



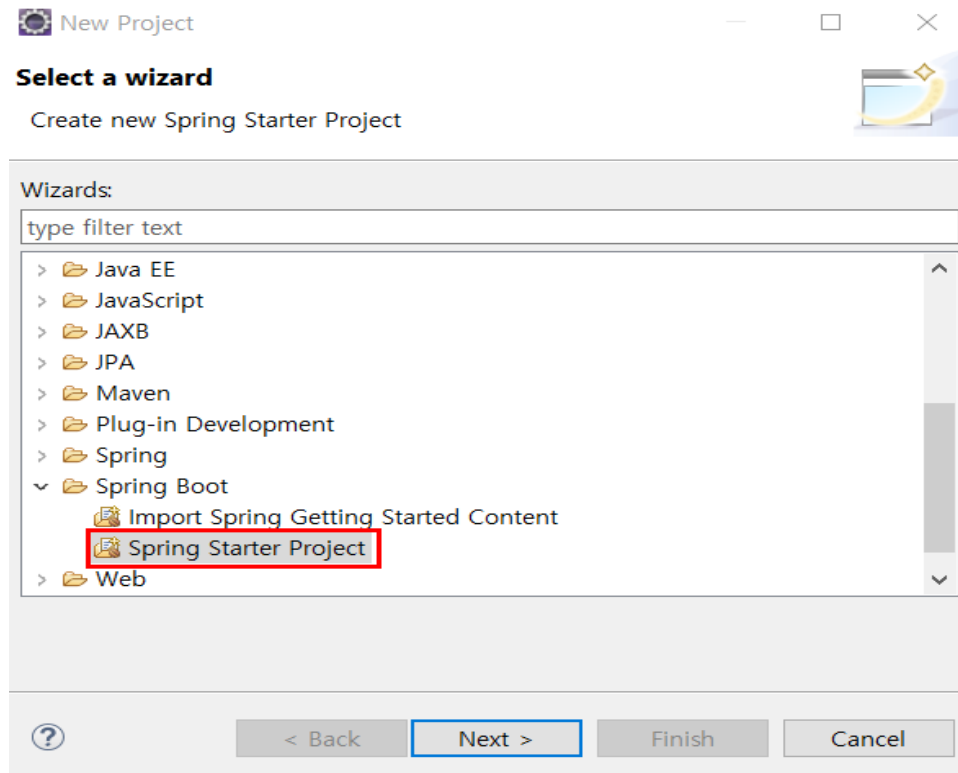
Lombok

안화수

spring boot 프로젝트

❖ boot02 project 생성

[File] – New - Project



spring boot 프로젝트

❖ boot02 project 생성

➤ Name : **boot02**

➤ Type : **Maven**, Gradle

➤ Packaging : **War**, Jar

New Spring Starter Project

Service URL:

Name:

☒ Use default location

Location:

Type: Packaging:

Java Version: Language:

Group:

Artifact:

Version:

Description:

Package:

Working sets

☐ Add project to working sets

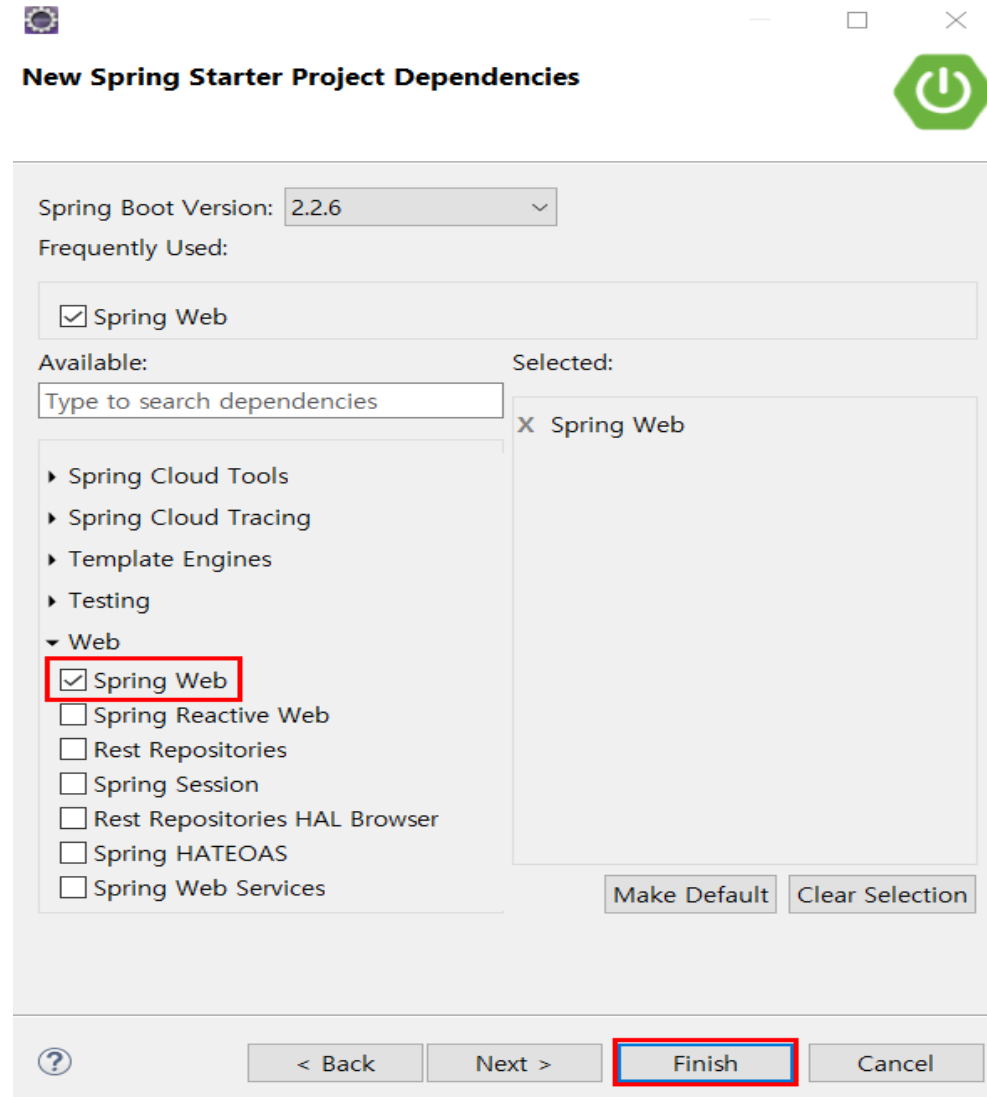
Working sets:

spring boot 프로젝트

❖ boot02 project 생성

➤ Web

Spring Web 체크



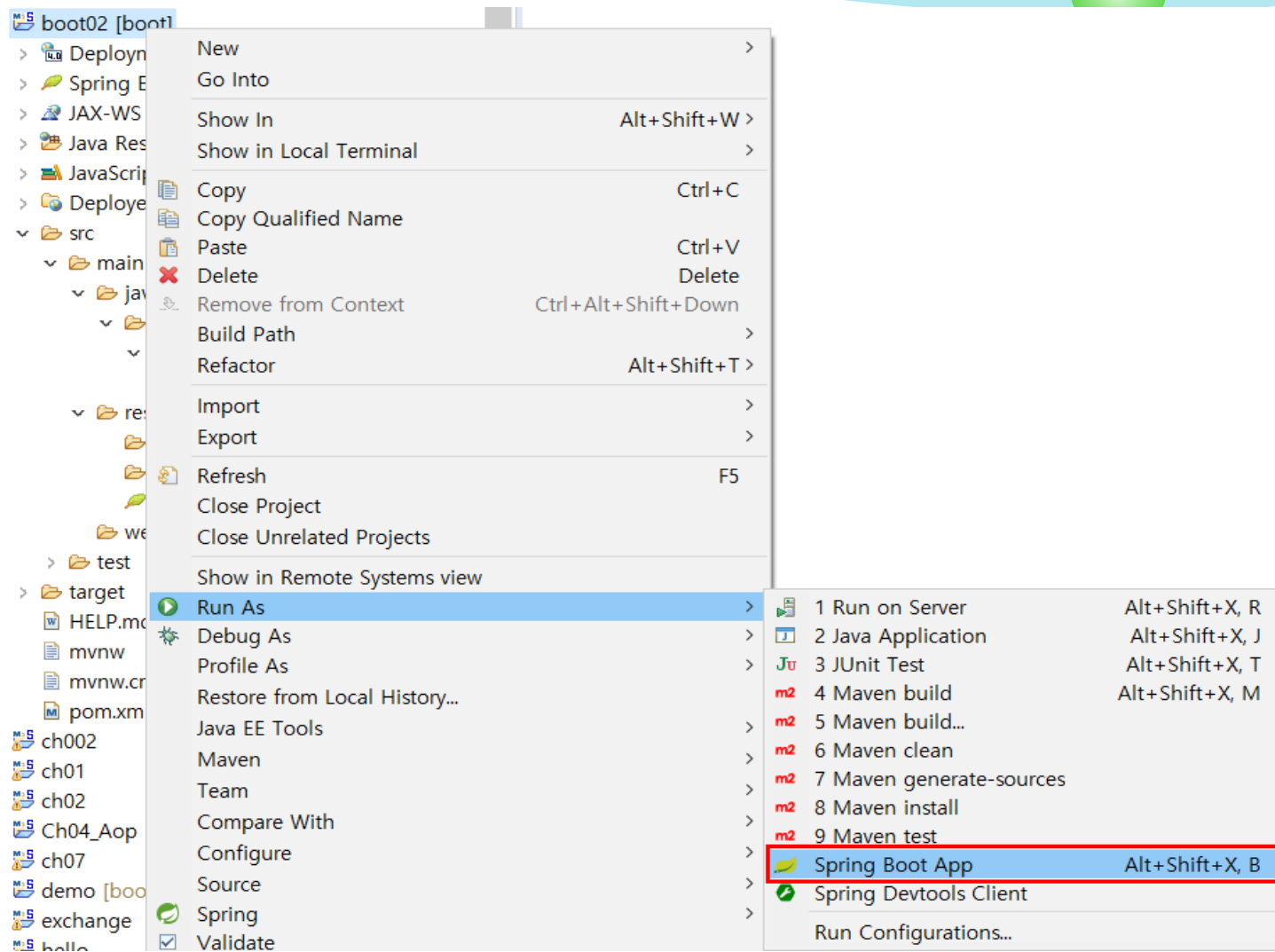
spring boot 서버 실행

❖ boot02 project 실행

boot02 - 오른마우스 클릭

- Run As

- Spring Boot App



spring boot 서버 실행

boot02 project 실행 결과

다음과 같이 출력되면 성공~!!

```

  .
 / \ / \
( ( ) \
 \ \   )
  '   |
=====|_=====
:: Spring Boot ::      (v2.2.9.RELEASE)

2020-08-25 15:26:50.480 INFO 10868 --- [
2020-08-25 15:26:50.483 INFO 10868 --- [
2020-08-25 15:26:51.444 INFO 10868 --- [
2020-08-25 15:26:51.455 INFO 10868 --- [
2020-08-25 15:26:51.456 INFO 10868 --- [
2020-08-25 15:26:51.723 INFO 10868 --- [
2020-08-25 15:26:51.727 INFO 10868 --- [
2020-08-25 15:26:51.727 INFO 10868 --- [
2020-08-25 15:26:51.889 INFO 10868 --- [
2020-08-25 15:26:52.051 INFO 10868 --- [
2020-08-25 15:26:52.054 INFO 10868 --- [

main] com.example.demo.Boot01Application
main] com.example.demo.Boot01Application
main] o.s.b.w.embedded.tomcat.TomcatWebServer
main] org.apache.catalina.core.StandardService
main] org.apache.catalina.core.StandardEngine
main] org.apache.jasper.servlet.TldScanner
main] o.a.c.c.C.[Tomcat].[localhost].[/]
main] w.s.c.ServletWebServerApplicationContext
main] o.s.s.concurrent.ThreadPoolTaskExecutor
main] o.s.b.w.embedded.tomcat.TomcatWebServer
main] com.example.demo.Boot01Application

```

의존 라이브러리 추가

❖ pom.xml

pom.xml 파일에 필요한 의존 라이브러리 추가한다.

```
<dependencies>
```

```
    <!-- 내장 Tomcat 실행시 jsp 파일을 사용하기 위한 의존 라이브러리 -->
```

```
    <dependency>
```

```
        <groupId>org.apache.tomcat.embed</groupId>
```

```
        <artifactId>tomcat-embed-jasper</artifactId>
```

```
        <scope>provided</scope>
```

```
    </dependency>
```

```
    <!-- jstl -->
```

```
    <dependency>
```

```
        <groupId>javax.servlet</groupId>
```

```
        <artifactId>jstl</artifactId>
```

```
    </dependency>
```

```
</dependencies>
```

환경 설정파일 수정

❖ 환경 설정 파일 수정

main / resources / **application.properties**

port

server.port=80

prefix and suffix

spring.mvc.view.prefix=/WEB-INF/views/

spring.mvc.view.suffix=.jsp

❖ webapp / **WEB-INF** / **views** 폴더 생성

DTO 생성

❖ DTO 생성

main / java / com / example / demo / model – Member.java

```
package com.example.demo.model;
```

```
public class Member {  
    private String id;  
    private String passwd;  
  
    public String getId() {  
        return id;  
    }  
    public void setId(String id) {  
        this.id = id;  
    }  
    public String getPasswd() {  
        return passwd;  
    }  
    public void setPasswd(String passwd) {  
        this.passwd = passwd;  
    }  
}
```

Controller 생성

❖ Controller 생성

/main/java/com/example/demo/controller – SampleController.java 생성

```
package com.example.demo.controller;

import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.RequestMapping;
import com.example.demo.model.Member;

@Controller
public class SampleController {
    @RequestMapping("/")
    public String main() {
        return "main";
    }
    @RequestMapping("send")
    public String send(Member member, Model model) {
        model.addAttribute("member", member);
        return "result";
    }
}
```

View 생성

❖ View 생성

webapp / WEB-INF / views - main.jsp 생성

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>로그인</title>
</head>
<body>
<form method="post" action="send">
    ID   : <input type="text" name="id"> <br>
    pass : <input type="text" name="passwd"> <br>
        <input type="submit" value="가입">
</form>
</body>
</html>
```

View 생성

❖ View 생성

webapp / WEB-INF / views - result.jsp 생성

```
<%@ 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>

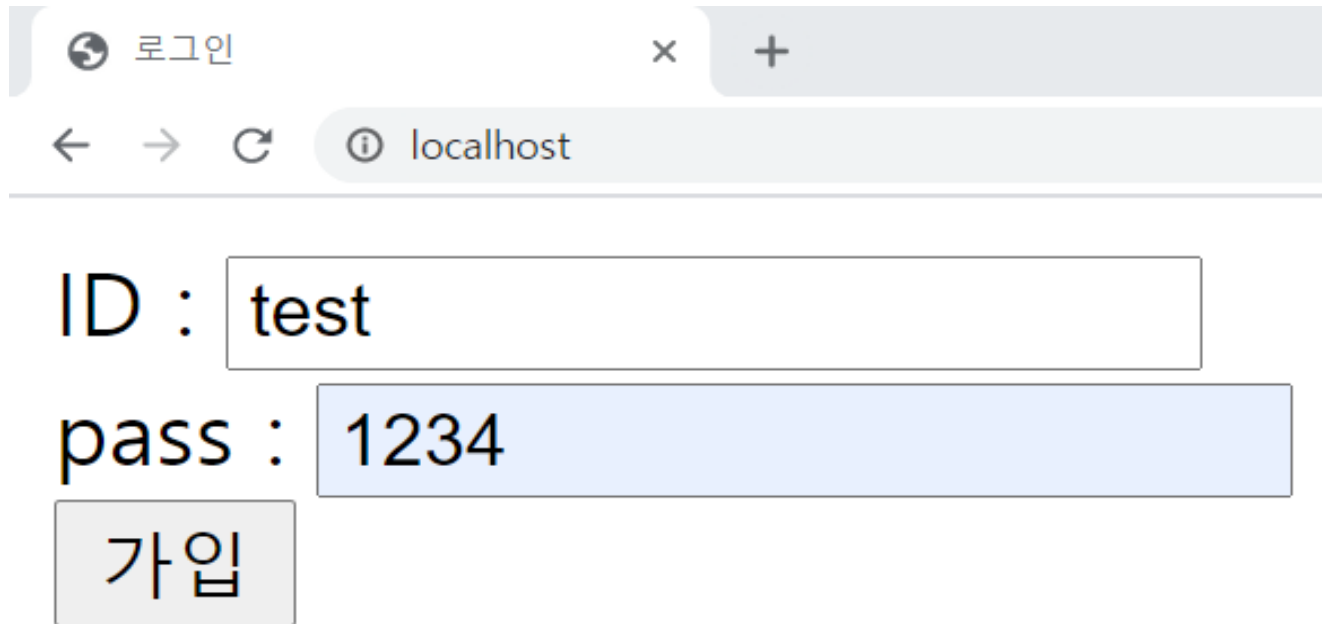
ID : ${member.id} <br>
pass : ${member.passwd} <br>

</body>
</html>
```

boot02 프로젝트 접속

❖ boot02 프로젝트 접속

boot02 project를 중지하고, 재시작 한 후에 웹브라우저에 `http://localhost` 아래와 같이 요청한다.



The screenshot shows a web browser window with a single tab titled '로그인' (Login). The address bar displays 'localhost'. The login form contains two input fields: 'ID' with the value 'test' and 'pass' with the value '1234'. Below these fields is a button labeled '가입' (Join).

로그인

localhost

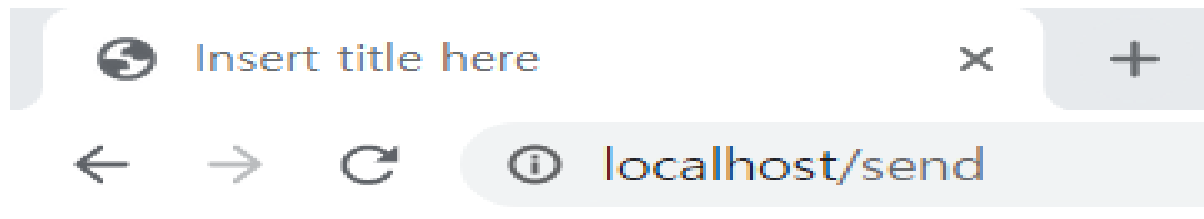
ID : test

pass : 1234

가입

boot02 프로젝트 처리 결과

❖ boot02 프로젝트 처리 결과



ID : test

pass : 1234



Lombok

Lombok

❖ Lombok

lombok 라이브러리는 java 라이브러리중 하나로, 멤버 변수에 대한 getter / setter method, toString(), Equals() 등과 생성자 코드를 불필요하게 반복적으로 만들었지만, lombok 라이브러리를 사용하면 Annotation(어노테이션) 기반으로 자동으로 메소드를 생성해 주는 라이브러리 이다.

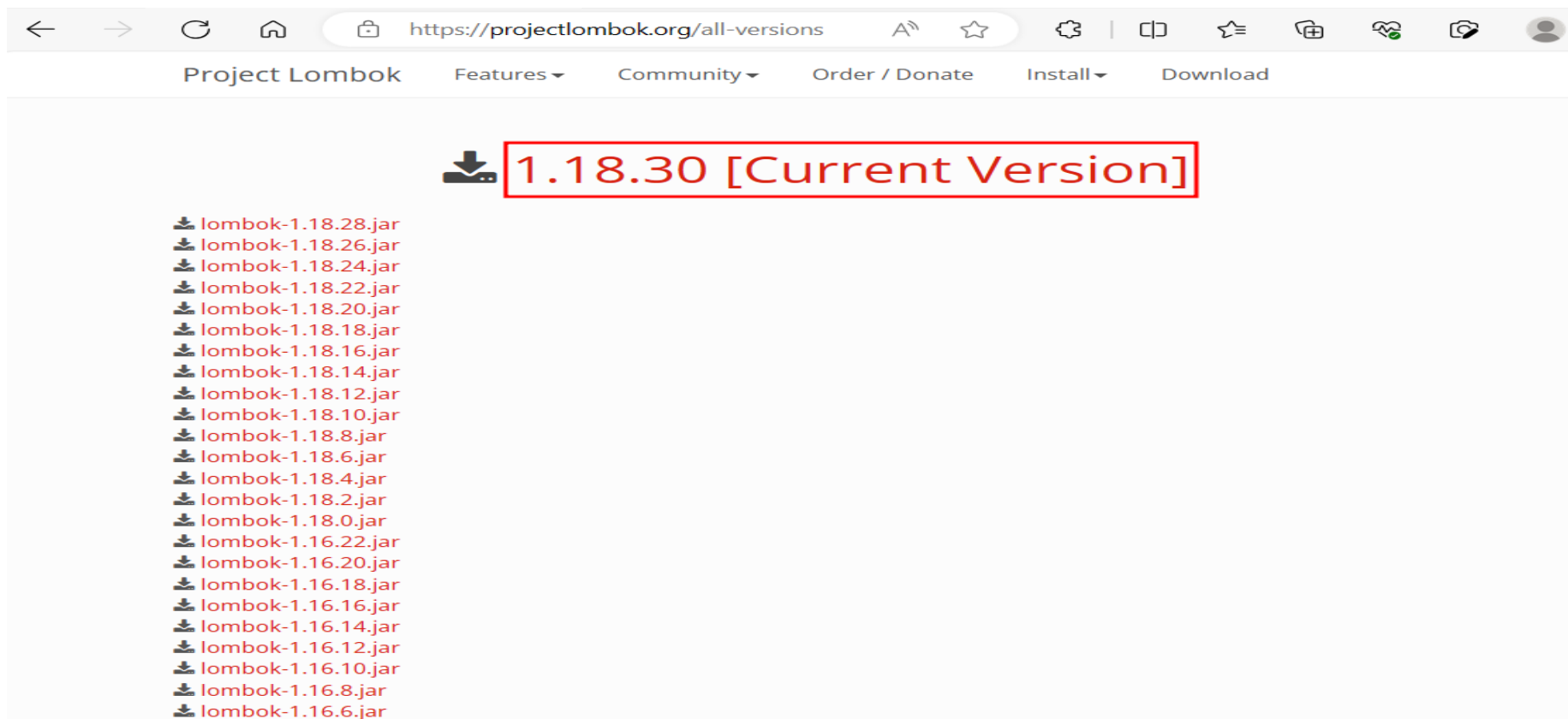
lombok 라이브러리를 사용하면 DTO(Data Transfer Object) 같은 클래스에서 getter 와 setter 메소드를 자동으로 생성해 준다.

boot02 프로젝트에 lombok 라이브러리를 적용 시켜보자.

Lombok 환경 구축

1. Eclipse 나 STS 에 Lombok 라이브러리 설정

- 1) Eclipse 나 STS에서 Lombok을 이용하기 위해서는 Lombok 사이트(<http://projectlombok.org/all-versions>)에서 lombok-1.18.30.jar 버전 파일을 다운로드 받는다.



Lombok 환경 구축

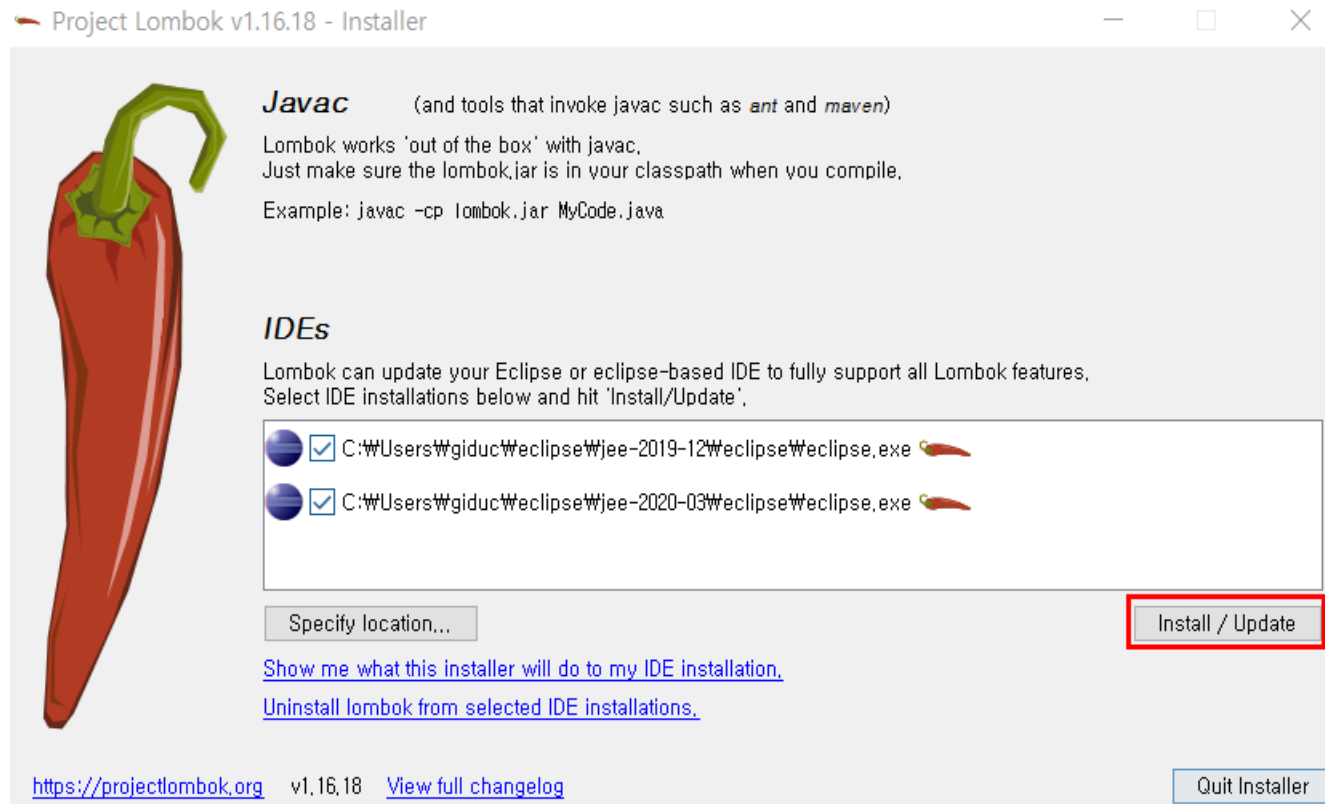
2) 다운로드 받은 lombok-1.18.30.jar 파일 실행 : 주의 - Eclipse or STS가 종료된 상태에서 실행한다.

```
c:\W> cd C:\Users\Wgiduc\Downloads
```

다운로드 받은 위치로 이동

```
c:\W> java -jar lombok.jar
```

lombok 파일 실행




Lombok 환경 구축

3) lombok-1.18.30.jar 파일 실행 결과 확인

lombok 이 설치되고 나면 Eclipse 나 STS 에 lombok.jar 파일이 추가 되어 있는 것을 확인 할 수 있다.

(C:) > programs > spring-tool-suite-3.9.9 > sts-bundle > sts-3.9.9.RELEASE

<input type="checkbox"/> 이름	수정한 날짜	유형	크기
configuration	2020-08-25 오후 1...	파일 폴더	
dropins	2019-06-18 오전 1...	파일 폴더	
features	2020-06-14 오후 2...	파일 폴더	
p2	2019-08-06 오전 1...	파일 폴더	
plugins	2020-06-14 오후 2...	파일 폴더	
readme	2019-08-05 오후 1...	파일 폴더	
.eclipseproduct	2019-06-18 오전 1...	ECLIPSEPRODUCT...	1KB
artifacts	2020-06-14 오후 2...	XML 문서	347KB
eclipsec	2019-06-18 오전 1...	응용 프로그램	120KB
license	2019-06-18 오전 9...	텍스트 문서	12KB
 lombok	2020-08-25 오후 9...	ALZip JAR File	1,407KB
open-source-licenses	2019-06-18 오전 1...	텍스트 문서	1,582KB
<input type="checkbox"/> STS	2019-06-18 오전 1...	응용 프로그램	408KB
STS	2020-08-25 오후 9...	구성 설정	1KB
system_catalog	2020-07-09 오전 6...	XML 문서	47KB

Lombok 환경 구축

2. Lombok이 필요한 프로젝트에 의존 라이브러리 추가

boot02 프로젝트의 pom.xml 파일에 Lombok 라이브러리를 추가 한다.

```
<dependencies>
    <!-- lombok -->
    <dependency>
        <groupId>org.projectlombok</groupId>
        <artifactId>lombok</artifactId>
        <optional>true</optional>
    </dependency>
</dependencies>
```

Lombok 환경 구축

3. DTO 클래스에 annotation 설정

Member.java 파일에 getter, setter 메소드를 만들지 않고, @Getter, @Setter annotation으로 처리한다.

main / java / com / example / demo / model – Member.java

```
package com.example.demo.model;
```

```
import lombok.Getter;
```

```
import lombok.Setter;
```

```
@Getter
```

```
@Setter
```

```
@Data
```

```
public class Member {
```

```
    private String id;
```

```
    private String passwd;
```

```
}
```

Lombok의 어노테이션

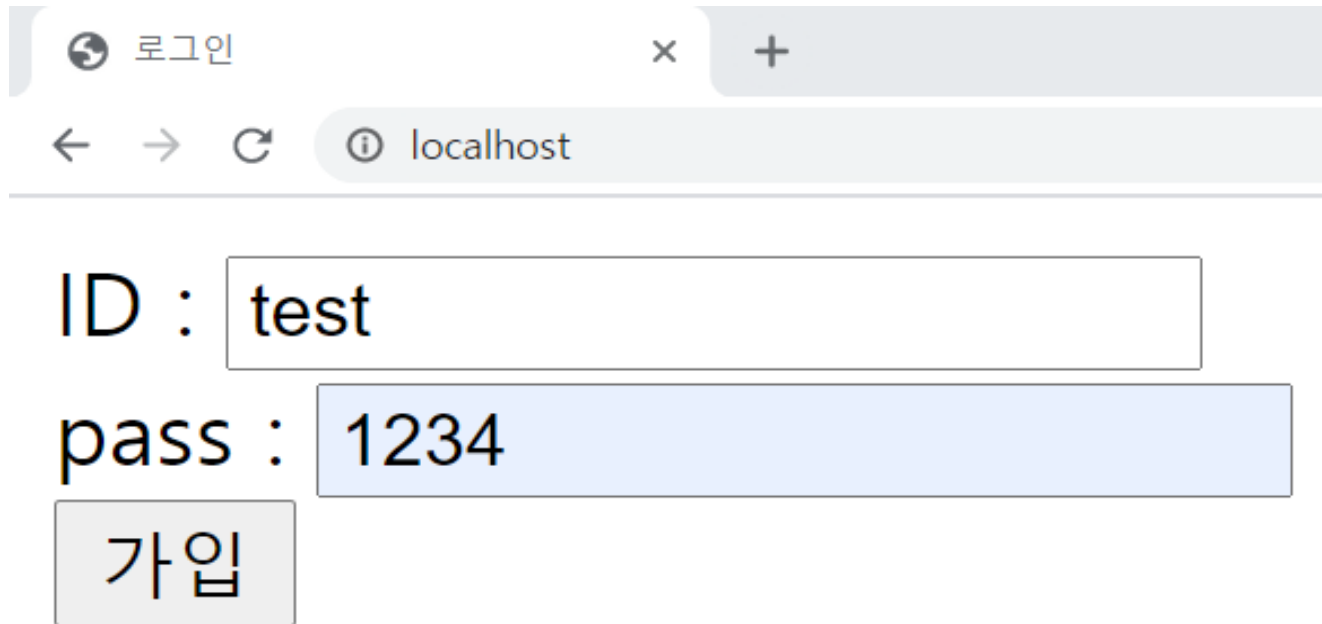
❖ Lombok 의 주요 어노테이션

@Getter	코드가 컴파일 될 때 속성들에 대해서 getter 메소드를 생성
@Setter	코드가 컴파일 될 때 속성들에 대해서 setter 메소드를 생성
@ToString	코드가 컴파일 될 때 속성들에 대해서 toString 메소드를 생성
@Data	@Getter , @Setter , @ToString , @EqualsAndHashCode , @RequiredArgsConstructor 를 한번에 설정해주는 어노테이션

boot02 프로젝트 접속

❖ boot02 프로젝트 접속

boot02 project를 중지하고, 재시작 한 후에 웹브라우저에 `http://localhost` 아래와 같이 요청한다.



The screenshot shows a web browser window with a single tab titled '로그인' (Login). The address bar displays 'localhost'. The login form contains two input fields: 'ID' with the value 'test' and 'pass' with the value '1234'. Below these fields is a button labeled '가입' (Join).

로그인

localhost

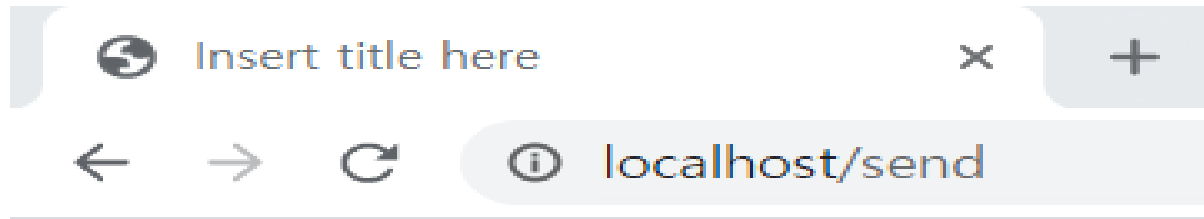
ID : test

pass : 1234

가입

boot02 프로젝트 처리 결과

❖ boot02 프로젝트 처리 결과



ID : test

pass : 1234

Lombok 라이브러리를 사용했을때 같은 결과로 출력되면 성공~!!