2024-2 오픈소스 개발 프로젝트01

SW중심대학사업단 강재구

(학연산 742호, 010-2278-8192, kangjk@cbnu.ac.kr)

01

충북대학교
SW중심대학사업단

01. 스프링 부트 개발 준비하기

01

스프링 부트 개발 준비하기

1-01 스프링 부트란?

- **스프링 부트(Spring Boot)**
  - 웹 프로그램(웹 애플리케이션)을 쉽고 빠르게 만들 수 있도록 도와주는 자바의 웹 프레임워크
  - 스프링 부트는 스프링(Spring) 프레임워크에 톰캣(Tomcat)이라는 서버를 내장하고 여러 편의 기능들을 추가하여 개발자들 사이에서 꾸준히 인기를 누리고 있음

1-01 스프링 부트란?

- **웹 프레임워크란?**
  - 웹 프로그램을 완성하려면 쿠키나 세션 처리, 로그인/로그아웃 처리, 권한 처리, 데이터베이스 처리 등 만들어야 할 기능이 많음
  - 하지만 웹 프레임워크를 사용하면 이런 기능을 일일이 만들 필요가 없음. 웹 프레임워크에는 그런 기능들이 이미 만들어져 있기 때문
  - 쉽게 말해 웹 프레임워크는 웹 프로그램을 만들기 위한 스타터 키트
  - 자바로 만든 웹 프레임워크 중 하나가 바로 스프링 부트

1-01 스프링 부트란?

▪ 웹 프레임워크란?

- 스프링 부트의 몇 가지 규칙만 익히면 기존에 자바로 웹 프로그램을 작성하는 방식보다 빠르게 웹 프로그램을 만들 수 있음

- 크롬이나 사파리와 같은 웹 브라우저에 'Hello World'를 출력하려면 다음과 같은 클래스 하나만 작성하면 됨

스프링 부트의 빠른 개발 속도를 보여 주는 예

```
@Controller
public class HelloController {
    @GetMapping("/")
    @ResponseBody
    public String hello() {
        return "Hello World";
    }
}
```

1-01 스프링 부트란?

- **스프링 부트를 배워야 하는 이유**
  - **스프링 부트는 튼튼한 웹 프레임워크이다**
    - 개발자가 웹 프로그램을 만들 때 어렵게 느끼는 기능 중 하나는 바로 보안 기능
    - 웹 사이트를 괴롭히는 사람의 공격에 개발자 홀로 신속하게 대응하기는 무척 어려운 일
    - 스프링 부트가 이런 보안 공격을 기본으로 아주 잘 막아줌
    - 예를 들어 SQL 인젝션, XSS(cross-site scripting), CSRF(Cross-Site Request Forgery), 클릭재킹(clickjacking)과 같은 보안 공격을 막음
    - 스프링 부트를 사용하면 이런 보안 공격을 막아 주는 코드를 짤 필요가 없음

1-01 스프링 부트란?

▪ 스프링 부트를 배워야 하는 이유

▪ 스프링 부트에는 여러 기능이 준비되어 있다

- 스프링 부트는 2012년에 등장하여 10년 이상의 세월을 감내한 '베테랑' 웹 프레임워크
- 무수히 많은 기능이 추가되고 또 다듬어져
  스프링 부트에는 웹 프로그램을 개발하는 데 필요한 도구와 기능이 대부분 준비됨

▪ 스프링 부트는 WAS가 필요없다

- 스프링 부트 대신 스프링만 사용하여 웹 애플리케이션을 개발한다면
  실행할 수 있는 톰캣과 같은 WAS(Web Application Server)가 필요함
- WAS의 종류는 매우 다양하며 설정 방식도 제각각이어서 공부해야 할 내용도 상당함
- 하지만 스프링 부트에는 톰캣 서버가 내장되어 있고 설정도 자동 적용되기 때문에
  WAS에 대해서 전혀 신경 쓸 필요가 없음
- 심지어 배포되는 jar 파일에도 톰캣 서버가 내장되어 실행되므로
  서로 다른 WAS들로 인해 발생되는 문제들도 사라짐

1-01 스프링 부트란?

- **스프링 부트를 배워야 하는 이유**
  - **스프링 부트는 설정이 쉽다**
    - 스프링 부트가 등장하기 전 개발자들은 스프링을 사용하여 웹 애플리케이션을 개발했는데 스프링의 복잡한 설정 때문에 개발자들은 많은 어려움을 겪음
    - 스프링 부트는 스프링의 복잡한 설정을 자동화·단순화함
  - **스프링 부트는 재미있다**
    - 스프링 부트로 웹 프로그램을 만드는 것이 게임을 하는 것보다 재미있음

1-02 스프링 부트 개발 환경 준비하기

JDK 설치하기

- 자바 프로그래밍을 하려면 꼭 필요한 JDK(Java Development Kit)를 설치해야 함
- JDK는 자바로 코드를 실행하는 도구와 코드를 번역하는 컴파일러 등으로 이루어짐
- JDK를 내려받을 수 있는 URL은 다음과 같음(검색창에서 'JDK Download' 로 검색)

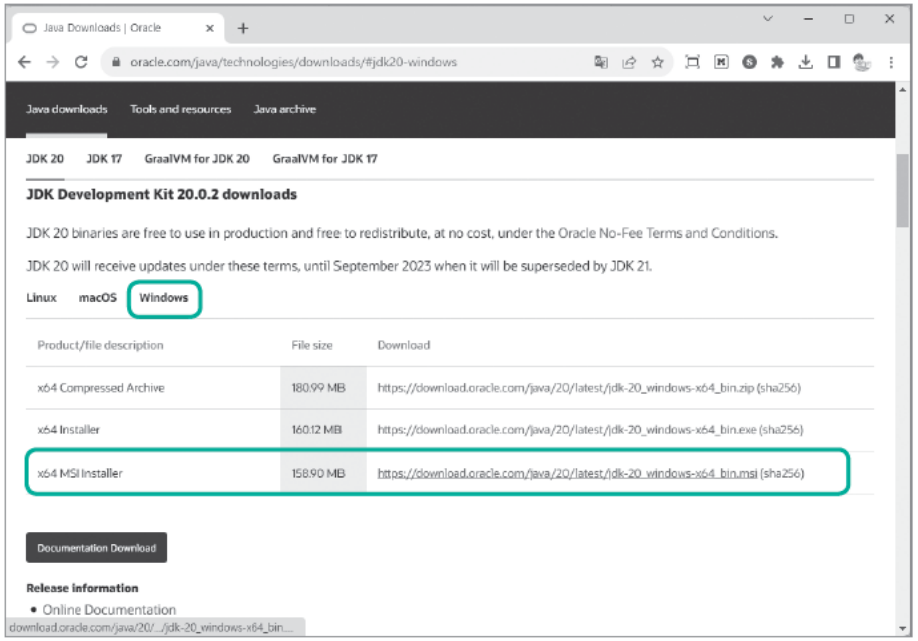
```
https://www.oracle.com/java/technologies/downloads/
```

1-02 스프링 부트 개발 환경 준비하기

▪ JDK 설치하기

▪ 윈도우에 JDK 설치하기

1. 윈도우에 설치한다면 [Windows] 탭을 선택한 후, [x64 MSI Installer] JDK를 내려받기

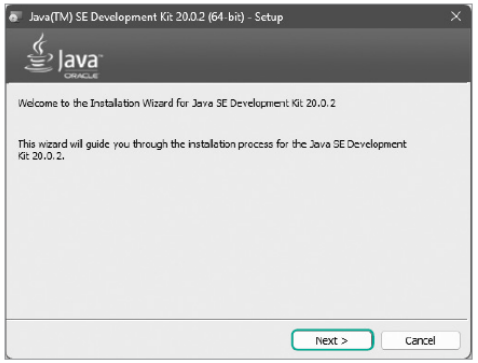


1-02 스프링 부트 개발 환경 준비하기

▪ JDK 설치하기

▪ 윈도우에 JDK 설치하기

2. 오른쪽 아이콘을 더블클릭하여 설치하기
3. Setup 창이 뜨면 [Next] 버튼을 클릭하고, [Close] 버튼이 나올 때까지 설치를 진행



2. JDK를 설치했다면 아마도 다음과 비슷한 디렉터리에 설치됨

```
c:\program files\java\jdk-20
```

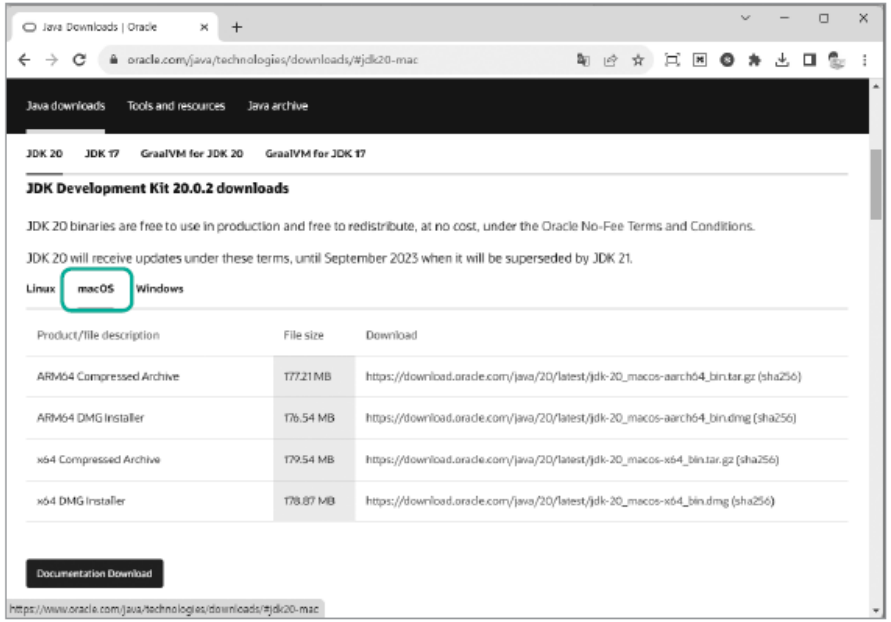
JDK 버전에 따라 다르다.

1-02 스프링 부트 개발 환경 준비하기

▪ JDK 설치하기

▪ macOS에 JDK 설치하기

1. macOS 환경에 설치한다면 [macOS] 탭을 선택한 후, 자신의 환경에 맞는 dmg 파일을 내려받음



1-02 스프링 부트 개발 환경 준비하기

▪ JDK 설치하기

▪ macOS에 JDK 설치하기

2. 다음 아이콘을 차례로 더블클릭하여 설치
3. JDK 설치 화면이 뜨면 [계속] 버튼을 클릭하고 설치를 진행

2. dmg 파일을 설치한 후 터미널에서 java -version 명령을 입력 및 실행하여 자바가 설치되었는지 확인

```
pahkey@mymac ~ % java -version   ← 이 부분을 입력한다.
java version "20.0.2"
(... 생략 ...)
```

1-02 스프링 부트 개발 환경 준비하기

▪ STS 설치하기

- JDK를 설치했다면 이번에는 스프링 부트 프로그램을 작성할 수 있도록 도와주는 도구인 STS(Spring Tool Suite)를 설치해야 함
- 문서 작성을 도와주는 도구로 MS 워드나 한글 프로그램이 있는 것처럼 자바 프로그램을 작성할 수 있도록 도와주는 도구들이 있음
- 이러한 도구를 IDE(Intergrated Development Environment) 또는 통합 개발 환경이라고 함
- 스프링 부트 IDE 중 가장 많이 추천하는 것은 STS.
  STS는 스프링 개발에 최적화된 에디터로 이클립스 기반으로 제작됨

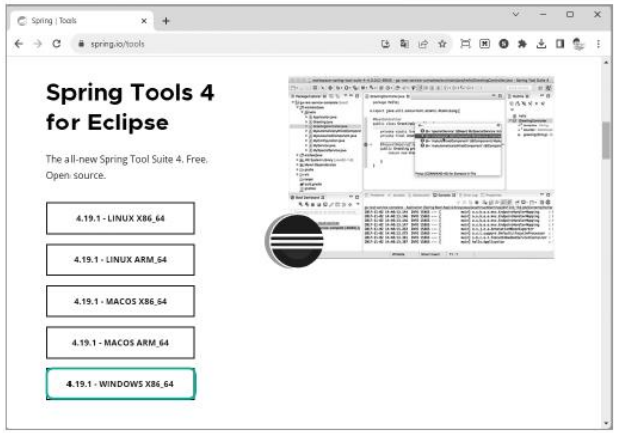
1-02 스프링 부트 개발 환경 준비하기

▪ STS 설치하기

1. STS를 내려받을 수 있는 URL은 다음과 같음

```
https://spring.io/tools
```

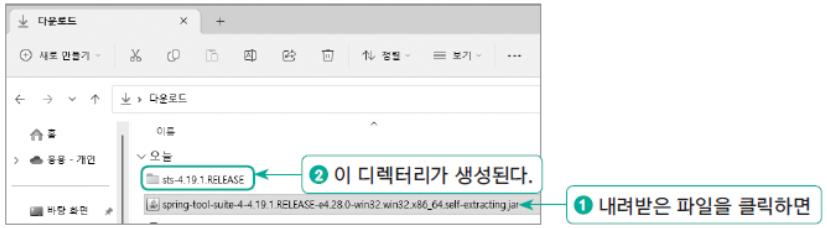
1. 윈도우에 설치한다면 [WINDOWS X86_64] 버튼을 눌러 STS 프로그램을 내려받자



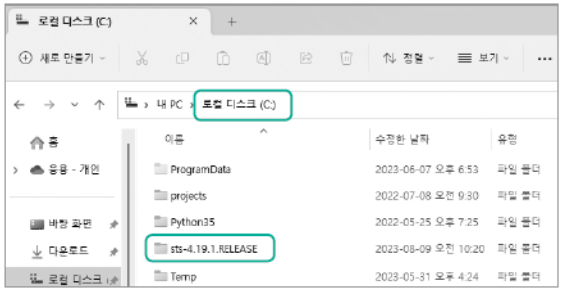
1-02 스프링 부트 개발 환경 준비하기

▪ STS 설치하기

3. 내려받은 파일을 더블클릭하여 실행.
그러면 해당 파일이 있는 위치에
'sts-4.19.1.RELEASE'라는 이름의 디렉터리가 생성



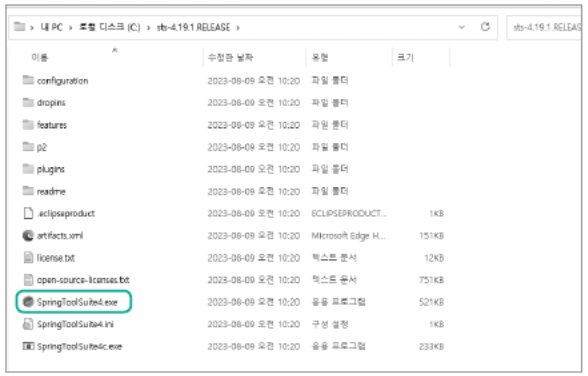
3. 새롭게 생성된 디렉터리를
C:₩ 디렉터리로 이동시켜
STS 설치를 마무리함



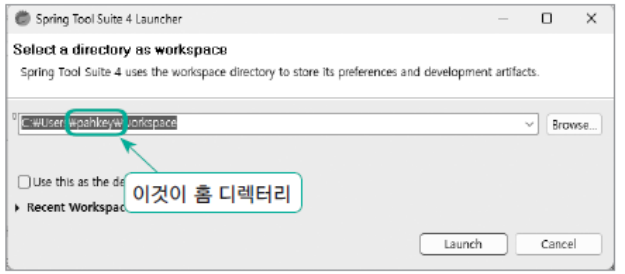
1-02 스프링 부트 개발 환경 준비하기

▪ STS 실행하기

1. STS를 설치한 디렉터리에서 SpringToolSuite4.exe 파일을 실행



1. 설치한 STS를 실행하면 가장 먼저 STS의 작업 공간(workspace) 디렉터리를 설정하는 창이 나타남



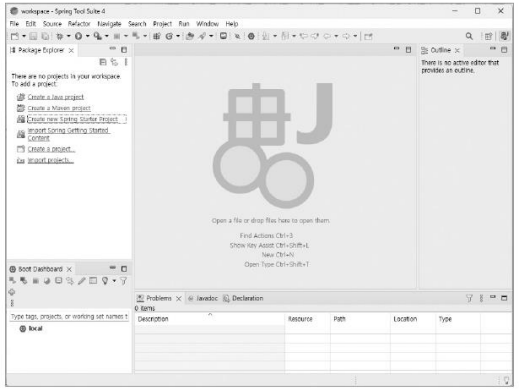
1-02 스프링 부트 개발 환경 준비하기

▪ STS 실행하기

2. 앞으로 STS로 작성하는 모든 파일은 이 디렉터리 안에 있음.
   사용자 홈 디렉터리 안에 workspace 디렉터리를 지정한 후 [Launch]를 클릭.

   이때 여기서 지정한 디렉터리(C:₩Users₩(사용자 홈 디렉터리)₩workspace)는
   앞으로 표시되는 파일들의 루트 디렉터리가 된다는 점을 기억하자

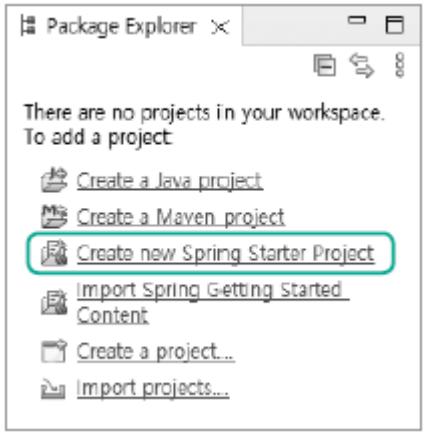
3. 그러면 STS가 실행됨



1-02 스프링 부트 개발 환경 준비하기

▪ 스프링 부트 프로젝트 만들기

1. 이번에는 STS 왼쪽에 표시된 'Create new Spring Starter Project'를 클릭해 스프링 부트 프로젝트를 생성해 보자

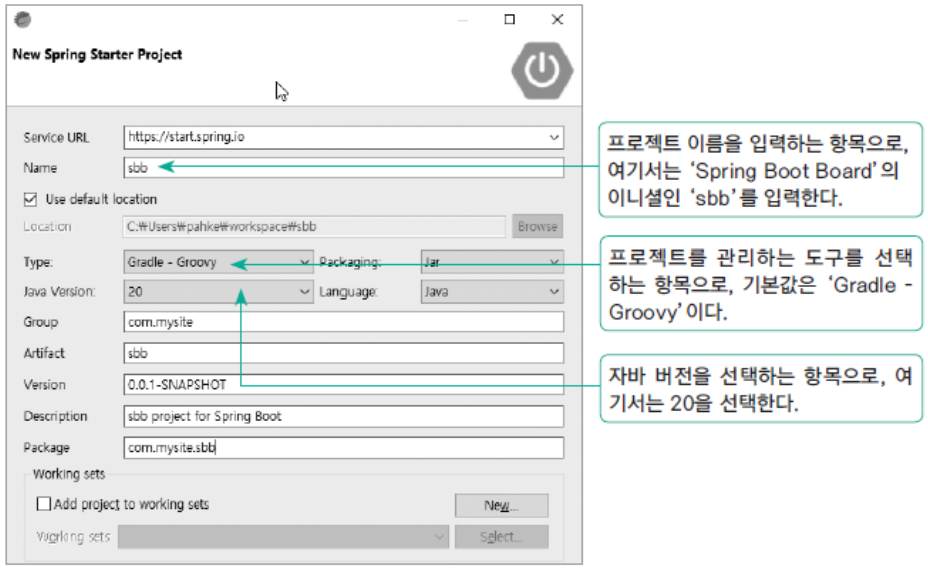


1. 'Create new Spring Starter Project'를 클릭하면 다음과 같은 설정 화면이 나타남. 이 부분은 매우 중요하므로 각 항목이 무엇인지 살펴보고 주의 깊게 입력해 보자

1-02 스프링 부트 개발 환경 준비하기

▪ 스프링 부트 프로젝트 만들기

2.

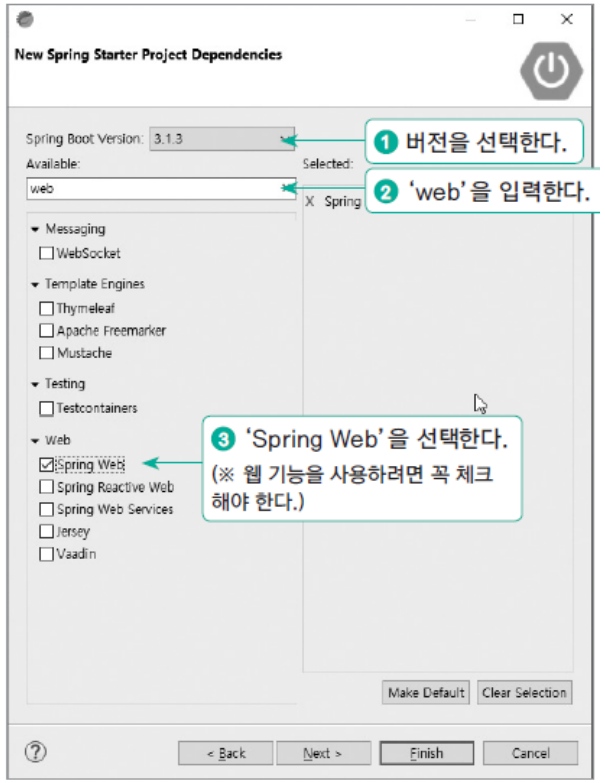


그 외에 Group, Artifact, Desrciption, Package 등은
이후 예제를 수월하게 진행하기 위해 위와 동일하게 설정하는 것을 추천.
이와 같이 설정했으면 [Next]를 클릭함

1-02 스프링 부트 개발 환경 준비하기

▪ 스프링 부트 프로젝트 만들기

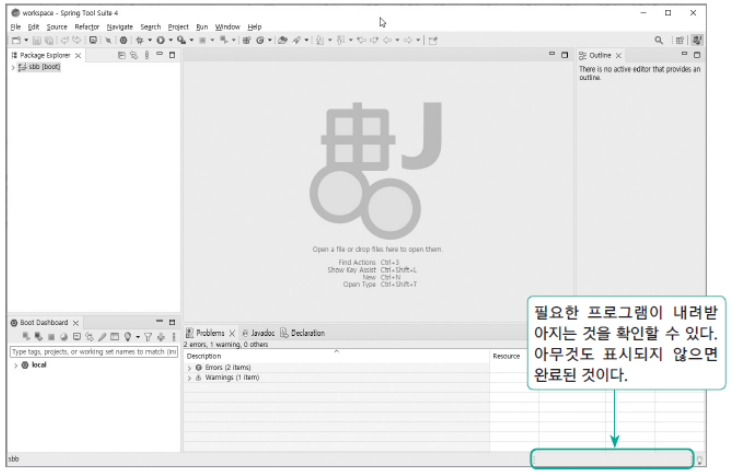
3. 스프링 부트 버전을 선택하는 화면이 나타남



1-02 스프링 부트 개발 환경 준비하기

▪ 스프링 부트 프로젝트 만들기

4. [Finish]를 클릭해 프로젝트를 생성했다면 STS가 실행됨

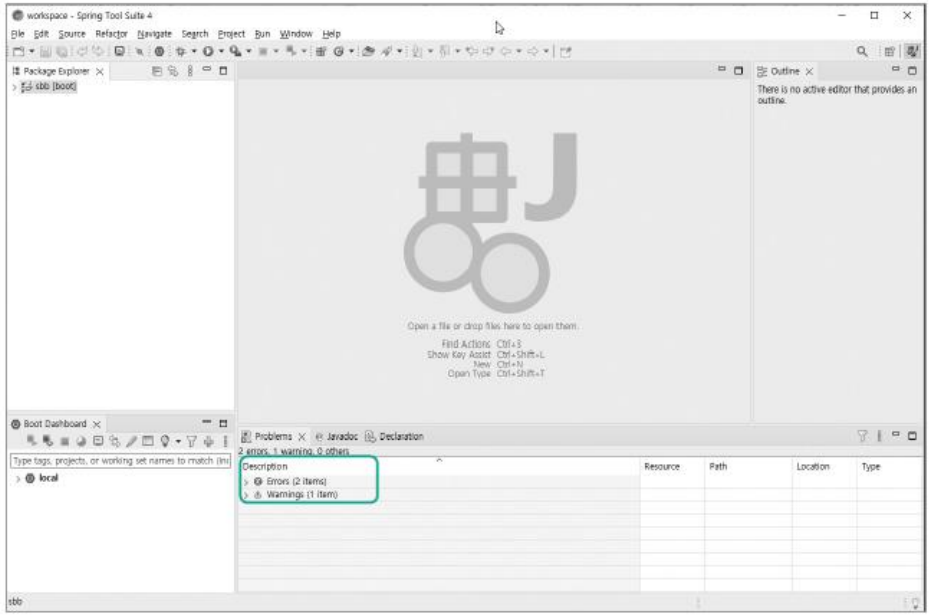


화면 왼쪽 상단에 sbb 프로젝트가 생성된 것을 확인할 수 있음.
프로젝트를 생성하면 프로젝트에 필요한 프로그램들이 자동으로 내려받아 짐.
조금 기다리면 내려받기가 완료됨

1-02 스프링 부트 개발 환경 준비하기

▪ 스프링 부트 프로젝트 만들기

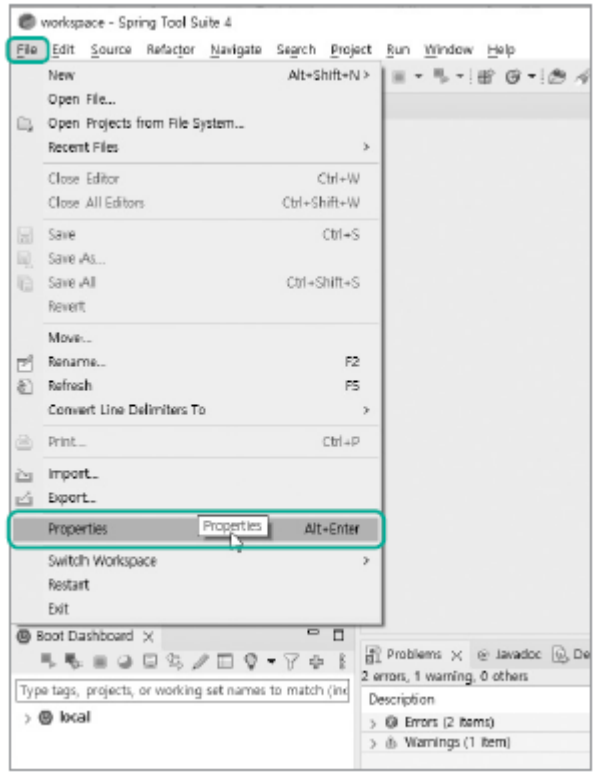
5. 내려받기를 완료한 후에 프로젝트에 오류가 발생할 수 있음.
   오류는 STS에 JDK가 제대로 설정되지 않아 발생함. 오류를 해결해 보자



1-02 스프링 부트 개발 환경 준비하기

▪ 스프링 부트 프로젝트 만들기

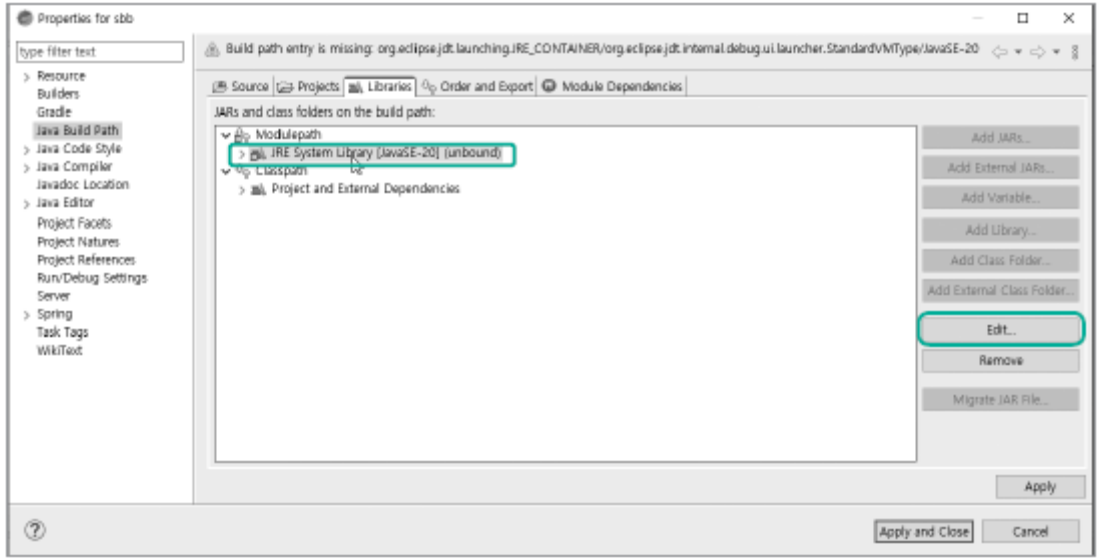
6. STS 상단의 [File → Properties]를 차례로 클



1-02 스프링 부트 개발 환경 준비하기

▪ 스프링 부트 프로젝트 만들기

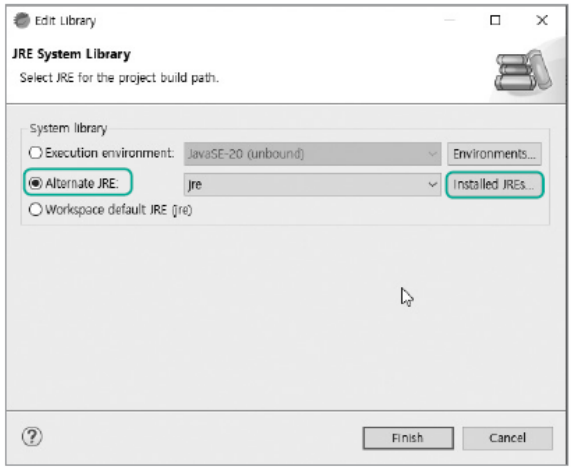
7. 다음 창에서 [Java Build Path → Libraries]을 차례로 클릭하고 오류 표시가 있는 'JRE System Library [JavaSE-20] (unbound)' 항목을 선택한 후 [Edit] 버튼을 클릭



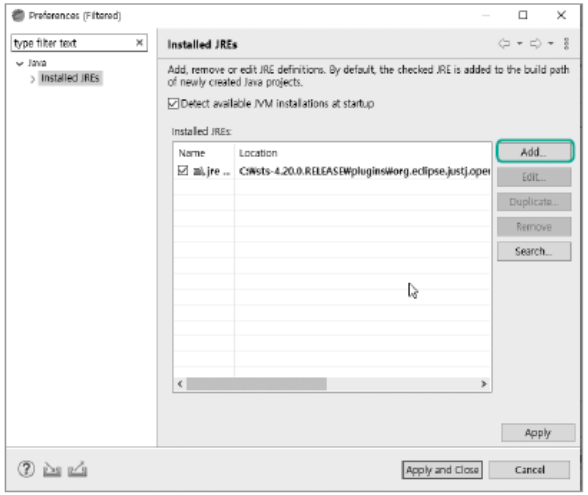
1-02 스프링 부트 개발 환경 준비하기

▪ 스프링 부트 프로젝트 만들기

8. 이어 등장한 창에서는 'Alternate JRE'를 선택한 후 [Installed JREs...] 버튼 클릭



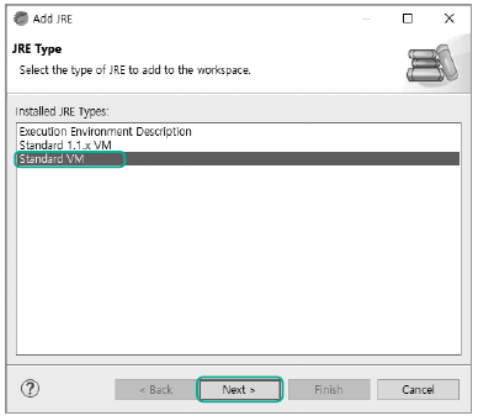
8. [Add] 버튼 클릭



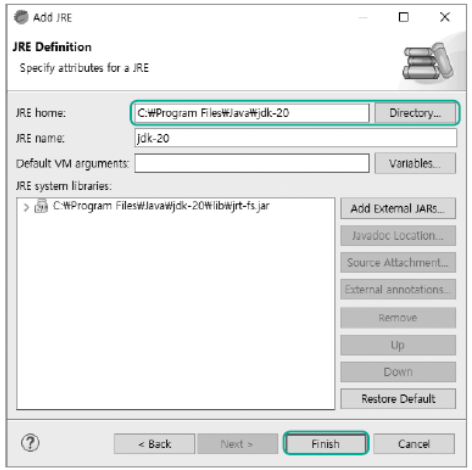
1-02 스프링 부트 개발 환경 준비하기

▪ 스프링 부트 프로젝트 만들기

10. 다음 화면에서 'Standard VM'을 선택하고 [Next] 버튼을 클릭



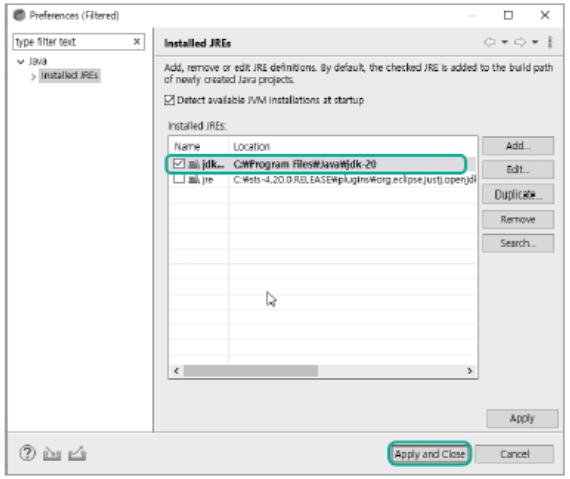
10. 다음 화면에서 [Directory...] 버튼을 클릭하여
앞서 JDK를 설치한 경로(여기서는
C:₩Program Files₩Java₩jdk-20)를 선택한 후
[Finish] 버튼을 클릭하여 jdk-20을 등록



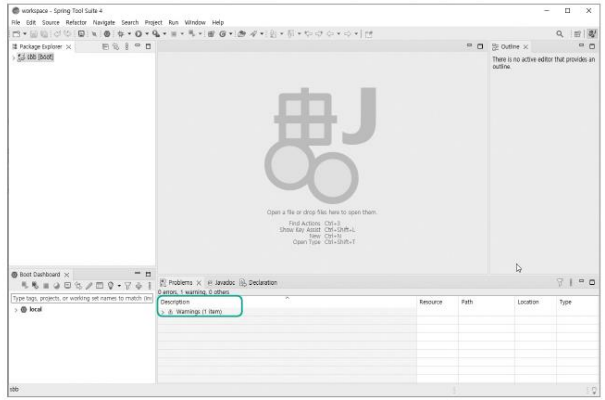
1-02 스프링 부트 개발 환경 준비하기

▪ 스프링 부트 프로젝트 만들기

12. 마지막으로 다음 화면에서 새로 등록한 'jdk-20' 항목을 선택한 후, [Apply and Close] 버튼을 클릭



12. 그러면 다음 화면과 같이 오류가 사라짐

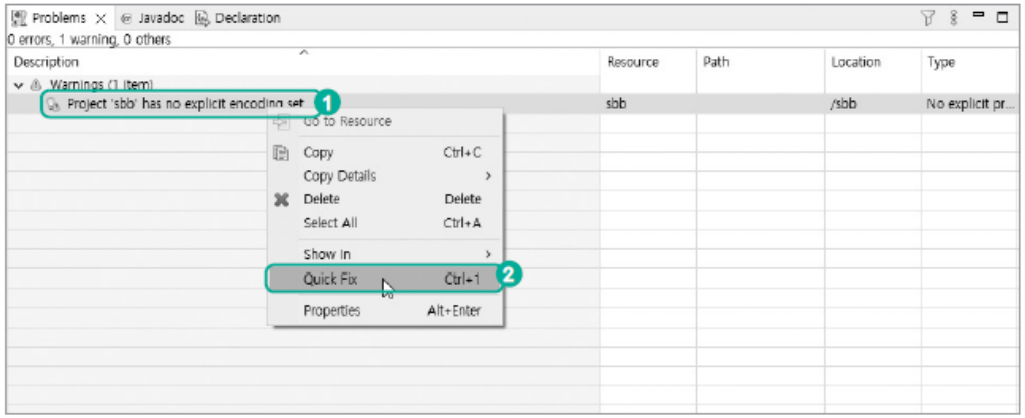


1-02 스프링 부트 개발 환경 준비하기

▪ 스프링 부트 프로젝트 만들기

13. 하지만 프로젝트의 인코딩 방식을 아직 설정하지 않았기 때문에 'Warnings (1 item)'이라는 경고 항목이 보임

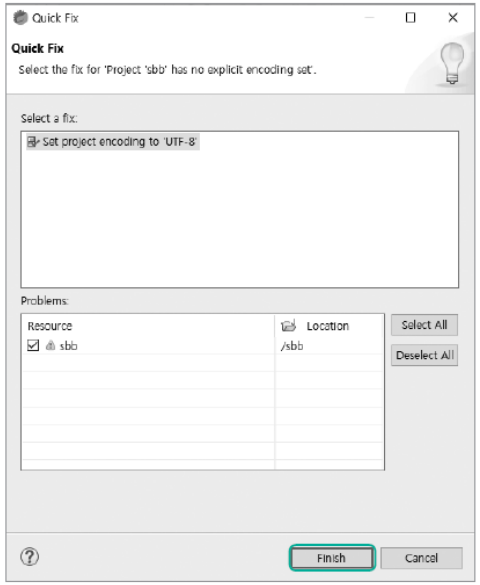
13. 'Warnings (1 item)'을 클릭하면 'Project 'sbb' has no explicit encoding set'이라는 문구가 보임. 이 문구를 선택한 후 마우스 오른쪽 버튼을 클릭하고 [Quick Fix]를 클릭해 보자



1-02 스프링 부트 개발 환경 준비하기

▪ 스프링 부트 프로젝트 만들기

15. Quick Fix 창에서 다음과 같은 설정을 확인한 후 [Finish] 버튼을 클릭하여, 프로젝트의 인코딩 방식을 UTF-8로 설정해 보자.



이제 프로젝트에 표시되던 경고 메시지가 사라진 것을 확인할 수 있음

1-03 스프링 부트 맛보기

스프링 부트를 사용하기 위한 준비를 마쳤으니
이번에는 브라우저 주소 창에 'http://localhost:8080/hello'라는 URL을 입력했을 때
브라우저 화면에 'Hello World'라는 문구를 출력하는 웹 프로그램을 작성해
스프링 부트를 살짝 맛보자.

이 프로그램이 동작하려면 컴퓨터(localhost)가
웹 서버가 되어 8080 포트에서 실행되어야 하고,

http://localhost:8080/hello를 통해
서버에 요청이 발생하면 'Hello World' 문장이 브라우저 화면에 출력되어야 함.

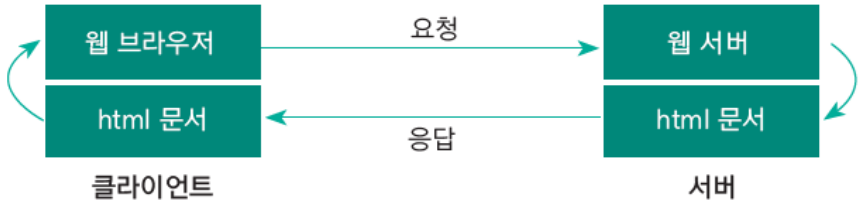
다소 어렵게 느껴지지만 이 프로그램을 스프링 부트로
얼마나 간편하게 만들 수 있는지 알아보자

1-03 스프링 부트 맛보기

▪ 웹 서비스는 어떻게 동작할까?

▪ 클라이언트와 서버 구조 이해하기

- 클라이언트는 자주 사용하는 브라우저(크롬, 사파리 등)를 말하고, 서버는 브라우저로 접속 가능한 원격 컴퓨터를 의미



- 크롬 브라우저에서 서버에 요청을 보낼 때는 서버의 주소(IP 주소) 또는 서버의 주소를 대체할 수 있는 도메인명을 알아야 함

1-03 스프링 부트 맛보기

▪ 웹 서비스는 어떻게 동작할까?

▪ IP 주소와 포트 이해하기

- 서버는 브라우저로 접속할 수 있는 웹 서비스뿐만 아니라 FTP, 이메일 서비스 등도 운용할 수 있음
- 하지만 보통 서비스별로 다른 IP 주소를 사용하지는 않음.
  왜냐하면 포트로 이러한 서비스들을 구분할 수 있기 때문
- 포트(port)는 네트워크 서비스를 구분하는 번호로, 하나의 서버 주소에서 포트를 사용하여 매우 많은 서비스를 운용할 수 있는데 대표적인 서비스의 종류는 다음과 같음

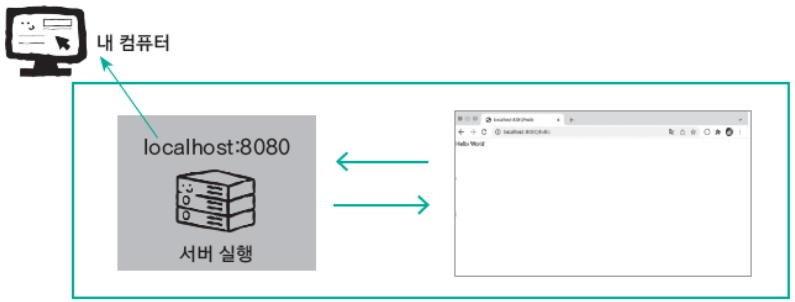
프로토콜	서비스 내용	포트
HTTP	웹 서비스	80
HTTPS	SSL을 적용한 웹 서비스	443
FTP	파일 전송 서비스	21
SSH, SFTP	보안이 강화된 TELNET(텔넷), FTP 서비스	22
TELNET	원격 서버 접속 서비스	23
SMTP	메일 전송 서비스	25

1-03 스프링 부트 맛보기

▪ 웹 서비스는 어떻게 동작할까?

▪ localhost:8080 이해하기

- 웹 개발에서 자주 등장하는 localhost:8080을 알아보자
- localhost:8080에서 먼저 localhost(로컬 호스트)라는 도메인명은 127.0.0.1이라는 IP 주소를 의미하며, 127.0.0.1 IP 주소는 내 컴퓨터를 의미함
- 사용하는 컴퓨터를 가리키는 말이며, 8080은 8080번 포트로 서비스를 운용한다는 의미
- 정리하자면, localhost:8080는 내 컴퓨터(localhost)에 8080번 포트로 실행된 서비스를 의미

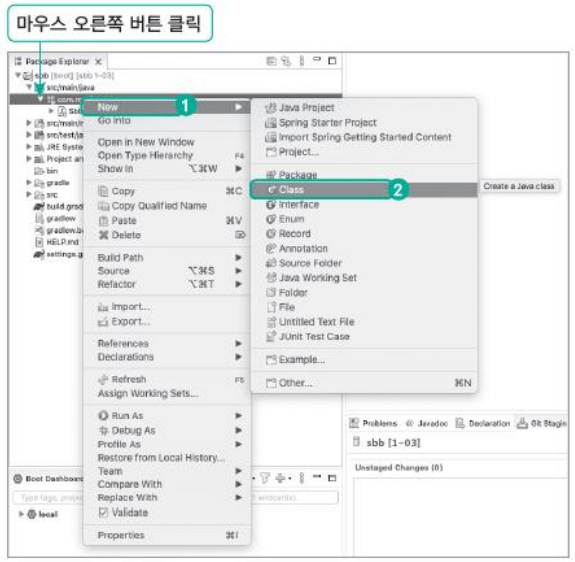


1-03 스프링 부트 맛보기

▪ 컨트롤러 만들기

- http://localhost:8080/hello와 같은 브라우저의 요청을 처리하려면 컨트롤러(controller)가 필요
- 컨트롤러는 서버에 전달된 클라이언트의 요청을 처리하는 자바 클래스
- 이러한 컨트롤러를 한번 만들어 보자

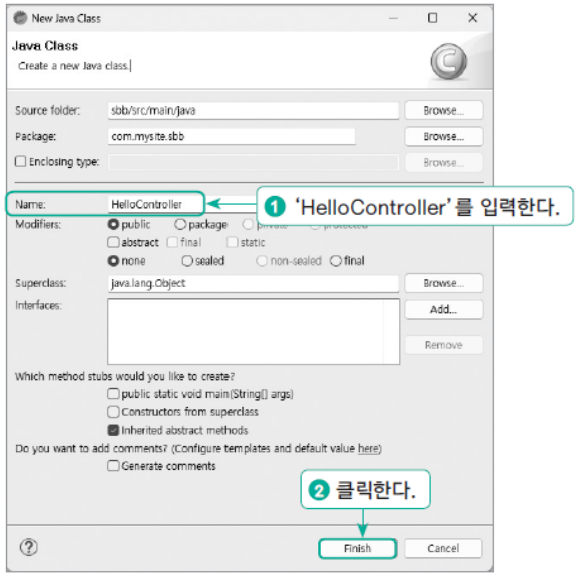
1. ‘com.mysite.sbb’ 패키지를 선택한 후,
   마우스 오른쪽 버튼을 누르고 [New → Class]를 클



1-03 스프링 부트 맛보기

▪ 컨트롤러 만들기

2. 다음과 같은 화면이 나타나면 Name 항목에 'HelloController'를 입력한 후 [Finish] 클릭



2. 화면에 HelloController.java 파일이 생성됨

• HelloController.java

```
package com.mysite.sbb;
public class HelloController {
}
```

1-03 스프링 부트 맛보기

▪ 컨트롤러 만들기

4. 하지만 지금 작성한 HelloController는 클래스 선언만 있고 내용은 없는 '껍데기' 클래스이므로 컨트롤러의 기능을 갖추려면 다음과 같이 수정해야 함

• HelloController.java

```
package com.mysite.sbb;

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

@Controller
public class HelloController {
    @GetMapping("/hello")
    @ResponseBody
    public String hello() {
        return "Hello World";
    }
}
```

HelloController 클래스가 컨트롤러의 기능을 수행한다는 것을 알려 준다.

클라이언트의 요청으로 hello 메서드가 실행됨을 알려 준다.

hello 메서드의 출력값 그대로 리턴할 것임을 알려 준다.

스프링 부트의 import 문은 다른 클래스, 패키지, 라이브러리 등을 사용할 때 관련 요소를 가져오는 역할을 해. 따로 설명하지 않더라도 이러한 역할을 한다는 점을 기억해 둬!

1-03 스프링 부트 맛보기

▪ 컨트롤러 만들기

4. 클래스명 위에 적용된 @Controller 애너테이션은 HelloController 클래스가 컨트롤러의 기능을 수행한다는 의미.
애너테이션이 있어야 스프링 부트 프레임워크가 컨트롤러로 인식함.

   hello 메서드에 적용된 @GetMapping("/hello") 애너테이션은 http://localhost:8080/hello URL 요청이 발생하면 hello 메서드가 실행됨을 의미.
즉, /hello URL과 hello 메서드를 매핑하는 역할을 함.

   이때 URL명과 메서드명이 동일할 필요는 없음.
즉 /hello URL일 때 메서드명을 hello가 아닌 hello2와 같이 써도 상관없음

1-03 스프링 부트 맛보기

▪ 컨트롤러 만들기

4. 또한 Get 방식의 URL 요청을 위해 GetMapping을 사용하고
Post 방식의 URL 요청을 위해서는 PostMapping을 사용함.

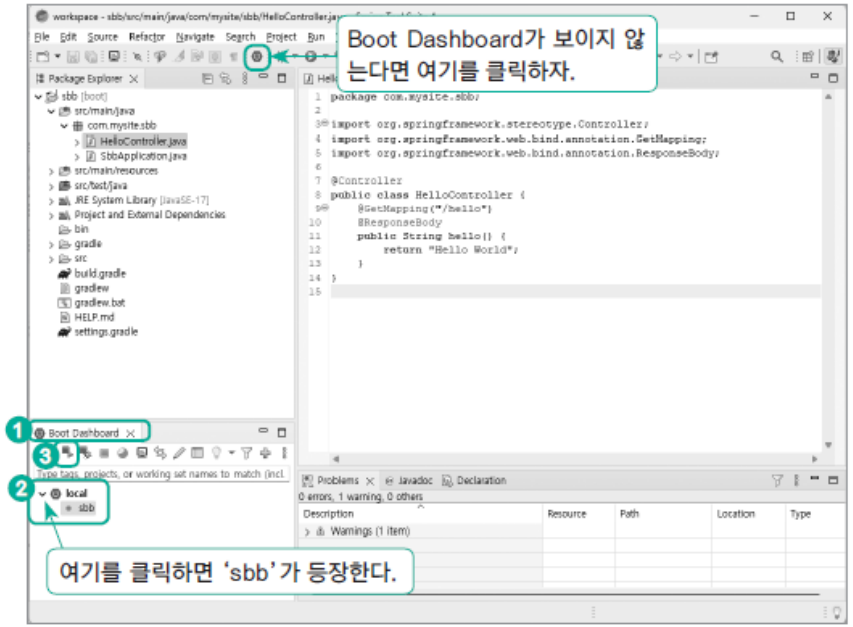
   그리고 @ResponseBody 애너테이션은 hello 메서드의 출력 결과가
문자열 그 자체임을 나타냄.

   hello 메서드는 'Hello World' 문자열을 리턴하므로
결과로 'Hello World' 문자열이 출력됨

1-03 스프링 부트 맛보기

▪ 로컬 서버 실행하기

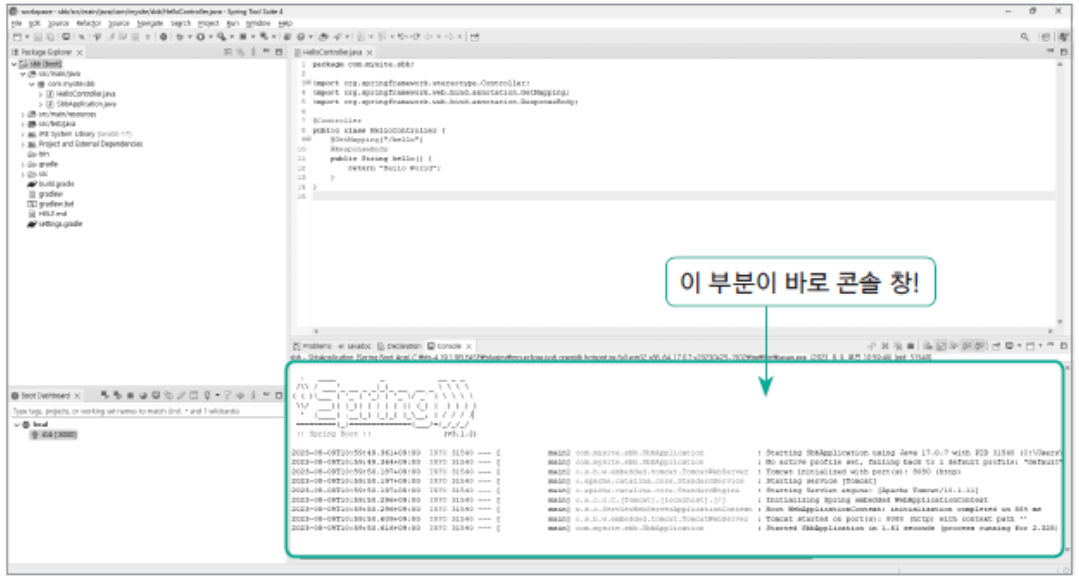
1. 로컬 서버는 다음과 같은 순서로 실행함



1-03 스프링 부트 맛보기

▪ 로컬 서버 실행하기

2. 순서대로 진행했다면 로컬 서버가 실행되고 STS 콘솔 창에 로컬 서버가 8080 포트로 실행되었다는 메시지가 출력될 것

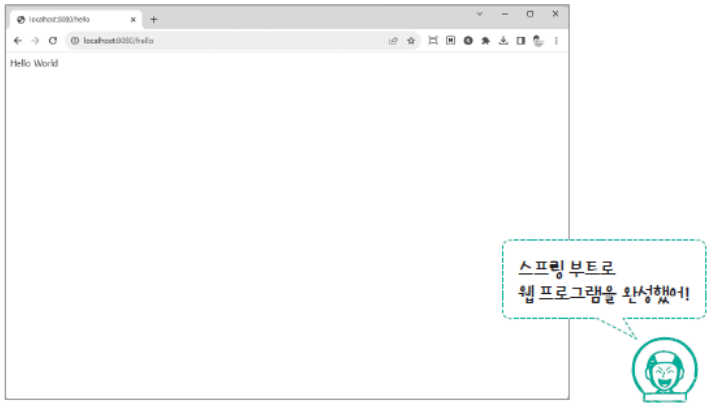


1-03 스프링 부트 맛보기

▪ 브라우저로 확인하기

로컬 서버를 실행하였으니 이번에는 HelloController가
브라우저의 요청을 처리하는지 확인해 보자.

구글 크롬(Google Chrome)과 같은 브라우저를 실행하고
주소 창에 http://localhost: 8080/hello를 입력해 보자



1-03 스프링 부트 맛보기

▪ 브라우저로 확인하기

이와 같이 /hello URL이 요청되면
컨트롤러인 HelloController의 /hello URL과 매핑된 hello 메서드가 호출되고

'Hello World' 문자열이 브라우저에 출력되는 것을 확인할 수 있음

이렇게 스프링 부트로 웹 프로그램을 완성해 봄

1-04 스프링 부트 도구 설치하기

웹 프로그램 개발을 도와주는 스프링 부트의 도구(라이브러리)에 대해 알아보자.

이 도구들은 앞으로 스프링 부트를 통해 웹 프로그램을 개발할 때
다소 귀찮을 수 있는 작업들을 좀 더 간편하고 빠르게 처리할 수 있도록 도와 줌

- **Spring Boot Devtools 설치하기**
  - Spring Boot Devtools 라이브러리를 STS에 추가해 보자.
    Sping Boot Devtools를 추가하면 서버를 매번 재시작하지 않고도
    수정한 내용이 반영됨

1-04 스프링 부트 도구 설치하기

▪ Spring Boot Devtools 설치하기

1. 앞에서 작성한 HelloController.java를 다음과 같이 수정해 보자

• HelloController.java

```
package com.mysite.sbb;

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

@Controller
public class HelloController {
    @GetMapping("/hello")
    @ResponseBody
    public String hello() {
        return "Hello SBB ";
    }
}
```

1-04 스프링 부트 도구 설치하기

▪ Spring Boot Devtools 설치하기

1. 출력하는 문자열을 'Hello World'에서 'Hello SBB'로 변경함.

   하지만 이렇게 수정하고 http://localhost:8080/hello URL을 호출하면
   여전히 'Hello World'가 출력됨.

   왜냐하면 이와 같이 프로그램이 변경되더라도 별도의 과정 없이는
   로컬 서버가 변경된 클래스를 즉시 반영하지 않기 때문.

   그래서 프로그램을 간단히 수정하더라도 변경된 사항을 확인하기 위해
   매번 서버를 재시작해야 하므로 개발 과정이 꽤 번거로움

1-04 스프링 부트 도구 설치하기

▪ Spring Boot Devtools 설치하기

2. 이러한 문제를 해결하려면 Spring Boot Devtools를 설치해야 함.

   Spring Boot Devtools를 설치하면 서버를 재시작하지 않아도
   클래스를 변경할 때 서버가 자동으로 재가동됨.

   Spring Boot Devtools를 사용하려면 Spring Boot Devtools를
   그레이들(Gradle)로 설치해야 함.

   다음과 같이 STS 화면 왼쪽에서 build.gradle 파일을 찾아 수정하자

1-04 스프링 부트 도구 설치하기

▪ Spring Boot Devtools 설치하기

2.

• build.gradle

```
plugins {
    id 'java'
    id 'org.springframework.boot' version '3.1.3'
    id 'io.spring.dependency-management' version '1.1.3'
}

group = 'com.mysite'
version = '0.0.1-SNAPSHOT'
sourceCompatibility = '20'

repositories {
    mavenCentral()
}

dependencies {
    implementation 'org.springframework.boot:spring-boot-starter-web'
    testImplementation 'org.springframework.boot:spring-boot-starter-test'
    developmentOnly 'org.springframework.boot:spring-boot-devtools'
}

tasks.named('test') {
    useJUnitPlatform()
}
```

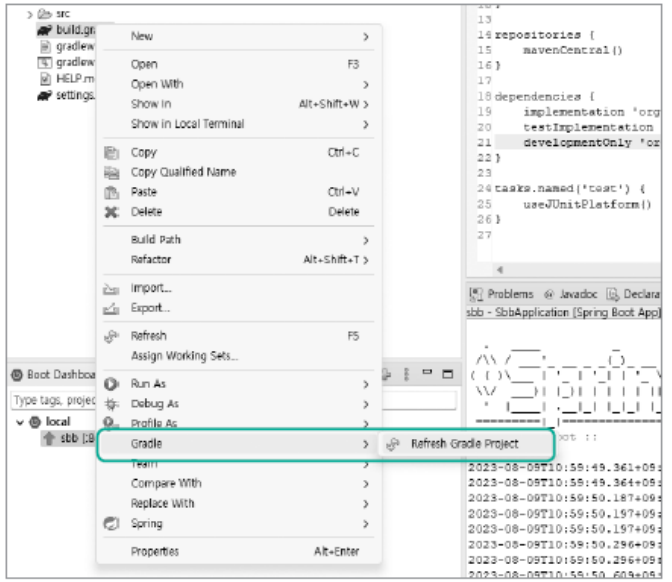
Spring Boot Devtools 추가

해당 라이브러리는 개발 환경에만 적용된다는 의미로, 운영 환경에 배포되는 jar, war 파일에는 이 라이브러리가 포함되지 않는다.

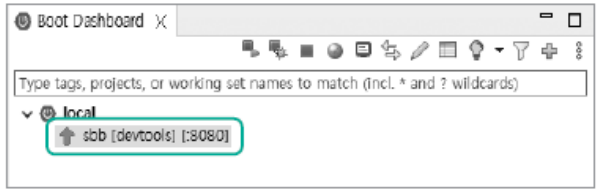
1-04 스프링 부트 도구 설치하기

▪ Spring Boot Devtools 설치하기

3. build.gradle 파일에 작성한 내용을 적용하려면
build.gradle 파일을 선택한 후 마우스 오른쪽 버튼을 눌
[Gradle → Refresh Gradle Project]를 클릭하여
필요한 라이브러리를 설치해야 함



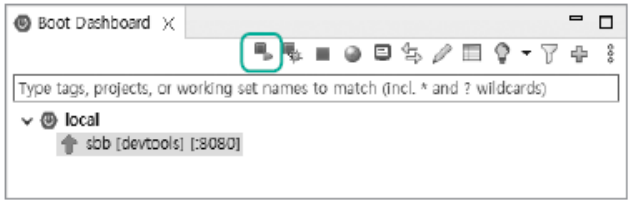
3. 설치가 완료되면 Boot Dashboard의 서버명이
sbb에서 sbb [devtools]로 바뀜



1-04 스프링 부트 도구 설치하기

▪ Spring Boot Devtools 설치하기

5. 서버를 재시작하자



5. 이제 Spring Boot Devtools가 적용되었으니
브라우저에서 다시 http://localhost:8080/hello를 호출해 보자.
서버를 재시작했으므로 'Hello SBB'가 출력됨.

서버 재시작 없이도 변경 사항이 적용되는지 확인하기 위해
출력할 문자열을 다음과 같이 다시 변경해 보자.

1-04 스프링 부트 도구 설치하기

▪ Spring Boot Devtools 설치하기

6.

• HelloController.java

```
package com.mysite.sbb;

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

@Controller

public class HelloController {
    @GetMapping("/hello")
    @ResponseBody
    public String hello() {
        return "Hello Spring Boot Board";
    }
}
```

1-04 스프링 부트 도구 설치하기

▪ Spring Boot Devtools 설치하기

6. 수정 사항이 잘 반영되는지 http://localhost:8080/hello을 호출해 확인해 보자.
서버를 재시작하지 않아도 'Hello Spring Boot Board'가 출력됨.
문자열이 잘 출력될 것임

1-04 스프링 부트 도구 설치하기

▪ 롬복 설치하기

롬복(Lombok) 라이브러리는 소스 코드를 작성할 때
자바 클래스에 애너테이션을 사용하여 자주 쓰는
Getter 메서드, Setter 메서드, 생성자 등을 자동으로 만들어 주는 도구임.

이제 SBB 프로그램을 만들면서 게시물과 관련된 데이터를 처리하기 위해
엔티티 클래스나 DTO 클래스 등을 사용해야 하는데
그러기 위해서는 먼저 이 클래스들의 속성값을 읽고 저장하는
Getter, Setter 메서드를 만들어야 함.

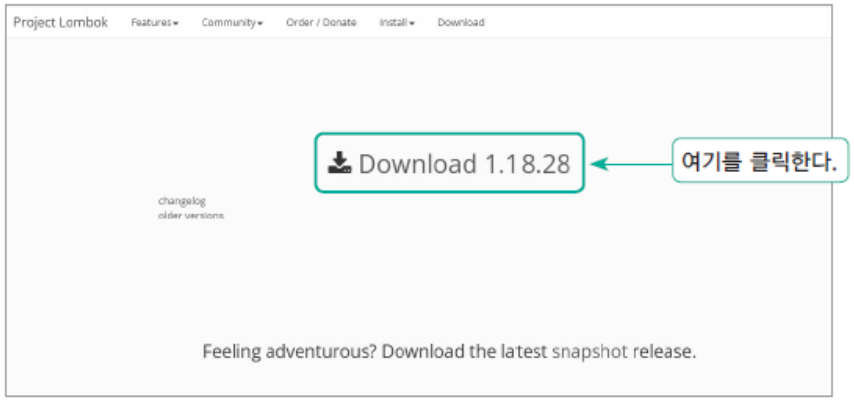
물론 Getter, Setter 메서드를 직접 작성해도 되지만
롬복을 사용하면 좀 더 짧고 깔끔한 소스 코드를 만들 수 있음

1-04 스프링 부트 도구 설치하기

▪ 롬복 설치하기

1. 롬복을 사용하려면 먼저 플러그인을 설치해야 함.
   다음 URL에서 롬복 플러그인을 내려받자

```
https://projectlombok.org/download
```



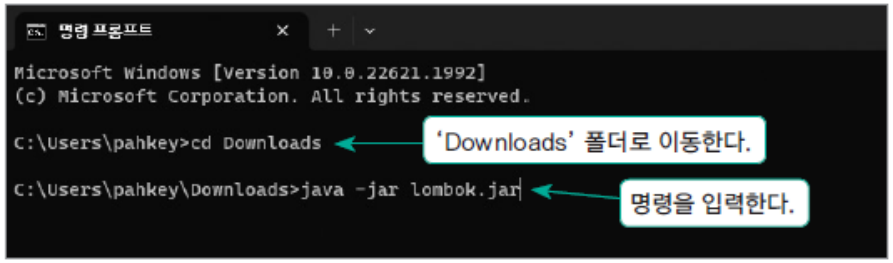
1-04 스프링 부트 도구 설치하기

▪ 롬복 설치하기

2. 내려받은 lombok.jar 파일을 명령 프롬프트 창에서 다음과 같이 설치하자. lombok.jar 파일이 있는 위치로 이동한 후, 다음 명령을 실행해야 함.

   여기서는 'Downloads' 폴더에 lombok.jar 파일이 있으므로, 다음과 같이 입력함

```
java -jar lombok.jar
```



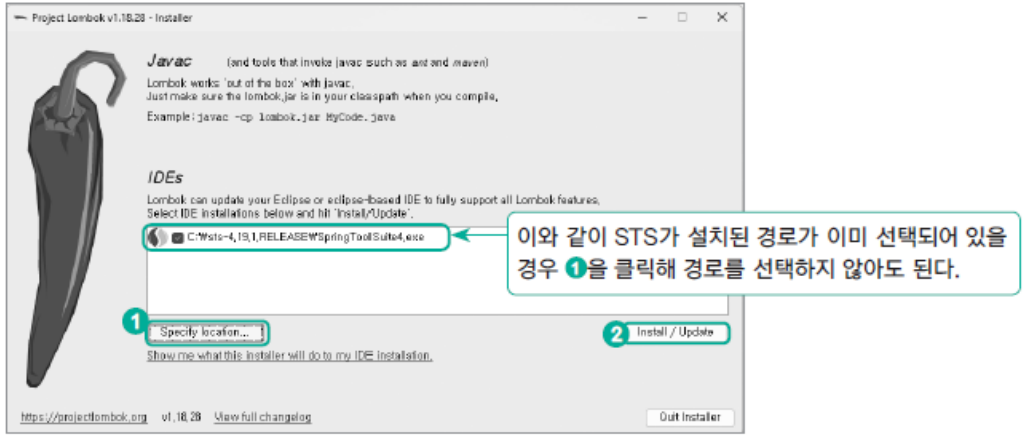
1-04 스프링 부트 도구 설치하기

▪ 롬복 설치하기

3. 다음과 같은 설치 창이 등장했다면 [Specify location]을 클릭해 롬복 플러그인을 사용할 IDE인 STS가 설치된 경로를 선택함

   (이미 올바른 경로가 선택되어 있다면 이 과정은 생략해도 됨).

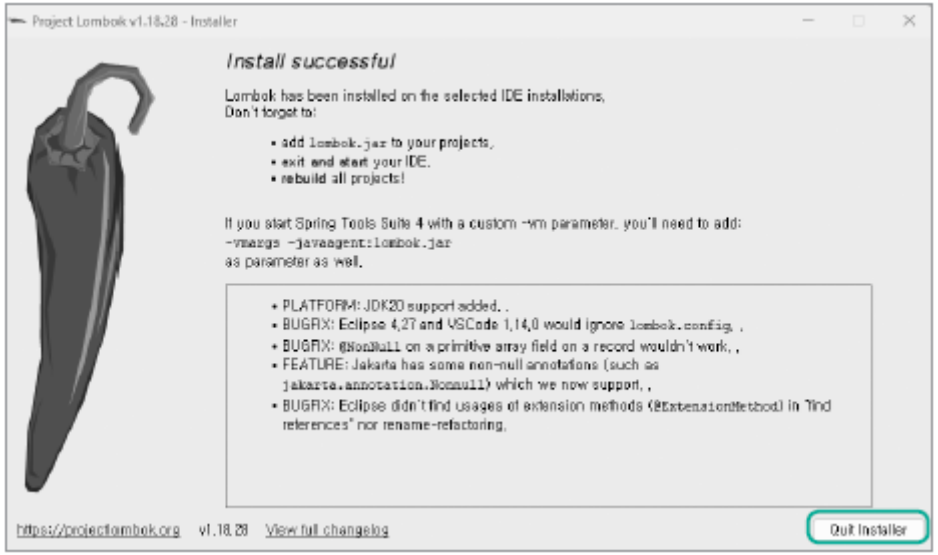
   그다음 [Install / Update]를 클릭해 롬복 플러그인을 설치함



1-04 스프링 부트 도구 설치하기

▪ 롬복 설치하기

4. 설치가 성공적으로 완료되었음을 알리는 문구가 뜬다면 [Quit Installer]를 클릭해 설치 프로그램을 종료함



1-04 스프링 부트 도구 설치하기

▪ 롬복 설치하기

5. 만약 STS가 활성화되어 있다면 종료하고 다시 시작한 후 build.gradle 파일을 다음과 같이 수정하자

• build.gradle

```
(... 생략 ...)

dependencies {
    implementation 'org.springframework.boot:spring-boot-starter-web'
    testImplementation 'org.springframework.boot:spring-boot-starter-test'
    developmentOnly 'org.springframework.boot:spring-boot-devtools'
    compileOnly 'org.projectlombok:lombok'
    annotationProcessor 'org.projectlombok:lombok'
}
(... 생략 ...)
```

이 부분을 추가로 작성하여 롬복 라이브러리를 설치한다.

컴파일 단계에서만 해당 라이브러리를 사용함을 의미한다.

컴파일 단계에서 애너테이션을 분석하고 처리할 때 사용함을 의미한다.

1-04 스프링 부트 도구 설치하기

▪ 롬복 설치하기

▪ 롬복으로 Getter, Setter 메서드 만들기

- [com.mysite.sbb]에 마우스 오른쪽 버튼을 누르고 [New → Class]를 클릭해 Hello Lombok 클래스를 만들어 보자
- 그리고 다음과 같이 소스 코드를 작성하여 롬복이 정상적으로 동작하는지 확인해 보자
- 다음 코드를 작성했을 때 오류가 없어야 함

• HelloLombok.java

```
package com.mysite.sbb;

import lombok.Getter;
import lombok.Setter;

@Getter
@Setter
public class HelloLombok {
    private String hello;
    private int lombok;
```

클래스의 속성을 추가한다.

```
    public static void main(String[] args) {
        HelloLombok helloLombok = new HelloLombok();
        helloLombok.setHello("헬로");
        helloLombok.setLombok(5);

        System.out.println(helloLombok.getHello());
        System.out.println(helloLombok.getLombok());
    }
}
```

1-04 스프링 부트 도구 설치하기

▪ 롬복 설치하기

▪ 롬복으로 Getter, Setter 메서드 만들기

- HelloLombok 클래스에 'hello', 'lombok' 이렇게 2개의 속성을 추가한 후
  클래스명 바로 위에 @Getter, @Setter라는 애너테이션을 적용했더니
  Getter와 Setter 메서드를 따로 작성하지 않아도 setHello, setLombok, getHello, getLombok 등의 메서드를 사용할 수 있게 되었음

- 즉, 롬복을 활용하면 속성에 대한 Setter, Getter 메서드를 별도로 작성하지 않아도 됨

- 다음과 같이 Getter, Setter 메서드를 직접 작성한 코드와 비교하며 롬복의 편리함을 느껴 보자.
  16줄로 완성했던 코드가 24줄로 표현됨

1-04 스프링 부트 도구 설치하기

Getter, Setter를 직접 작성한 코드

```
package com.mysite.sbb;

public class HelloLombok {
    private String hello;
    private int lombok;

    public void setHello(String hello) {
        this.hello = hello;
    }

    public void setLombok(int lombok) {
        this.lombok = lombok
    }

    public String getHello() {
        return this.hello;
    }

    public int getLombok() {
        return this.lombok;
    }

    public static void main(String[] args) {
        HelloLombok helloLombok = new HelloLombok();
        helloLombok.setHello("헬로");
        helloLombok.setLombok(5);

        System.out.println(helloLombok.getHello());
        System.out.println(helloLombok.getLombok());
    }
}
```

1-04 스프링 부트 도구 설치하기

▪ 롬복 설치하기

▪ 롬복으로 생성자 만들기

- 이번에는 HelloLombok 클래스를 다음과 같이 수정해 보자

• HelloLombok.java

```
package com.mysite.sbb;

import lombok.Getter;
import lombok.RequiredArgsConstructor;

@RequiredArgsConstructor
@Getter
public class HelloLombok {
    private final String hello;
    private final int lombok;

    public static void main(String[] args) {
        HelloLombok helloLombok = new HelloLombok("헬로", 5);
        System.out.println(helloLombok.getHello());
        System.out.println(helloLombok.getLombok());
    }
}
```

1-04 스프링 부트 도구 설치하기

▪ 롬복 설치하기

▪ 롬복으로 생성자 만들기

- 이와 같이 hello, lombok 속성에 final을 추가하고 @RequiredArgsConstructor 애너테이션을 적용하면 해당 속성(hello와 lombok)을 필요로 하는 생성자가 롬복에 의해 자동으로 생성됨
- 즉, 롬복을 활용하면 필요한 생성자를 자동으로 만들어 줌
- 다음과 같이 생성자를 직접 작성한 코드와 비교하며 롬복의 편리함을 느껴 보자

1-04 스프링 부트 도구 설치하기

▪ 롬복 설치하기

▪ 롬복으로 생성자 만들기

생성자를 직접 작성한 코드

```
package com.mysite.sbb;
import lombok.Getter;

@Getter
public class HelloLombok {
    private final String hello;
    private final int lombok;

    public HelloLombok(String hello, int lombok) {
        this.hello = hello;
        this.lombok = lombok;
    }

    public static void main(String[] args) {
        HelloLombok helloLombok = new HelloLombok("헬로", 5);
        System.out.println(helloLombok.getHello());
        System.out.println(helloLombok.getLombok());
    }
}
```

롬복을 사용하지 않는다면 이와 같이 생성자를 직접 작성해야 한다.

되 / 새 / 김 / 문 / 제

P.49~50

Q1. 간단한 웹 프로그램 만들기

Q2. 롬복으로 메서드 만들기