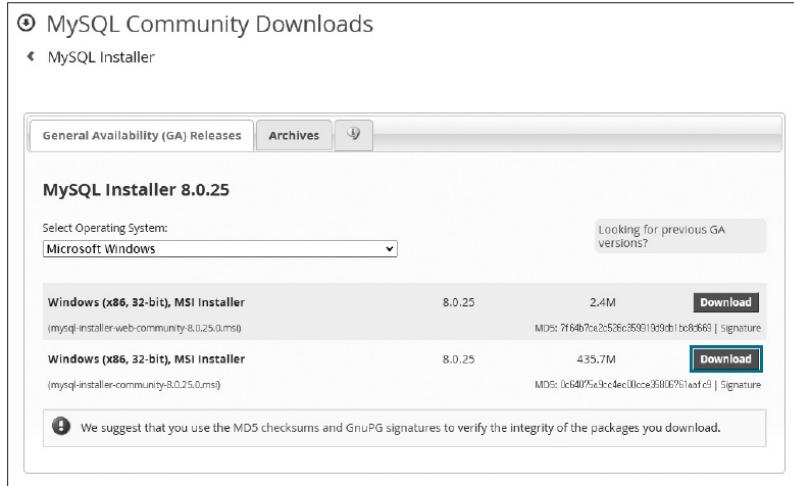
```
1 // 20210714211542
2 // http://localhost:8000/test
3
4 {
5     "name": "hoon",
6     "age": 20
7 }
```

[그림 1-35] Lombok 라이브러리 적용 후 유저 정보 출력

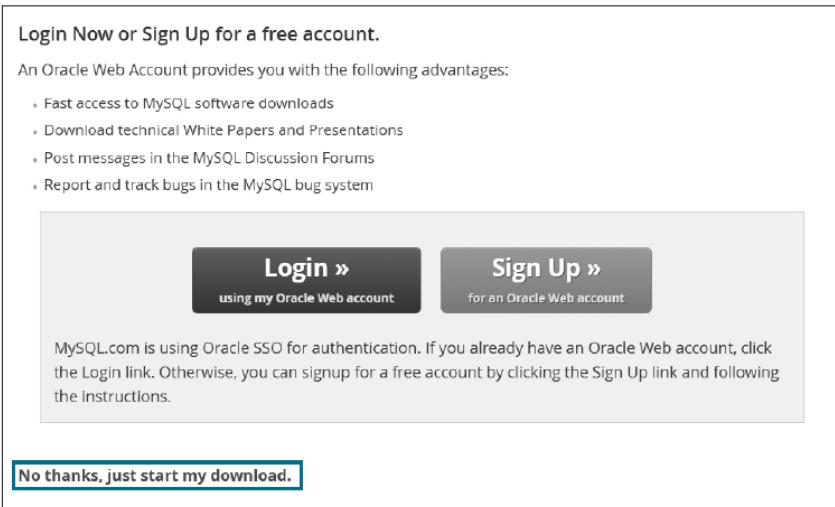
1.6 MySQL 설치하기



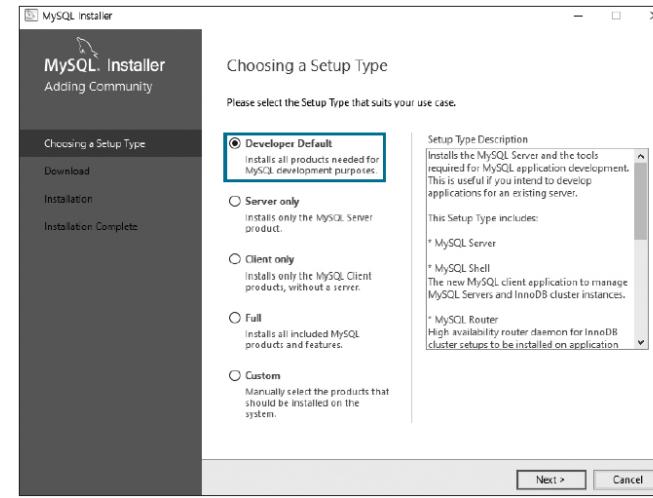
[그림 1-37] MySQL 설치 1단계



[그림 1-38] MySQL 설치 2단계

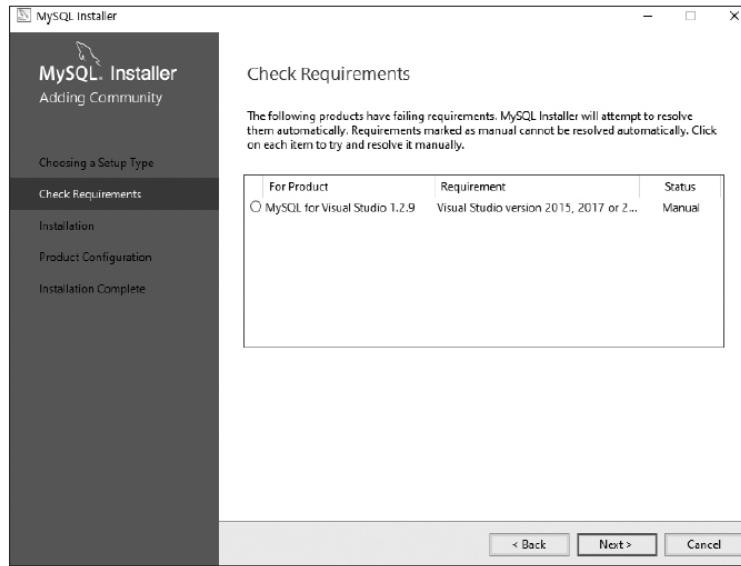


[그림 1-39] MySQL 설치 3단계

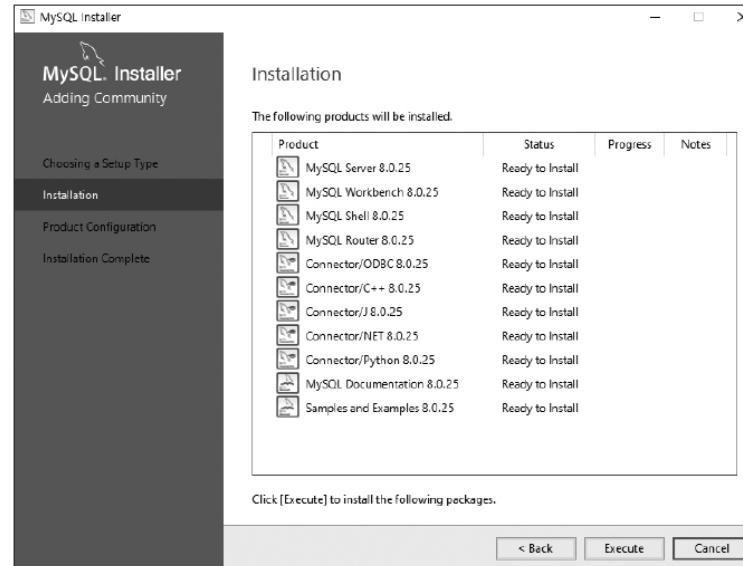


[그림 1-40] MySQL 설치 4단계

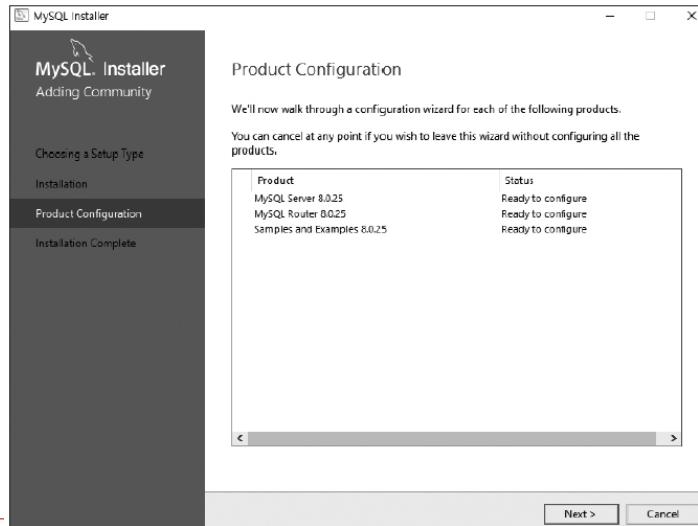
1.6 MySQL 설치하기



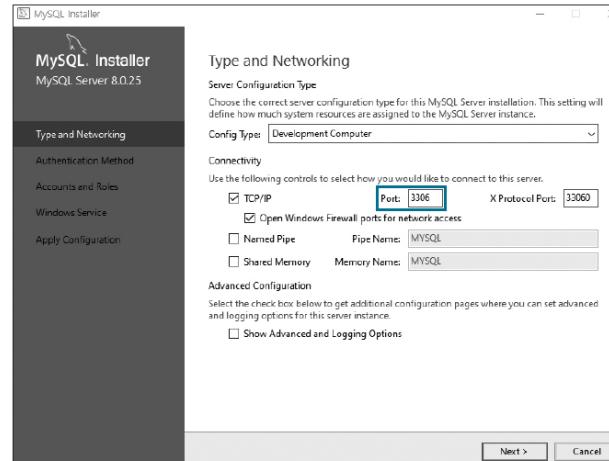
[그림 1-41] MySQL 설치 5단계



[그림 1-42] MySQL 설치 6단계

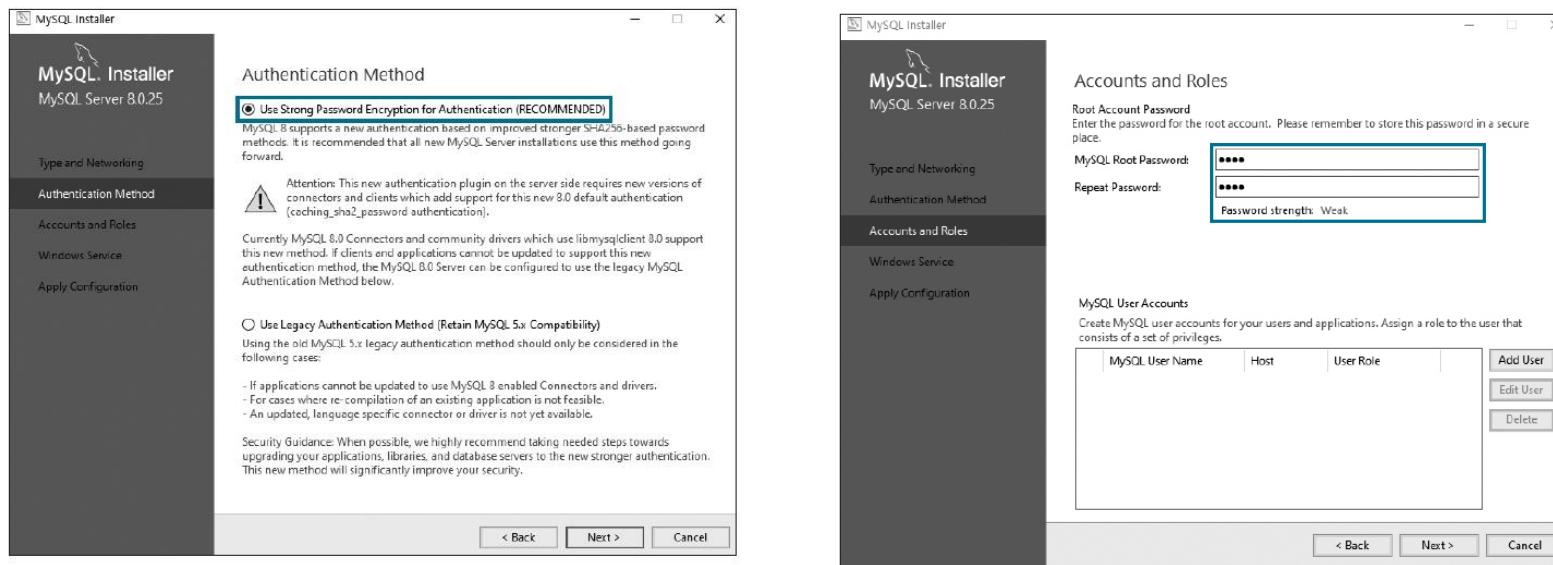


[그림 1-43] MySQL 설치 7단계

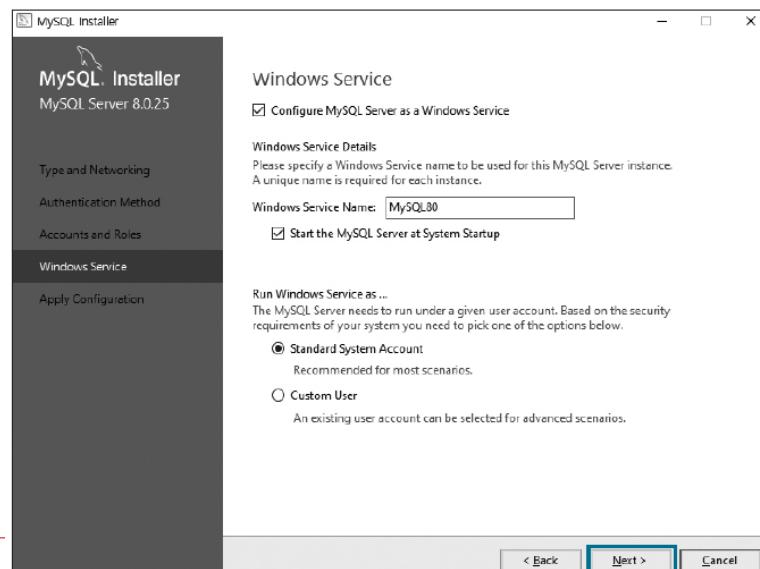


[그림 1-44] MySQL 설치 8단계

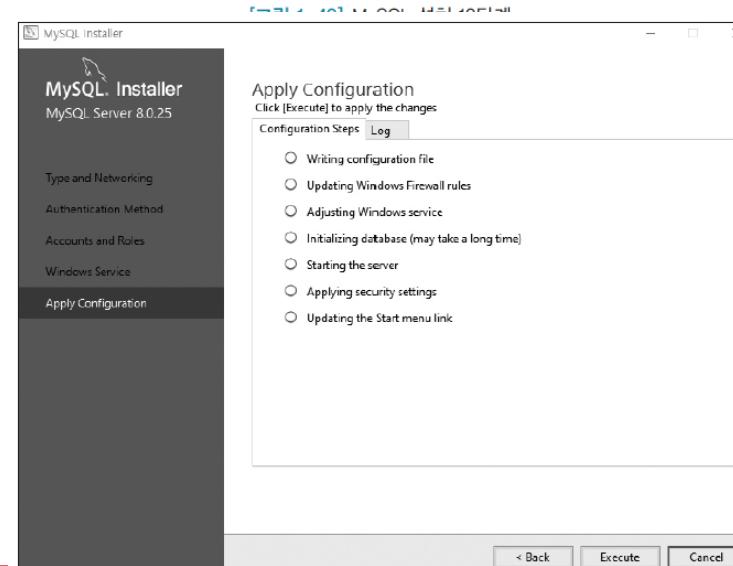
1.6 MySQL 설치하기



[그림 1-45] MySQL 설치 9단계

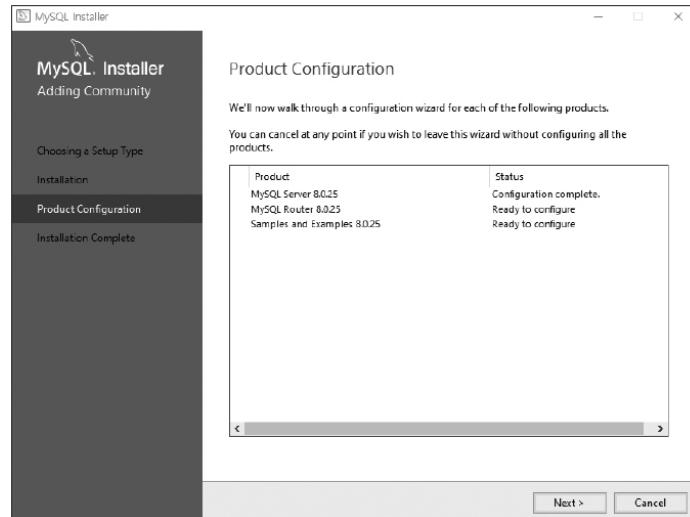


[그림 1-47] MySQL 설치 11단계

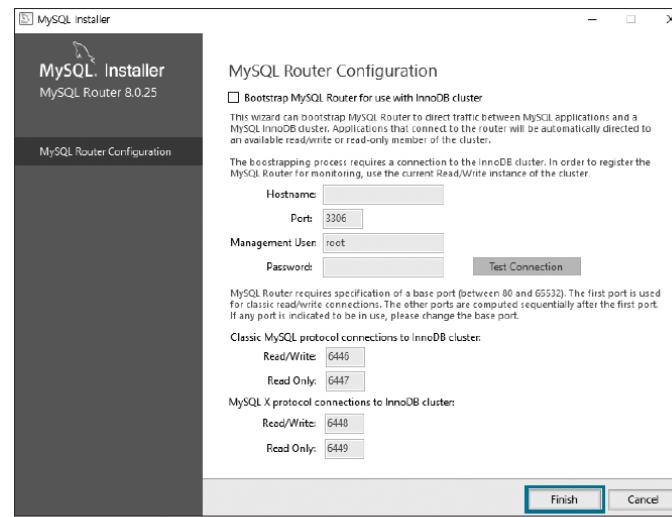


[그림 1-48] MySQL 설치 12단계

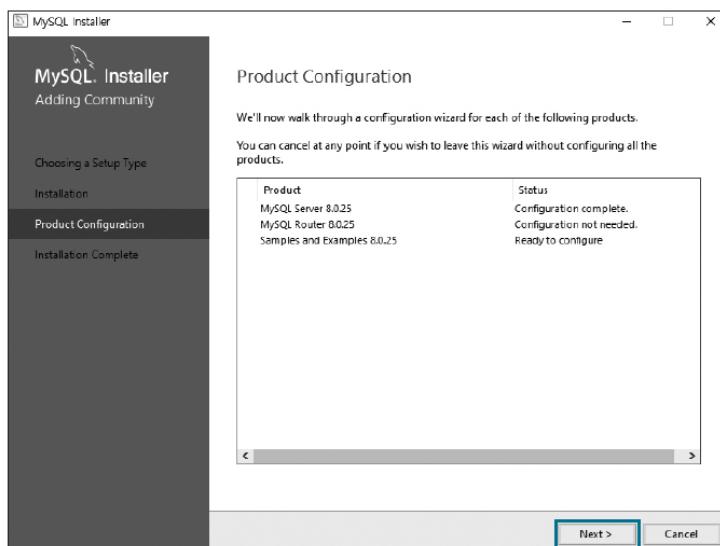
1.6 MySQL 설치하기



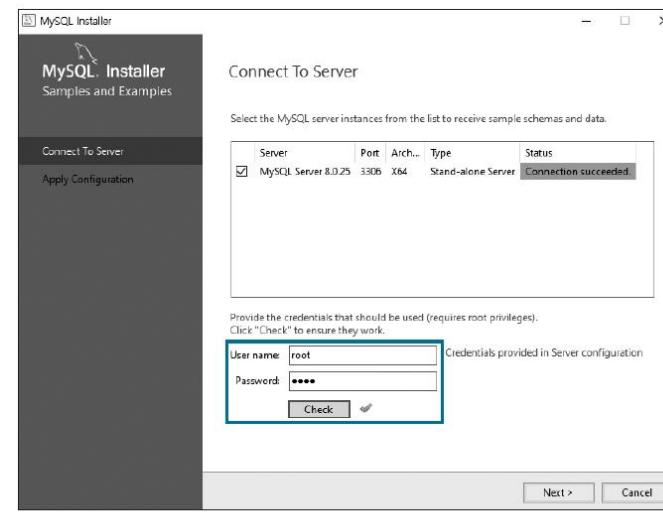
[그림 1-49] MySQL 설치 13단계



[그림 1-50] MySQL 설치 14단계

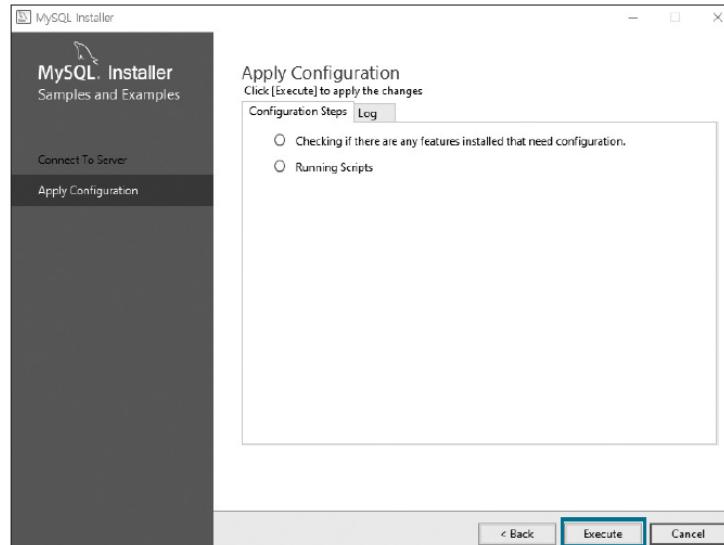


[그림 1-51] MySQL 설치 15단계

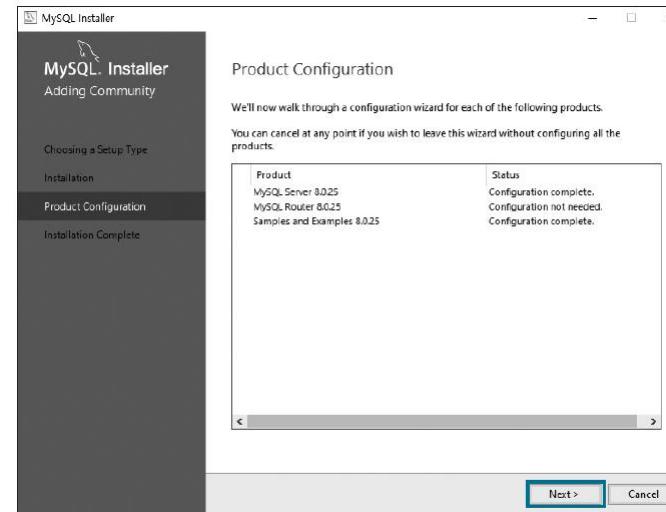


[그림 1-52] MySQL 설치 16단계

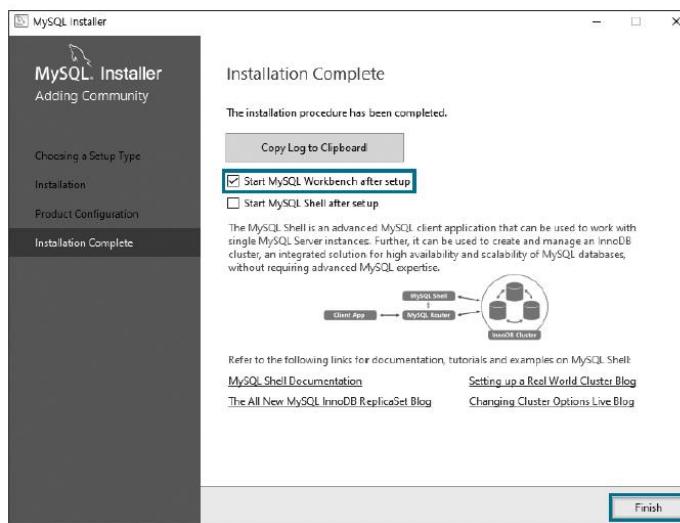
1.6 MySQL 설치하기



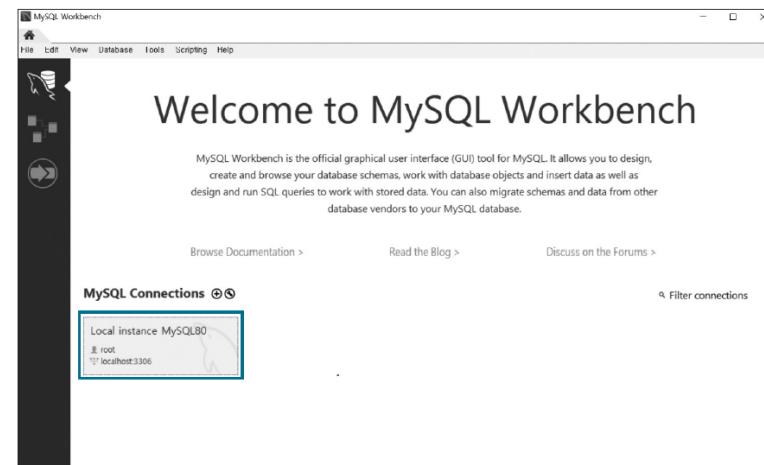
[그림 1-53] MySQL 설치 17단계



[그림 1-54] MySQL 설치 18단계



[그림 1-55] MySQL 설치 19단계

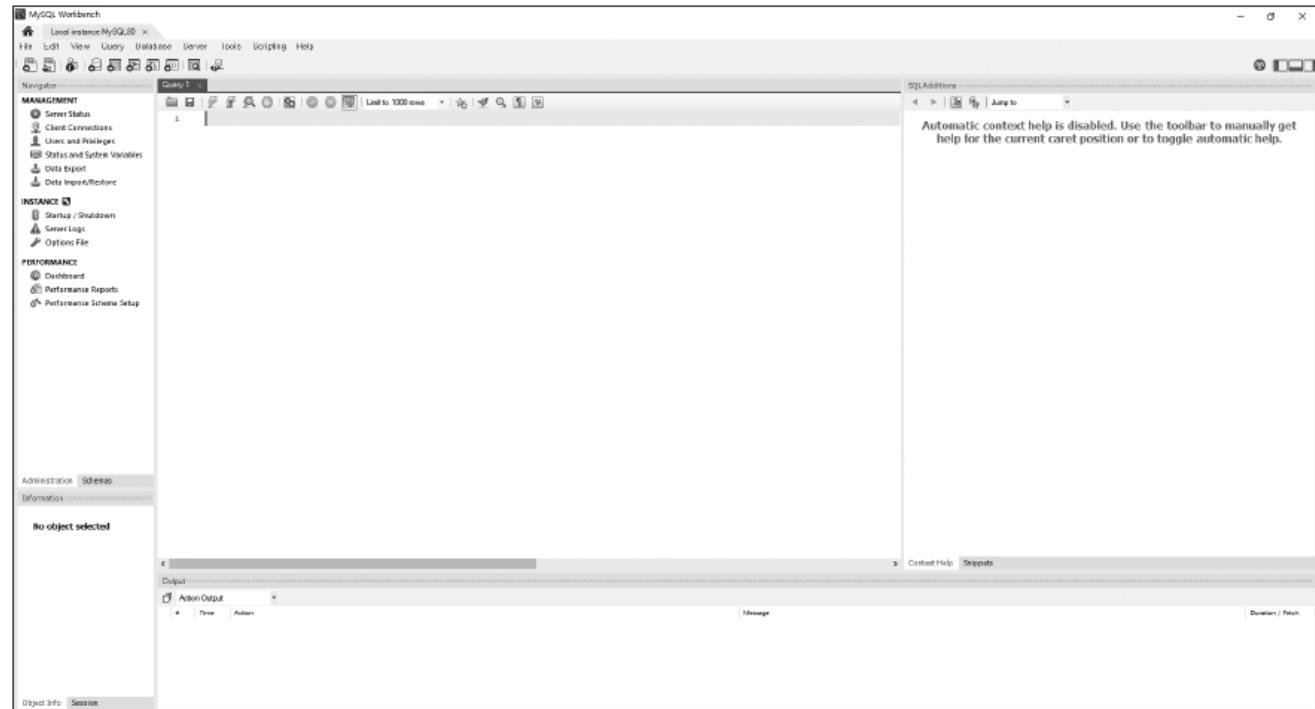


[그림 1-56] MySQL 접속

1.6 MySQL 설치하기



[그림 1-57] MySQL 로그인



[그림 1-58] MySQL 로그인 성공