

# 1장 스프링 부트 시작하기

출처: 코딩 자율학습 스프링부트3 자바 백엔드 개발 입문, 홍팍, 길벗, 2023

# 라이브러리 vs. 프레임워크

- **라이브러리(library)**
  - 자주 사용되는 기능을 재사용하도록 정리해 놓은 코드들의 집합
- **프레임워크.framework)**
  - 랄프 존슨(Ralph Johnson)
    - ✓ 소프트웨어의 구체적인 부분에 해당하는 설계와 구현을 재사용이 가능하게끔 일련의 협업화 된 형태로 클래스들을 제공하는 것
  - 기능
    - ✓ 애플리케이션의 틀과 구조 제공
    - ✓ 확장이 가능한 기반 코드 제공
    - ✓ 개발된 애플리케이션 코드를 제어

# 프레임워크의 장단점

- **장점**
  - 시간과 비용이 절약되어 생산성이 좋아짐
  - 개발자가 반복 작업에서 실수하기 쉬운 부분을 커버하여 소프트웨어의 품질 (Quality)을 향상
  - 유지보수가 좋아짐
- **단점**
  - 프레임워크에 있는 코드를 학습하는데 시간이 오래 걸림
  - 제약사항이 발생함
    - ✓ 프레임워크 제작자가 설계한 구조를 유지해야 하므로 개발자가 유연하게 개발하는데 한계가 있음

# 스프링(Spring)

- 자바 엔터프라이즈 개발을 편하게 해주는 오픈소스 경량급 애플리케이션 프레임워크
  - 애플리케이션 프레임워크
    - ✓ 애플리케이션 개발의 전 과정을 빠르고 편리하고 효율적으로 만들어 줌
  - 경량급(lightweight)
    - ✓ 단순한 개발툴과 기본적인 개발환경으로 엔터프라이즈 개발에 필요한 주요 기능을 갖춘 애플리케이션 개발이 가능
  - 자바로 엔터프라이즈 개발을 편하게
    - ✓ 엔터프라이즈 개발의 복잡함을 제거
    - ✓ 실수하기 쉬운 로우레벨(low-level) 기술에 신경을 많이 안 쓰게 함
    - ✓ 애플리케이션의 핵심인 비즈니스 로직을 빠르고 효과적으로 구현하게 함

# 스프링 부트(Spring boot)란

- 자바 웹 프로그램을 더욱 쉽고 빠르게 만들기 위해 개발된 오픈 소스 프레임워크
  - 자바 웹 프로그램을 만들기 위한 기능과 도구 모음
  - 원하는 기능을 조립함으로 쉽게 빠르게 자바 웹 프로그램을 만듦
- 스프링 프레임워크(Spring framework)를 개선함
  - 개발 환경 설정을 간소화
    - ✓ 미리 설정된 스타터 프로젝트로 외부 라이브러리를 최적화해 제공
    - ✓ 사용자가 직접 외부라이브러리를 찾아 연동하지 않아도 됨  
→ 개발 시간 단축, 생산성 향상
    - ✓ Cf. 스프링: 애플리케이션을 구축할 때, XML 환경 설정과 종속성 설정이 필요
  - 웹 애플리케이션 서버 내장
    - ✓ WAS(Web Application Server)인 톰캣을 내장
    - ✓ 웹 어플리케이션을 jar 파일로 간편하게 배포함
- 개발자가 개발에만 더 집중할 수 있음

# 스프링 부트의 주요 특징

- **환경 설정 자동화**
  - 개발자가 복잡한 설정을 직접 하지 않고, 실행 환경에 맞춰 필요한 설정을 자동으로 구성
- **종속성 관리 자동화**
  - 개발자가 라이브러리 버전을 지정하지 않아도, 호환되는 버전들을 자동으로 관리
- **설정 파일 외부화**
  - 애플리케이션 설정 값을 코드 외부에서 정의하고 관리하여 개발, 테스트, 운영 환경마다 다른 설정을 적용하여 유연하게 동작하도록 함
- **라이브러리 버전 관리 자동화**
  - build.gradle에 스프링 부트 버전을 설정하면, 스프링 라이브러리 뿐만 아니라 의존 관계 라이브러리들도 호환되는 버전으로 자동으로 다운로드하고 관리
- **독립형 애플리케이션 생성**
  - 웹 서버 톰캣(tomcat)이 애플리케이션에 내장되어 하나의 실행 가능한 jar 파일로 배포되므로, 별도의 설치나 복잡한 설정 없이 독립적으로 실행됨
- **프로덕션 지원**
  - 애플리케이션의 운영 모니터링을 위한 통계(metrics), 상태 정검을 제공

# 스프링 vs 스프링 부트

- **스프링**
  - 유연성과 확장성이 뛰어남
  - 설정이 복잡하고, 진입 장벽이 높음
  - 대규모 시스템, 세밀한 설정이 필요한 경우에 사용
- **스프링 부트**
  - 스프링을 기반으로 하되, 자동 설정과 내장 서버를 통해 개발자 더 빠르고 쉽게 애플리케이션을 만들 수 있도록 도움
  - 빠른 개발, 마이크로 서비스, 클라우드 환경에 적합
    - ✓ 마이크로 서비스 철학: 작고 독립적인 서비스의 빠른 개발과 배포
    - ✓ 클라우드 환경의 핵심은 확장성, 독립성, 자동화, 경량화

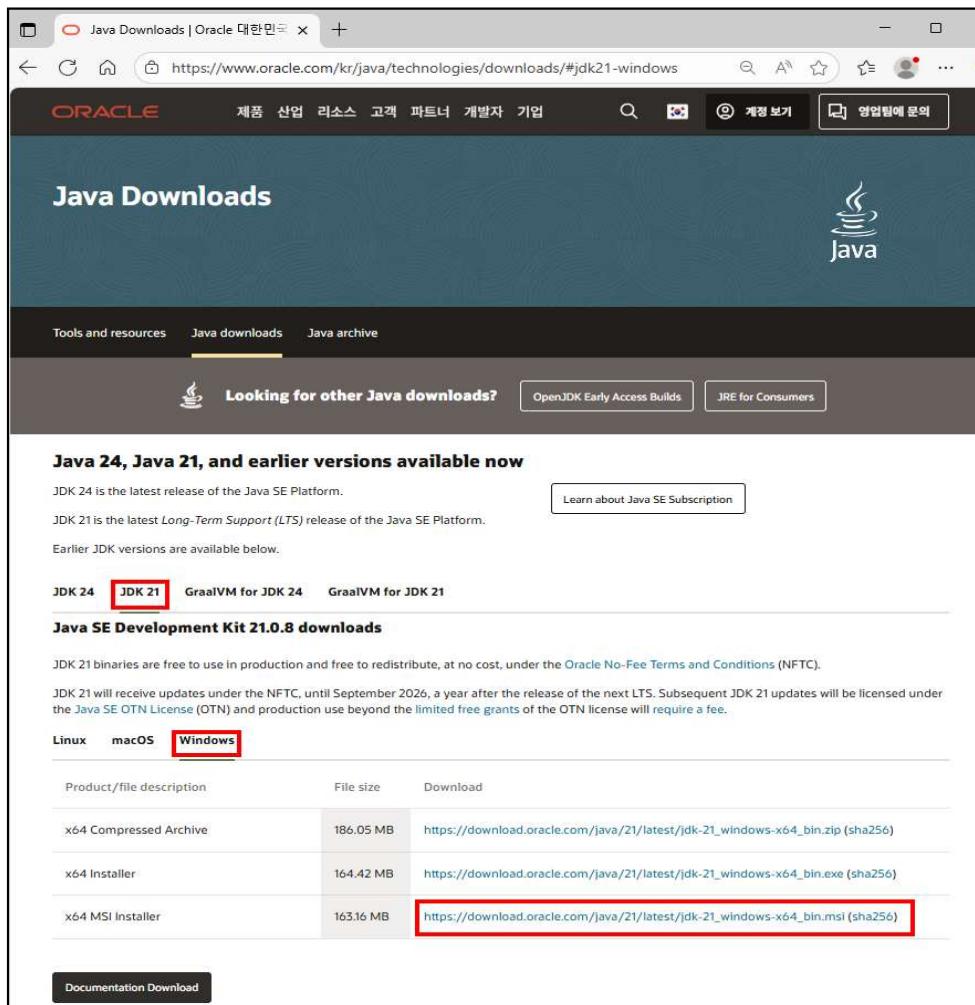
# 스프링 부트 개발 환경 설정하기

- 1. JDK 설치
  - 1.8 이상(수업에서, jdk 21 사용)
- 2. IDE 설치
  - IntelliJ IDEA Community
- 3. 스프링 부트 프로젝트 만들기

# JDK 설치

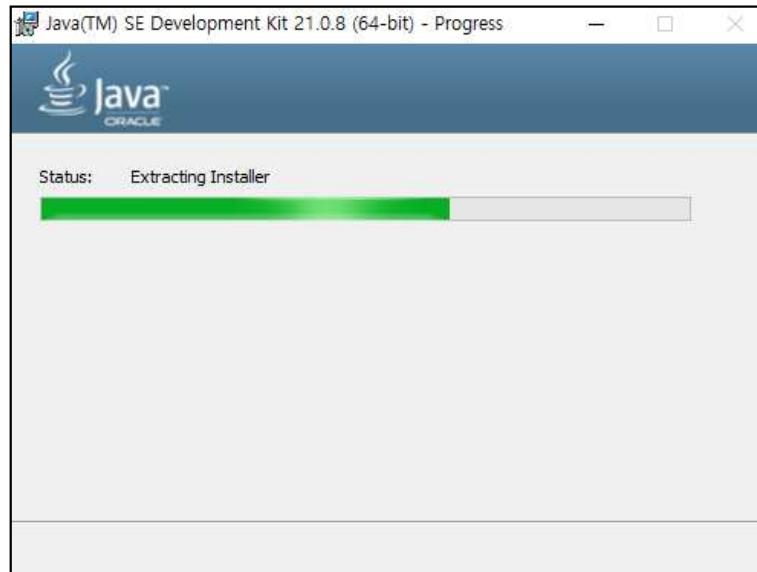
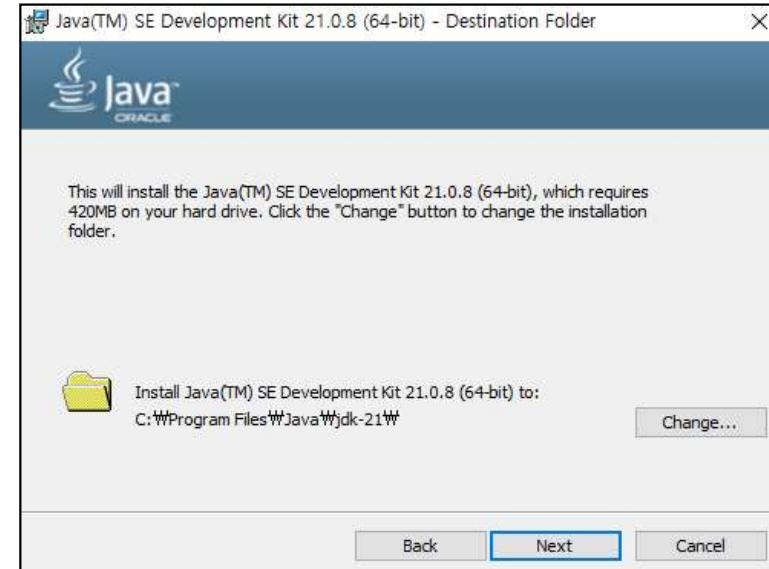
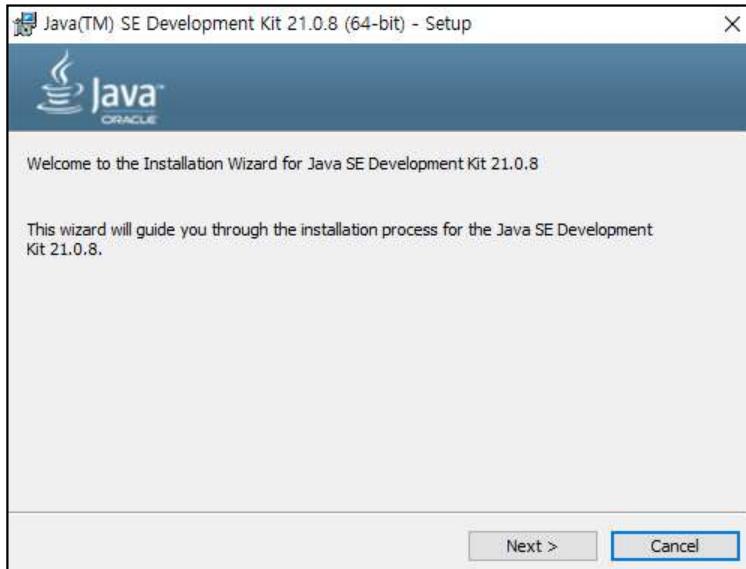
- 다운로드: JDK 21 버전 설치

- <https://www.oracle.com/kr/java/technologies/downloads/#java21>
- 사용하는 운영체제에 맞는 jdk 다운로드 (Linux, macOS, Windows 중 택일)
  - ✓ Windows를 사용할 때, x64 MSI Installer 다운로드



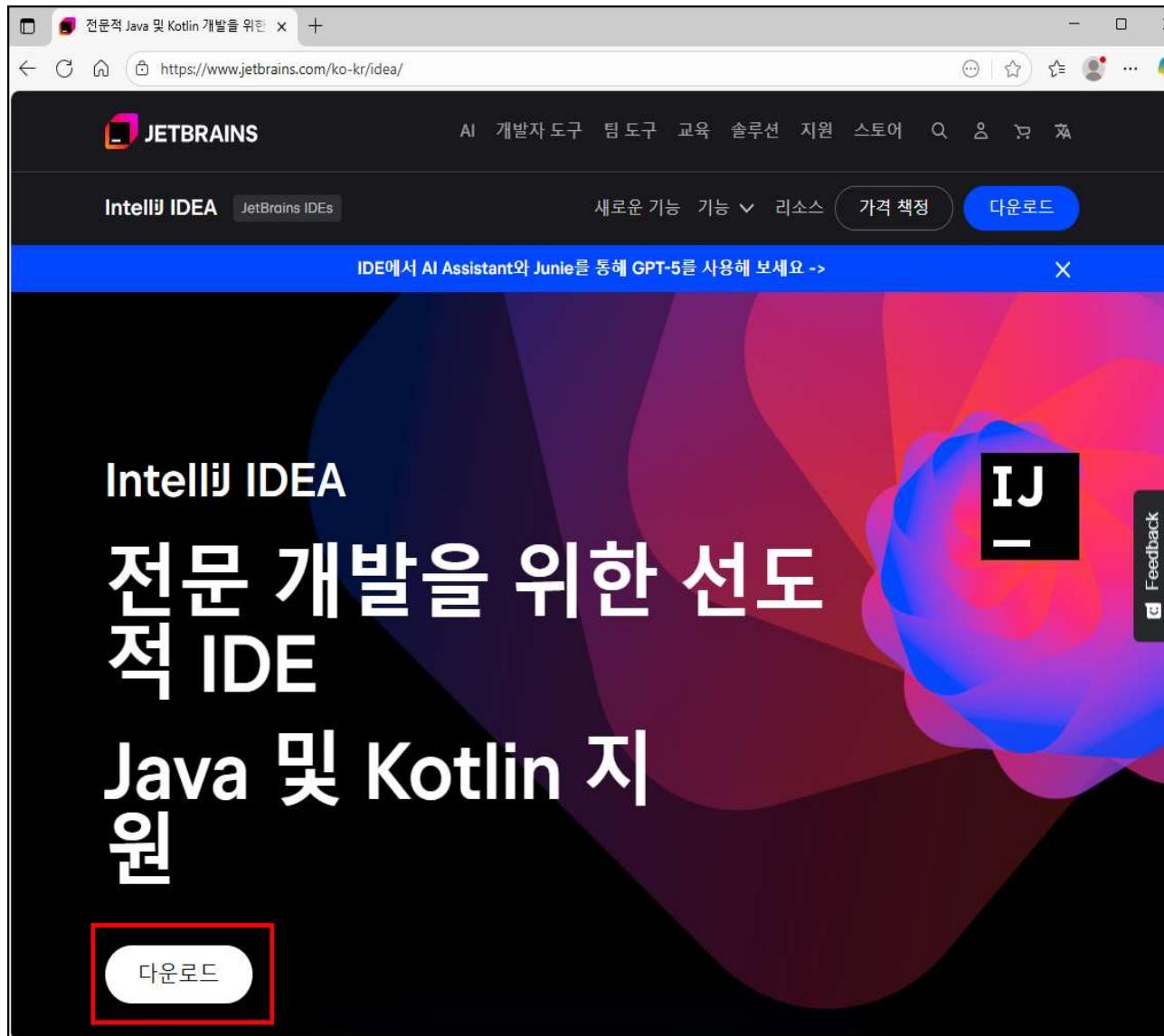
# JDK 설치

- jdk-21\_windows-x64\_bin.msi 설치



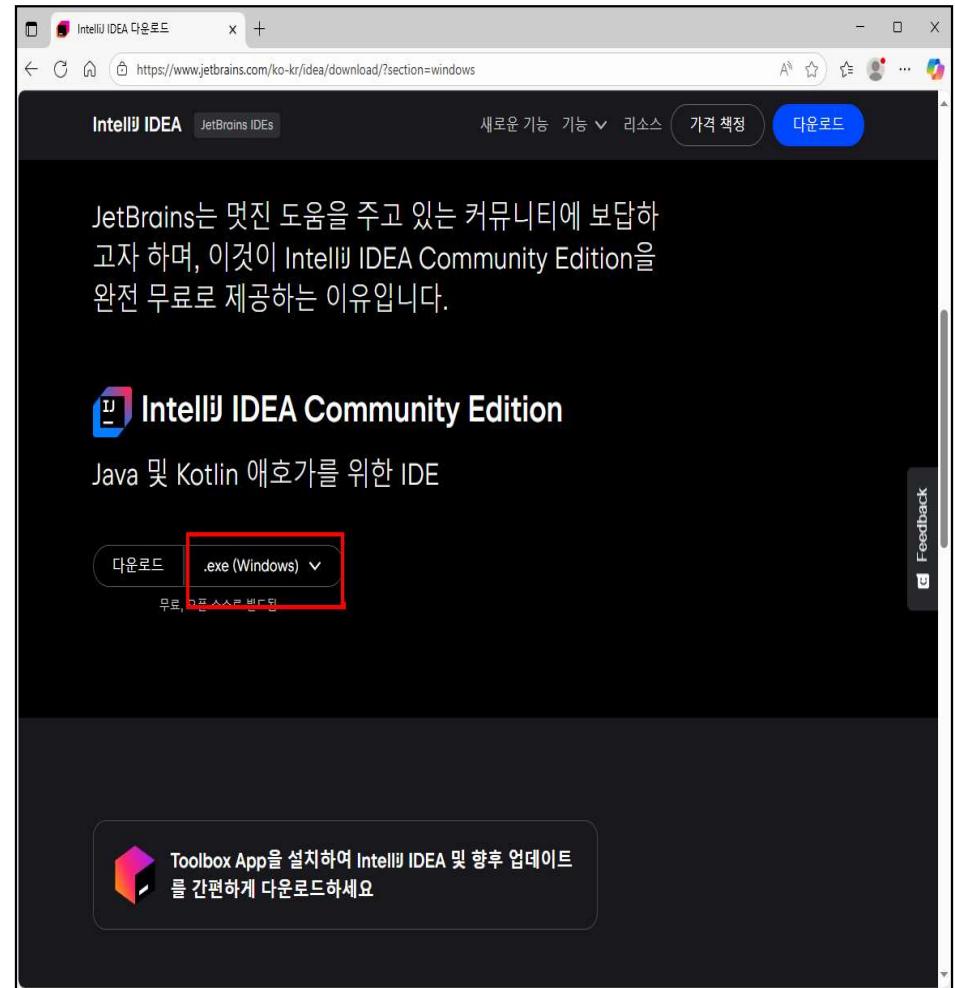
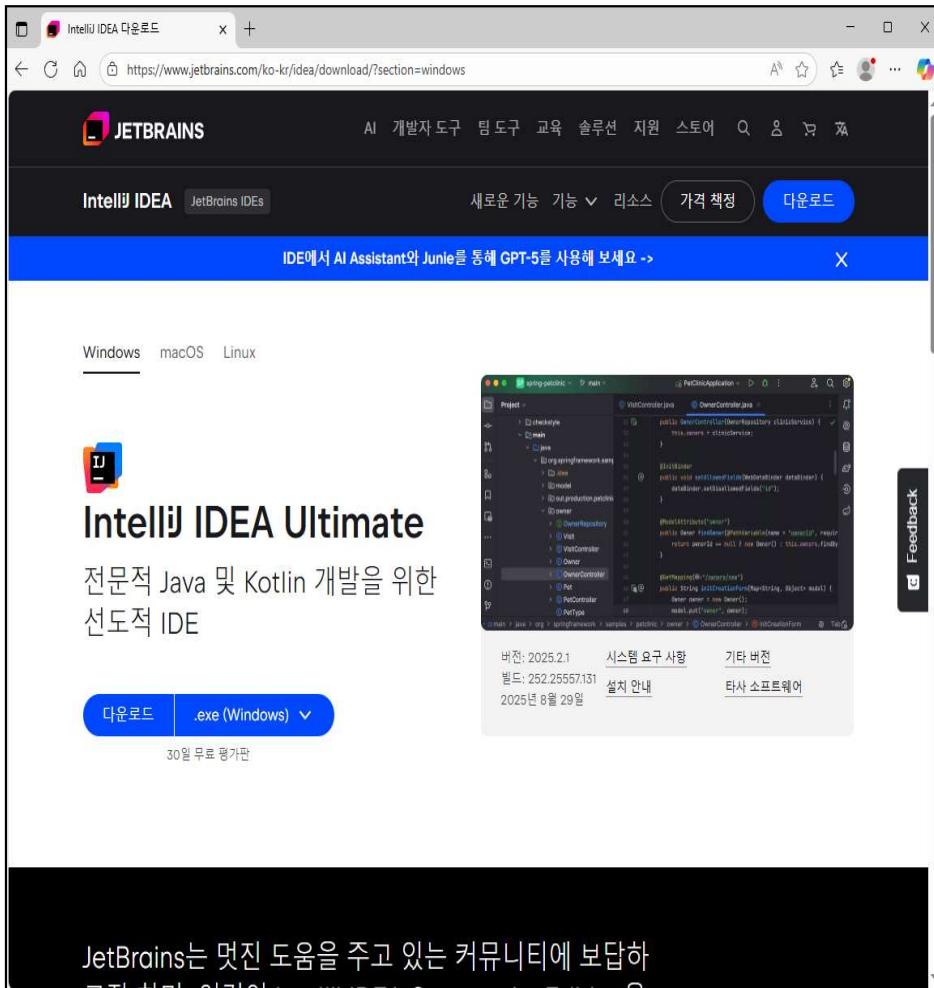
# IntelliJ IDEA Community 설치

- <https://www.jetbrains.com/ko-kr/idea/>



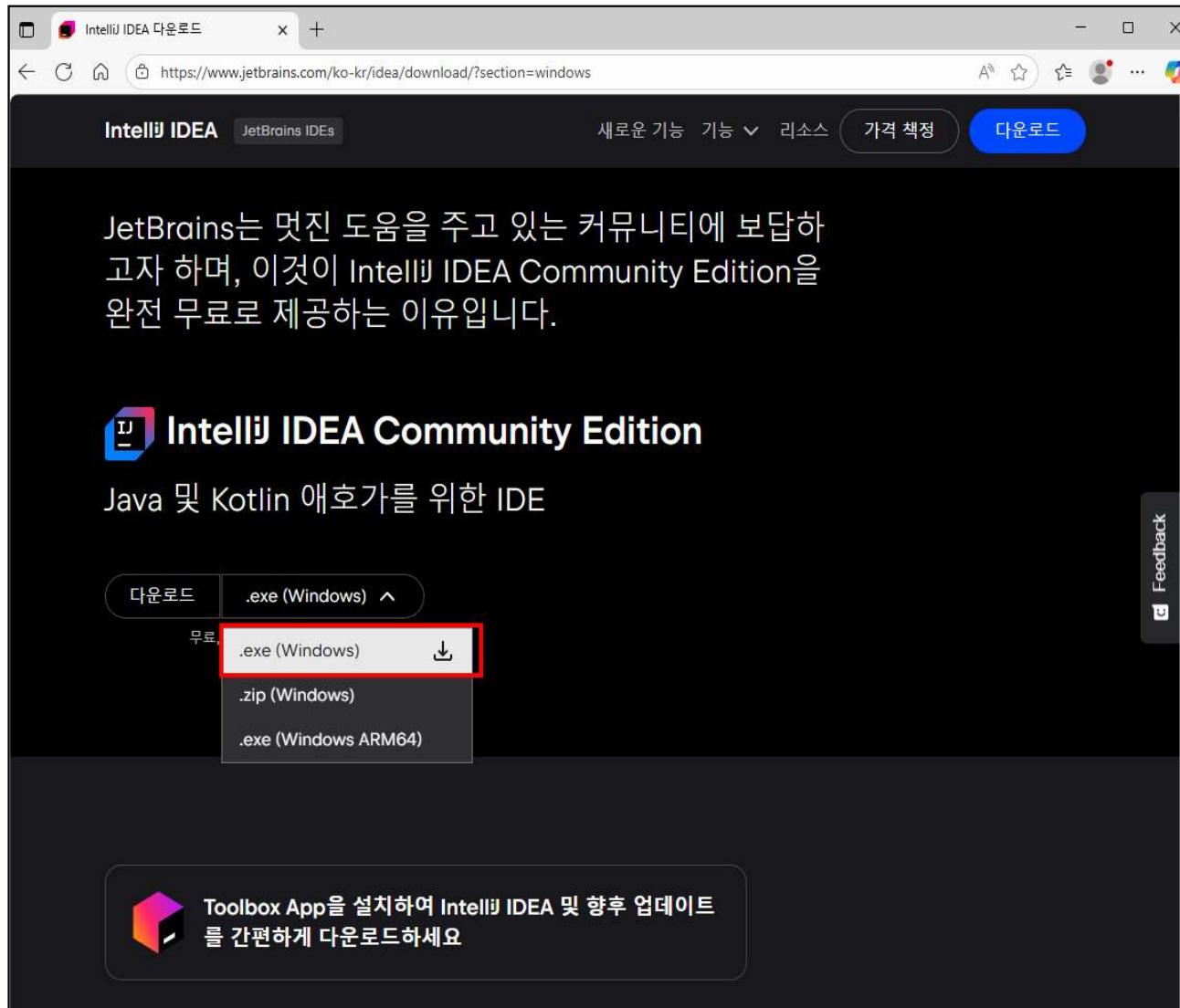
# IntelliJ IDEA Community 설치

- IntelliJ IDEA Ultimate 버전이 아닌, 웹페이지를 하단으로 스크롤하여 IntelliJ IDEA Community Edition의 다운로드를 클릭
  - 학생의 경우, IntelliJ IDEA Ultimate academy 버전을 사용해 볼 수 있음



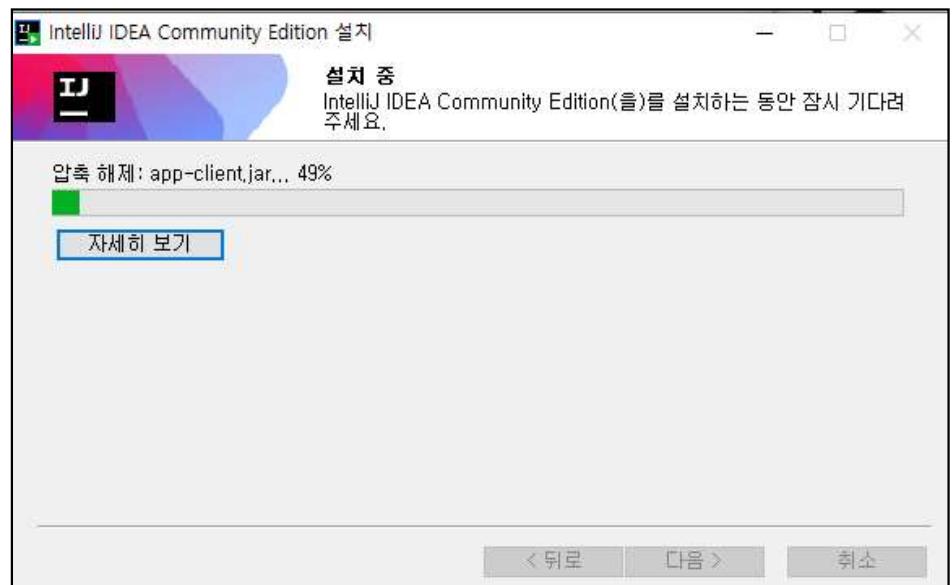
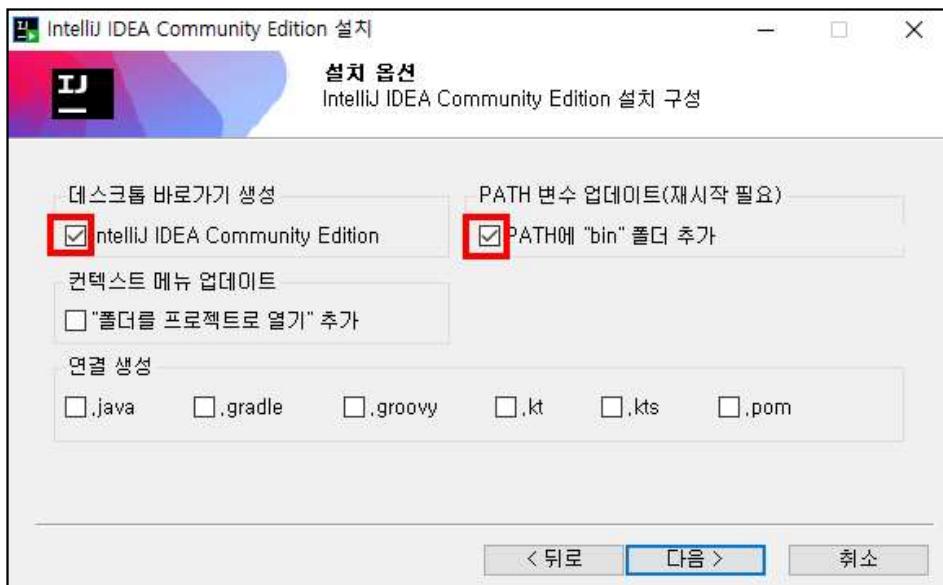
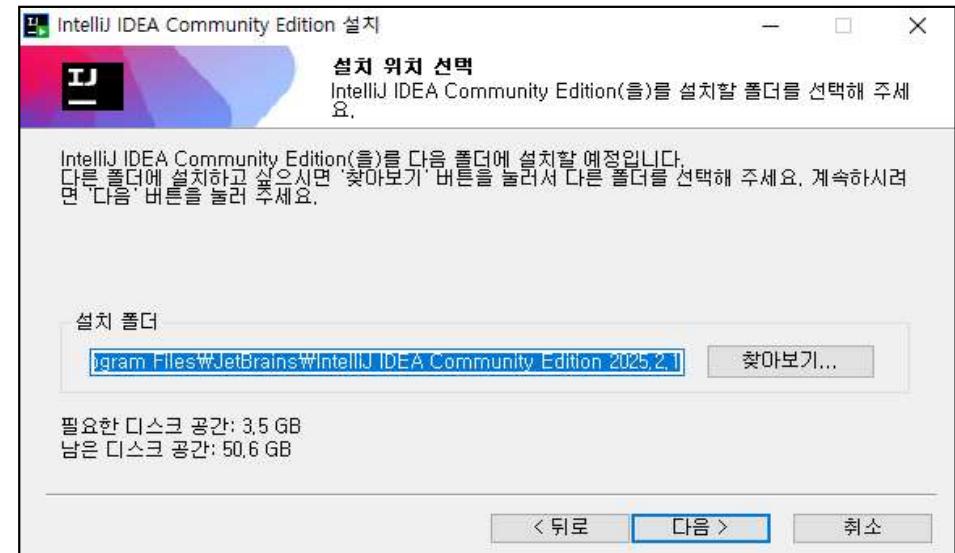
# IntelliJ IDEA Community 설치

- 운영체제에 맞는 설치파일을 다운로드
  - 예시 .exe(windows) 선택



# IntelliJ IDEA Community 설치

- idealC-2024.2.1.exe 설치



# IntelliJ IDEA Community 설치

- 설치 완료



# 스프링 부트 프로젝트 만들기

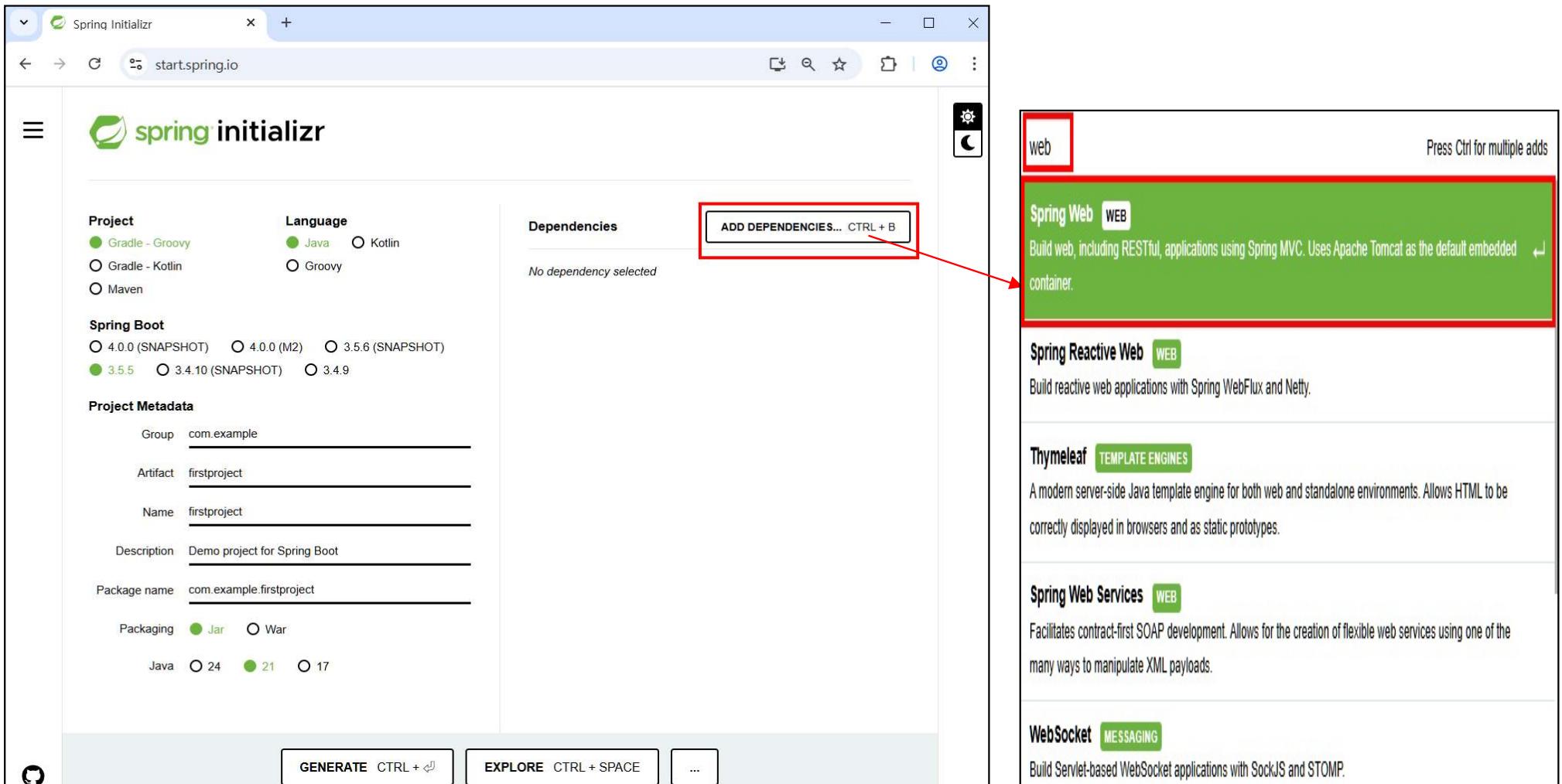
- 스프링 부트는 Spring Initializer를 사용해 쉽게 프로젝트를 생성
- <https://start.spring.io> 접속
- 프로젝트 세부 항목 설정
  - Project: Gradle - Groovy
  - Language: Java
  - Spring Boot: 3.5.5 (없으면 기본값 설정)
  - Packaging: Jar
  - Java: 21

The screenshot shows the Spring Initializr web interface with the following configuration:

- Project**:
  - Gradle - Groovy
  - Gradle - Kotlin
  - Maven
- Language**:
  - Java
  - Kotlin
  - Groovy
- Spring Boot**:
  - 3.5.5
  - 4.0.0 (SNAPSHOT)
  - 4.0.0 (M2)
  - 3.5.6 (SNAPSHOT)
  - 3.4.10 (SNAPSHOT)
  - 3.4.9
- Project Metadata**:
  - Group: com.example
  - Artifact: firstproject (highlighted with a red box)
  - Name: firstproject
  - Description: Demo project for Spring Boot
  - Package name: com.example.firstproject (highlighted with a red box)
- Packaging**:
  - Jar
  - War
- Java**:
  - 24
  - 21
  - 17

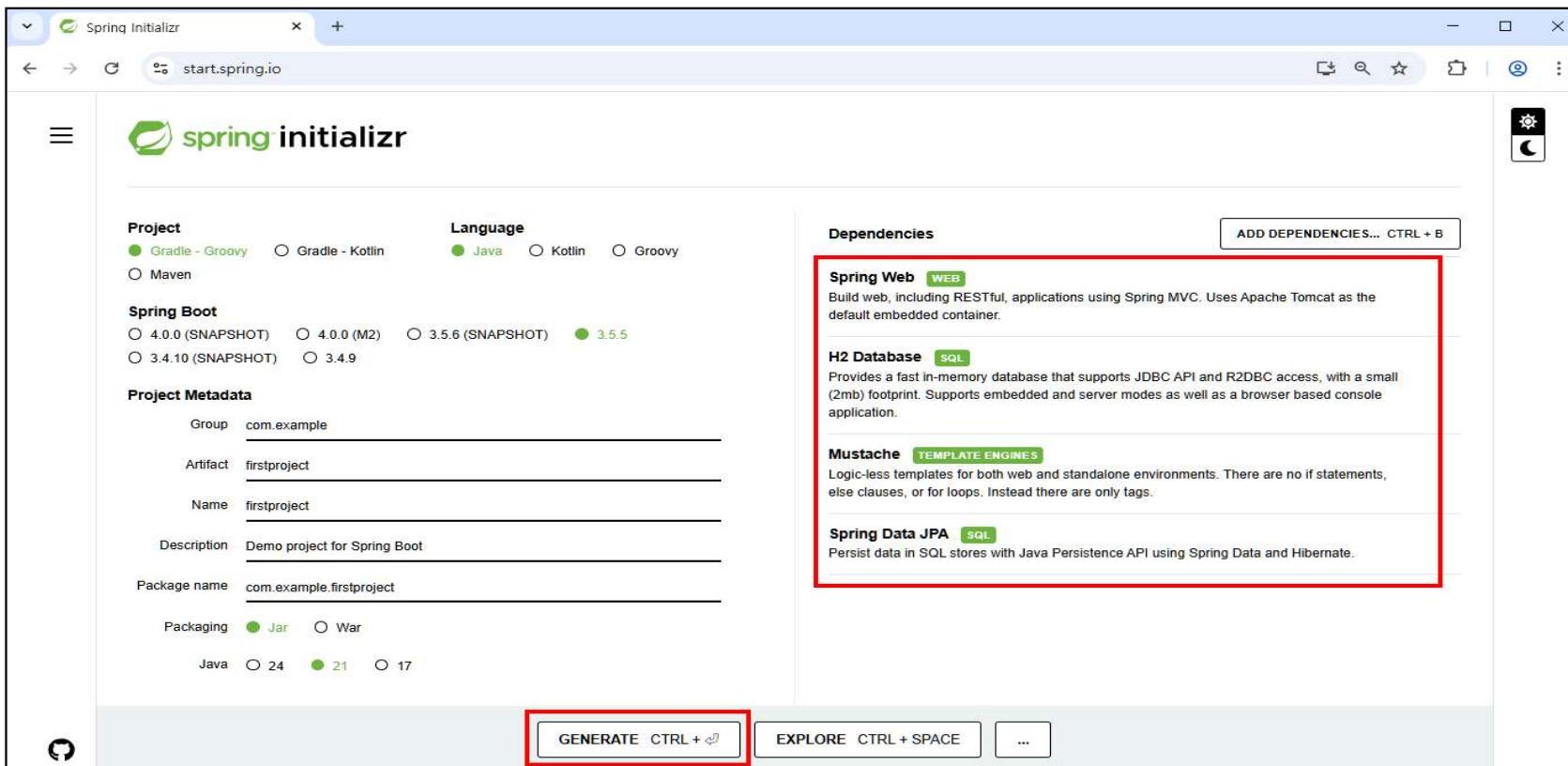
# 스프링 부트 프로젝트 만들기

- [ADD DEPENDENCIES… CTRL+B] 클릭
- 검색: ‘Web’ 입력 - ‘Spring Web’ 선택



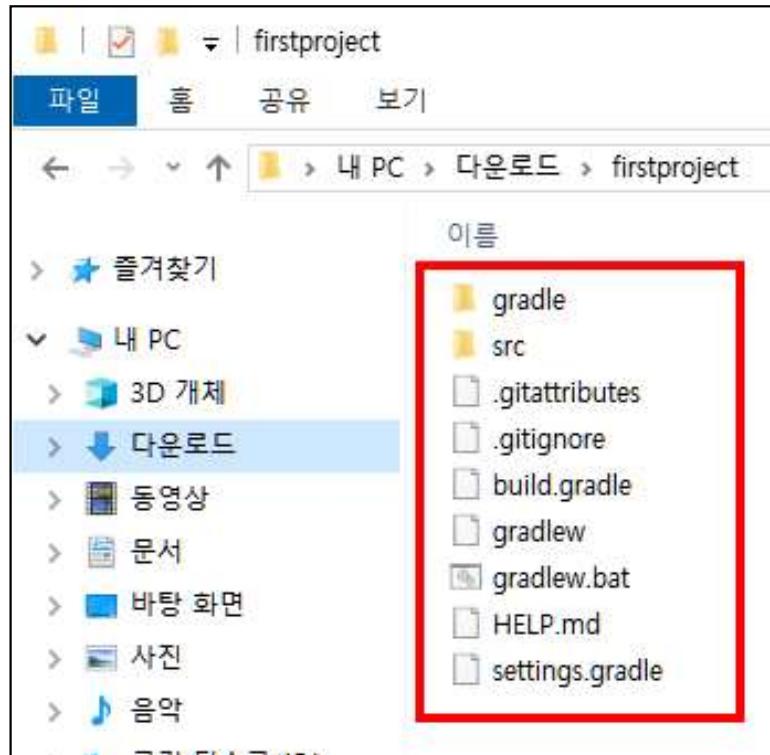
# 스프링 부트 프로젝트 만들기

- [ADD DEPENDENCIES… CTRL+B] 클릭 후,  
같은 방법으로, 아래의 세 가지 도구를 추가
  - H2 Database
  - Mustache
  - Spring Data JPA
- [Generate] 클릭하여 프로젝트를 도운로드함



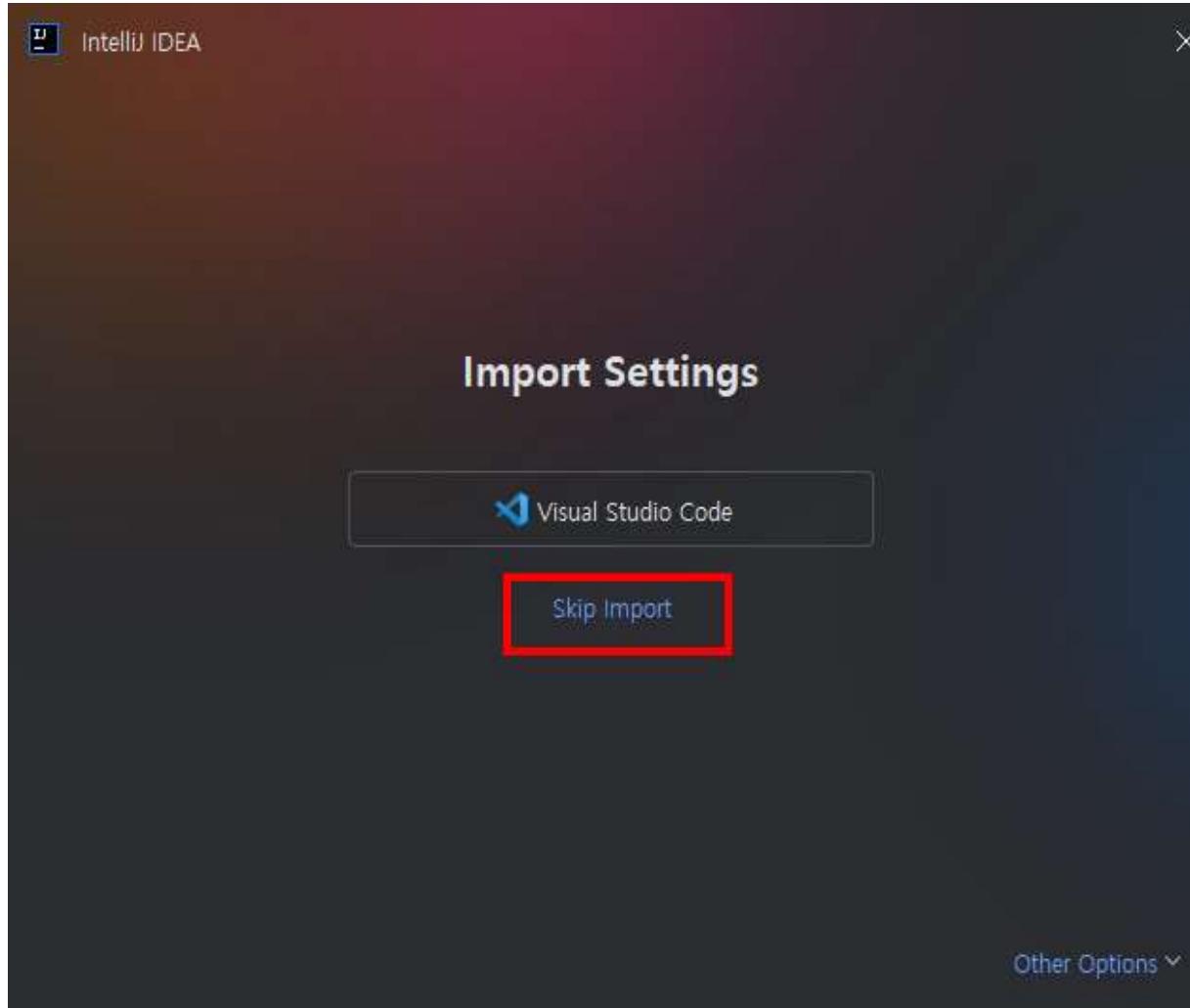
# 스프링 부트 프로젝트 만들기

- 다운로드한 firstproject.zip 파일의 압축을 푼다.
- firstproject 디렉토리를 자신의 작업 디렉토리에 이동



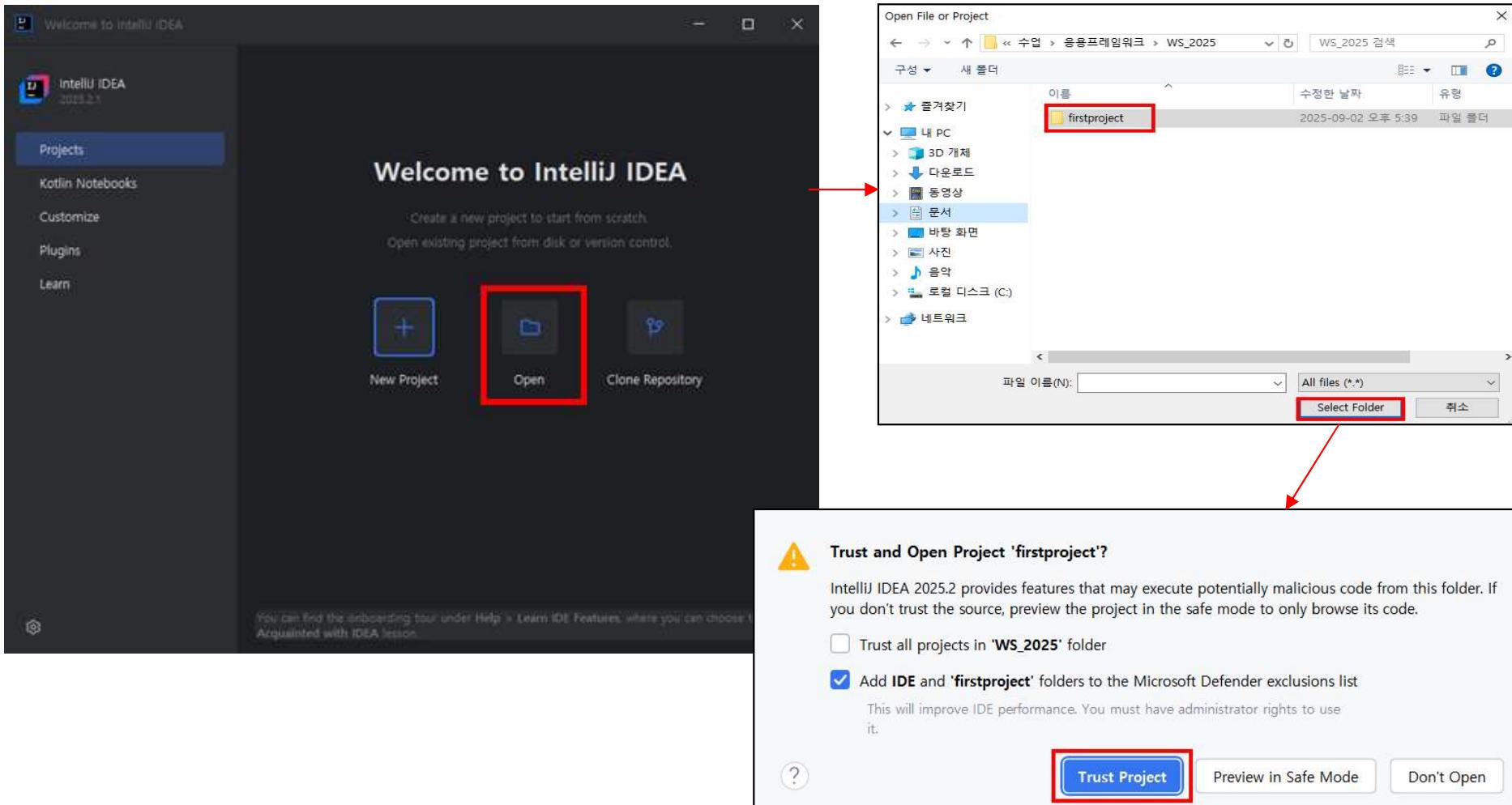
# IntelliJ IDEA 실행

- 바탕화면 IntelliJ 아이콘  더블클릭
- [Import Settings] - [Skip Import] 클릭



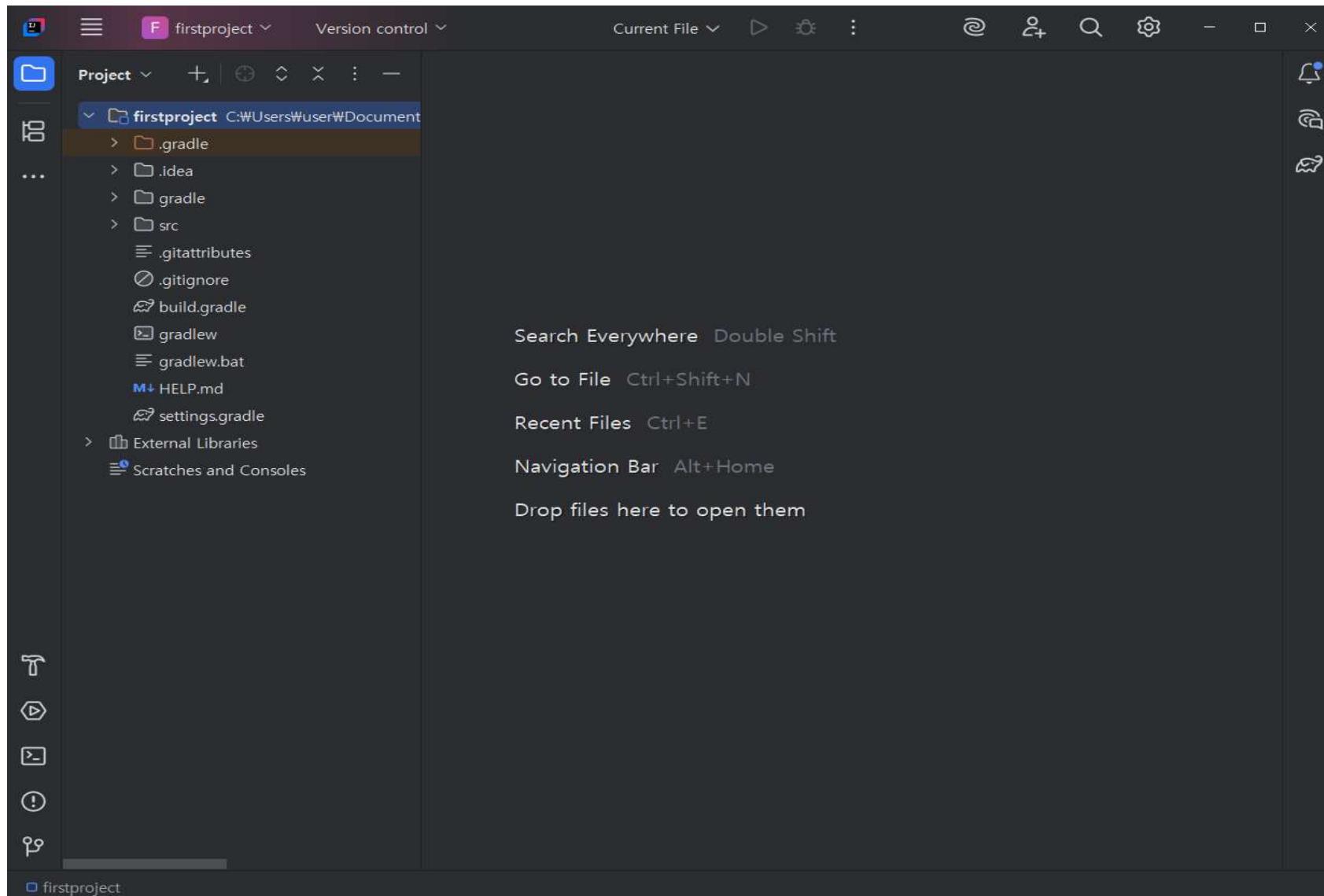
# firstproject 열기

- [Open] - firstproject 디렉토리 선택
- 프로젝트를 [Trust] 를 묻는 창에, [Trust Project] 클릭



# firstproject 열기

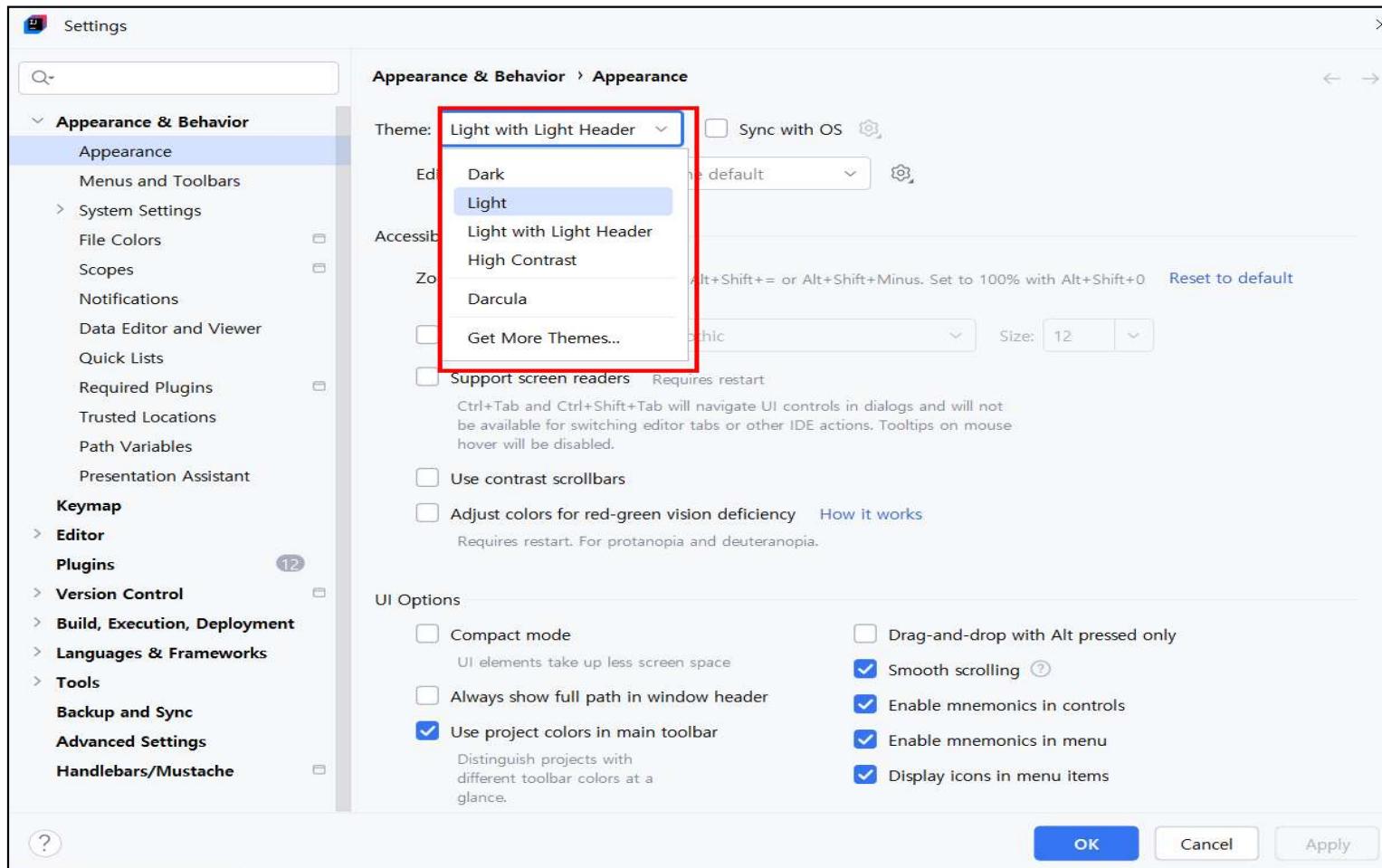
- 프로젝트 열기 성공



# IntelliJ 화면 테마 바꾸기

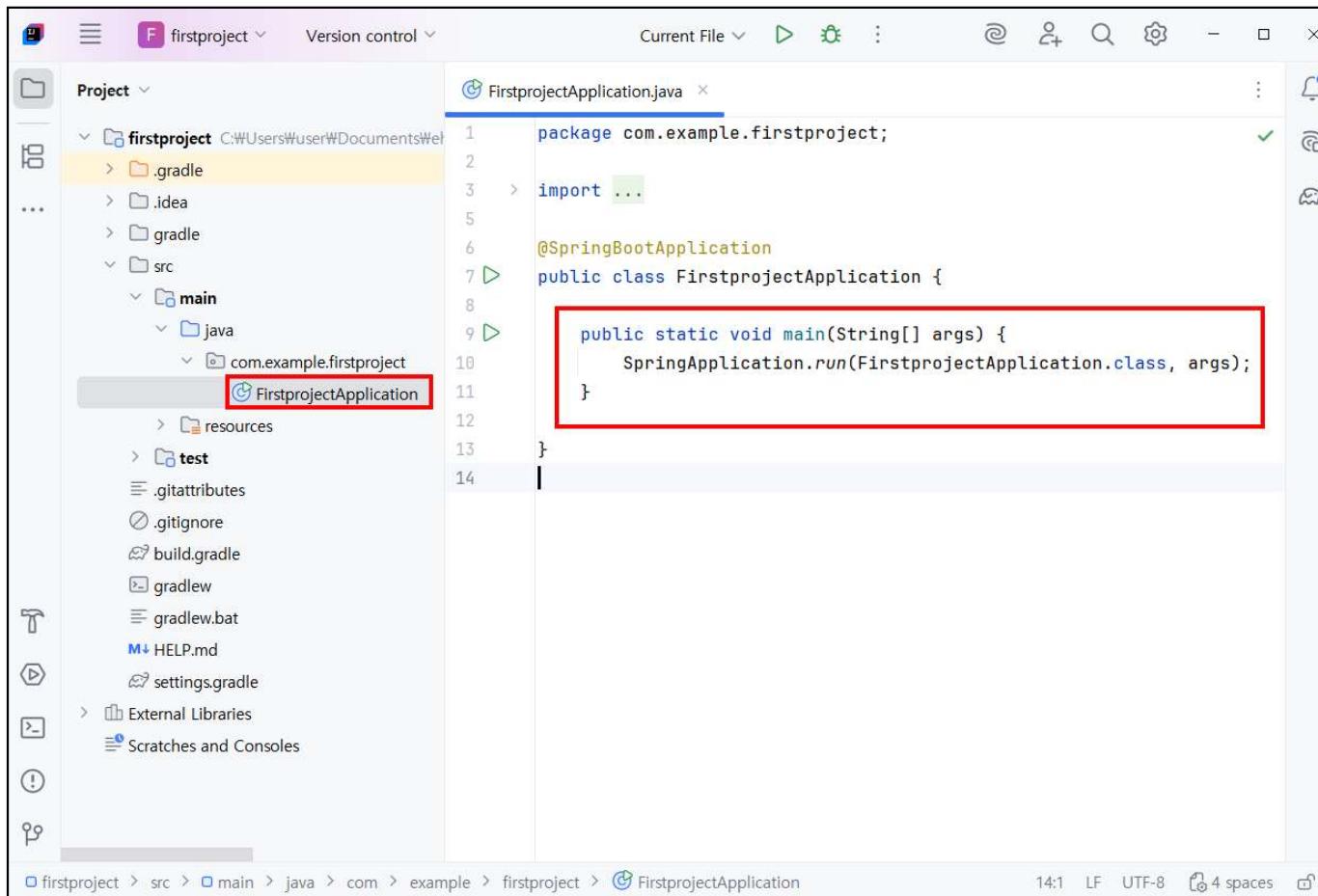
- [File-Settings] - Appearance & Behavior - Appearance

원하는 Theme 선택



# firstproject 실행

- [firstproject-src-main-java] 디렉토리 내에 com.example.firstproject 패키지 내에 FirstprojectApplication 열기
  - main() 메소드 확인



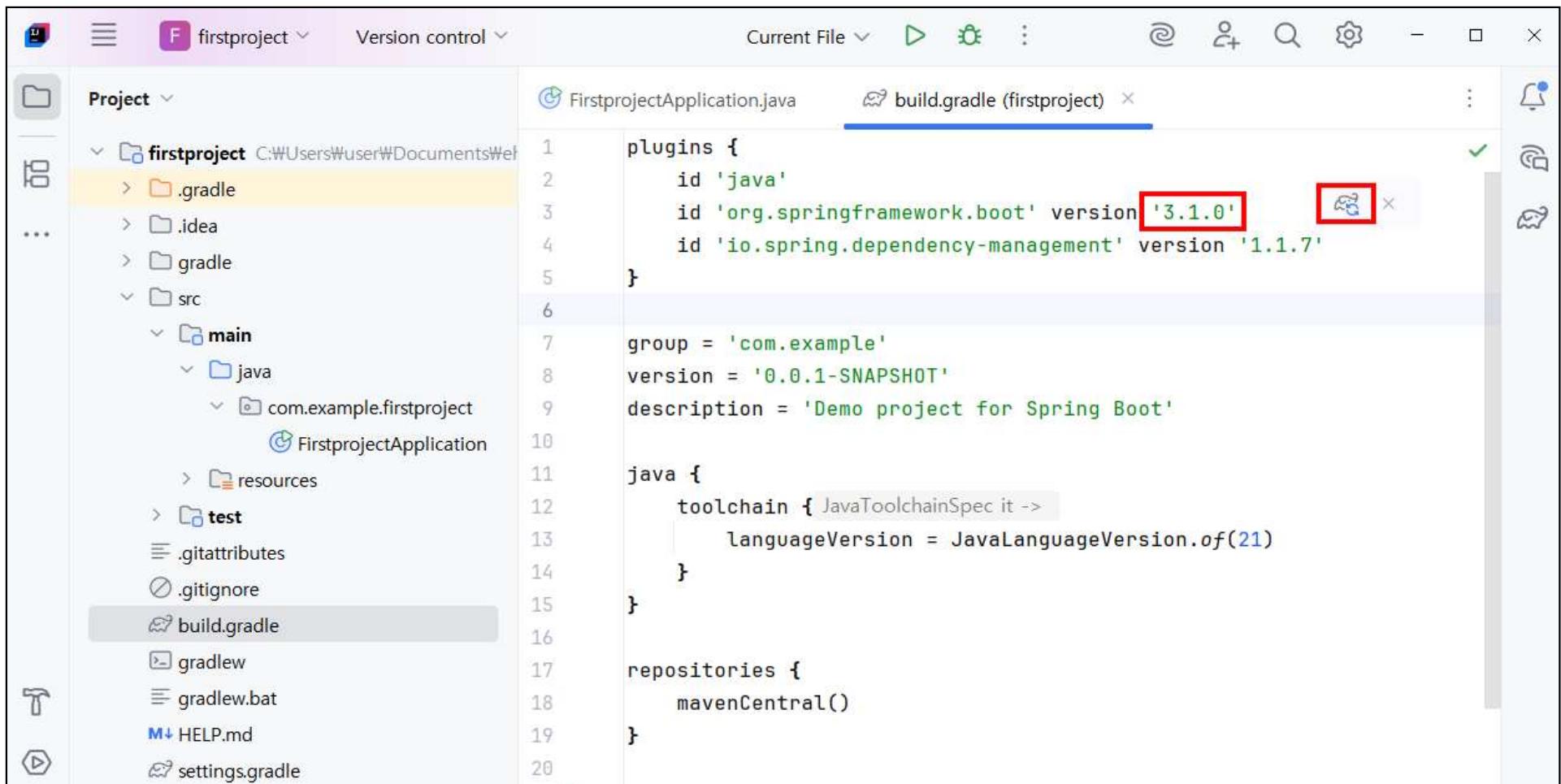
```
package com.example.firstproject;
import ...;
@SpringBootApplication
public class FirstprojectApplication {
    public static void main(String[] args) {
        SpringApplication.run(FirstprojectApplication.class, args);
    }
}
```

The screenshot shows the IntelliJ IDEA interface with the project 'firstproject' open. The 'FirstprojectApplication.java' file is selected in the editor. The main() method is highlighted with a red box. The code is as follows:

```
package com.example.firstproject;
import ...;
@SpringBootApplication
public class FirstprojectApplication {
    public static void main(String[] args) {
        SpringApplication.run(FirstprojectApplication.class, args);
    }
}
```

# 스프링 부트 버전 바꾸기

- [firstproject-build.gradle] 더블 클릭
- plugins 에서  
‘org.springframework.boot’ version ‘3.1.0’ 으로 수정 후,  
코끼리 아이콘 🐄 클릭(상태 바를 보면, 변경한 스프링 부트를



```
plugins {
    id 'java'
    id 'org.springframework.boot' version '3.1.0' // Version highlighted with a red box
    id 'io.spring.dependency-management' version '1.1.7'
}

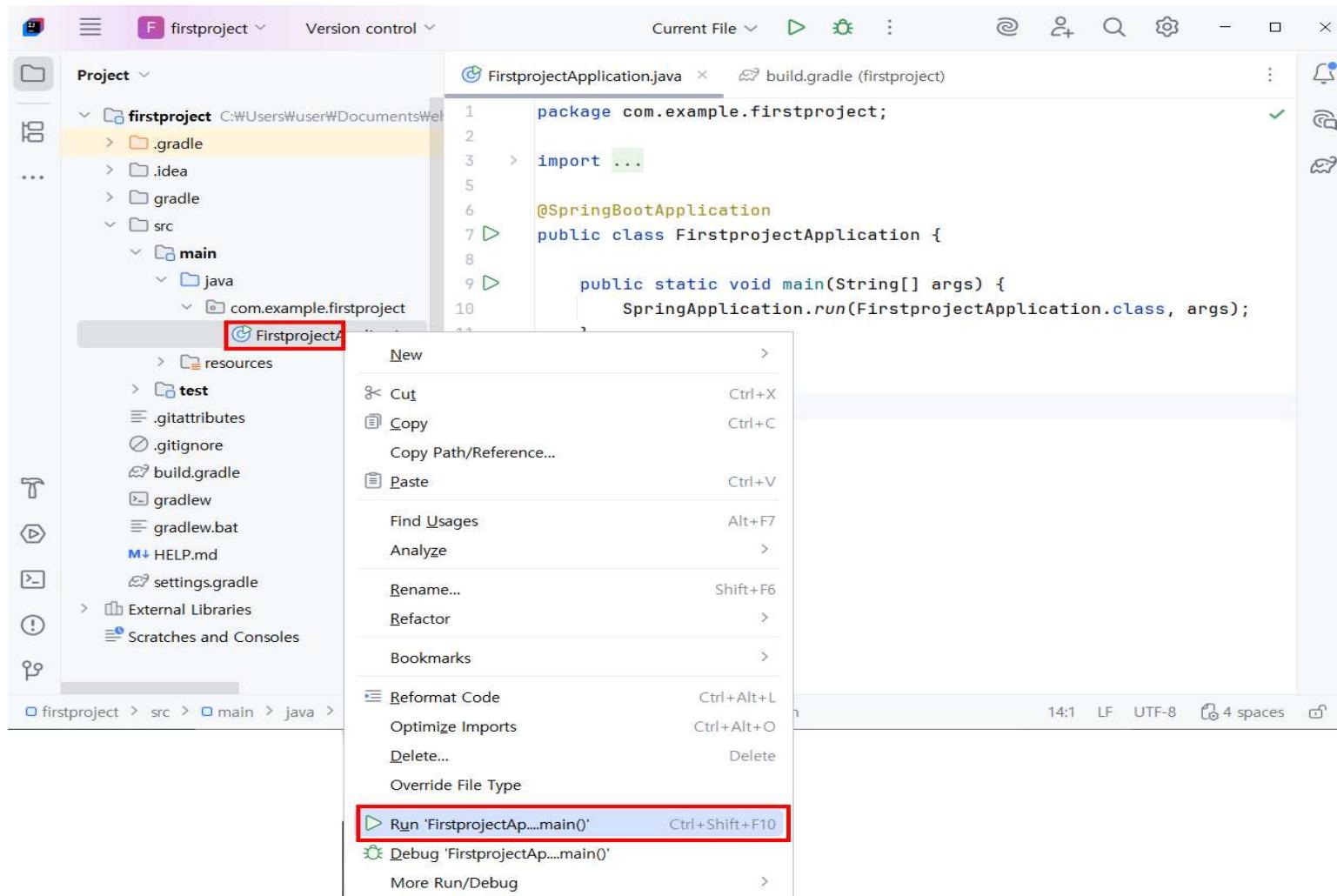
group = 'com.example'
version = '0.0.1-SNAPSHOT'
description = 'Demo project for Spring Boot'

java {
    toolchain { JavaToolchainSpec it ->
        languageVersion = JavaLanguageVersion.of(21)
    }
}

repositories {
    mavenCentral()
}
```

# firstproject 실행하기

- FirstprojectApplicaiton에서, 마우스 오른쪽 버튼 클릭하여 메뉴가 뜨면, [Run FirstprojectAp..main()] 클릭
  - 만약, 윈도우 보안 경고가 뜨면 [액세스 허용] 클릭



# firstproject 실행하기

- FirstprojectApplication 실행 성공 확인
  - 도구 창 모음에 [Run] 탭이 생성됨
  - Tomcat 서버 실행 확인: ‘Tomcat started on port(s): 8080…’
  - 출력 된 문장에서 ‘Started FirstprojectApplication…’ 확인

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

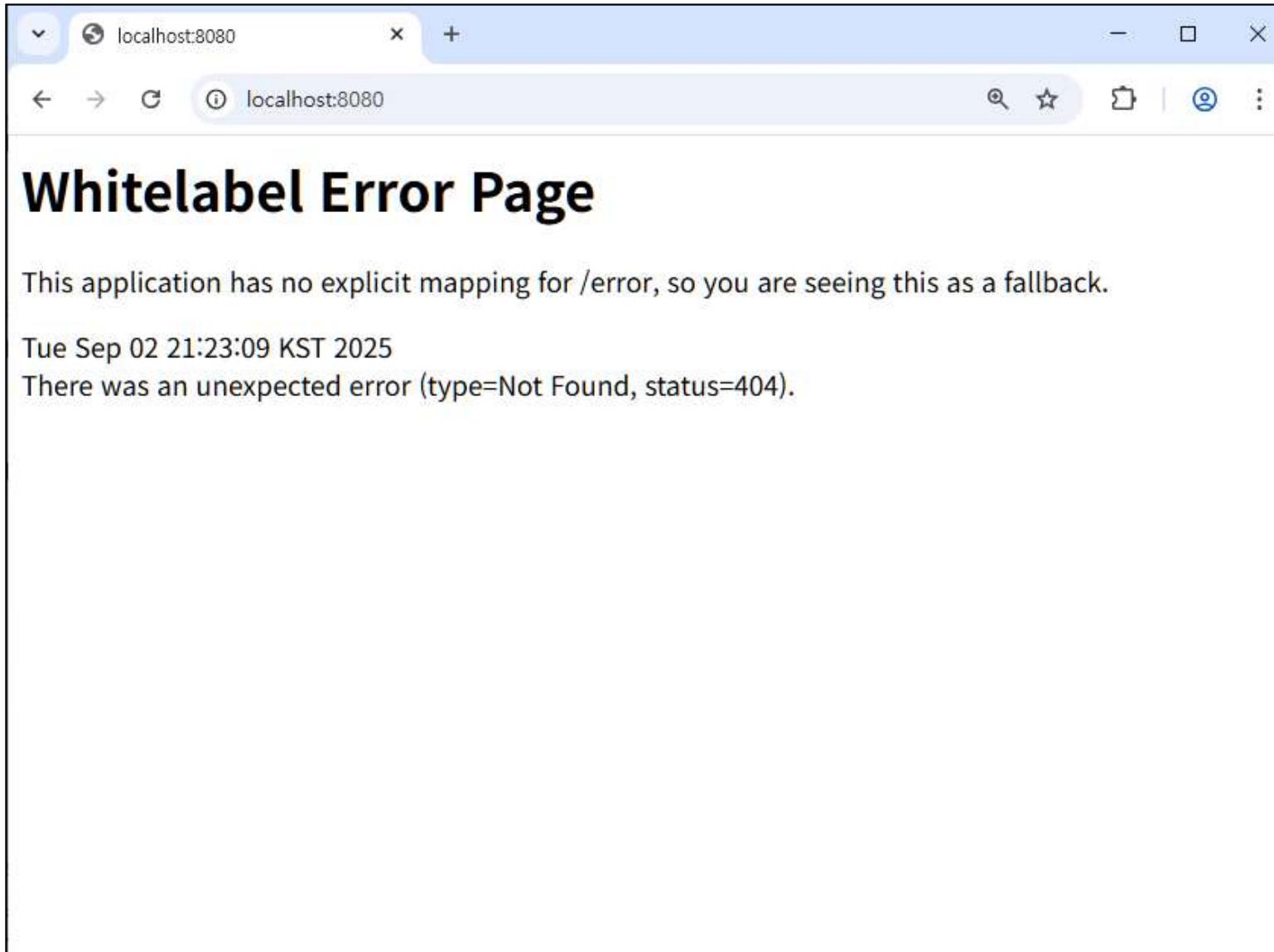
- Project View:** Shows the project structure under "firstproject". The "build" folder is selected.
- Code Editor:** Displays the `FirstprojectApplication.java` file content:package com.example.firstproject;  
import ...  
@SpringBootApplication  
public class FirstprojectApplication {  
 public static void main(String[] args) {  
 SpringApplication.run(FirstprojectApplication.class, args);  
 }  
}
- Run Tool Window:** Shows the "Run" tab with the configuration "firstproject [com.example.firstproject.FirstprojectApplication.main...]" and its status as "1 min, 24 sec".
- Output Log:** Shows the console output with the following entries:

```
...  
o.e.l.j.p.1.0.Slf4jContainerInitializer : Initialized SLF4J LoggerFactory.  
o.e.j.p.1.0.EntityManagerFactoryBean : Initialized JPA EntityManagerFactory for persistence unit 'default'  
BaseConfiguration$JpaWebConfiguration : spring.jpa.open-in-view is enabled by default. Therefore, database queries may  
.b.w.embedded.tomcat.TomcatWebServer : Tomcat started on port(s): 8080 (http) with context path ''  
.f.FirstprojectApplication : Started FirstprojectApplication in 7.36 seconds (process running for 8.179)
```

The last two lines are highlighted with a red box.

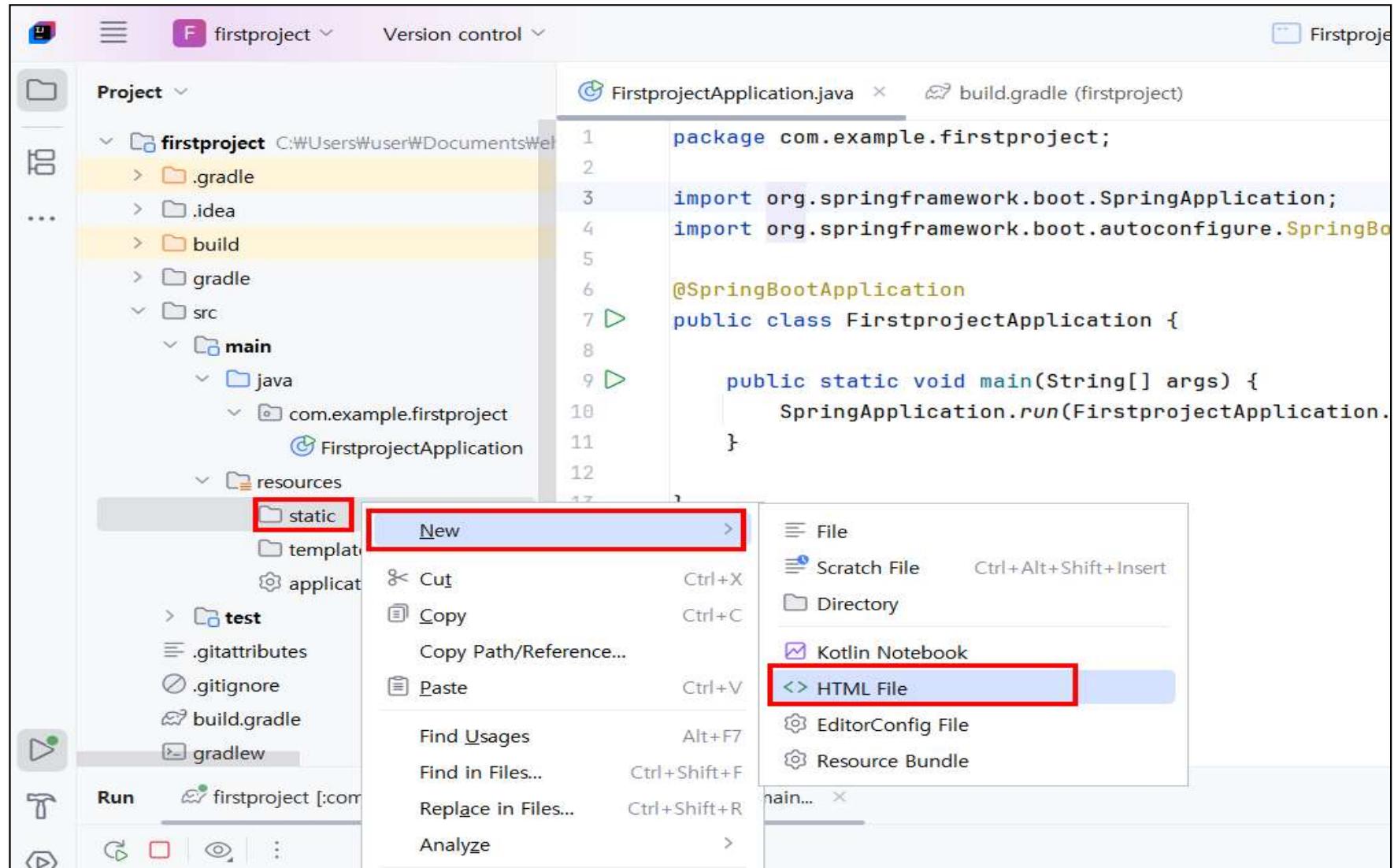
# 스프링 부트 프로젝터가 실행된 서버에 접속

- 웹 브라우저 URL
  - localhost:8080 입력



# Hello World! 웹 페이지 만들기

- [src - main - resources - static]에서 마우스 오른쪽 버튼 클릭
- 나타난 컨텍스트 메뉴에서 [New - HTML File]



# Hello World! 웹 페이지 만들기

- hello.html 생성 및 편집

The screenshot shows a development environment with a 'New HTML File' dialog open at the top, containing the text '<> hello.html'. Below it, the IDE's interface is visible, featuring a 'Project' sidebar on the left and a code editor on the right.

**Project Sidebar:**

- Project: firstproject
- Modules: firstproject
- Files:
  - .gradle
  - .idea
  - build
  - gradle
  - src
    - main
      - java
        - com.example.firstproject
      - resources
        - static
          - hello.html
        - templates
        - application.properties
    - test
  - .gitattributes
  - .gitignore
  - build.gradle

**Code Editor:**

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Hello</title>
</head>
<body>
<h1>Hello World!</h1>
</body>
</html>
```

The line containing the text '<h1>Hello World!</h1>' is highlighted with a red box.

# Hello.html 페이지 요청과 응답

- 서버 재실행
  - FirstprojectApplicaiton에서, 마우스 오른쪽 버튼 클릭하여 메뉴가 뜨면, [Run FirstprojectAp..main()] 클릭
- 웹 브라우저 주소 입력: localhost:8080/hello.html

