

스프링 부트 웹 프로젝트

chapter01

프로젝트를 위한 준비

제공된 자료는 훈련생의 수업을 돕기 위한 것으로, 타인과 공유하시면 안됩니다.

Contents

part.1

스프링 프레임워크란?

part.2

개발 환경 설정

part.3

프로젝트 생성하고 실행하기

part.4

프로젝트 살펴보기

스프링 프레임워크

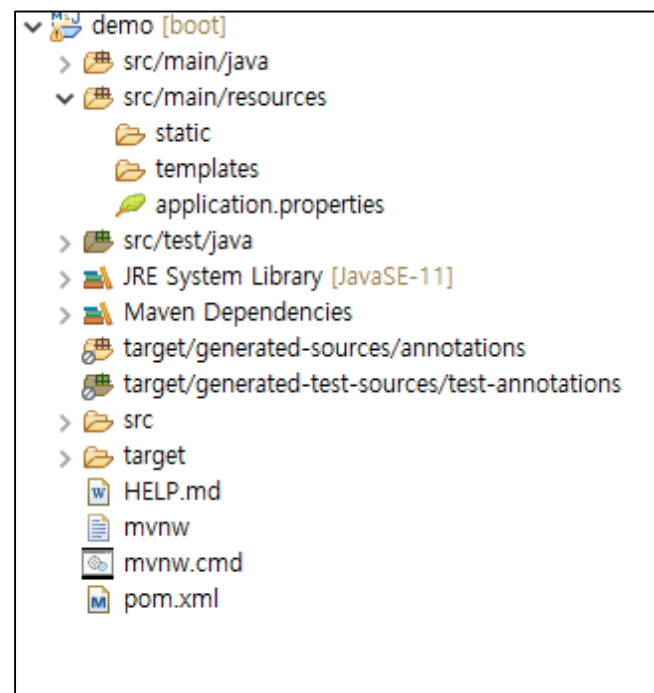
프레임워크란?

프레임워크는 프로젝트의 기본 구조를 자동으로 생성해주는 도구이다.

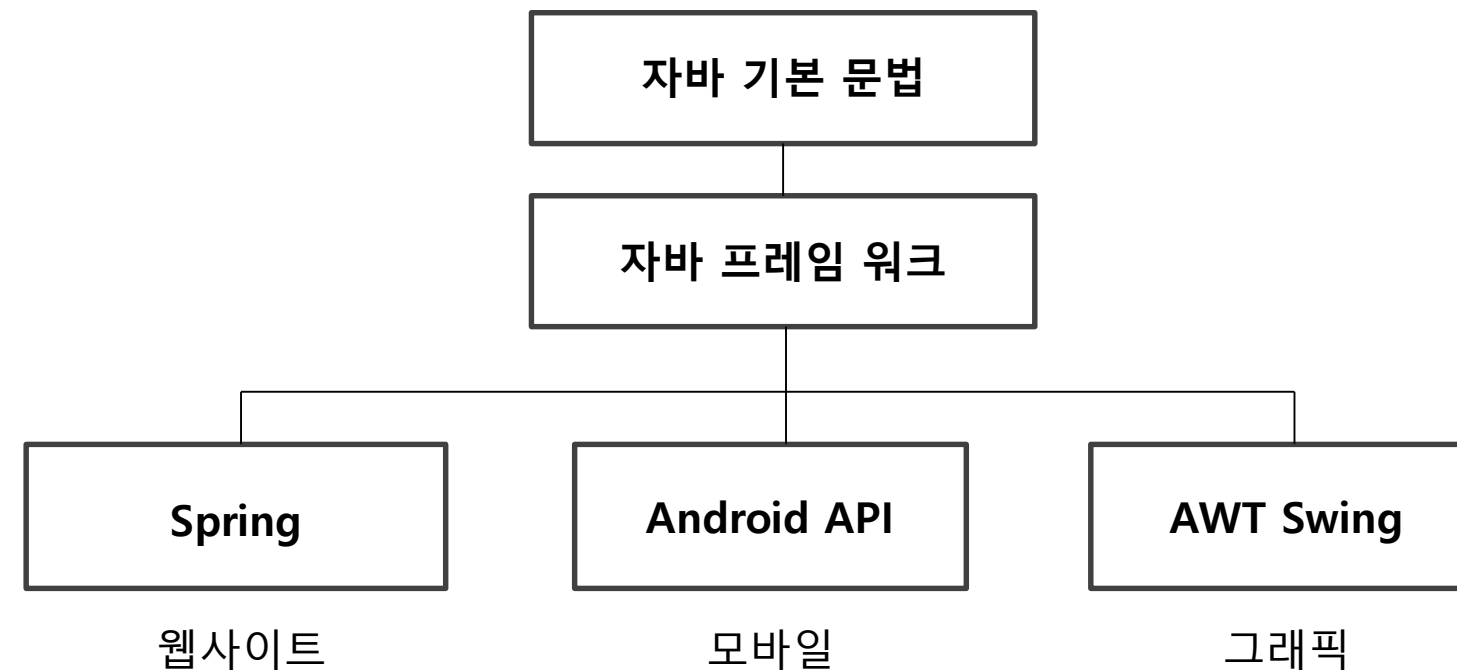
이를 통해 개발자는 생성된 구조에 필요한 코드를 추가하여 프로그램을 완성할 수 있다.

프레임워크는 개발자가 프로젝트 구조를 고민하지 않고, 개발 목적에 맞게 바로 사용할 수 있도록 제공되는 '틀'이다.

스프링 프로젝트 구조 예시



프레임워크의 종류



라이브러리

라이브러리는 개발에 필요한 기능들을 모아놓은 것이다.

개발자는 프로젝트에서 특정 기능을 개발할 때, 라이브러리를 불러와 활용할 수 있다.

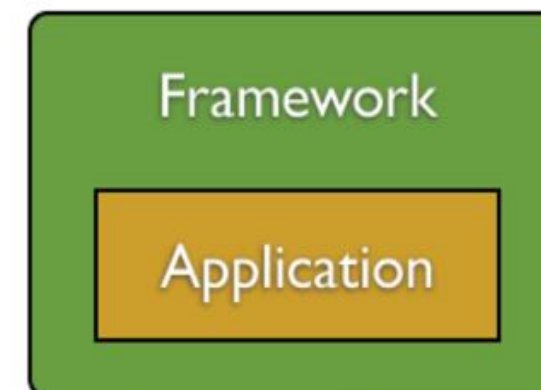
프레임워크

프레임워크는 개발에 필요한 라이브러리와 도구를 모아놓은 것이다.

라이브러리와 달리 어플리케이션의 전반적인 구조와 제어 흐름을 담당한다.



라이브러리의 구조



프레임워크의 구조

스프링 프레임워크란?

스프링은 웹 어플리케이션 개발을 위한 프레임워크로, 주로 웹 사이트 구축에 사용된다.

웹 어플리케이션 개발에 필요한 다양한 기술을 제공하며, 스프링의 핵심 개념은 의존성 주입과 MVC구조이다.

스프링의 주요 기술

- 핵심 기술: 의존성 주입(DI), 스프링 컨테이너, AOP(Aspect-Oriented Programming)
- 웹 관련 기술: MVC 패턴, REST API
- 데이터베이스 관련 기술: 트랜잭션 관리, JDBC, JPA
- 테스트 지원: 단위 테스트



스프링 프레임워크



스프링 부트



스프링 데이터



스프링 세션



스프링 시큐리티



스프링 Rest Docs



스프링 배치



스프링 클라우드

스프링 프레임워크 릴리즈

- 스프링 프레임워크 1.0 출시 - XML 설정 지원
- 스프링 프레임워크 2.0 출시 - 어노테이션을 활용한 설정 지원
- 스프링 프레임워크 3.0 출시 - 자바코드 설정 지원
- 스프링 프레임워크 4.0 출시 - Rest 방식의 컨트롤러 지원
- 스프링 부트 1.0 출시 - 자동 구성, 내장 톰캣 지원
- 스프링 부트 2.0 출시 - 시큐리티 기능 향상
- 스프링 프레임워크 5.0 출시 - 리액티브 프로그래밍 지원
- 스프링 프레임워크 6.0 출시 - 자바17이상 지원, AOP 엔진 도입
- 스프링 부트 3.0 출시 - 자바17이상 지원, AOP 엔진 도입

Spring Framework

Spring은 다양한 모듈과 기능을 제공하며, 프로젝트에 필요한 모듈을 선택하여 사용할 수 있다.
요구사항에 맞는 다양한 구성이 가능하지만, 그만큼 프로젝트 초기 설정이 복잡할 수 있다.

Spring Boot

Spring Boot는 Spring을 더 쉽게 사용할 수 있도록 만들어진 프레임워크이다.
스타터 패키지와 자동 설정 기능을 통해 프로젝트를 간단하게 설정할 수 있어, 빠르게 개발을 시작할 수 있다.



스프링 프레임워크

스프링 프레임워크와 스프링 부트의 차이

1. 내장된 Tomcat 서버 제공

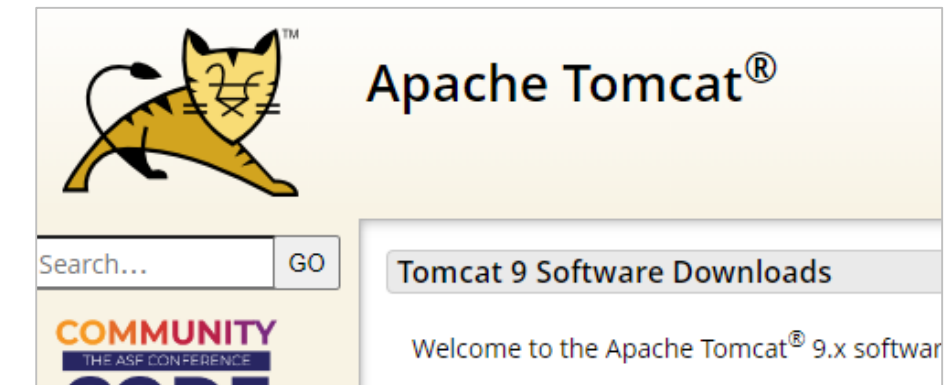
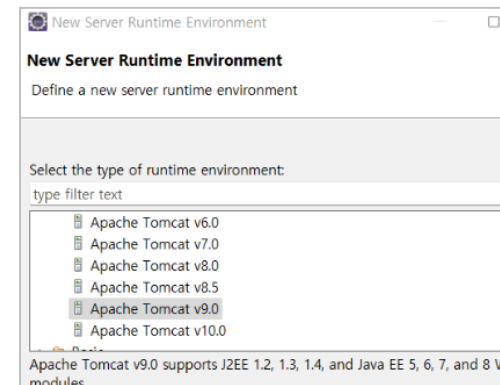
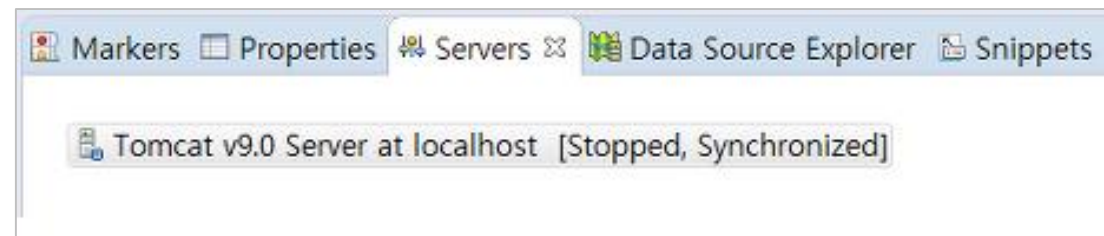
프로젝트를 실행하려면 웹 어플리케이션 서버(WAS)가 필요하다.

WAS는 Tomcat, Jeus 등과 같은 프로그램으로, 웹 어플리케이션의 실행 환경을 제공한다.

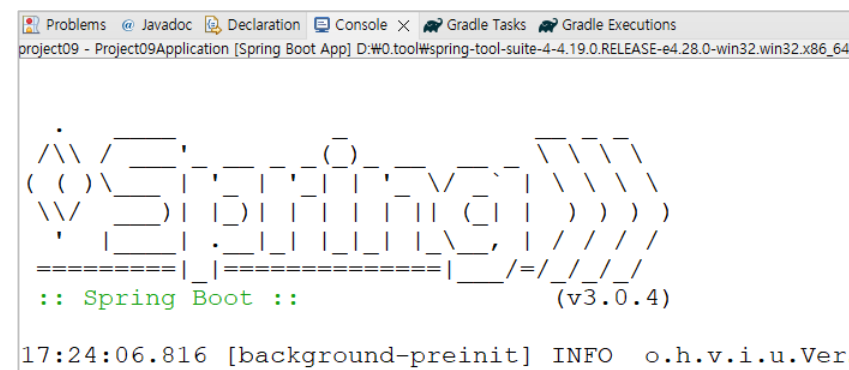
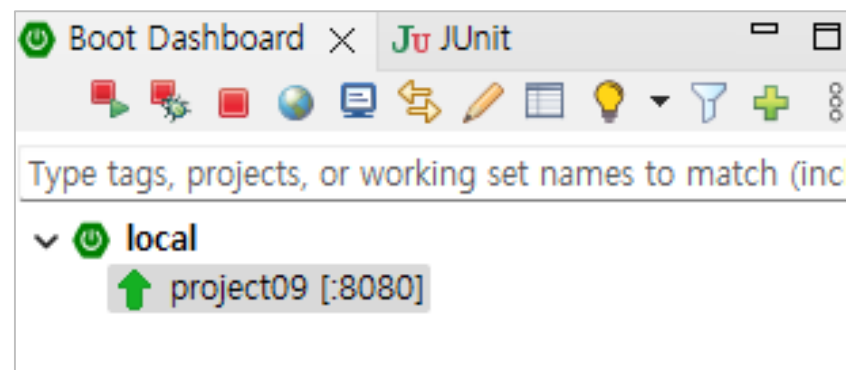
과거에는 WAS를 직접 설치하고, 소스코드를 빌드한 후에 WAS에 배포하는 복잡한 과정이 필요했다.

하지만 Spring Boot는 내장된 Tomcat 서버를 가지고 있어서, 서버를 설치할 필요없이 어플리케이션을 바로 실행할 수 있다.

스프링 프레임워크 프로젝트 실행 과정



스프링 부트 프로젝트 실행 과정



2. 라이브러리 자동 관리

스프링 프레임워크에서는 프로젝트에 필요한 라이브러리의 버전을 개발자가 직접 설정해야 한다.

각 라이브러리 간의 호환성을 고려하여 설정해야 하며, 때로는 버전 충돌이 발생하기도 한다.

반면, 스프링 부트는 스타터 패키지를 통해 프로젝트에 필요한 라이브러리와 버전을 자동으로 관리한다.

이를 통해 라이브러리 충돌 문제를 줄이고, 개발 속도를 높일 수 있다.

개발자는 버전 관리 문제에 신경 쓰지 않고 더 빠르게 개발을 진행 할 수 있다.

스프링 프레임워크의 dependency 관리

```
<dependency>
  <groupId>org.springframework</groupId>
  <artifactId>spring-web</artifactId>
  <version>5.3.5</version>
</dependency>
<dependency>
  <groupId>org.springframework</groupId>
  <artifactId>spring-webmvc</artifactId>
  <version>5.3.5</version>
</dependency>
```

스프링 부트의 dependency 관리

```
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-web'
    compileOnly 'org.projectlombok:lombok'
```

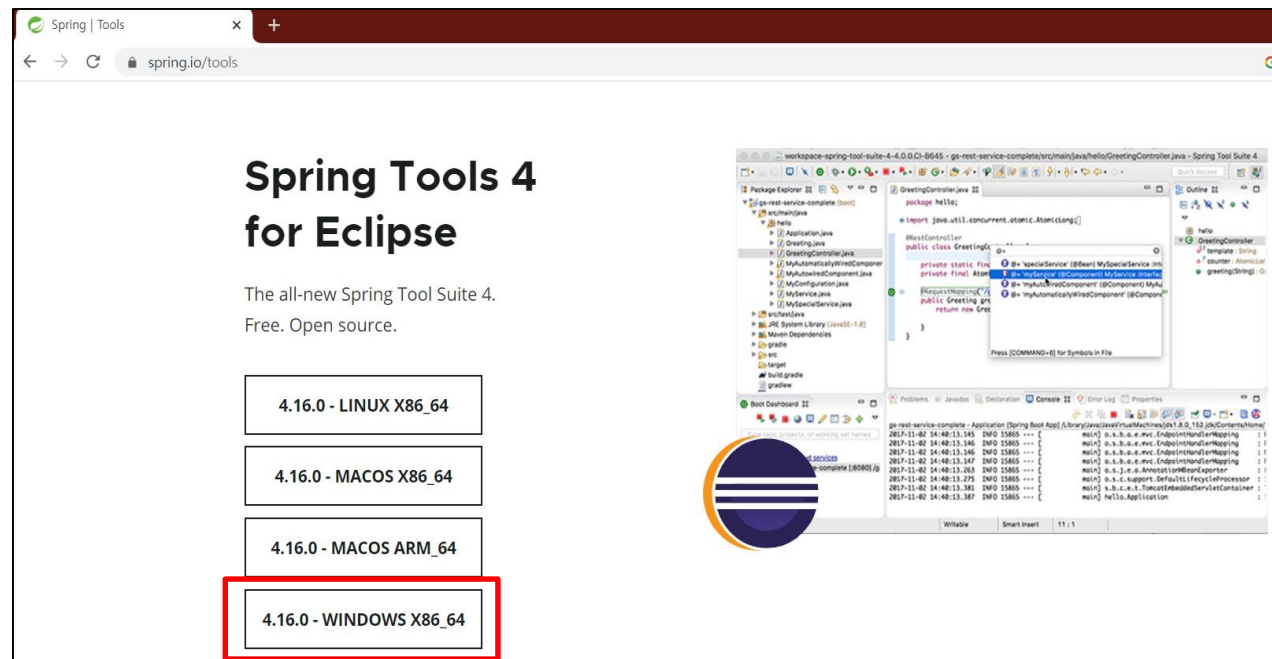
STS4(Spring Tool Suite 4)

STS4는 이클립스를 기반으로 만들어진 개발환경으로, 스프링 프레임워크 관련 플러그인이 추가되었다. 스프링 부트 프로젝트를 개발할 수 있도록 다양한 기능을 제공한다.

STS 설치

1. 스프링웹사이트에 접속해서 STS를 다운로드한다.

<https://spring.io/tools>



2. 다운로드 파일을 본인 폴더로 옮긴다.

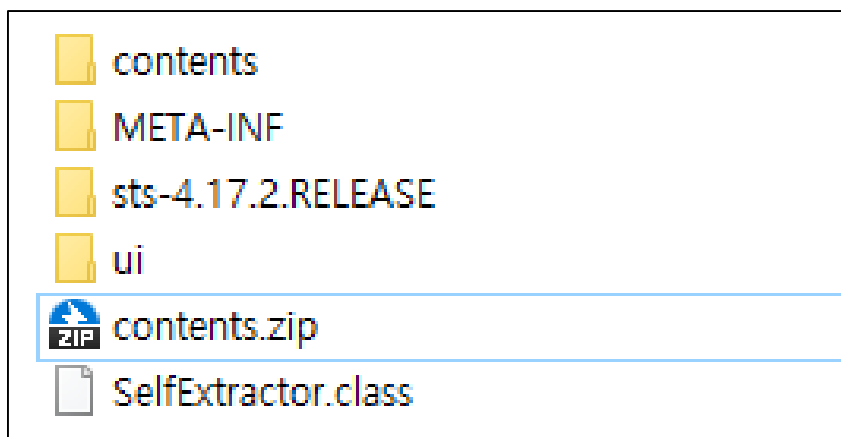
3. 다운로드 파일을 압축해제 한다.

알집으로 압축해제 하면 파일명이 길어서 에러가 발생한다.

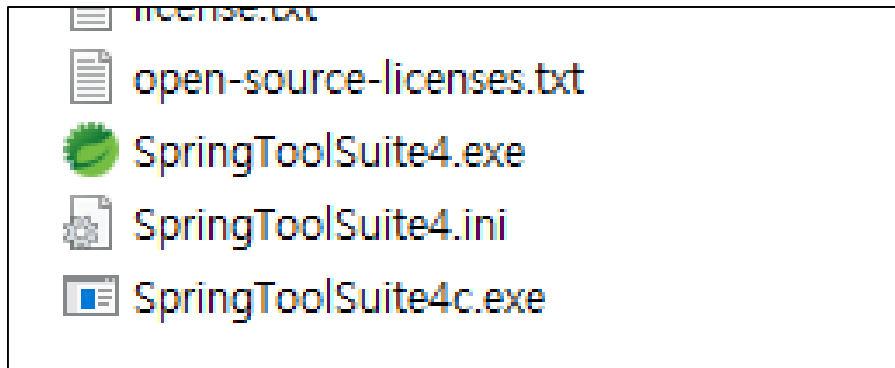
반디집을 사용하여 압축해제 한다.

 spring-tool-suite-4-4.17.2.RELEASE-e4.26.0-win32.win32.x86_64.self-extracting.jar

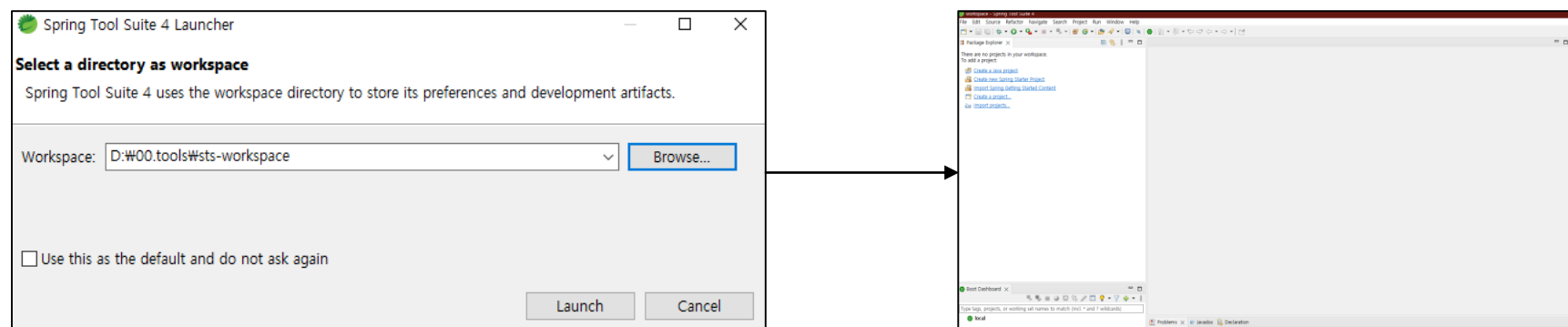
4. 폴더에 있는 contents.zip을 찾아서 압축해제 한다.



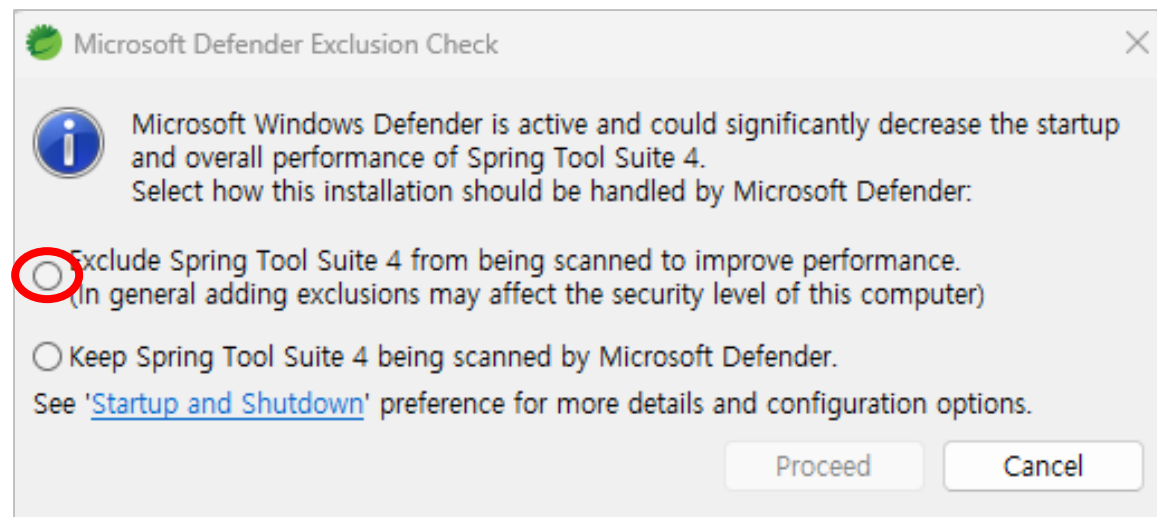
5. contents 폴더 밑에 SpringToolSuite4.exe파일을 찾아서, 바로가기를 만든다.
6. 바로가기 파일을 본인이 찾기 쉬운 위치에 놓는다.
7. 바로가기 파일을 실행한다.



8. STS가 시작하면 workspace 폴더 위치를 선택하고 [launch] 버튼을 클릭한다.



9. STS가 실행되면, Microsoft Windows Defender 메시지 창이 나타난다.
이때 첫번째 옵션을 선택한다.
- 1) Windows Defender가 STS4를 스캔하지 않게 되어 STS4의 속도가 빨라진다.
이 옵션을 선택하면 시스템 보안이 약간 낮아진다.
- 2) Windows Defender가 계속 STS4를 스캔하면 보안은 유지되지만, 속도가 느려질 수 있다.



롬복(Lombok) 플러그인을 추가하는 이유

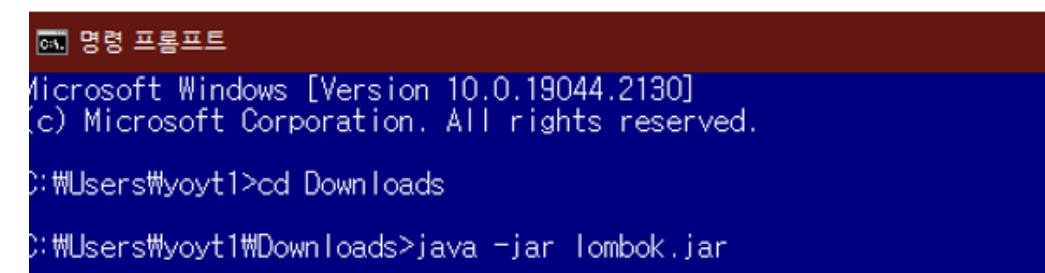
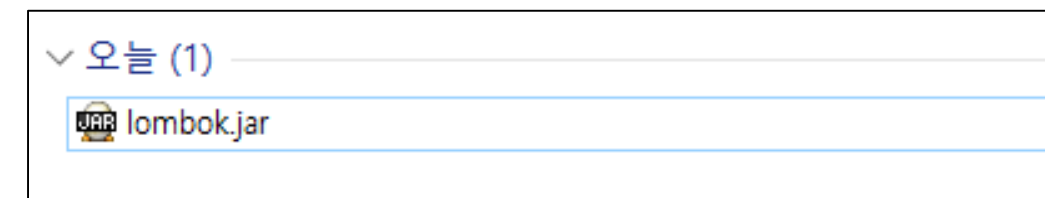
롬복은 개발자가 자주 사용하는 getter/setter, toString(), 생성자 등을 메소드를 자동으로 생성해주는 기능을 제공하는 플러그인이다.

롬복을 사용하면 반복적인 코딩 작업을 줄일 수 있다.

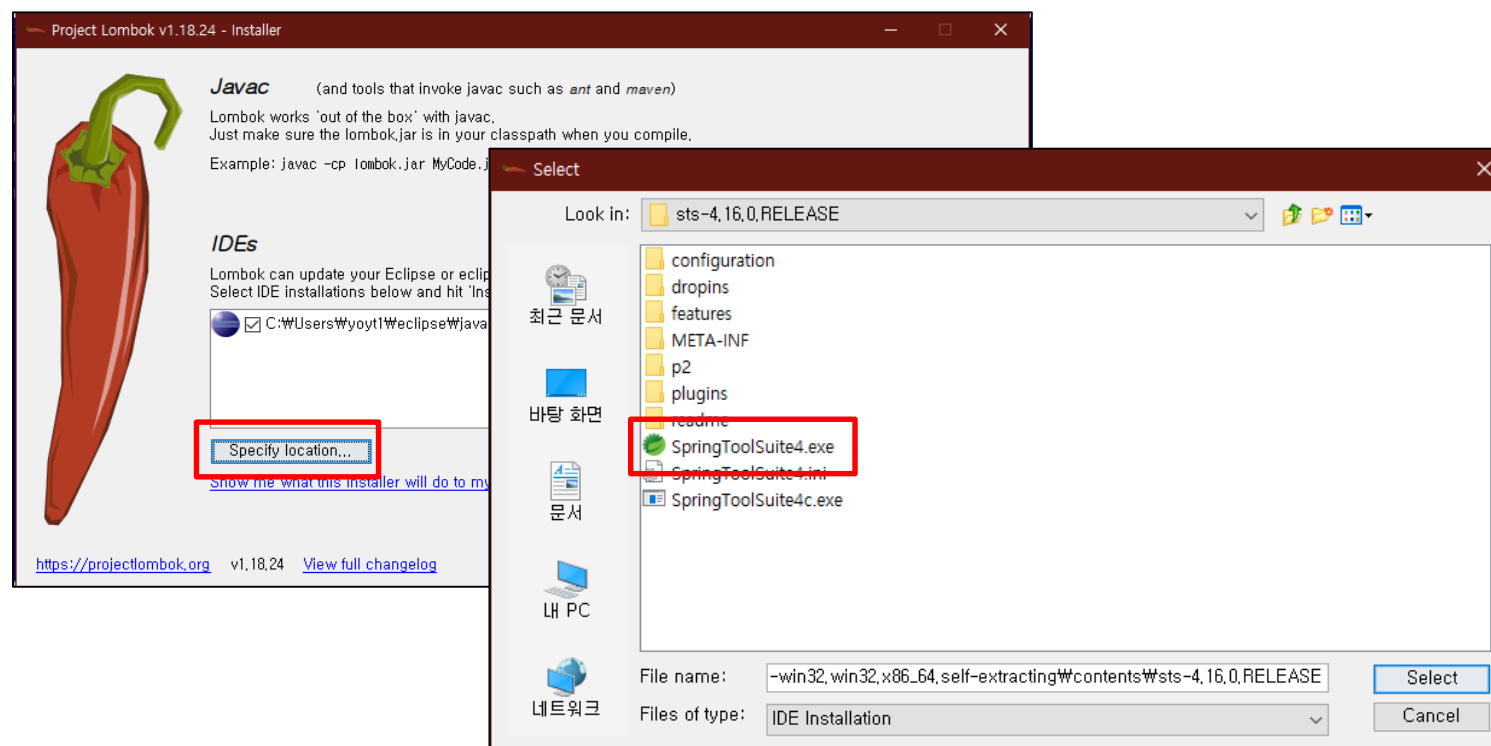
롬복 설치

1. 롬복 웹사이트를 접속해서 롬복을 다운로드한다.
2. 파일을 본일 폴더로 옮긴다.
3. STS를 종료하고 CMD를 실행한다.
4. lombok.jar 파일이 있는 위치로 이동한다.
5. `java -jar lombok.jar` 명령을 실행한다.

<https://projectlombok.org/download>



6. [Specify location..] 버튼을 클릭한다.
SpringToolSuite4.exe 파일을 선택한다.



7. IDEs 설치대상에서 이클립스 체크박스를 해제한다. STS 체크박스를 선택한다.

[Install] 버튼을 클릭한다.



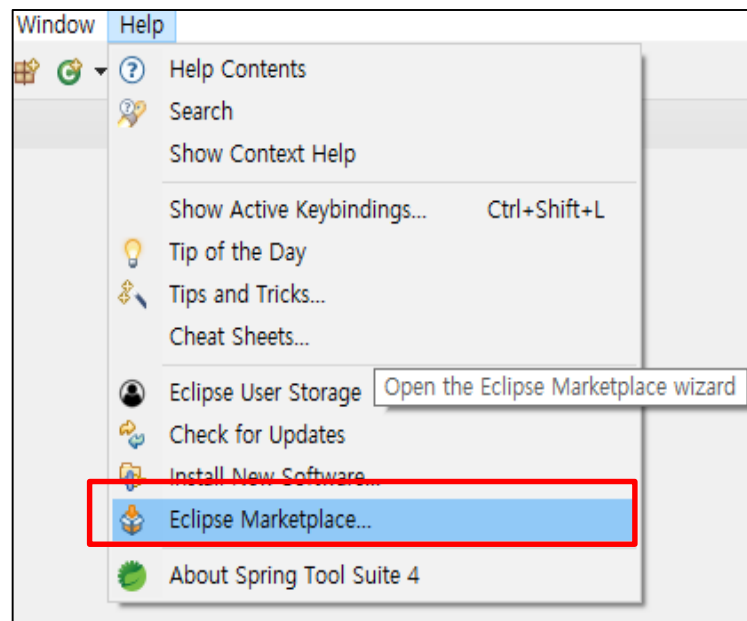
8. 설치가 끝나면 이제 STS에서 롬복을 사용할 수 있다.



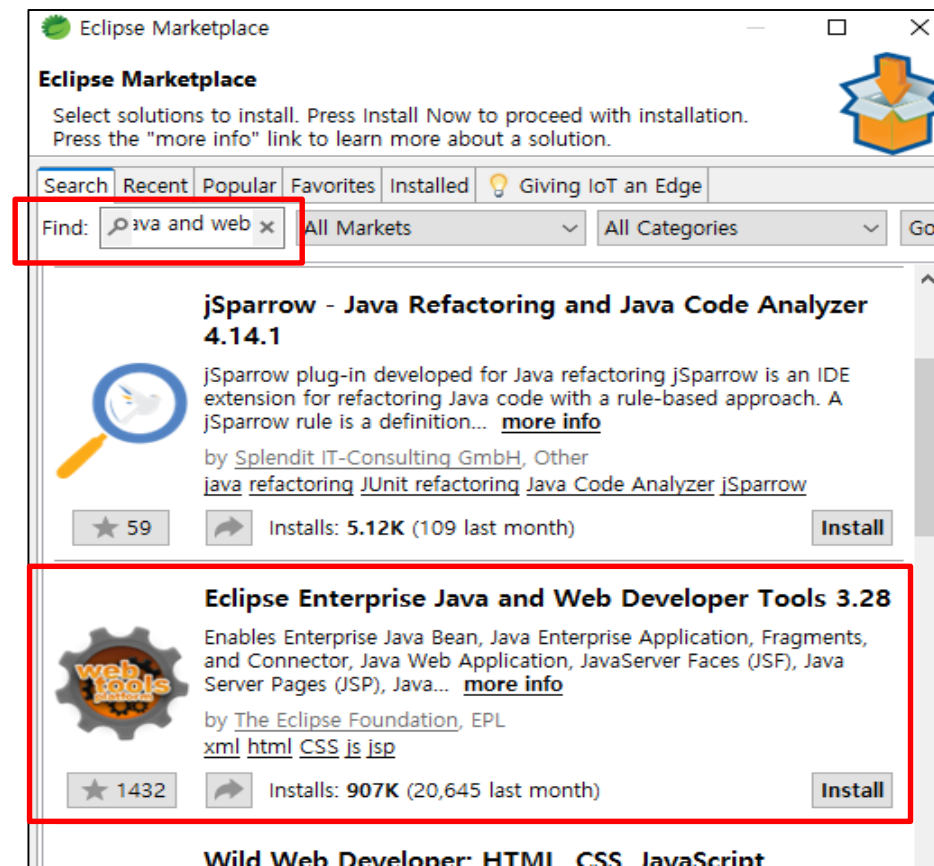
Web 플러그인을 추가하는 이유

해당 플러그인을 설치하면 HTML, CSS, JavaScript 등 웹 관련 파일을 사용할 수 있다.

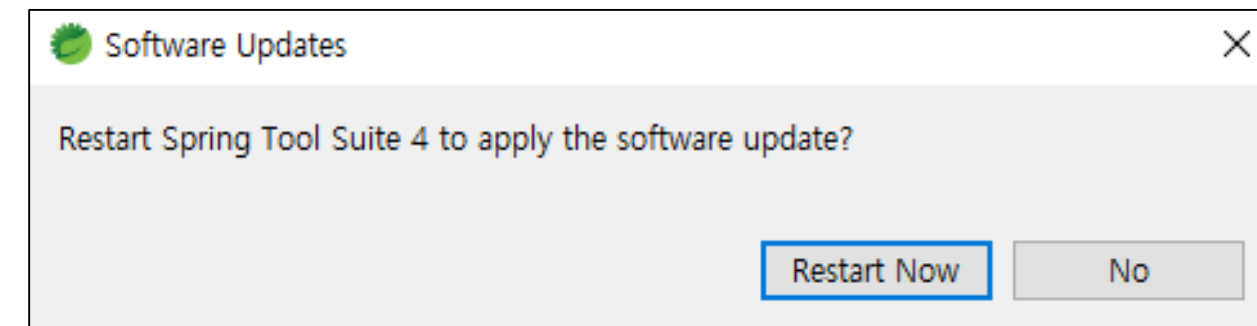
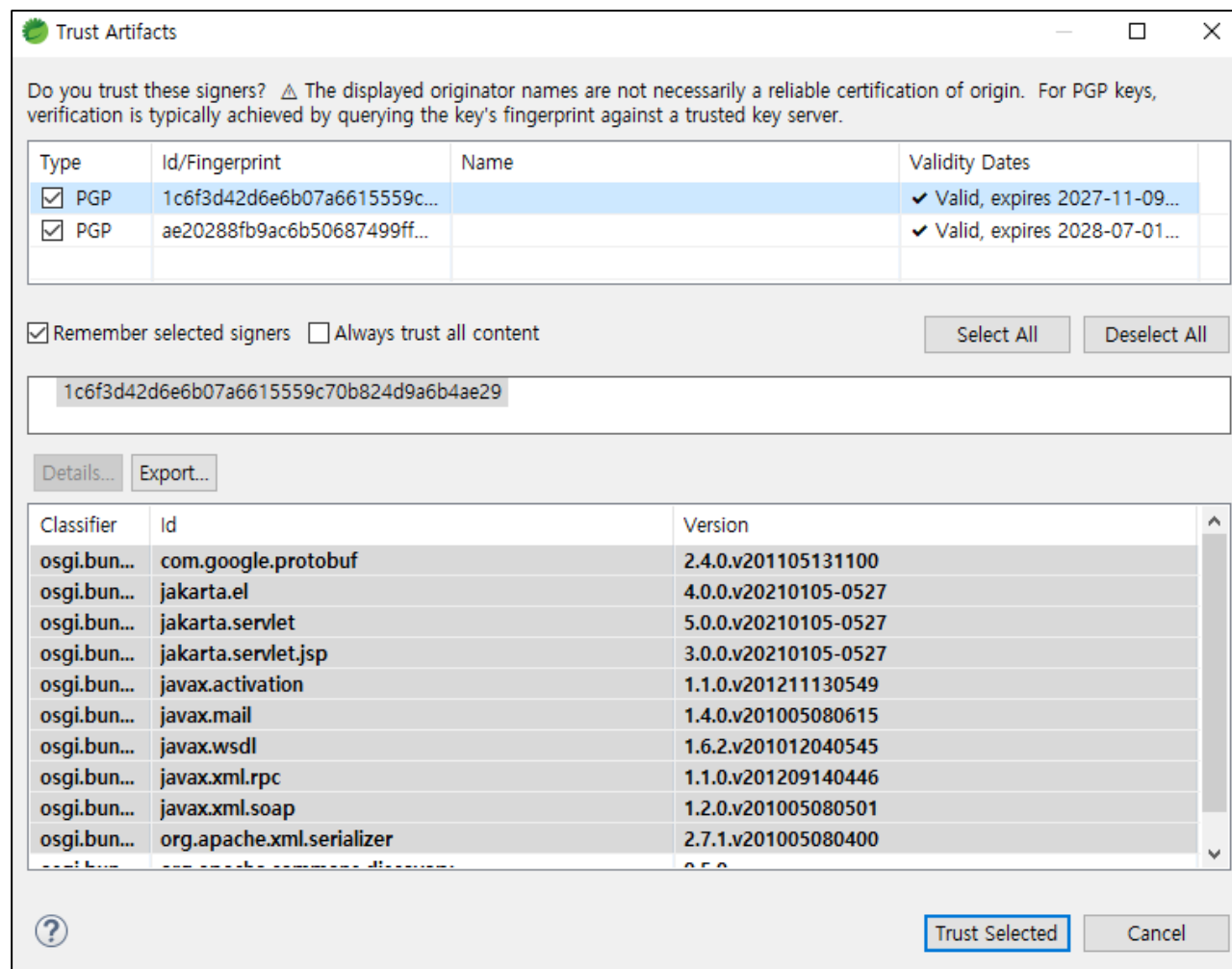
1. [Help – Eclipse Marketplace] 메뉴를 클릭한다.



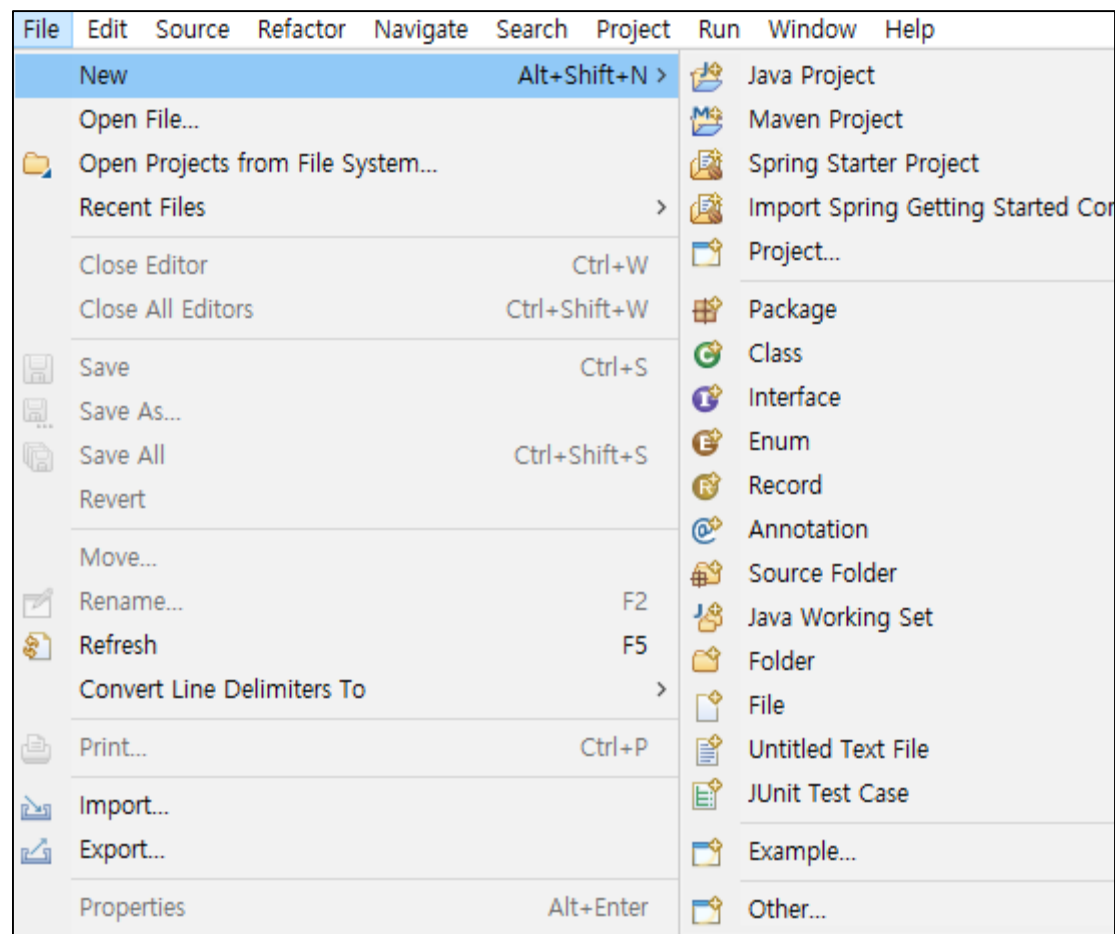
2. [java and web]를 검색한다.



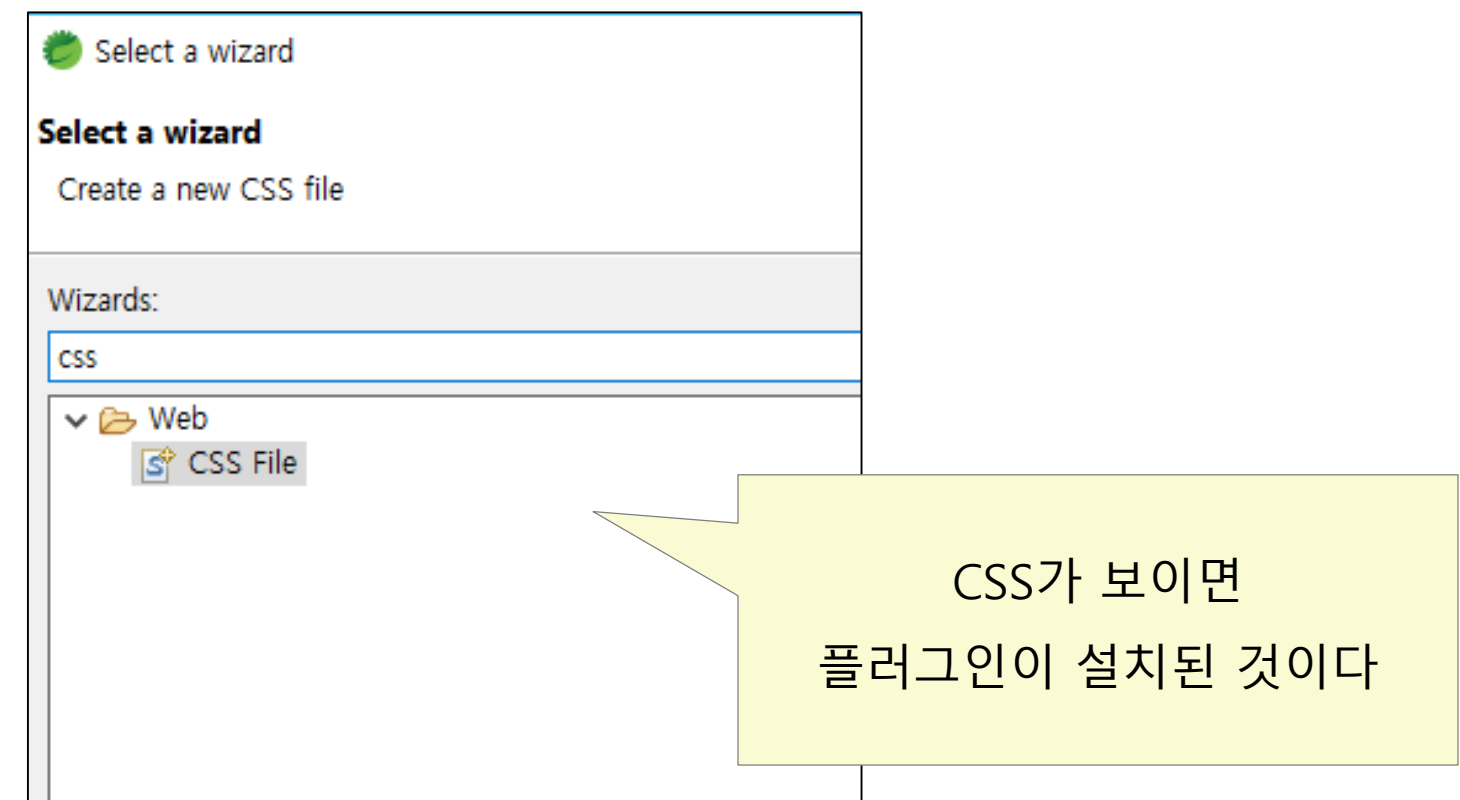
3. 설치 중간에 아래와 같은 화면이 뜨면, 모든 항목에 체크하고 [Trust Selected] 버튼을 누른다.
4. 설치가 끝나면 재시작 버튼을 누른다.



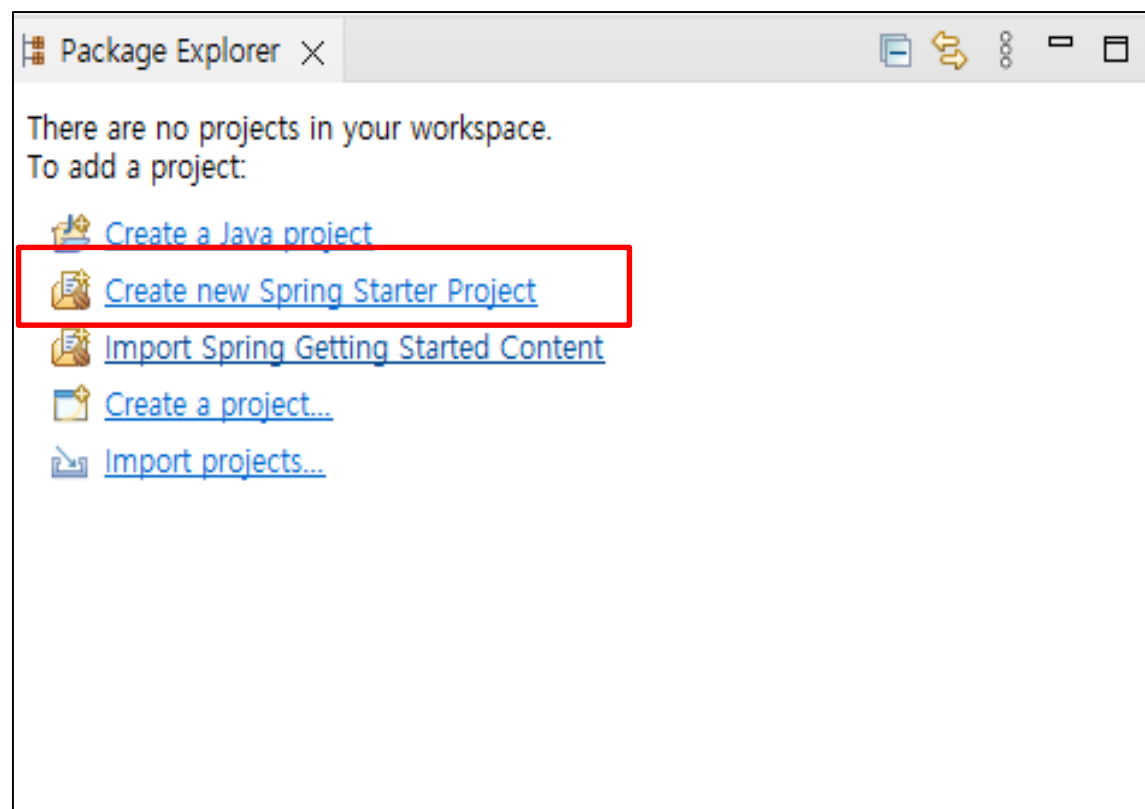
5. [File-New-Others] 메뉴를 클릭한다.



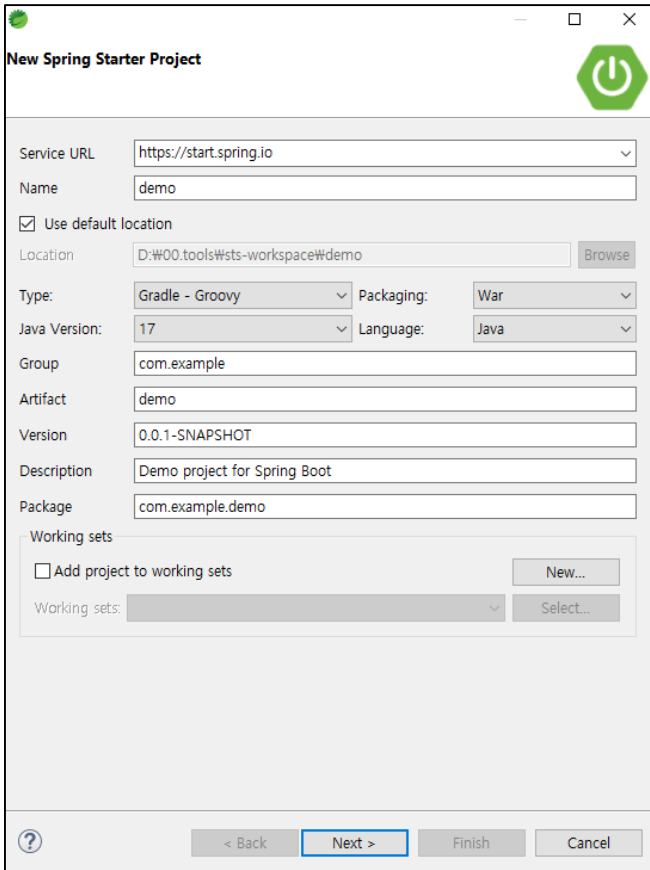
6. [css]를 검색한다.



1. [Create new Spring Starter Project] 버튼을 클릭하여 스프링부트 프로젝트를 생성한다.



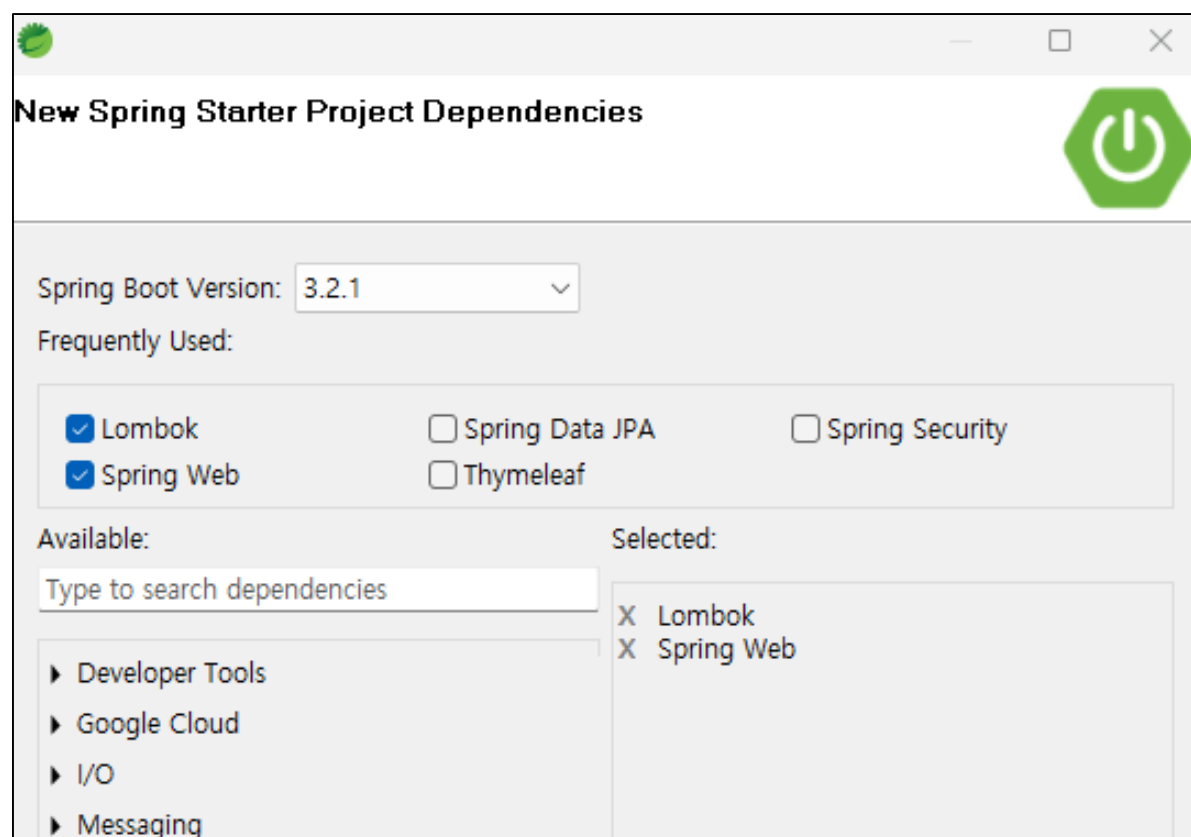
2. 다음과 같이 프로젝트 구조를 설정한다.
- Name, Group, Artifact는 프로젝트의 이름과 관련된 항목이다.
- *SpringBoot는 JAR 패키징을 권장한다.
- 하지만 외부 서버를 사용하거나, JSP가 포함된 경우에는 War를 사용하는게 좋다.



Type	라이브러리 관리 도구 (Gradle)
Packaging	패키징 파일 형식 (Jar)
Language	프로그래밍 언어 (Java)
JavaVersion	자바 버전 (17)
Group	프로젝트를 관리하는 그룹. 프로젝트 이름 같이 프로젝트를 구별하는 식별자 Ex. 회사이름
Artifact	빌드 결과물 파일의 이름
Version	빌드 결과물 파일의 버전
Description	프로젝트 설명
Package	기본 패키지명

3. 프로젝트에 필요한 라이브러리를 추가한다.

만약 외부 라이브러리가 필요하다면 별도로 추가해야 한다



Lombok, SpringWeb 선택

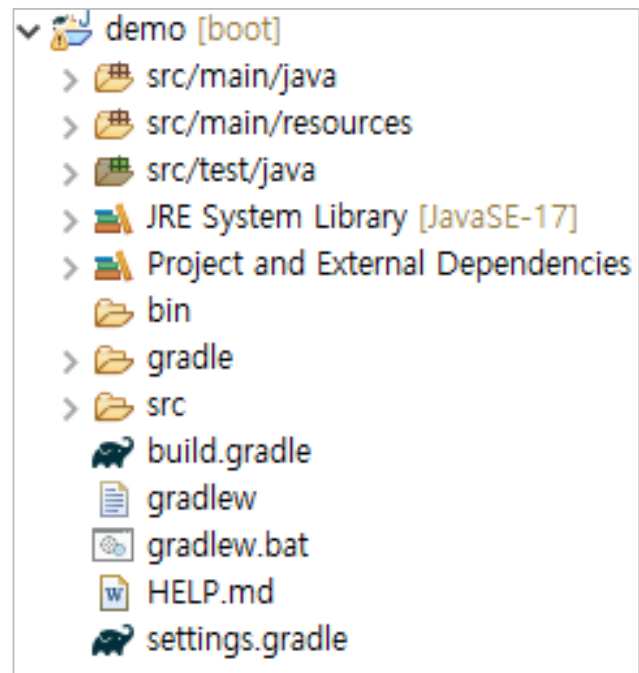
프로젝트 생성

프로젝트 생성

4. 정상적으로 프로젝트가 생성되었는지 확인한다.

프로젝트 생성 시, 필요한 라이브러리들이 원격 저장소에서 다운로드되기 때문에 시간이 오래 걸린다.

생성된 프로젝트의 ~Application 클래스를 보면 @SpringBootApplication이라는 어노테이션이 붙어있다.
이 클래스는 main 함수를 포함하고 있으며, 프로젝트를 실행하는 부분이다.



생성된 프로젝트

```
@SpringBootApplication
public class DemoApplication {

    public static void main(String[] args) {
        SpringApplication.run(DemoApplication.class, args);
    }

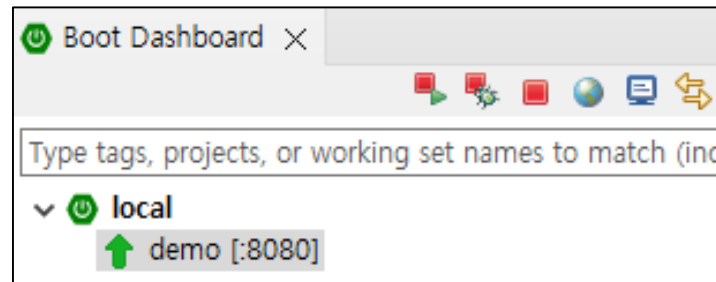
}
```

메인 클래스

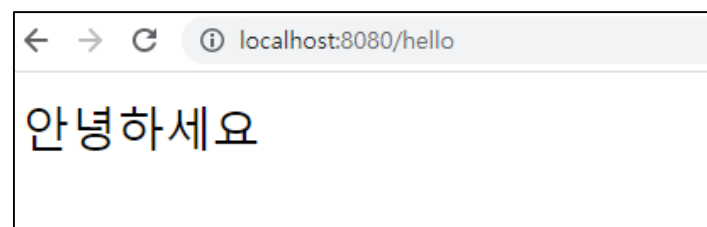
5. TestController 클래스를 생성한다.

```
@RestController
public class TestController {
    @GetMapping("/hello")
    public String hello() {
        return "안녕하세요";
    }
}
```

6. 스프링부트를 실행한다.

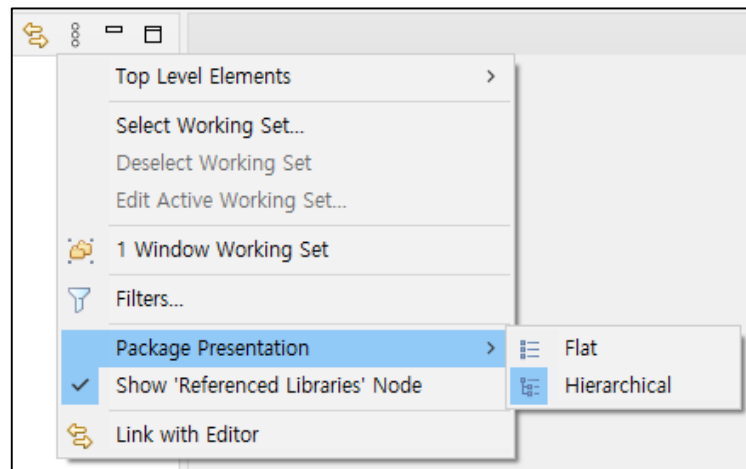


7. 브라우저로 'localhost:8080/hello'를 호출한다.

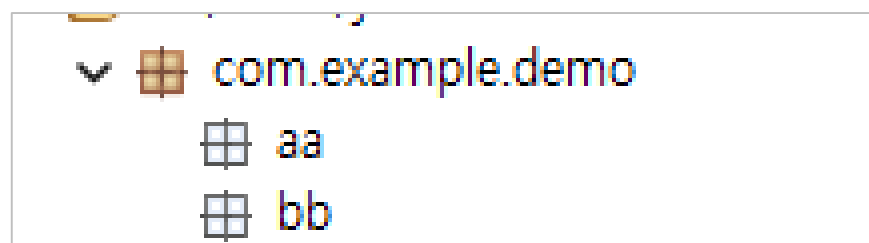


컨트롤러의 hello함수가 호출되어
"안녕하세요" 문자열이 전달되었다

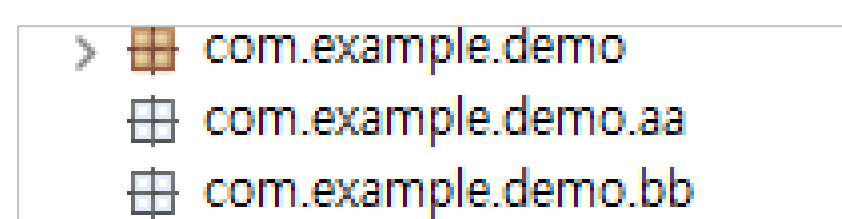
1. Package 바에서 [...] 점세개 버튼을 누른다.
2. [Package Presentation] 메뉴를 누르고 [Hierarchical] 를 선택한다.
Flat 방식은 폴더 구조를 나열하는 방식이기 때문에 전체 구조를 파악하기 어렵다.



3. 이제 패키지 표시가 전체경로로 표시되는지 확인한다.



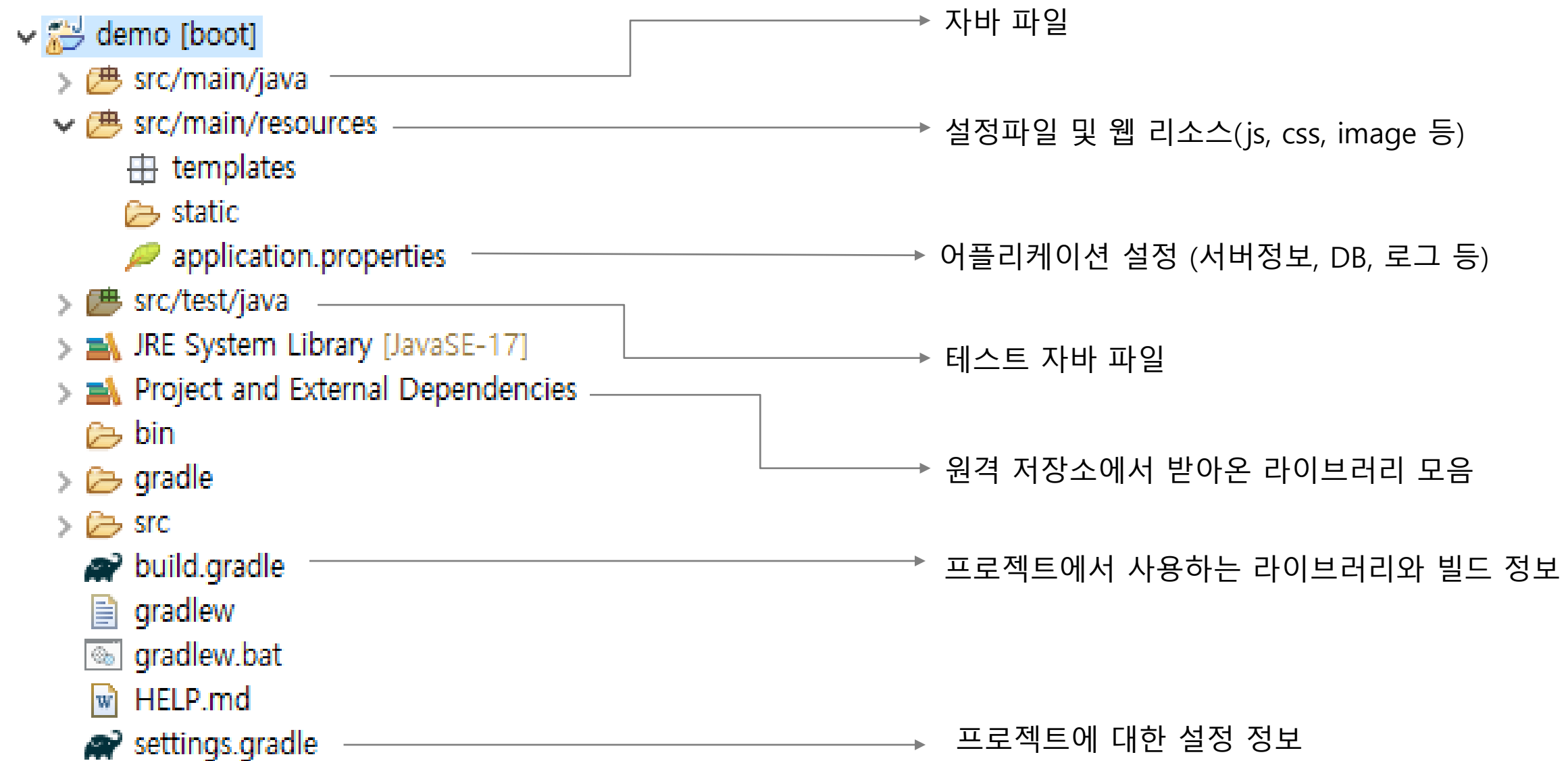
Hierarchical 방식



Flat 방식

프로젝트 살펴보기

프로젝트 구조



build.gradle

이 파일은 프로젝트의 구조, 빌드 방법, 사용할 라이브러리를 정의하는 파일이다.

dependencies 영역에서는 프로젝트 생성 시 추가한 라이브러리를 확인할 수 있다.

```
plugins {  
    id 'java'  
    id 'war'  
    id 'org.springframework.boot' version '3.0.2'  
    id 'io.spring.dependency-management' version '1.1.0'  
}  
  
group = 'com.example'  
version = '0.0.1-SNAPSHOT'  
sourceCompatibility = '17'  
  
configurations {  
    compileOnly {  
        extendsFrom annotationProcessor  
    }  
}  
  
repositories {  
    mavenCentral()  
}  
  
dependencies {  
    implementation 'org.springframework.boot:spring-boot-starter-thymeleaf'  
    implementation 'org.springframework.boot:spring-boot-starter-web'  
    compileOnly 'org.projectlombok:lombok'  
    annotationProcessor 'org.projectlombok:lombok'  
    providedRuntime 'org.springframework.boot:spring-boot-starter-tomcat'  
    testImplementation 'org.springframework.boot:spring-boot-starter-test'  
}
```

언어, 패키징형식, 스프링 버전

프로젝트 메타정보

라이브러리가 있는 원격 저장소

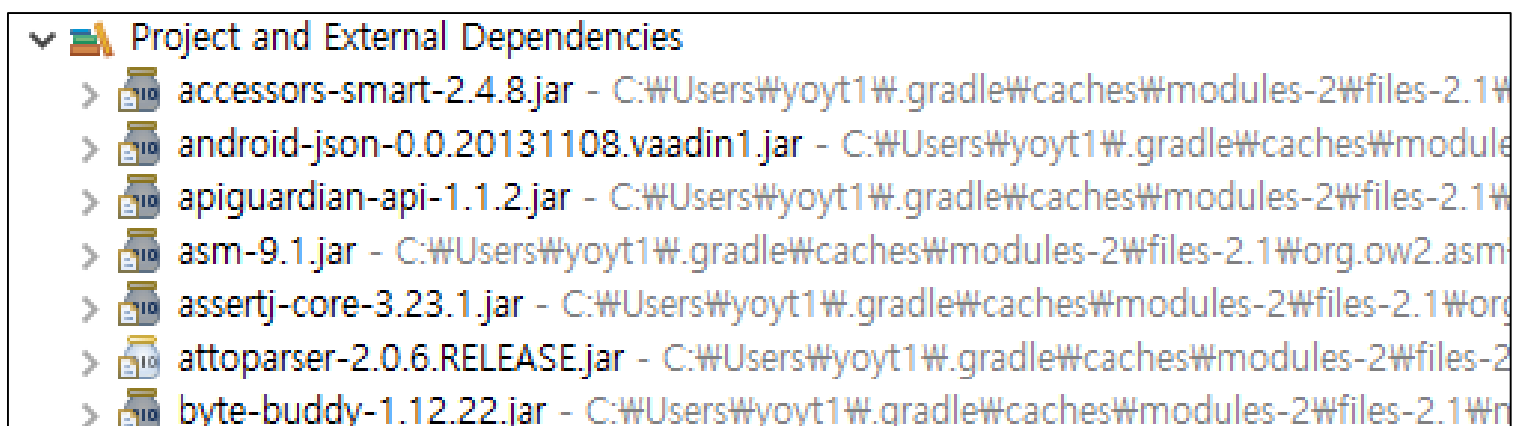
사용중인 라이브러리

Gradle

Gradle은 프로젝트를 관리하는 도구이다.

Gradle을 사용하면 원격저장소에서 프로젝트에 필요한 라이브러리를 다운로드하고, 프로젝트를 패키징 파일로 빌드할 수 있다.

다운로드된 라이브러리들은 .gradle 폴더에 저장된다.

















.gradle 폴더

SpringBoot 는 프로젝트 생성 시 자동으로 라이브러리가 추가 하는 등 자동화된 부분이 많다.

특히, Spring Web에는 JSON 데이터를 변환하는 기능이 제공하는 Jackson 라이브러리가 기본으로 포함되어 있다.

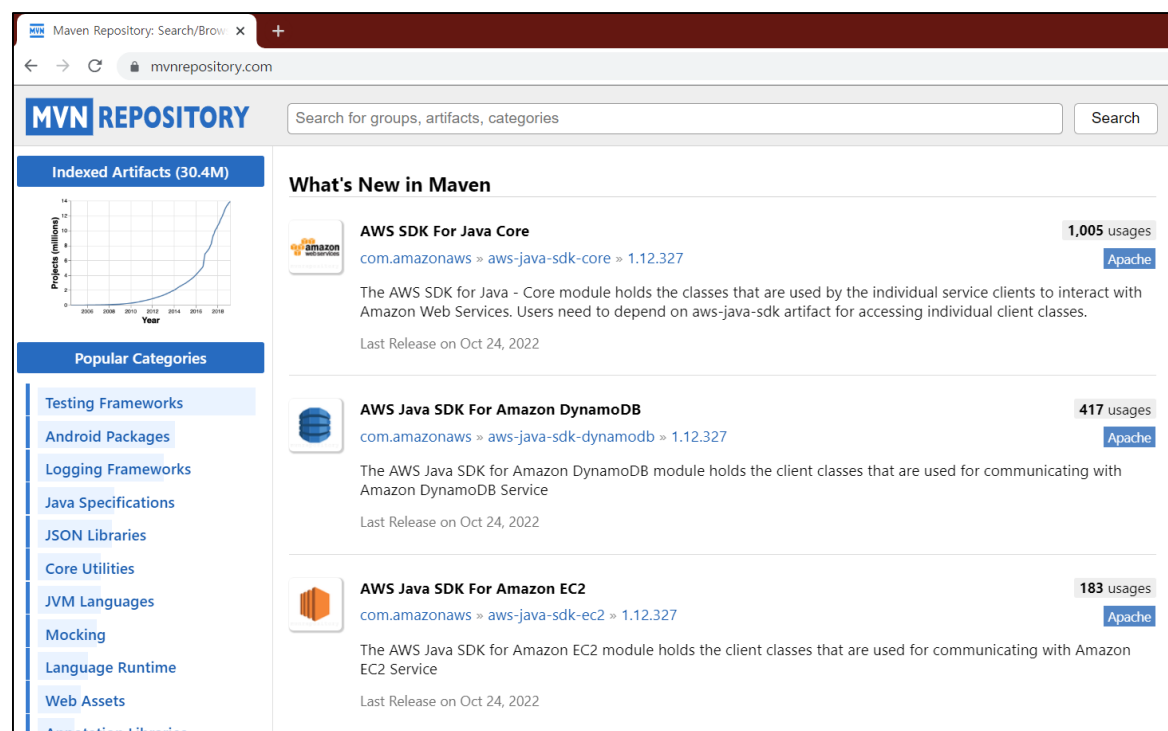
이와 달리, 스프링 프레임워크는 수동으로 설정해야 하는 부분이 많다.

- ▼  Project and External Dependencies
 - >  accessors-smart-2.5.1.jar - C:\Users\Wimjiyeo
 - >  android-json-0.0.20131108.vaadin1.jar - C:\
 - >  apiguardian-api-1.1.2.jar - C:\Users\Wimjiyeo
 - >  asm-9.6.jar - C:\Users\Wimjiyeon\gradle\ca
 - >  assertj-core-3.25.3.jar - C:\Users\Wimjiyeon\
 - >  awaitility-4.2.2.jar - C:\Users\Wimjiyeon\gra
 - >  byte-buddy-1.14.19.jar - C:\Users\Wimjiyeon\
 - >  byte-buddy-agent-1.14.19.jar - C:\Users\Wim
 - >  hamcrest-2.2.jar - C:\Users\Wimjiyeon\gradl
 - >  jackson-annotations-2.17.2.jar - C:\Users\Win
 - >  jackson-core-2.17.2.jar - C:\Users\Wimjiyeon\
 - >  jackson-databind-2.17.2.jar - C:\Users\Wimjiy
 - >  jackson-datatype-idk8-2.17.2.jar - C:\Users\Wimjiy

스프링 프로젝트는 메이븐 리파지토리에서 필요한 라이브러리를 가져온다.

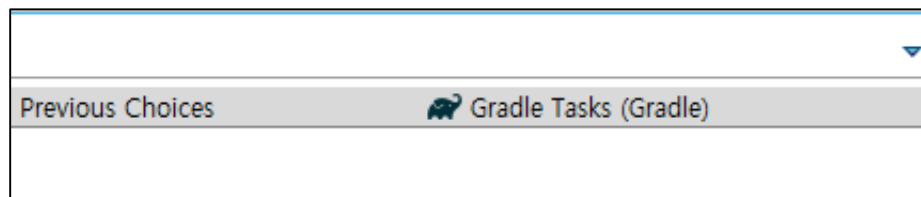
이 사이트에서는 프로젝트에서 사용할 수 있는 다양한 라이브러리를 제공한다.

<https://mvnrepository.com/>

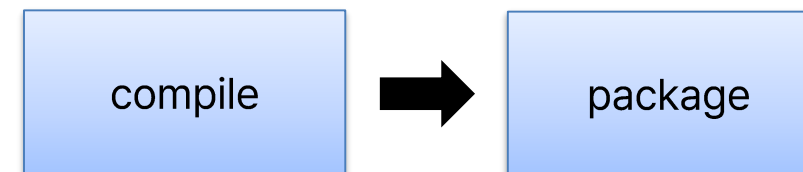
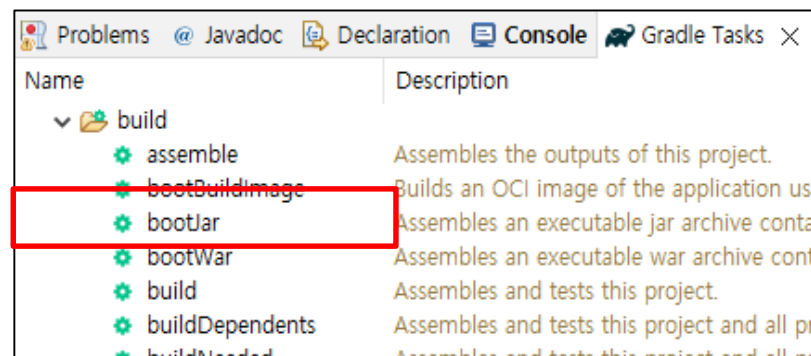


Gradle 빌드 도구를 사용하여 스프링 부트 프로젝트 실행 파일을 만든다.

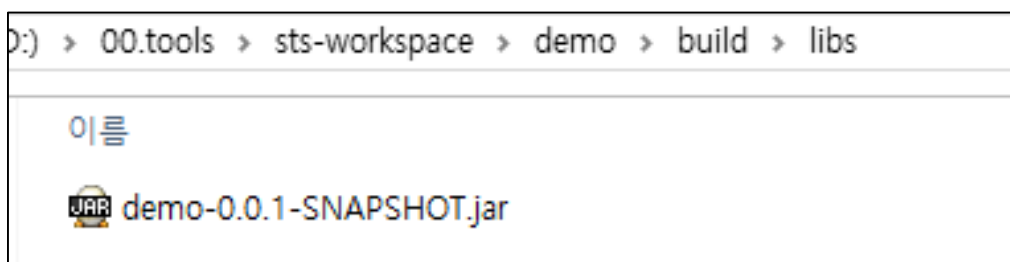
1. 퍼스펙티브 메뉴에서 "gradle task"를 검색한다.



2. [bootjar] 메뉴를 실행하여 컴파일과 패키징을 수행한다.



3. 프로젝트 > build > libs 폴더 밑에서 패키지 파일이 만들어졌는지 확인한다.



1. CMD를 실행한다.
2. Jar파일이 있는 위치로 이동한다.
3. "java -jar demo~.jar" 를 입력하여 프로젝트를 실행한다.

```

C:\>명령 프롬프트 - java -jar demo-0.0.1-SNAPSHOT.jar

D:\00.tools\sts-workspace\demo\build\libs>java -jar demo-0.0.1-SNAPSHOT.jar

  ____  _
 / ___|| | | |
| |___| |_| |
 \___ \|  _/
      |_|

:: Spring Boot ::
                 (v3.0.2)

2023-02-06T20:45:21.887+09:00 INFO 3500 --- [main] com.example.demo.DemoApplication : Starting DemoApplication using
ava 17.0.6 with PID 3500 (D:\00.tools\sts-workspace\demo\build\libs\demo-0.0.1-SNAPSHOT.jar started by yoyt1 in D:\00.tools\sts-workspace\
demo\build\libs)
2023-02-06T20:45:21.891+09:00 INFO 3500 --- [main] com.example.demo.DemoApplication : No active profile set, falling b
ack to 1 default profile: "default"
2023-02-06T20:45:23.355+09:00 INFO 3500 --- [main] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat initialized with port(s)
8080 (http)
2023-02-06T20:45:23.372+09:00 INFO 3500 --- [main] o.apache.catalina.core.StandardService : Starting service [Tomcat]
2023-02-06T20:45:23.373+09:00 INFO 3500 --- [main] o.apache.catalina.core.StandardEngine : Starting Servlet engine: [Apache
Tomcat/10.1.5]
2023-02-06T20:45:23.516+09:00 INFO 3500 --- [main] o.a.c.c.C.[Tomcat].[localhost].[/] : Initializing Spring embedded Web
ApplicationContext
2023-02-06T20:45:23.521+09:00 INFO 3500 --- [main] w.s.c.ServletWebServerApplicationContext : Root WebApplicationContext: ini
tialization completed in 1513 ms
2023-02-06T20:45:24.093+09:00 INFO 3500 --- [main] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat started on port(s): 8080
(http) with context path ''
2023-02-06T20:45:24.160+09:00 INFO 3500 --- [main] com.example.demo.DemoApplication : Started DemoApplication in 2.82
seconds (process running for 3.461)

```